Universidad de Cantabria

Departamento de Ingeniería de Comunicaciones

TESIS DOCTORAL

Uso de técnicas de Network Coding y Multi-path para la mejora de las comunicaciones sobre redes malladas inalámbricas

Autor: *David Gómez Fernández*
Director: *Ramón Agüero Calvo*

Santander, marzo de 2015

Certificado del director de la Tesis

D. Ramón Agüero Calvo, Profesor Contratado Doctor de la Universidad de Cantabria en el Área de Ingeniería Telemática,

Hace constar:

Que la Tesis titulada **"Uso de técnicas de Network Coding y Multi-path para la mejora de las comunicaciones sobre redes malladas inalámbricas"** ha sido realizada por **D. David Gómez Fernández** en el Departamento de Ingeniería de Comunicaciones de la Universidad de Cantabria bajo mi dirección, y que reúne las condiciones exigidas a los trabajos de doctorado.

Santander, marzo de 2015

Fdo. Ramón Agüero Calvo

Afiliación

Grupo de Ingeniería Telemática
Departamento de Ingeniería de Comunicaciones
Universidad de Cantabria

Este trabajo ha sido financiado por la beca *FPI BES*-2010-038455 del Ministerio de Economía y Competitividad.

Resumen

El mundo de las comunicaciones se caracteriza en la actualidad por un uso cada vez más extendido de dispositivos portátiles y móviles de gran capacidad (*smartphones*, *tablets*, ordenadores portátiles, etc.) para intercambiar información. La importancia de este fenómeno es tal que el tráfico generado por este tipo de terminales está empezando a superar al correspondiente a los accesos cableados, convirtiéndolos en el medio más utilizado para la conexión a Internet. Si bien es cierto que el uso más habitual de las tecnologías inalámbricas se sitúa en el conocido como último salto, hay que destacar la mayor importancia que adquieren día a día las topologías multi-salto. Vistas en un comienzo como solución de emergencia para extender, de una forma económica y rápida, la cobertura en situaciones donde la infraestructura de red tradicional no era suficiente, cada día surgen nuevos casos de uso en los que el despliegue de redes malladas inalámbricas aparece como una alternativa a los esquemas de transmisión más clásicos.

Por otro lado, los protocolos que se emplean sobre las redes de comunicaciones no evolucionan al mismo ritmo que éstas. El ejemplo más claro es *TCP*, solución de nivel de transporte más extendida, que no se ha adaptado a los cambios que se han ido produciendo en las redes. Desde sus primeras versiones, *TCP* fue concebido para trabajar sobre entornos cableados, en los que la aparición de errores de transmisión no era relevante, por lo que fue diseñado para reaccionar de manera óptima ante el principal problema de dichas redes: la congestión. Su interacción con los canales inalámbricos, con unas características radicalmente diferentes a las de los interfaces cableados, es muy complicada.

Para hacer frente a esta situación, se han realizado muchas propuestas. Desde el punto de vista del propio *TCP*, se han hecho numerosas modificaciones del mismo para sobreponerse a sus limitaciones, y mejorar sus pobres prestaciones sobre entornos inalámbricos. Al mismo tiempo, otras opciones consideran sustituir a *TCP* como solución de transporte orientada a la conexión. Entre las más interesantes se encuentra el protocolo *MPTCP*, llamado a ser la evolución natural de *TCP*, y que aprovecha una circunstancia cada vez más habitual en los dispositivos, que no es otra que la incorporación de múltiples interfaces con los que comunicarse. Por otra parte, las técnicas conocidas como *Network Coding* están llamadas a tener un papel fundamental en las comunicaciones del futuro. Su concepto básico propone un giro radical con respecto al encaminamiento tradicional, convirtiendo los elementos intermedios de la red en entidades activas, capaces de transformar el contenido de los mensajes a medida que los procesa.

Esta Tesis parte de los problemas observados en *TCP*, analizando la viabilidad de algunas de las soluciones más importantes que han surgido en los últimos tiempos. Este trabajo se vale de las técnicas de simulación, y los resultados obtenidos muestran las mejoras que suponen estas alternativas con respecto a los esquemas tradicionales vistos con *TCP*.

En primer lugar, se tratará de mitigar una de las principales carencias a la hora de emular el comportamiento de los canales inalámbricos en simuladores de red, ya que

se tiende a simplificar en exceso la operación de los mecanismos de nivel físico. En esta memoria se presentan dos alternativas, computacionalmente eficientes, capaces de reflejar adecuadamente la naturaleza de las transmisiones sobre este tipo de medios.

Posteriormente, se llevará a cabo un extenso análisis del protocolo *MPTCP* sobre dispositivos multi-interfaz en redes inalámbricas multi-salto. Se pondrá un especial interés en estudiar los mecanismos de control de flujo, que se encargan de balancear la carga entre las diferentes conexiones, en función de la calidad percibida por los enlaces. Asimismo, se extenderá el estudio a topologías más complejas, en las que la multiplexación del tráfico a través de diferentes rutas permite alcanzar cotas superiores de rendimiento.

Finalmente, se estudiará una solución diferente, que utiliza técnicas de *Network Coding* con ópticas diferentes. En primer lugar, se combinará el protocolo *TCP* con una solución *inter-flujo*, mediante la cual se mezclará el tráfico perteneciente a dos flujos en transmisiones únicas, maximizando la utilización del canal y, por tanto, incrementando el rendimiento final. Por otra parte, se utilizará un esquema formado por *UDP* y un mecanismo de codificación aleatoria de los datagramas de un único flujo, mediante el que se pretende enmascarar los errores producidos durante las transmisiones, ofreciendo un servicio fiable, equivalente en esencia al ofrecido por *TCP*.

Abstract

Communications are nowadays characterised by a growing use of high-capacity portable and mobile devices (smart-phones, tablets, laptops, etc.) to exchange information. The relevance of this phenomenon is behind the fact that the traffic generated by this type of terminals has already surpassed the one over traditional wired accesses, making them the most widespread alternative to connect to the Internet. Although it is true that the most frequent use of wireless technologies entails the so-called last hop, it is worth highlighting the importance recently given to multi-hop topologies. When initially proposed, they were seen as emergency solutions to extend, on a quick and economical way, the coverage under circumstances where the traditional network architecture was not enough. However, new use cases are looming everyday, where the deployment of wireless mesh networks appears as an interesting alternative to more legacy schemes.

On the other hand, the protocols that are used over communication networks do not evolve at the same pace. A clear example is *TCP*, the most important transport-level solution, which has not appropriately adapted to the network evolution. From its initial versions, *TCP* was conceived to perform well over wired deployments, where the appearance of transmission errors can be considered as negligible, and it was therefore designed to deal with the most relevant concern of such networks: congestion. Its interaction with wireless channels, with very different characteristics to those exhibited by wired interfaces, is thus rather complex.

In order to deal with this situation, a large number of proposals have been made. From the point of view of *TCP* itself, several modifications were specified to overcome its limitations, aiming to improve its poor performance over wireless links. At the same time, other options consider the substitution of *TCP* as the transport-level connection-oriented solution. Among the most interesting ones, we could highlight the *MPTCP* protocol, which is believed to stand as the natural evolution of *TCP*; it exploits a characteristic that is becoming more frequent in current devices, which incorporate multiple interfaces. On the other hand the so-called *Network Coding* techniques would likely play a key role in future communication scenarios. Their core concept proposes a radical change with respect to more traditional routing schemes, since they provide the intermediate nodes with the capacity of tweaking the packets as they are being processed, as active entities.

This PhD starts from the problems exhibited by *TCP* over wireless networks, assessing the feasibility of some of the most relevant solutions that have recently loomed to overcome them. This work exploits simulation techniques and the obtained results show the improvements brought about by these techniques, comparing them with the performance of the traditional *TCP*.

Before that, the PhD discusses a proposal to overcome one of the key shortcomings of network simulators, which are not able to accurately mimic the behaviour of real wireless channels, since they over-simplify the subjacent mechanisms. This work introduces two

different approaches, computationally efficient, able to appropriately reflect the characteristics of this type of links.

Afterwards, we will present a thorough analysis of the *MPTCP* protocol over wireless multi-hop networks, using multi-interface devices. Special attention will be paid to studying flow control mechanisms, which balance the load between various connections, based on the quality perceived by each of the links/paths. In addition, the analysis will be extended so as to cope with more complex topologies, where traffic multiplexing between different routes would bring about better performances.

Finally, a different solution, based on *Network Coding* techniques, will be studied, using two complementary approaches. First, the *TCP* protocol will be combined with an *inter-flow* solution, which will merge the segments belonging to two different flows in a unique transmission, maximizing the channel utilization and thus increasing the overall performance. Afterwards, we will introduce a scheme comprising the *UDP* protocol and a random coding mechanism, which we apply to packets belonging to one single flow; this combination is able to mask the transmission errors, thus offering a reliable service, comparable to that intrinsically offered by *TCP*.

Índice

Índice de Figuras

Índice de Tablas

1

Introducción

Desde el descubrimiento (a finales del siglo XIX) de que las ondas electromagnéticas podrían ser utilizadas para transmitir mensajes sobre el espectro radioeléctrico, sin necesitar la alternativa cableada que se había venido utilizando hasta la fecha, la presencia de este tipo de comunicaciones inalámbricas ha ido evolucionando a un ritmo cada vez mayor. Ya desde sus orígenes, la aparición de fenómenos tan populares como la radio difusión comercial (aún perdura, habiendo pasado de las tradicionales señales AM y FM hasta radio por *Internet*, etc.) han ido adquiriendo una importancia cada vez mayor. Posteriormente, la irrupción de las tecnologías celulares (cuyos primeros despliegues datan de finales de la década de 1970) ha supuesto un cambio drástico en la percepción de los usuarios hacia estos nuevos paradigmas de comunicación, de tal manera que, a día de hoy, haya más dispositivos móviles activos que seres humanos en el planeta. Un tercer gran paso en la evolución de las comunicaciones inalámbricas es la fusión entre este tipo de tecnologías con la conmutación de paquetes para el intercambio de datos, dando lugar a redes de nueva generación globales, también conocidas como *redes todo-IP*, en las que los dominios cableado e inalámbrico serán gestionados por una plataforma única y común.

De entre todas las posibilidades que aparecen con este tipo de tecnologías, la más predominante es la utilización de enlaces inalámbricos como conexión directa a un elemento de acceso a la red dorsal, lo que se conoce como *último salto inalámbrico*. No obstante, hay que tener en cuenta la presencia, cada vez mayor, de las redes malladas o multi-salto. En un mundo "conectado", cualquier dispositivo cuenta con la tecnología necesaria para comunicarse con otros, apareciendo nuevas formas de comunicación, como las conocidas *Machine to Machine (M2M)* o *Device to Device (D2D)*. En ellas, cada vez son más habituales las transmisiones directas entre dispositivos con *hardware* relativamente sencillo, como sensores, con otros equipos de mayores capacidades, que procesarán los datos emitidos por aquellos. Además, no es difícil encontrar despliegues inalámbricos en entornos donde la infraestructura de red tradicional no esté disponible, como zonas rurales o países en vías de desarrollo, redes vehiculares, y de sensores de diversa naturaleza.

Sin embargo, y a pesar de este giro con respecto al arquetipo más tradicional de red, *Transport Control Protocol (TCP)* se mantiene como el protocolo de nivel de transporte más

extendido, estando presente en multitud de aplicaciones de uso masivo (correo electrónico, transferencia de ficheros, navegación *web*, etc.). Su principal problema es que fue concebido en una época en la que la red estaba formada, prácticamente en su totalidad, por enlaces cableados, en los que los errores derivados de la transmisión tenían un impacto despreciable, lo que permitía atribuir toda pérdida a la congestión de nodos intermedios o routers.

Como contrapartida, las propiedades físicas de los medios inalámbricos los hacen especialmente propensos a inducir pérdidas de paquetes de manera aleatoria, debido principalmente a las condiciones hostiles que presenta el propio canal (obstáculos, interferencias, propagación multi-camino, etc.). Además, las limitaciones de memoria o velocidad de procesado en las etapas iniciales de Internet ya no son tan importantes, por lo que la situación actual es muy diferente. Por otro lado, hay que tener en cuenta que uno de los fenómenos más característicos de una transmisión inalámbrica es la correlación entre pérdidas, que muestran cierta predisposición a aparecer en forma de ráfagas. Estos errores consecutivos tendrán un impacto muy negativo sobre el rendimiento global de *TCP*, que reacciona asumiendo incorrectamente la congestión de los nodos intermedios de la red.

Con el paso de los años han ido apareciendo diferentes modificaciones del protocolo, que han tratado, con mayor o menor éxito, de mitigar la mala reacción de *TCP* ante la pérdida de segmentos (en la literatura se habla de que una tasa de pérdida de un 2 % podría reducir el rendimiento hasta en un 60 %). Ante el impacto causado por los entornos inalámbricos sobre el protocolo *TCP*, la comunidad científica ha presentado siempre un especial interés en la búsqueda de diferentes soluciones, que van desde versiones "mejoradas" del propio protocolo (principalmente los mecanismos de control de flujo y congestión) hasta el desarrollo de alternativas novedosas, diseñadas para combatir eficientemente las limitaciones de *TCP* sobre este tipo de escenarios.

1.1 Motivación y objetivos

Partiendo de los problemas que surgen de la combinación de redes inalámbricas y *TCP*, el objetivo principal de esta Tesis se sitúa en el estudio de diversas técnicas que buscan mejorar el rendimiento de *TCP* sobre entornos inalámbricos, utilizando dos aproximaciones diferentes: por un lado, se estudiará el comportamiento del protocolo *MPTCP*, que nace como una versión evolucionada de *TCP*, que posilibita la transferencia de información a través de varios subflujos de manera simultánea. Aprovechando el hecho de que los dispositivos portátiles incluyen de manera habitual múltiples interfaces, permitiéndoles la conexión a través de diferentes tecnologías, aparece la posibilidad de dividir el tráfico en varios subflujos, lo que permitiría mejorar las prestaciones del *TCP* tradicional, tanto en su rendimiento como en su robustez. Por otra parte, se utilizará una nueva filosofía en la operación y funcionalidad de los nodos intermedios, los cuales no sólo redirigirán la información sin más, sino que además tendrán la capacidad de procesarla y combinarla, en función de una serie de criterios y restricciones. A estas técnicas se las conoce en la literatura

como *Network Coding (NC)*. En concreto, se estudiarán dos soluciones de caracteerísticas opuestas: en primer lugar se combinarán los paquetes pertenecientes a diferentes flujos (NC inter-flujo); por otro lado, como segunda alternativa se contemplará el uso de esquemas de codificación aleatorios para mezclar el tráfico correspondiente a un único flujo.

Hay que destacar que la mayor parte de los desarrollos y análisis llevados a cabo en el marco de esta Tesis se han hecho sobre el simulador `ns-3` como herramienta básica. Uno de los principales problemas que se les achaca a este tipo de plataformas es que no son capaces de reflejar de manera fidedigna los efectos observados en transmisiones sobre entornos inalámbricos reales, condicionando en gran medida la validez de los resultados obtenidos. Esta Tesis también presenta un exhaustivo análisis del comportamiento de un enlace inalámbrico en un entorno de interiores, basado en el estándar *Institute of Electrical and Electronics Engineers (IEEE) 802.11*. Dicho estudio se utilizará para proponer dos modelos de canal alternativos que, utilizando enfoques diferentes, son capaces de reflejar el comportamiento observado.

Teniendo en cuenta los aspectos mencionados anteriormente, este trabajo afronta una serie de objetivos concretos, que aparecen enumerados a continuación.

- Caracterizar el comportamiento de un canal inalámbrico real basado en el estándar *IEEE 802.11*, tanto experimental como analíticamente. Así, se establecerá el rendimiento máximo sobre un enlace inalámbrico punto a punto, corroborándolo posteriormente con medidas reales. Los resultados obtenidos servirán además como secuencias de entrenamiento para configurar los modelos de canal propuestos en el marco de la presente Tesis.

- Implementar, validar y evaluar dos modelos de canal sobre la plataforma `ns-3`, poniendo de manifiesto sus ventajas sobre las soluciones que normalmente se encuentran dentro de los simuladores de red. Para reflejar el comportamiento a ráfagas observado se plantean dos alternativas. La primera de ellas utiliza un proceso oculto de Markov, en el que las transiciones entre los estados de la cadena que lo define emulan las condiciones cambiantes del medio radio. Por otra parte, el segundo de los modelos propuestos se basará en la utilización de un filtro *auto-regresivo* para estimar la calidad de la señal recibida.

- Adaptar una implementación del protocolo *MPTCP* para que funcione correctamente sobre una versión más reciente del simulador, manteniendo la arquitectura propuesta originalmente en la recomendación del *Internet Engineering Task Force (IETF)* correspondiente. Tras ello, se validará su operación a través de simulaciones sobre topologías sencillas, para verificar la validez de los resultados y, por tanto, de la propia implementación. Posteriormente, se estudia la influencia de algunos de los mecanismos de control de congestión que se proponen en el protocolo, su capacidad para distribuir el tráfico simultáneamente a través de múltiples interfaces y, por último,

la habilidad para balancear la carga adecuadamente cuando alguno de los enlaces presente un peor comportamiento.

- Extender el análisis del protocolo *MPTCP* a topologías más complejas. Para ello, se hará uso de tres algoritmos de encaminamiento multi-camino, que establecen múltiples rutas disjuntas entre dos nodos. En primer lugar, se caracterizarán las principales diferencias entre dichos algoritmos, a partir de métricas como el número de caminos encontrados o su longitud. Finalmente, se utilizará el simulador para comprobar las prestaciones que ofrece *MPTCP*, corroborando que es capaz de mejorar el enfoque tradicional de una única ruta empleado por *TCP*.

- Diseñar un módulo independiente, localizado entre los niveles de red y de transporte, que permita la integración escalable de nuevas funcionalidades, que será utilizado para implementar los componentes de *NC*. Se creará un sistema que intercepte los flujos durante su procesado habitual y se habilitará además la creación de técnicas *cross-layer*, conectando la nueva entidad con diferentes puntos de los niveles inferiores.

- Diseñar, implementar, validar y evaluar un protocolo *NC inter-flujo*, que combina segmentos pertenecientes a diferentes conexiones *TCP*, en los nodos intermedios habilitados para tal efecto. Caracterizar de manera detallada el comportamiento de dicho protocolo bajo diferentes configuraciones del mismo, utilizando un conjunto de escenarios canónicos.

- Proponer, evaluar y validar algoritmos para localizar potenciales nodos codificadores en redes malladas inalámbricas, en las que los nodos se despliegan aleatoriamente. Los resultados se utilizarán para cuantificar, utilizando el simulador `ns-3`, el beneficio del protocolo *NC inter-flujo*, al compararle con esquemas de transmisión tradicionales.

- Diseñar, implementar, validar y evaluar un protocolo *NC intra-flujo*. En este caso, a diferencia del anterior, los flujos serán procesados de manera independiente; así, el propio nodo fuente combina paquetes de una misma conexión. Utilizando *User Datagram Protocol (UDP)* como solución a nivel de transporte, este esquema explota la codificación aleatoria de bloques de tamaño fijo para garantizar la recepción, ordenada, de toda la información transmitida por la fuente. Además, se estudiarán los beneficios que pueden aportar la recodificación en los nodos intermedios, que procesan y transforman el contenido de los paquetes codificados que reciben.

Conviene destacar finalmente que toda contribución derivada de esta Tesis, ya sea en forma de código fuente, manual o resultado, está regida por una política *open-source*, y se ha puesto a disposición de la comunidad investigadora, de manera que pueda ser utilizada.

Además, esta Tesis se encuentra enmarcada dentro de dos proyectos del *Plan Nacional de I+D+i*: *C3SEM*, "***C**ognitive, **C**ooperative **C**ommunications and autonomous **SE**rvice **M**anagement*" (TEC2009-1458-C02-01) y *COSAIF*, "***CO**nnectivity as a **S**ervice: **A**ccess for the **I**nternet of the **F**uture*" (TEC2012-38754-C02-02).

1.2 Estructura de la memoria

Para afrontar de manera adecuada los objetivos enumerados anteriormente, el contenido de este documento se divide en tres grandes bloques. Para proporcionar una visión más clara de la estructura de la memoria, la Figura 1.1 clasifica y resume los principales puntos que se van a discutir.

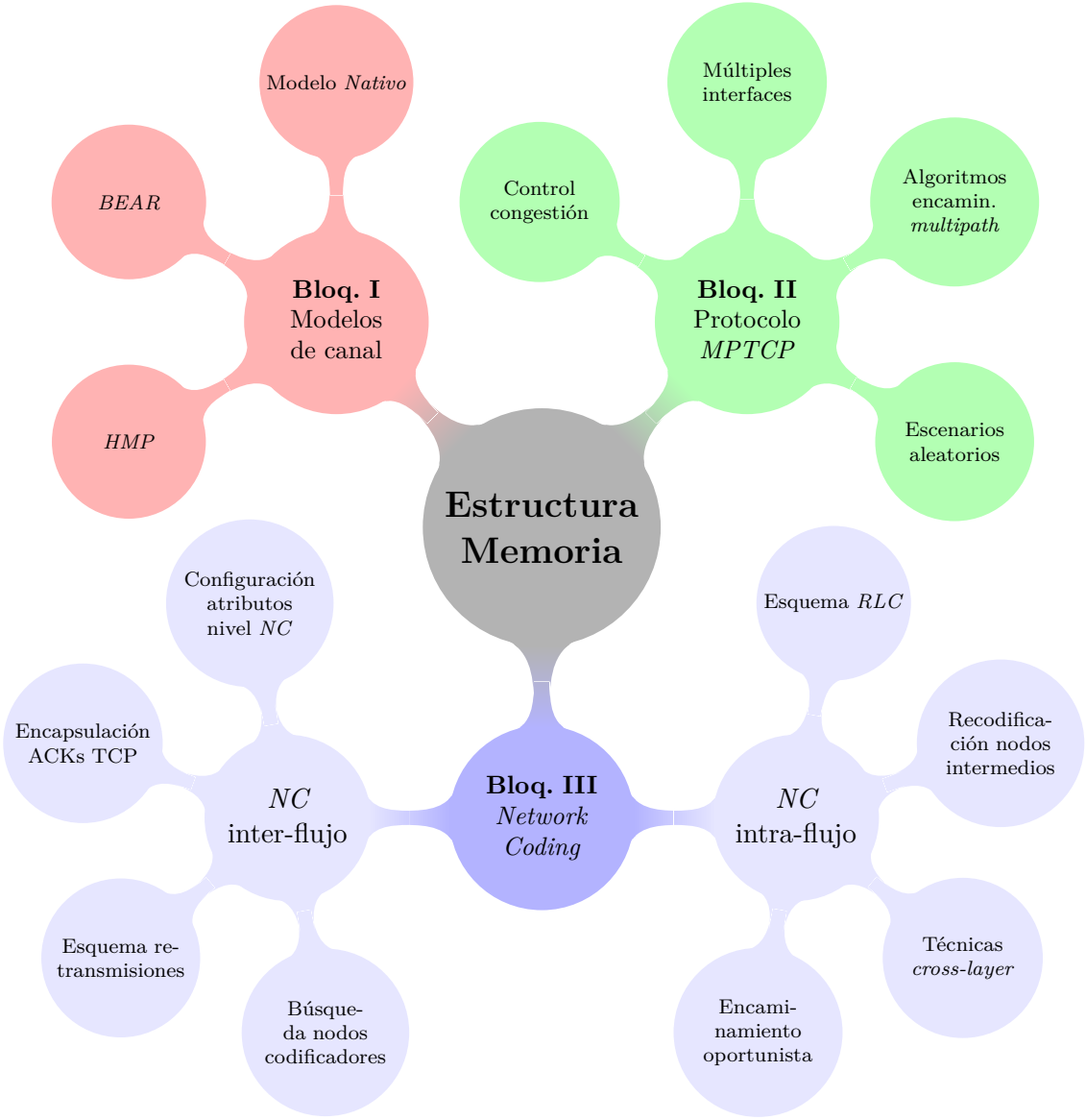


Figura 1.1: Esquema-resumen del contenido de la tesis

- En primer lugar, el **Bloque I** (**Capítulo 2**) pone de manifiesto la incapacidad de los modelos de canal más extendidos para reflejar, de manera fidedigna, el comportamiento observado sobre canales inalámbricos reales, en donde las tramas erróneas

tienden a aparecer agrupadas (a ráfagas). Se exponen dos enfoques diferentes en el marco del simulador `ns-3`: el primero de ellos se apoya en la utilización de procesos ocultos de Markov, mientras que la segunda alternativa hace uso de un filtro auto-regresivo para introducir cierta componente de memoria en la estimación de la calidad del enlace radio.

- Posteriormente, el **Bloque II** (**Capítulo 3**) presenta un estudio detallado de las principales características del protocolo *MPTCP* y su implementación en el marco del simulador `ns-3`. Tomando como punto de partida una versión limitada del protocolo, se llevará a cabo una extensa campaña de simulaciones para caracterizar sus prestaciones, haciendo especial hincapié en los mecanismos de control de congestión que incorpora, así como en la capacidad que tiene para balancear la carga ante situaciones negativas en alguna de sus subconexiones. En paralelo, se ha llevado a cabo un estudio de índole más teórica que, apoyándose en la teoría de grafos, permite la obtención de múltiples rutas entre dos nodos, permitiendo multiplexar simultáneamente el tráfico entre ellas.

- El **Bloque III** introduce la utilización las técnicas *NC* como alternativa para mejorar el rendimiento de *TCP* sobre entornos inalámbricos. En primer lugar, el **Capítulo 4** presenta los conceptos más relevantes, profundizando en las contribuciones que se centran en redes malladas inalámbricas. Se identifican dos grandes alternativas para afrontar los procesos de codificación. En primer lugar, el **Capítulo 5** presenta una solución en la que los nodos intermedios o *routers* combinan segmentos de flujos diferentes. Por otro lado, el **Capítulo 6** enfoca el problema desde otro punto de vista, ya que las operaciones de codificación se llevarán a cabo entre datagramas *UDP* del mismo flujo (*intra-flujo*). Se estudia un esquema de transmisión de bloques codificados aleatoriamente, mediante operaciones algebraicas sobre cuerpos finitos, que permite enmascarar los errores producidos sobre enlaces de baja calidad.

- Finalmente, el **Capítulo 7** concluye la memoria, resumiendo las principales aportaciones que se han realizado durante el desarrollo de esta Tesis. También describe las líneas de investigación que se han abierto a raíz de este trabajo, sugiriendo nuevas áreas de estudio.

Bloque I

Canales inalámbricos

2

Modelado de un canal inalámbrico *IEEE 802.11*

El mundo de las comunicaciones inalámbricas se encuentra inmerso en una continua evolución, habiéndose convertido, por derecho propio, en una parte esencial en el día a día de la sociedad actual. En concreto, el nacimiento de las tecnologías basadas en el estándar *IEEE 802.11* ha dado lugar a un nuevo nicho de mercado impensable hace no mucho tiempo atrás. A pesar de no poder competir con su equivalente cableado en términos de rendimiento (el canal inalámbrico siempre se mostrará más hostil que aquél basado en un medio guiado), las ventajas inherentes a este tipo de comunicaciones (e.g. flexibilidad, movilidad, reducción de costes de infraestructura, etc.) hacen que su empleo se haya asentado de forma paulatina, viendo como su factor de penetración presenta unos niveles realmente elevados. La importancia de este estándar (y sus derivados) es tal, que la cantidad y variedad de dispositivos que soportan esta tecnología es inabarcable: desde los clásicos equipos portátiles, precursores en el uso de interfaces *WiFi*[1] (fueron además los grandes responsables en la eliminación de la "tiranía del cable"), pasando por los terminales móviles (denominados comercialmente *smartphones* en sus versiones con mayor capacidad), que se han convertido en auténticos ordenadores de bolsillo, hasta dispositivos de lo más variopinto, como pueden ser básculas, pulseras cuantificadoras, consolas portátiles, impresoras, tarjetas de memoria, y así un largo etcétera.

Como consecuencia directa de este rotundo éxito, la comunidad científica necesita abastecerse de las herramientas necesarias para poder entender de manera detallada el comportamiento de este tipo de comunicaciones. A primera vista, la opción más lógica para su estudio parece la experimentación en entornos reales, donde la utilización de equipamiento permite la obtención más rápida y directa de resultados. No obstante, las limitaciones asociadas a este tipo de análisis (e.g. escalabilidad, reproducibilidad, entre otros factores) hace que muchas veces haya que recurrir a otras opciones más viables a

[1]El término *Wireless Fidelity (WiFi)*, utilizado a la hora de referirse al estándar *IEEE 802.11*, es tanto (o más) conocido que el propio código que define a la recomendación, sobre todo en aquellos entornos alejados del campo técnico de las telecomunicaciones.

la hora de generar datos útiles. En este punto entra en juego la metodología basada en la simulación, cuyo principal objetivo es el de complementar el uso de caracterizaciones reales, cuando éstas no proporcionan soluciones viables a los problemas planteados.

Por otra parte, uno de los argumentos más clásicos a la hora de postularse en contra de las plataformas simulación se sustenta en la baja precisión que la mayoría de los modelos de propagación es capaz de proporcionar a la hora de replicar el comportamiento observado sobre un canal de comunicaciones real, donde normalmente se opta por "simplificar" el modelo con el fin de obtener una mayor eficiencia computacional o, en su defecto, simulaciones más rápidas. En este punto merece la pena recalcar la división existente en el mundo de los simuladores y, en concreto, para este tipo de modelado: por una parte, se mantiene la línea más purista, centrada en los *efectos físicos* producidos durante las transmisiones, utilizando avanzadas (y complejas) técnicas que garantizan altos niveles de precisión (como por ejemplo, el modelado de rayos o la teoría electromagnética). Sin embargo, el tiempo que requieren para generar resultados es demasiado elevado, lo que, dependiendo de la red a analizar, hace que su utilización no sea la más adecuada. En el otro lado de la balanza aparecen los simuladores de red, cuyo eje central gira en torno a los protocolos, algoritmos y mecanismos propios de los niveles superiores. Dentro de esta categoría, suele ser habitual el análisis de escenarios con una gran cantidad de nodos, siendo la escalabilidad uno de sus principales requisitos y objetivos, ya que pueden darse casos con cientos e incluso miles de nodos desplegados sobre el escenario. Por lo tanto, se antoja imprescindible que los mecanismos utilizados en los niveles inferiores (físico y de enlace) sean capaces de reflejar un comportamiento similar al observado en el mundo real, manteniéndose al mismo tiempo dentro de unos márgenes de tiempo de simulación razonables. En definitiva, el modelo de canal óptimo será en el que mejor se adecúe a cada contexto, proporcionando la mejor relación entre precisión y complejidad.

La creciente capacidad computacional de los dispositivos deriva en el desarrollo de técnicas cada vez más complejas para replicar el comportamiento observado en un entorno real, impensable hace no muchos años, donde las limitaciones impuestas por la propia tecnología restringían la libertad de desarrolladores e investigadores para alcanzar sus objetivos. Como consecuencia directa han ido surgiendo a lo largo de los años nuevos y potentes simuladores, entre los que destaca `ns-3`, utilizado en el desarrollo de la presente Tesis. Sucesor natural del abandonado[2] `ns-2`, permite a los investigadores la sencilla inclusión de nuevos módulos gracias a su gran flexibilidad y modularización. Sin embargo, y a pesar de su incipiente popularidad (aunque `ns-2` se mantiene como el simulador con mayor número anual de publicaciones, `ns-3` mantiene una tendencia creciente y cada vez más usuarios lo utilizan), los modelos de propagación disponibles en las versiones oficiales del simulador (y en general, de todos) siguen estando lejos de ofrecer un comportamiento realista, incapaces de emular adecuadamente aquellos fenómenos que se observan en un canal inalámbrico real.

[2]Su última actualización en los repositorios oficiales data de *mayo del 2009*.

En la actualidad pueden encontrarse redes *IEEE 802.11* en prácticamente cualquier entorno (de hecho, es incluso más complicado encontrar espacios libres de ellas, sobre todo si la densidad de población es elevada), incluidos grandes espacios abiertos, lugares públicos o naves industriales. Sin embargo, los escenarios interiores (marcados principalmente por el uso doméstico o de oficina) aparecen como los entornos en los que más habitualmente se encuentran este tipo de redes. Entre sus propiedades más características destacan los factores hostiles que presentan, como muros, puertas, inmobiliario, gente moviéndose, por citar sólo algunos ejemplos. Sin duda alguna, la contribución de todos estos elementos no hará sino penalizar el rendimiento de las comunicaciones inalámbricas, dando lugar a un claro comportamiento "a ráfagas", donde los eventos que derivan en la pérdida de tramas no son independientes, poniéndose de manifiesto un claro efecto "memoria". Es en este punto donde aparece la necesidad de proporcionar un acercamiento a este comportamiento tan característico, a través de mecanismos que sean capaces de reflejar fielmente los fenómenos producidos en escenarios interiores. En concreto, en este trabajo se presentan dos aproximaciones diferentes al modelado del canal inalámbrico: en primer lugar, se utilizarán las herramientas matemáticas conocidas como procesos ocultos de Markov para adaptar un modelo a las diferentes condiciones que pueden darse en un canal inalámbrico. Es preciso destacar la principal diferencia con otras propuestas que comparten el principio de utilización de *Hidden Markov Processs (HMPs)*: mientras que en éstas su funcionalidad se restringe estrictamente a la secuencia de tramas utilizada para configurar sus parámetros fundamentales, en el caso propuesto en esta Tesis los procesos presentan una nueva caracterización temporal, permitiendo desacoplarlos de las trazas que los modelan y proporcionando un rendimiento elástico que se adapta a diferentes configuraciones de tráfico. Por otro lado, a partir del trabajo llevado a cabo originalmente por Agüero *et al.* [Agüe10], se introduce *Bursty Error model based on an Auto-Regressive filter (BEAR)*, el cual hace uso de un filtro auto-regresivo, popularmente conocido por su acrónimo, *Auto-Regressive (AR)*, para estimar la calidad de la señal recibida, introduciendo así el factor memoria usualmente ignorado por los modelos más tradicionales.

El presente capítulo se centrará en el estudio y la caracterización de un canal inalámbrico basado en el estándar *IEEE 802.11*, que será utilizado como punto de partida para el resto del desarrollo del trabajo. Se detallarán los aspectos descritos a continuación:

- En primer lugar se procederá a estudiar analíticamente el comportamiento de una transmisión producida a través de un canal inalámbrico *IEEE 802.11*, utilizando para ello los parámetros definidos en el estándar [IEEE802.11]. Se utilizará la topología más básica como referencia, donde un nodo transmisor se comunicará directamente con un receptor, en un escenario "limpio" y libre de interferencias. Con este sencillo análisis se establecerán los umbrales máximos de rendimiento alcanzables por la recomendación *IEEE 802.11b*, obtenido en enlaces punto a punto sin presencia de errores de propagación.

- Posteriormente, replicando los resultados obtenidos en la etapa anterior, se completará una campaña *experimental* sobre un escenario real, que será posteriormente utilizada para evaluar la fidelidad de los modelos de canal diseñados con el fin de emular los fenómenos observados en este tipo de entornos.
- Tras la caracterización del canal inalámbrico en un entorno real, se procederá a describir los *dos modelos* diseñados a partir de los resultados obtenidos en la campaña experimental. Como se ha mencionado anteriormente, el primero de ellos, *HMP*, basará su funcionamiento en el uso de procesos ocultos de Markov, mientras que el segundo, *BEAR*, se centra en la estimación de la relación señal a ruido recibida con la ayuda de un filtro *AR*.
- Con un criterio puramente comparativo, se incorpora al análisis una *tercera solución*, incluida en el conjunto de opciones disponibles en la plataforma de simulación `ns-3` y definida como modelo *Nativo*. Su configuración se establecerá para mantener un cierto paralelismo con *HMP* y *BEAR*.
- Una vez descritas todas las opciones a la hora de imitar el comportamiento del canal inalámbrico, se llevará a cabo un *profundo análisis sistemático* bajo el simulador `ns-3` que permita caracterizar y comparar las prestaciones obtenidas con estos tres modelos, tanto en términos de rendimiento como de eficiencia computacional.

2.1 Comportamiento de un canal *IEEE 802.11* en escenarios interiores

Antes de proceder a introducir las aproximaciones realizadas para emular un canal inalámbrico *real*, en este punto se tratará de analizar su comportamiento a través de dos vertientes complementarias: primero, se obtendrán analíticamente los valores máximos de rendimiento en un escenario saturado ideal; posteriormente, y bajo un proceso experimental, se caracterizará el comportamiento real de un canal inalámbrico en un escenario en interiores. A partir de esta caracterización, se definirá un marco de referencia con el que comparar la fidelidad mostrada por los diferentes modelos de canal presentados posteriormente. Además, los resultados obtenidos en esta campaña de medidas tendrá una funcionalidad *dual*, ya que serán utilizados para configurar adecuadamente a los dos modelos de canal que se proponen en esta Tesis: *HMP* y *BEAR*.

2.1.1. Caracterización analítica

En un primer acercamiento, se caracterizan los límites del rendimiento alcanzados por un enlace directo entre dos nodos (con un único salto entre transmisor y receptor), sin que se produzcan pérdidas en el canal y asumiendo que no haya ninguna fuente externa

Tabla 2.1: Parámetros del control de acceso al medio *DCF* en *IEEE 802.11b*

Parámetro	Valor
Duración *slot* (Δ)	20 μseg
SIFS	10 μseg
DIFS	$2\,\Delta + SIFS = 50\ \mu seg$
CW_{min}	32
CW_{max}	1024
Retransmisiones *MAC* (R)	3
Preámbulo *PLCP* (largo)	192 μseg
R_{datos}	11 *Mbps*
R_{ACK}	2 *Mbps*

de interferencia. En el marco concreto de la presente Tesis, los niveles inferiores (físico y de enlace) se ajustarán a la recomendación *IEEE 802.11b*, cuyos principales parámetros se recogen en la Tabla 2.1.

El estándar *IEEE 802.11* establece como esquema de acceso al medio por defecto el método *Distributed Control Function (DCF)*, función básica que proporciona un servicio de tipo *best-effort*. Se trata de un mecanismo distribuido que se basa en el protocolo *Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA)*, típicamente utilizado en canales compartidos con acceso basado en contienda. Su funcionamiento se resume en la Figura 2.1: cuando una estación quiere transmitir una trama, monitoriza la actividad del canal; si lo encuentra libre durante un intervalo de tiempo denominado *Distributed Inter-Frame Space (DIFS)*, la estación pasa a una fase de contienda. Por el contrario, si el medio está ocupado, la estación continúa escuchando al canal hasta que lo encuentre libre durante un tiempo *DIFS*. En ese momento, inicializa un temporizador, esperando un número aleatorio de ranuras temporales o *slots* antes de comenzar con su transmisión. Dicho temporizador se va reduciendo en tanto en cuanto el canal siga libre, congelándose cuando deje de estarlo y volviéndose a reactivar al liberarse nuevamente durante un intervalo *DIFS*. Cuando el temporizador llega a cero, la estación podrá iniciar la transmisión. El número de ranuras o *slots de backoff* a esperar es una variable aleatoria uniformemente distribuida en el intervalo $[0, CW - 1]$, donde *Contention Window (CW)* es la ventana de contención (en un primer intento de transmisión, $CW = CW_{min}$), que se duplica cada vez que se retransmite una trama, hasta alcanzar el valor máximo, CW_{max}, según un algoritmo exponencial binario, o hasta sobrepasar el número máximo de retransmisiones, lo que conlleva a descartar la trama. Dichos límites, así como la duración de la ranura, dependen de la especificación de capa física utilizada y de los valores predefinidos por el fabricante (ver Tabla 2.1). Tras la correcta recepción de una trama, el receptor envía el correspondiente reconocimiento o *ACK*, después de esperar un tiempo de guarda denominado *Short Inter-Frame Space (SIFS)*, que debe ser menor que el *DIFS*, evitando así que otra estación comience a transmitir información antes de la recepción del *ACK* en el transmisor (típicamente, $DIFS = 2\,\Delta + SIFS$). Una vez recibido éste, si la estación transmisora tiene más

tramas por enviar, deberá esperar de nuevo un tiempo aleatorio de *backoff*, incluso si el medio se encuentra libre después de un *DIFS*, para evitar la captura del canal. Aunque las diferentes extensiones del estándar han ido añadiendo técnicas adicionales que modifican el comportamiento del acceso al medio (e.g. envío de tramas agregadas, transmisión de una ráfaga sin necesidad de volver a contender por el acceso al canal, reconocimiento en bloques, etc.), el estándar *IEEE 802.11* mantiene la definición del mecanismo *DCF* en su versión más básica.

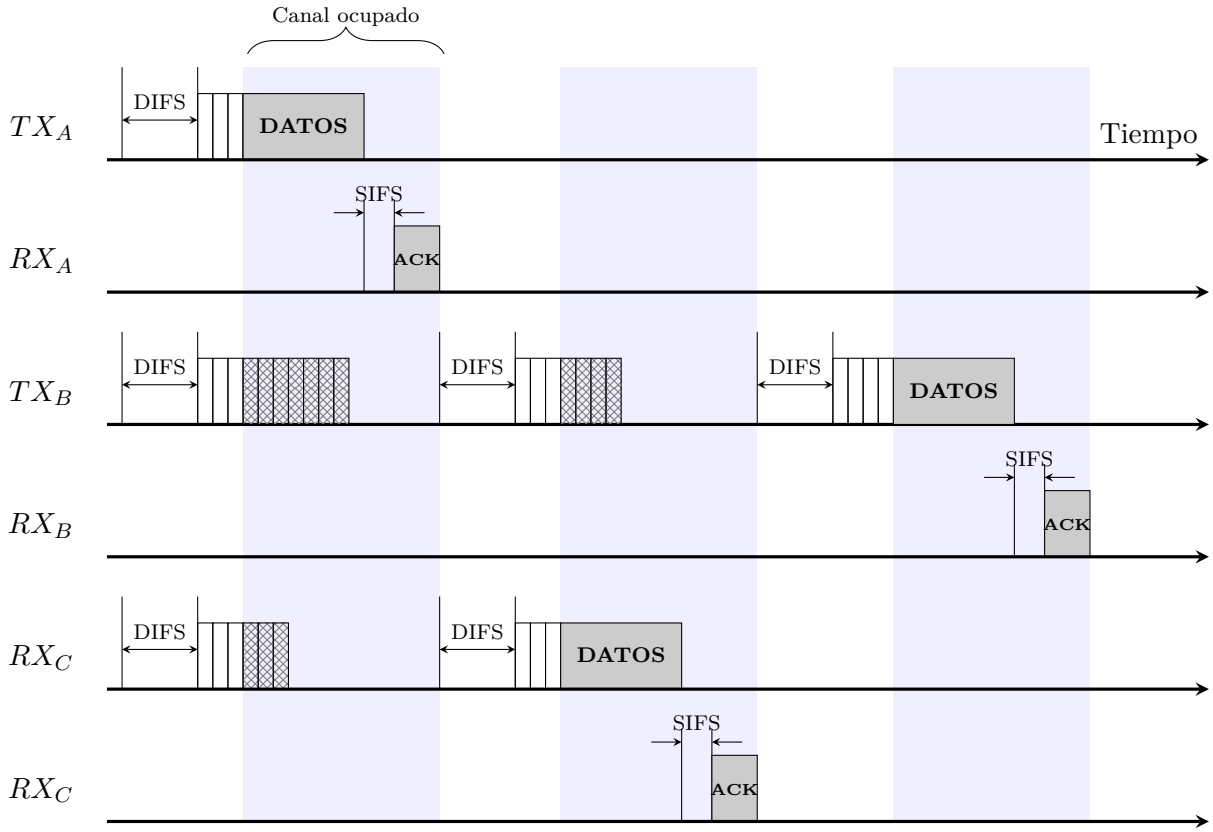


Figura 2.1: Mecanismo de acceso al medio *DCF* en *IEEE 802.11*

En base al esquema descrito hasta este punto, se puede modelar analíticamente el retardo medio introducido por los niveles inferiores del estándar *IEEE 802.11b*. Este intervalo abarcará desde el primer instante en el que el nodo escucha al canal hasta que recibe el último *bit* correspondiente al *ACK* de la estación receptora. Asumiendo que no existen otras estaciones contendiendo por el acceso al canal, la no presencia de interferencias externas y que no se producen errores derivados de la transmisión, el retardo medio, $\overline{\tau_{ideal}}$, se obtiene, en función de la longitud de la información transmitida, a partir de la Ecuación (2.1). Como ya se han descrito anteriormente, los parámetros *DIFS* y *SIFS* marcan los tiempos de guarda previos al acceso al canal para transmitir las tramas de datos y de reconocimiento, respectivamente. $\overline{BO_i}$, por su parte, representa el valor medio de la

ventana de contención en función del número de retransmisión de la trama[3]. Además, el nivel físico define el protocolo *Physical Layer Convergence Protocol (PLCP)*, que añade una nueva cabecera a la trama utilizada para transportar la información relativa a las tasas binarias utilizadas y la longitud de la trama. En total, consta de 24 octetos, teniendo como particularidad que son transmitidos a la menor velocidad posible, 1 *Mbps*, suponiendo un total de 192 μs en cada trama (hay que tener en cuenta que se envía una cabecera *PLCP* asociada a la trama de datos y otra para el *ACK*). Otro elemento a transmitir está ligado a las cabeceras introducidas por los diversos protocolos que intervienen en la comunicación, introduciendo una sobrecarga adicional. En el caso que se está analizando, que utiliza *UDP* como nivel de transporte, este *overhead* se corresponde con los 28 *bytes* de la cabecera de nivel *Medium Access Control (MAC)* del estándar *IEEE 802.11*, 8 del protocolo *Logical Link Control (LLC)*, 20 del nivel *Internet Protocol (IP)* y otros 8 octetos correspondientes a *UDP*. Posteriormente se tiene en cuenta la porción de información correspondiente a los datos, utilizando (al igual que en el proceso de envío de las cabeceras) una tasa binaria R_{datos}, necesitando un tiempo total[4] dado por $(t_{TX})^{datos}$. Tras la transmisión de la trama de datos, se procederá al envío de un *ACK* por parte del receptor, cuyo proceso consta de la espera durante el intervalo de guarda *SIFS*, el tiempo t_{PLCP} y la duración asociada al envío del reconocimiento, con una longitud de 14 octetos de longitud. Es importante destacar que esta transmisión se realiza a una tasa binaria R_{ACK} menor que la empleada para las tramas de datos.

$$\begin{aligned}\overline{\tau_{ideal}}(L) = DIFS + \overline{BO_0} + t_{PLCP} + \ t_{TX_{Cabeceras}}(L_{OH}) + t_{TX_{datos}}(L) + \\ + SIFS + t_{PLCP} + t_{TX_{ACK}}\end{aligned} \tag{2.1}$$

Desde otro punto de vista, estas contribuciones se pueden clasificar en dos grupos: por una parte se encuentran los tiempos cuya duración es conocida a priori (deterministas), $\tau_{det}(L)$, cuyas únicas variables son las longitudes de las cabeceras añadidas por los diferentes protocolos y la propia información que proviene del nivel de aplicación. Por otro lado, la elección del número de ranuras a esperar (ventana de contención) se modelará con una variable aleatoria uniforme discreta, por lo que su valor medio se obtiene fácilmente a través de la Ecuación (2.2), donde se aprecia el crecimiento exponencial binario tras cada retransmisión. Se trata de una función con dos posibles valores: en primer lugar, tras un intento de transmisión fallido, el tamaño de la ventana de contención se duplicará hasta *(1)* alcanzar el valor CW_{max} o *(2)* superar el número de reintentos; por otra parte, en el

[3]En el caso concreto de un canal ideal, no será necesario retransmitir trama alguna, por lo que se utiliza únicamente el valor $\overline{BO_0}$.

[4]Este campo dependerá en buena parte de la longitud recibida de los niveles superiores, por lo que podrá ser calculado como $(t_{TX})^{datos} = (L_{datos} \cdot 8)/R_{datos}$

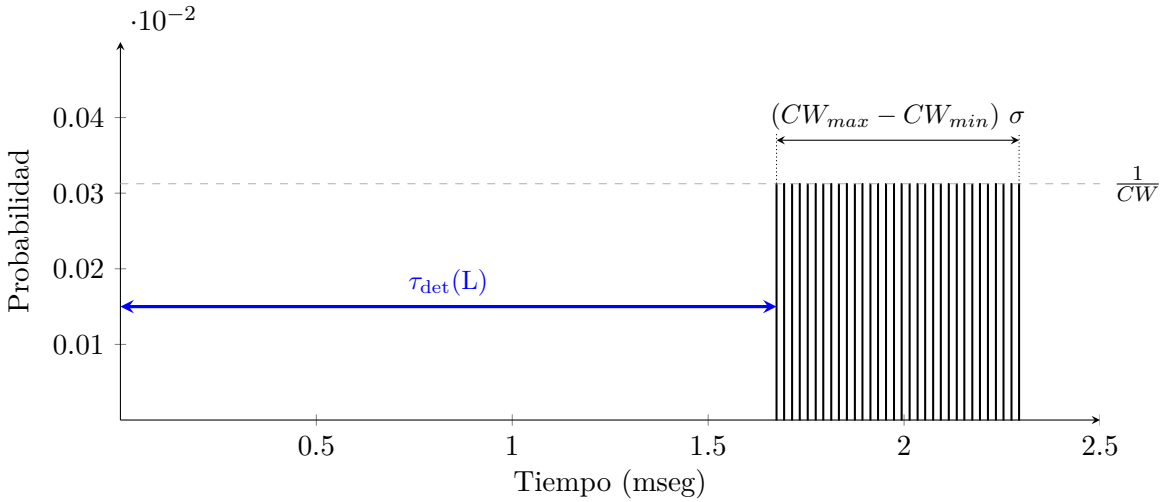


Figura 2.2: Función densidad de probabilidad del retardo de un enlace ideal *IEEE 802.11*

supuesto caso de que se cumpla la condición *(1)*, pero queden más intentos de retransmisión pendientes, la ventana mantendrá el valor de CW_{max} durante el resto de iteraciones.

$$\overline{BO_i} = \begin{cases} \dfrac{CW_{min} \cdot 2^i}{2} \cdot \Delta - 1, & CW_{min} \cdot 2^i < CW_{max},\ i < R \\ \dfrac{CW_{max}}{2} \cdot \Delta - 1, & CW_{min} \cdot 2^i \geq CW_{max},\ i < R \end{cases} \tag{2.2}$$

Conocida la naturaleza estocática que modela este proceso de transmisión en condiciones ideales, se puede modelar asimismo el *jitter* o la varianza del retardo, tal y como muestra la Ecuación (2.3).

$$\sigma^2 = \frac{1}{CW_{min}} \sum_{s=0}^{CW_{min}-1} (\tau_{det}(L) + s \cdot \Delta)^2 - (\overline{\tau_{ideal}}(L))^2 = \frac{((CW_{min} - 1) \cdot \Delta)^2}{12} \tag{2.3}$$

Una vez definidas todas las contribuciones que forman una transmisión en un enlace ideal, la Figura 2.2 ilustra la *función densidad de probabilidad (fdp)* del retardo medio del esquema *DCF*, donde se aprecia claramente la componente aleatoria inducida por el mecanismo de *backoff*.

Además, si la tasa de llegada de paquetes de información a los niveles inferiores es igual o mayor que la propia capacidad del canal, se podrá asumir que existen condiciones de saturación, por lo que siempre habrá, como mínimo, una trama esperando a ser transmitida en la cola de salida del nivel *MAC*. En estas situaciones, en las que no se producen tiempos

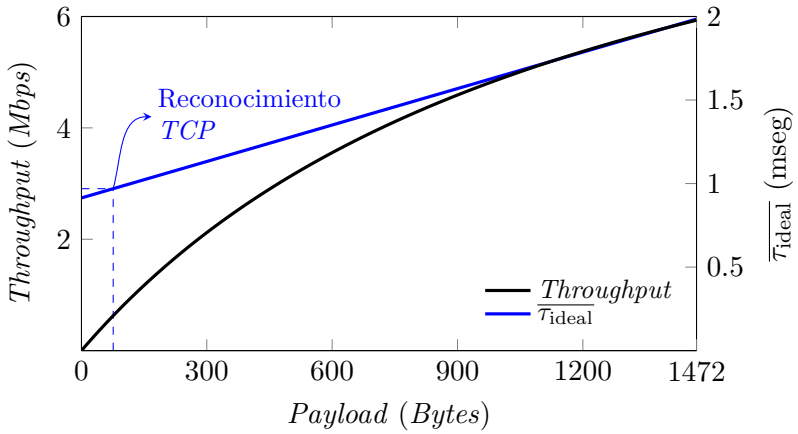


Figura 2.3: Caracterización del rendimiento máximo (*throughput* y retardo) en un enlace *IEEE 802.11* (canal ideal)

ociosos en el medio, se podrá establecer una relación directa entre el retardo medio ($\overline{\tau_{ideal}}$) obtenido en la Ecuación (2.1) y el *throughput*, tal y como define la Ecuación (2.4).

$$Throughput(L) = \frac{L \cdot 8}{\overline{\tau_{ideal}}(L)} \tag{2.4}$$

La Figura 2.3 representa los valores máximos de rendimiento en función de la longitud de los datos útiles. Como es lógico, a medida que esta longitud aumenta, el *throughput* observado será también mayor, debido a que el porcentaje de sobrecarga se reduce (por ejemplo, con 1472 octetos de información el *overhead* introducido será del 47 %). A modo de ejemplo, se muestra el retardo medio asociado a la transmisión de un reconocimiento *TCP*, elemento que tendrá una importancia notable en futuros análisis.

Sin embargo, un escenario real presenta condiciones adversas que pueden derivar en que una trama no llegue correctamente a su destino. Como se ha podido deducir de la descripción realizada hasta el momento, si la estación transmisora no recibe un *ACK* transcurrido un tiempo dado, la trama es retransmitida, de acuerdo con las reglas establecidas en el proceso de *backoff* exponencial binario. Por lo tanto, al retardo medio ideal ($\overline{\tau_{ideal}}$) obtenido habría que añadirle una nueva contribución, ϵ , de naturaleza aleatoria, que venga a modelar los efectos derivados de los errores en la transmisión. Como consecuencia, la expresión para el retardo medio sobre un canal real será la que se muestra en la Ecuación (2.5). Más adelante se estudiará el impacto de estos errores de manera más exhaustiva.

$$\overline{\tau_{real}} = \overline{\tau_{ideal}} + \epsilon\,(\text{Colisiones, propagación, interferencias}) \tag{2.5}$$

Tabla 2.2: Parámetros de las tarjetas empleadas durante el experimento

Parámetro	Valor
Tasa binaria (datos)	11 *Mbps* (*CCK*)
Tasa binaria (control)	2 *Mbps* (*DQPSK*)
RTS/CTS	Desactivado
Número de retransmisiones *MAC*	3
Modo	*Ad-Hoc* (*pseudo-IBSS*)

2.1.2. Caracterización experimental

A continuación se describirá, con un mayor nivel de detalle, el procedimiento empleado para "cuantificar" las transmisiones sobre el medio inalámbrico. La campaña de medidas consistió en la utilización de dos equipos portátiles con el sistema operativo `Linux` instalado. El proceso a seguir para la realización del experimento se describe a continuación: uno de los nodos hizo las veces de transmisor (TX), mientras que el otro ejercía el rol de receptor (RX). Ambos terminales utilizaban tarjetas inalámbricas modelo "*WaveLAN 11 Mbps Lucent/Orinoco PCMCIA*", basadas en la extensión del estándar *IEEE 802.11b* [IEEE802.11], cuyos principales parámetros se resumen en la Tabla 2.2.

La Figura 1 en [Agüe10] muestra un plano del escenario en cuestión, en el que la distancia (en línea recta) entre los equipos transmisor y receptor es de, aproximadamente, 15 metros. Esta disposición emula un típico escenario de interiores, como puede ser una oficina en la que el router de acceso y los equipos se encuentran en diferentes salas, por lo que no hay línea de vista entre los terminales, existiendo además múltiples obstáculos (e.g. paredes, objetos metálicos como escaleras o estanterías, etc.) entre ambos. Del mismo modo, no se restringió el paso a la zona durante la realización de los experimentos, por lo que la presencia de personas podría afectar de algún modo al proceso de medida. Otro factor a tener en cuenta es que, durante la realización de las medidas, no existía ningún tipo de fuente de interferencia co-canal, ya fuere otra celda *IEEE 802.11* interferente, otro equipo conectado a la misma red en infraestructura o cualquier otro tipo de dispositivo que trabaje en la banda de los 2.4 *GHz*.

En lo que respecta a la configuración del software, se utilizó la aplicación `Iperf` [Iperf] para sintetizar tráfico a nivel de aplicación, enviando paquetes hacia los niveles inferiores a una tasa mayor que la soportada por el canal[5], con lo que se aseguraba la presencia de al menos un paquete esperando a ser transmitido en el *buffer* del nivel *MAC*. De este modo, puede afirmarse que el cuello de botella del sistema es el propio canal inalámbrico, del cual se desea conocer su rendimiento máximo. La capa de aplicación generaba un total de 10000 paquetes de 1472 *bytes* de longitud (límite máximo sin fragmentación de un datagrama *UDP* con un *Maximum Transfer Unit*, *MTU*, de 1500 *bytes*, definido por

[5]Sin llegar a desbordar los *buffers* de las capas más bajas, asegurando que toda pérdida fue debida a la propagación a través del medio inalámbrico.

Tabla 2.3: Rendimiento *UDP* de una transmisión sobre un enlace saturado (15 medidas independientes, ordenadas de forma ascendente)

#	*Thput* [*Mbps*]	*FER*	*PER*	*EFB*	
				Med.	*Máx.*
1	0.82	0.814	0.5	15.975	2759
2	1.34	0.709	0.314	5.929	1035
3	1.49	0.676	0.297	7.502	1229
4	2.32	0.53	0.146	3.644	1927
5	2.33	0.517	0.179	6.217	821
6	2.72	0.465	0.105	3.014	383
7	3.58	0.331	0.058	2.6	258
8	3.76	0.301	0.059	3.408	259
9	3.80	0.298	0.127	4.836	219
10	4.00	0.268	0.044	2.767	320
11	4.04	0.261	0.05	3.065	321
12	4.79	0.163	0.025	2.633	144
13	5.50	0.069	0.012	3.136	75
14	5.96	0.014	0.002	2.84	16
15	5.99	0.013	0.001	1.361	7

Ethernet) a la estación receptora, a una velocidad binaria de 11 *Mbps* (tasa bruta en el nivel de aplicación de la estación transmisora), provocando en el canal una situación de saturación.

Para poder monitorizar de manera individual cada uno de los eventos producidos en la recepción de las tramas, el controlador de las tarjetas inalámbricas tuvo que ser modificado, permitiendo capturar y registrar los estadísticos más relevantes de la transmisión en un fichero traza (en el nodo receptor). En concreto, se recogieron los siguientes datos: *(1)* instante en el que recibió la trama (*timestamp*), *(2)* un valor binario que establecía si la trama había llegado al receptor de manera correcta o corrupta (en otras palabras, si se había producido un fallo en la comprobación del *Cyclic Redundance Check*, *CRC*) y *(3)* la calidad de la señal recibida, medida a partir de la relación señal a ruido o *Signal to Noise Ratio (SNR)*. Con respecto a este último parámetro, es importante remarcar dos aspectos: por un lado, su estimación se realiza durante la fase de recepción de la cabecera *PLCP*, sacrificando la precisión que se lograría con un cálculo a lo largo de toda la trama (aunque el coste, tanto computacional como energético, sería mayor); además, las tarjetas empleadas utilizan un valor entero, reduciendo la exactitud del proceso. Entre todos los experimentos realizados, la Tabla 2.3 resume los resultados más característicos de las 15 medidas que presentaron un mayor interés. En particular, el rendimiento de las transmisiones se interpreta a partir de los parámetros que se describen a continuación.

- ***Throughput* (caudal eficaz)**. Métrica global de rendimiento, obtenida a partir de la división del número de *bits* de información entre el tiempo total de la transmisión. A través de este estadístico se refleja la tasa útil medida en el receptor. Concretamente, los valores representados en este capítulo miden el *throughput* obtenido en el nivel de enlace, asumiendo que la sobrecarga introducida por las cabeceras de las capas superiores es información útil (conocido popularmente como *payload*).

- **Tasa de error de trama - *Frame Error Rate (FER)***. Valor que refleja el porcentaje de recepciones erróneas (medido en el nivel físico) frente al número total de tramas recibidas en el nodo. Es importante destacar en este punto la estrecha relación que mantiene este parámetro con el *throughput*, ya que la pérdida de tramas penalizará el rendimiento de las transmisiones.

- **Tasa de error de paquete - *Packet Error Rate (PER)***. Este estadístico, medido a nivel de red, refleja el porcentaje de paquetes que el mecanismo de retransmisión del estándar *IEEE 802.11* no ha sido capaz de recuperar, dando lugar a la pérdida definitiva de la información. Como se ha comentado anteriormente, el nivel *MAC* del estándar *IEEE 802.11* define/hace uso de un esquema de retransmisión, en el que, si tras la transmisión de una trama *unicast*, el nodo origen no recibe una confirmación (*ACK*) por parte del receptor, procederá a la retransmisión del mensaje hasta un número máximo de intentos (3 en este caso), tras los cuales, de seguir persistiendo el error, el paquete será descartado. La consecuencia es la pérdida definitiva de la información en el lado del receptor.

- **Ráfagas de tramas erróneas - *Erroneous Frame Burst (EFB)***. Debido a los desvanecimientos aleatorios producidos en el medio inalámbrico, un enlace puede presentar unas condiciones "hostiles" durante un intervalo determinado de tiempo, provocando la pérdida de un número significativo de tramas consecutivas. En consecuencia, la presencia de errores durante la comunicación no aparece de manera aislada, sino que presenta una cierta "memoria", poniendo de manifiesto que existe una correlación entre errores consecutivos. Este efecto conlleva una penalización de las comunicaciones en términos de rendimiento, sobre todo en protocolos orientados a la conexión, como *TCP*, como se verá más adelante.

Observando la Tabla 2.3, la característica más relevante es la gran variabilidad de resultados, tanto en términos de *throughput* como de *FER* y *PER*, a pesar de que las condiciones del experimento permanecen (aparentemente) inalteradas. Esta "aleatoriedad" puede ser achacada a los desvanecimientos que aparecen en el medio inalámbrico, fruto de la presencia de obstáculos entre los terminales o la recepción de múltiples ecos de la señal, consecuencia de la propagación multi-camino. Pueden encontrarse desde medidas que presentan unas condiciones prácticamente ideales (e.g. medidas #14 y #15), hasta rendimientos radicalmente opuestos, donde el canal se comporta de una forma extremadamente hostil (por ejemplo, en la medida #1 la mitad de los datagramas enviados no son recibidos por la aplicación en el destino). El citado factor de *memoria* se vislumbra claramente

con la presencia de largas ráfagas de errores, incluso en aquellos casos cuyo rendimiento puede considerarse bueno. Se puede afirmar que realmente existe una dependencia entre recepciones consecutivas.

El gran objetivo de los modelos de canal que se definen en este capítulo es el de ofrecer un comportamiento que se acerque en la mayor medida posible al observado en estos experimentos. Por lo tanto, deberán ser capaces de reflejar la poca predecibilidad que lo caracteriza, poniendo especial atención en replicar la presencia de las ráfagas de tramas erróneas, factor que la mayor parte de los modelos proporcionados por los simuladores de redes[6] *no tienen en cuenta* o, al menos, no le dan la relevancia que merece.

2.1.3. Impacto sobre el comportamiento de *TCP*

Caracterizar el rendimiento neto del canal inalámbrico, sin presencia de mecanismos que interfieran la operación nativa de los niveles inferiores definidos en la recomendación *IEEE 802.11b*, permitirá asimilar mejor aquellos casos en los que se utilicen protocolos con procesos adicionales, ya que asienta las bases que serán utilizadas como marco de referencia (umbrales de rendimiento, ráfagas de errores, etc.). Dado que el eje central de esta Tesis se mueve alrededor de la mejora de las prestaciones del protocolo *TCP* en entornos inalámbricos, parece razonable estudiar su comportamiento en un escenario real.

Manteniendo las mismas condiciones que las empleadas en las pruebas realizadas con *UDP*[7], se ha repetido nuevamente el proceso anterior, con el objetivo de estudiar el comportamiento de *TCP* sobre este tipo de redes. Hay que tener en cuenta que, en el momento en el que fue definida las primera versión de *TCP* (primera mitad de los años 70) [Post81], la inmensa mayoría de las redes de conmutación de paquetes estaban establecidas sobre entornos cableados. La principal diferencia entre este tipo de medio guiado y la transmisión inalámbrica es que, en el primero de los casos, el impacto producido por las pérdidas debidas a la propagación puede considerarse como un elemento testimonial, cuyo efecto no era tenido en cuenta. Por lo tanto, a la hora de diseñar los mecanismos de control de flujo y congestión propios de *TCP*, la principal (y única) causa de la pérdida de un segmento se vinculaba a la congestión producida en los *routers* a lo largo de la red, que tenían que descartar paquetes cuando la tasa de llegada era superior a su capacidad para dar salida a los mismos (fenómeno que causaba el desbordamiento de los *buffer*).

Lastrado por estos motivos históricos, *TCP* es incapaz de distinguir la causa real de la pérdida de un segmento (congestión o propagación), interpretando siempre la aparición

[6] Aunque es cierto que existen herramientas de capa física que cuentan con unos niveles elevados de precisión, su coste computacional desaconseja su uso en entornos de simulación a nivel de red.

[7] Con la única salvedad de que, para este experimento, se ha utilizado *TCP*, en su variante "Reno", como solución a nivel de transporte. Además, junto a este protocolo, se han habilitado las funciones de *Timestamp* y de *Reconocimiento Selectivo* (*Selective ACK, SACK* [Math96]), reduciendo el tamaño del *payload* a 1448 *bytes* para mantener el *Maximum Transfer Unit (MTU)* de 1500 *bytes* utilizado en el caso de *UDP*.

de congestión y reaccionando a través de los mecanismos de control de flujo (reducción de la ventana de congestión, retransmisión rápida, etc.). El problema inherente a esta falta de criterio penaliza principalmente a aquellos medios de transmisión en los que predominan los errores producidos por la propagación a través del canal (inalámbrico, en este caso), donde el rendimiento se ve fuertemente perjudicado.

Es preciso destacar que el estudio de este protocolo (que es el más utilizado en *Internet*) y su relación con el medio inalámbrico ha suscitado siempre un gran interés para la comunidad científica, que desde hace tiempo ha tratado de buscar soluciones, ya sea en forma de mejora del protocolo básico, o mediante el empleo de técnicas adicionales. A continuación se describen brevemente algunas de las propuestas más relevantes.

1. El tremendo impacto producido por la eclosión de las redes de acceso inalámbricas derivó en la aparición continua de modificaciones a la recomendación original de *TCP* [Post81]. Entre las más destacables se encuentran *TCP Reno* [Stev97], utilizado en la campaña de medidas descrita en esta sección, *TCP New Reno* [Hend12], una actualización de la anterior, que mejora su respuesta ante situaciones de pérdida múltiple de segmentos [Berk05]. Ésta será la versión utilizada en el simulador para el desarrollo de todos los procesos de medida[8]. Entrando en las implementaciones más utilizadas por los sistemas operativos más populares en la actualidad, *TCP Compound* [Srid08] es la versión oficial de los sistemas de *Microsoft* (desde *Windows Vista*), mientras que para los sistemas *UNIX* (concretamente, de la versión 2.6.19 del *Kernel* en adelante), *TCP Cubic* [Ha08] es la opción preferida.

2. Otro de los principales focos de interés consiste en la utilización de mecanismos adicionales al propio protocolo, capaces de interactuar con la operación del mismo para mitigar sus principales deficiencias. A modo de ejemplo, una de las soluciones más populares es el protocolo *Indirect TCP (I-TCP)* [Bakr95], que se basa en "partir" una única sesión de nivel de transporte en varias, en función de la tecnología subyacente; así, utiliza de manera independiente una conexión para la red cableada (típicamente asociada a la red dorsal), y otra para la alternativa inalámbrica (red de acceso), ocultando esta última al transmisor. Otra aproximación diferente, definida originalmente por el protocolo *Snoop* [Amir95], propone la utilización de nodos intermedios o *proxies*, típicamente estaciones base o puntos de acceso, que almacenan los segmentos no confirmados por el nodo destino. Al mismo tiempo, monitorizan los reconocimientos enviados, retransmitiendo los segmentos correspondientes tras la recepción de un *ACK* duplicado. De este modo, las retransmisiones no se producirán extremo a extremo, y los mecanismos de control de congestión no se verán tan perjudicados como con el comportamiento tradicional.

3. Por último, existe una alternativa más radical, que rompe con el modelo preestablecido, proponiendo nuevos protocolos de nivel de transporte que sean capaces de

[8]El motivo de la elección de *TCP New Reno* es que se trata de la modificación de *TCP* más reciente disponible en la versión del simulador en el momento de comenzar con la realización masiva de las Medidas.

sustituir a *TCP* en las redes actuales, ofreciendo el mismo tipo de transmisión segura extremo a extremo. Así, aparecen soluciones como el protocolo *MultiPath TCP (MPTCP)* [Ford13], que permite la multiplexación simultánea del tráfico a través de diferentes interfaces, negociados durante el establecimiento de la conexión. Sobre esta solución se entrará más en detalle en el Capítulo 3. Otro enfoque viene representado por el protocolo *Stream Control Transmission Protocol (SCTP)* [Stew07], el cual busca mantener una conexión activa en caso de que se produzca algún contratiempo (e.g. un traspaso entre celdas, rotura de un enlace, etc.), a través de la introducción de rutas redundantes o de respaldo.

Para comprender mejor los efectos de la propagación inalámbrica sobre *TCP*, se ha realizado una nueva campaña experimental. Para ello, se requiere la definición de nuevas métricas, descritas a continuación:

- **Triple *ACK* duplicado**. La primera versión del protocolo *TCP* [Post81] establece que la recepción de un segmento "fuera de orden" producirá, con carácter inmediato, la transmisión de un *ACK* que contenga el número de secuencia que el receptor espera obtener, dando lugar a un *ACK duplicado*, ya que la entidad transmisora ya había sido notificada con anterioridad de tal evento, y estaba por tanto esperando la confirmación de un número de secuencia posterior. Tras la recepción de tres de *ACKs duplicados*, el algoritmo de *Retransmisión Rápida* (*Fast Retransmit*) [Allm09] procederá a la retransmisión automática del segmento en cuestión.
- **Retransmisiones *TCP***. Número de segmentos *TCP* retransmitidos por la entidad transmisora. Hay dos tipos de eventos que dan lugar a una retransmisión: la recepción de un *triple ACK duplicado* o la expiración del temporizador conocido como *Retransmission TimeOut (RTO)*.
- **Número máximo de retransmisiones por segmento**. Número máximo de veces que un mismo segmento ha sido retransmitido en una conexión.
- **Tiempo máximo de inactividad**. Como consecuencia de los mecanismos de control de congestión definidos en el protocolo *TCP* y en sus sucesivas modificaciones, una entidad transmisora limita su tasa de envío (cesan las transmisiones durante un tiempo) cuando detecta una situación potencial de congestión. Este periodo puede alcanzar valores relativamente elevados, dando lugar a una penalización notable en el rendimiento de una transmisión *TCP*. Debido al esquema binario exponencial de *backoff* utilizado por *TCP*, cuanto mayor sea el número de retransmisiones por segmento, más largo será el intervalo durante el cual el nodo transmisor bloqueará el envío de segmentos hacia los niveles inferiores.

Como se observa en la Tabla 2.4, *TCP* muestra, al igual que *UDP*, un comportamiento altamente variable, a pesar de la supuesta estabilidad en las condiciones de los

Tabla 2.4: Rendimiento *TCP* sobre un enlace saturado (15 medidas independientes, ordenadas de manera ascendente)

#	*Thput [Mbps]*	*FER*	*PER*	*Trip ACK*	*Max Inact.*	*Rtx*	*Máx Retx*
1	0.55	0.418	0.071	154	28.2	620	9
2	1.67	0.36	0.034	103	39.7	264	9
3	1.19	0.292	0.041	68	8.3	314	6
4	1.31	0.212	0.038	39	8.0	321	7
5	2.23	0.279	0.033	99	2.1	278	5
6	2.39	0.255	0.029	61	1.3	217	5
7	2.43	0.318	0.022	108	1.9	185	5
8	2.86	0.171	0	0	0.8	1	1
9	3.17	0.143	0.011	23	2.1	84	4
10	3.23	0.153	0.013	36	2.2	103	4
11	3.5	0.09	0.015	21	1.7	120	4
12	3.55	0.186	0.023	30	0.6	177	3
13	3.67	0.105	0.016	17	0.8	138	3
14	4.36	0.052	0	45	0.2	1	1
15	4.85	0.025	0	0	0.0	0	0

experimentos. Existen medidas en las que no se produjo la pérdida de ningún segmento, siendo suficiente el mecanismo de retransmisión del estándar *IEEE 802.11* para enmascarar los errores de trama (e.g. #14 y #15), por lo que no fueron necesarias las retransmisiones en el nivel de transporte. El *throughput* obtenido en estos dos casos se encuentra cercano a los 5 *Mbps*, suponiendo éste el umbral superior de rendimiento en una transmisión sobre un enlace inalámbrico *IEEE 802.11b* [Vasu03; Garc04]. Por otra parte, en las filas superiores se observan resultados cuyos valores de *throughput* son notablemente inferiores (incluso menores que *1 Mbps*), debido en su mayor parte a la aparición de largos periodos de inactividad en el nodo origen (por ejemplo, en la medida #2, la fuente llegó a permanecer sin transmitir durante aproximadamente 40 segundos). Este efecto encuentra su explicación en la pérdida de segmentos de manera consecutiva (en forma de ráfaga), lo que conduce a un importante incremento en el valor del *RTO*, que define el temporizador utilizado por los mecanismos de control de congestión de *TCP* en el nodo transmisor.

2.2 Estado del arte

2.2.1. Modelos de canal existentes

Normalmente, tras el asentamiento de una nueva tecnología, transcurre tiempo hasta que empiezan a aparecer modelos que permitan su estudio a través de los entornos de

simulación. La popularidad de este tipo de análisis lleva años creciendo a un ritmo elevado, ya que el tamaño y complejidad de las redes a analizar suele limitar la viabilidad a la hora de realizar campañas experimentales sobre escenarios reales. Como consecuencia más directa, la comunidad se ve forzada a recurrir a nuevas vías de estudio, por lo que sus esfuerzos se ven focalizados hacia el desarrollo de modelos de simulación que reflejen, con altos niveles de precisión, los fenómenos observados en el mundo real. Como se mostrará a continuación, una de las técnicas más empleadas para emular el comportamiento de los niveles inferiores, de naturaleza puramente *empírica*, se basa en una primera fase de observación, en la que se estudia el rendimiento sobre redes reales, a través de repetitivas campañas de medidas; posteriormente, los resultados obtenidos serán utilizados para dar forma a los modelos de canal que serán integrados posteriormente en diferentes plataformas de simulación.

Esta secuencia no ha sido diferente con la aparición de las tecnologías inalámbricas *no celulares*, categoría que tiene al estándar *IEEE 802.11* como su ejemplo más representativo. Los primeros trabajos que trataron de modelar el comportamiento de protocolos IP sobre redes de área local inalámbricas se sitúan en una ventana temporal anterior incluso a la aprobación del estándar *IEEE 802.11* (1997). En aquel entonces, algunos fabricantes, como *AT&T* o *Lucent* empezaban a introducir dispositivos inalámbricos como alternativa a los interfaces de red de área local cableados. Fruto de este trabajo surgió la marca *WaveLAN*, con un marcado carácter propietario. En esta etapa *pre-IEEE 802.11* ya se podían encontrar los primeros trabajos en hacer frente a la caracterización del canal inalámbrico, como es el caso de Eckhardt *et al.* [Eckh96], que evalúan el impacto producido por diferentes fuentes de interferencia y atenuación en la red objeto de análisis, que derivarán en la aparición de errores aleatorios durante una transmisión. Para ello se aprovechan de la información recogida en forma de traza en diversas estaciones monitoras, utilizando métricas como la relación de señal a ruido de las tramas recibidas o la presencia de errores en las mismas. Al mismo tiempo, Nguyen *et al.* [Nguy96], siguiendo una metodología similar, buscaron encontrar un modelo realista que pudiese emular el comportamiento de un canal inalámbrico. Tomando como punto de partida las tasas de error observadas en las medidas, así como la longitud de las ráfagas de errores, propusieron un modelo de Markov de 2 estados mejorado, en el que se sustituyeron las tradicionales distribuciones geométricas utilizadas para modelar el tiempo de permanencia en cada estado, dando lugar a unos resultados más precisos.

Al igual que los modelos de canal propuestos en este capítulo, otros trabajos se centran en la caracterización de un canal *IEEE 802.11b*, como por ejemplo el realizado por Ikkurthy y Labrador [Ikku02], en el que estudiaron los efectos de los errores aleatorios sobre transmisiones de vídeo codificadas (utilizando como *codec* el popular compresor *MPEG-4*). Durante el proceso de pruebas se modificaba el tamaño de los paquetes transmitidos, analizándose la respuesta en forma de ráfagas (tanto correctas como erróneas), caracterizándolas estadísticamente para un escenario donde los nodos se encontraban a una distancia de aproximadamente 22 metros (el experimento, al igual que el realizado en la Sección 2.1, se realizaba sobre un enlace punto a punto inalámbrico entre un cliente y

un servidor). Comparando los resultados con los obtenidos en [Nguy96] con una tasa de 2 *Mbps*, llegaron a la misma conclusión: *un simple modelo geométrico no es suficiente para modelar de manera precisa un canal real*. Además, en sus resultados demostraron que para tramas largas (1500 *bytes*), el 90 % de las ráfagas de errores tenían una longitud inferior a cuatro paquetes. Como principal limitación, en ningún momento indican el número de retransmisiones a nivel *MAC* empleadas durante el procedimiento, lo cual, unido a la utilización de ráfagas de paquetes erróneas, puede conducir a cierta confusión, ya que no se puede deducir si se refieren a las recepciones en el nivel físico o en el *IP*.

Uno de las aproximaciones más clásicas a la hora de emular el comportamiento errático de un canal a ráfagas tuvo su origen en los años sesenta, cuando los trabajos de Gilbert [Gilb60] y Elliott [Elli63] dieron lugar a una solución común que pasó a denominarse modelo de *Gilbert-Elliott*. Su principio se centra en el uso de una cadena de Markov de 2 estados. Desde entonces, han habido múltiples contribuciones que han tratado de estudiar su fiabilidad (e.g. [Mush89; Hoch99; Hass11]). Con el paso del tiempo, este modelo ha sobrevivido y se ha ido adaptando de manera sucesiva al estudio de las diferentes tecnologías inalámbricas que han ido surgiendo en el mundo de las telecomunicaciones, contando siempre con implementaciones desarrolladas para la mayor parte de los simuladores de red. No obstante, son numerosos los trabajos que afirman que la utilización de cadenas de Markov de 2 estados es insuficiente. Por ejemplo, en [Conv06] los autores demuestran que este modelo no es capaz de reflejar el auténtico comportamiento de un canal inalámbrico, sobre todo en franjas temporales en las que las tasas de error de trama sean especialmente altas. Estas limitaciones lastran la utilización de este tipo de modelos en ciertos tipos de aplicaciones, como las transmisiones de vídeo, donde la pérdida masiva de información en un corto espacio de tiempo puede tener consecuencias drásticas sobre la percepción final de un usuario. Manteniedo un discurso similar, en [Gand08] se utiliza un modelo *Gilbert-Elliott* para definir los parámetros más adecuados que reflejen la calidad percibida en una transmisión de voz que utiliza un esquema *Forward Error Correction (FEC)* adaptativo, aunque, al igual que en el caso anterior, los autores ponen de manifiesto la necesidad de profundizar en el diseño del modelo del canal si se pretende obtener comportamientos más realistas.

En un marco temporal más reciente, Barsocchi *et al.* [Bars09] parten de unos resultados obtenidos en un proceso experimental llevado a cabo en un entorno rural, tratando de componer un modelo de canal capaz de replicar las tasas de error observadas en estas pruebas. Finalmente, Cardoso *et al.* [Card09] se cuestionan sobre la idoneidad de la utilización de *cadenas de Markov* de 2 estados para imitar los procesos aleatorios que definen las pérdidas de tramas en canales *IEEE 802.11* reales. Como alternativa, proponen y evalúan un modelo novedoso basado en *procesos de Markov ocultos*, cuyos parámetros operacionales se configuran a partir de una serie de trazas provenientes de un conjunto de experimentos realizados bajo unas condiciones uniformes, donde la tasa binaria de la aplicación transmisora se mantenía constante, fijando un intervalo constante entre paquetes consecutivos de 10 milisegundos, valor que no asegura condiciones de saturación en un enlace punto a

punto *IEEE 802.11b*. Además, es importante destacar que en ningún momento tienen en cuenta (o al menos, no se menciona), el esquema de retransmisión *IEEE 802.11* empleado, obviando también las referencias a la topología empleada (e.g. la distancia entre los nodos) o la relación señal a ruido percibida en recepción. En cuanto a los resultados que presentan, tanto la tasa de error como la longitud de las ráfagas de errores son claramente inferiores a las que se han observado en el escenario analizado en esta Tesis. Por último, concluyen que un proceso oculto de Markov de 11 estados, modelado con un esquema de nacimiento y muerte, es capaz de reflejar (con un alto nivel de precisión) los dos primeros niveles de estadísticos que caracterizan las pérdidas de paquetes en sistemas reales. Como se discutirá más adelante, su modelo no es realmente capaz de adaptar su comportamiento a tráficos con características diferentes a las que fueron utilizadas a la hora de configurar los parámetros de los procesos ocultos de Markov, manteniendo una estrecha relación con las secuencias con las que fueron entrenados.

A tenor de las limitaciones existentes, sin la presencia de un modelo con una relación *precisión/complejidad*, se desarrolla en la Universidad de Cantabria una nueva propuesta cuyos cimientos difieren en gran medida con lo visto hasta el momento: *BEAR* [Agüe10]. Su operación se basa en la estimación de la calidad de la señal detectada en los terminales receptores, tomando como referencia los resultados observados durante una campaña de medidas realizada sobre un escenario real. Además, hace uso de un filtro auto-regresivo que introduce un efecto *memoria* al canal inalámbrico. En el mencionado trabajo se comparó su rendimiento con alternativas más clásicas, con un uso muy extendido en la comunidad investigadora, incluyendo un canal *Gilbert-Elliott* tradicional, mostrando unos resultados claramente superiores a todos ellos.

En lo que respecta a implementaciones desarrolladas para la plataforma `ns-3`, el simulador de redes basado en eventos discretos que está llamado a ser la referencia en el mundo de la investigación durante los próximos años, es preciso destacar en primer lugar aquellas implementaciones que se encuentran disponibles en los repositorios oficiales del simulador desde sus primeras versiones [Laca06; Pei09a; Pei09b]. Basan su funcionamiento en el cálculo de la probabilidad de error de *bit* (*Bit Error Rate*, *BER*) en función del parámetro E_b/N_0 (energía de *bit* entre densidad espectral de potencia de ruido). A pesar de que pueden conseguir resultados ciertamente cercanos a los vistos en canales reales tanto en términos de *FER* como de *throughput*, no son capaces de reflejar el efecto *memoria* del medio inalámbrico, según el cual los errores de transmisión tienden a agruparse y a aparecer en forma de *ráfaga*. Aún conociendo las limitaciones impuestas por este tipo de modelos deterministas, que optan incluso por simplificar aún más los fenómenos físicos que tienen lugar durante una transmisión, siguen siendo las soluciones más ampliamente utilizadas por la comunidad científica para obtener sus resultados, sobre todo en aquellas líneas de investigación que focalizan sus esfuerzos en los niveles superiores.

Otra aproximación diferente a la simulación de canales *IEEE 802.11* para el simulador `ns-3` viene dada en [Papa10], donde los autores se cuestionan la precisión de los simuladores de red a la hora de emular los fenómenos producidos en el nivel físico, defen-

diendo que, a pesar de que los niveles superiores presentan una correcta implementación, las simplificaciones y abstracciones llevadas a cabo en los niveles más bajos no permiten reflejar correctamente efectos como los desvanecimientos rápidos o selectivos (en frecuencia), ya que los modelos de canal gestionan la trama como unidad elemental (ignorando las variaciones producidas a nivel de *bit*). Como solución, proponen una nueva alternativa, para acercar la emulación de la capa física a un entorno de simulador de red, conocida como `PhySimWifi` [Mitt12], optimizada para trabajar con las modulaciones *Orthogonal Frequency Division Multiplex (OFDM)* utilizadas en algunas de las extensiones del estándar *IEEE 802.11* (e.g `a`, `g` o `n`). A pesar de que ofrece una mayor precisión, acercándose más al comportamiento observado en un canal real, la complejidad de los cálculos que lleva a cabo penaliza de manera significativa al tiempo de simulación, pudiendo llegar a ser prohibitivo en escenarios con un número elevado de nodos y/o con varios flujos de comunicación simultáneos.

En otra propuesta empírica, Al-Bado *et al.* [AlBa12] utilizan los resultados de su propia campaña de medidas en un escenario de interiores para proponer el modelo *Berlin Open Wireless Lab (BOWL)*, ajustado a la medida de la tasa de detección de trama o *Frame Detection Rate (FDR)*[9], configurándolo para diferentes modulaciones (y tasas) correspondientes a la extensión *IEEE 802.11g* (concretamente, 6, 24 y 54 *Mbps*). Aunque es cierto que el modelo mantiene la misma naturaleza empírica que los propuestos en el marco de esta Tesis, sus autores se centran en aspectos diferentes a las ráfagas de errores del canal. En concreto, ponen especial atención a la caracterización del efecto de las fuentes de interferencia y al efecto captura que tienen lugar en redes reales, utilizando como escenario un edificio de laboratorios en Berlín, donde se despliega una red inalámbrica con hasta 9 nodos activos. Teniendo en cuenta que *HMP* y *BEAR* no tienen en cuenta estos efectos, parece razonable afrontar la integración de ambos enfoques.

Resulta interesante destacar también el reciente trabajo llevado a cabo por Lertpratchya *et al.* [Lert14], donde los autores se centran exclusivamente en las ráfagas (tanto de tramas correctas como erróneas) producidas en el canal inalámbrico, adoptando un modelo estocástico dual (implementado en `ns-3`): en el primer caso, se utilizarán ficheros traza como secuencias de entrenamiento a la hora de ajustar la evolución de las ráfagas en función de los resultados observados en éstas, necesitando una gran cantidad de memoria para guardar la información generada; por otro lado, en el caso en el que no disponga de referencias, el patrón de comportamiento de las ráfagas quedará determinado por una serie de funciones de ajuste, basadas en probabilidades modeladas a partir de condiciones ideales. Se utilizan los valores históricos recogidos en cada enlace, ajustando la decisión final de si una trama es correcta o no en función del comportamiento previo. Además, al igual que en *BEAR*, para evitar que las tramas presenten una influencia posterior de manera indefinida, se ajusta un temporizador que limitará el intervalo en el que tienen efecto en el canal. Teniendo en cuenta las similitudes conceptuales con *BEAR*, ambos modelos emulan

[9]Se define de manera análoga a la tasa de error de trama utilizado en esta Tesis, es decir: $FDR = 1 - FER$.

la presencia de largas ráfagas de errores en el canal, las cuales comparan sus prestaciones en una serie de experimentos llevados a cabo en el simulador. En relación a los resultados presentados, se aprecia que el modelo propuesto presenta unos niveles de variabilidad aún más elevados que *BEAR*, aunque no se proporciona información alguna sobre algunos de los parámetros importantes del escenario, como la distancia entre los equipos utilizados en la etapa experimental o el número de retransmisiones a nivel de enlace. Además, el proceso de medidas se ajusta a las condiciones existentes en la captura de los ficheros traza, por lo que resultaría coherente repetir el análisis bajo condiciones más generales. Otra diferencia con respecto a *BEAR* es que el código fuente del modelo propuesto en este caso no se encuentra disponible, por lo que no es posible reproducir los resultados.

2.2.2. Simuladores de red basados en eventos discretos

La gran influencia de los entornos de la simulación a día de hoy se pone de manifiesto, entre otras cosas, por la cifra incalculable de publicaciones científicas que en algún punto de su desarrollo han utilizado estas plataformas. Una de las consecuencias directas es la enorme disgregación existente, pudiendo encontrarse un gran número de simuladores disponibles, buscando todos ellos especializarse o desmarcarse de algún modo respecto al resto. Un patrón común en todos estos casos es que, a pesar de contar con aproximaciones de gran precisión para los protocolos y mecanismos de los niveles superiores, tieden a "descuidar" el modelado de los fenómenos que tiene lugar en el medio físico, favoreciendo una mayor escalabilidad a costa de reducir la complejidad de las operaciones subyacentes. A continuación se listan algunos de las herramientas que cuentan con módulos para simular el comportamiento de escenarios basados en redes inalámbricas:

- `ns-2` [NS2]. Sin duda alguna, se trata del simulador de código libre dominante (aunque dejó de tener actualizaciones en el 2009), habiendo contado con una gran cantidad de desarrolladores y contribuciones externas. Como elementos negativos, se puede destacar una clara falta de modularidad y una gran elevada curva de aprendizaje.

- `ns-3` [NS3]. Heredero directo de `ns-2`, este simulador nace de la experiencia adquirida en éste, ya que varios de sus desarrolladores más importantes deciden partir "desde cero" (reutilizando ciertos componentes, como la emulación de los niveles inferiores del estándar *IEEE 802.11*) y crear un nuevo entorno de simulación especializado en el análisis de redes. Se trata de un proyecto muy activo que recientemente se ha visto ampliado con módulos de análisis de tecnologías de acceso inalámbricas como *IEEE 802.11n*, *Long Term Evolution (LTE)* o *IEEE 802.15.4*.

- `OMNet++` [OMNET]. Al contrario que los casos anteriores, `OMNet++` no es simulador de red en sí mismo, sino una herramiento basada en eventos discretos de propósito más general. Como punto negativo se destaca su escaso soporte, con una comunidad de usuarios menos activa.

- `SteelCentral NetModeler Suite` [Riverbed]. Antiguamente conocido como `Opnet`, este software fue adquirido por la compañía *Riverbed* en el año 2012, con lo que su nombre pasó a un segundo plano. Se trata de una alternativa *comercial*, por lo que cuenta con una gran base de *software* detrás, aunque es necesario disponer de una licencia para poder emplearlo. Como nota al margen, las universidades pueden beneficiarse de un programa de licencias gratuitas.

- `QualNet` [QualNet]. Se trata de otra *plataforma de pago*, que se autoproclama como la más rápida en el modelado de tráfico en tiempo real. Presenta módulos tanto para redes inalámbricas como cableadas y cuenta además con un interfaz gráfico que facilita su uso.

- `GloMoSim` [Zeng98] es un software de simulación de red tanto para entornos cableados como inalámbricos. Su característica más relevante es que ha sido diseñado para utilizar un lenguaje de programación especializado en la computación paralela, `Parsec`.

- `JSim` [JSim]. Conocido formalmente como `JavaSim`, se trata de un simulador composicional, centrado en el desarrollo a través una jerarquía de componentes autónomos, que utiliza `Java` como lenguaje principal de desarrollo.

- `Matlab` [Matlab]. Mucho más que un simulador de red, *Matlab* es una plataforma completa de software matemático empleado en un sinfín de especialidades. Aunque en su origen, en 1984, estuvo ligado al cálculo a través de expresiones matriciales, con el paso de los años se ha ido extendiendo y ahora es capaz de abarcar un gran número de utilidades. Con respecto a la simulación de redes, se puede destacar que ya se ha implementado la familia de protocolos *TCP/IP*, así como un entorno de emulación del comportamiento de un canal inalámbrico [MatWi]. Una de sus principales desventajas es que, al igual que el antiguo `Opnet` o `Qualnet`, se trata de un *software* de pago. Si bien pueden encontrarse alternativas similares a `Matlab` con licencia *GNU's Not Unix! (GNU)* (como por ejemplo Octave [Octave]), éstas no presentan las herramientas adecuadas para emular el comportamiento de redes.

Como ya se ha mencionado, el desarrollo de esta Tesis va a estar ligado al simulador `ns-3`, por lo que toda implementación está sujeta a la arquitectura del mismo. Los principales motivos de esta elección se pueden resumir en los siguientes puntos: en primer lugar, se trata de una plataforma gratuita y de código abierto, lo que otorga la flexibilidad necesaria para modificar libremente los módulos existentes a conveniencia, así como añadir nuevos elementos de una manera sencilla y directa. Otro punto a su favor pasa por la experiencia previa en el Grupo de Ingeniería Telemática de la Universidad de Cantabria con `ns-2`, por lo que el siguiente paso lógico era evolucionar hacia su más directo sucesor, ya que aquél lleva más de cinco años sin actualizarse. Además, puede encontrarse en Internet una amplia y activa comunidad de usuarios, haciendo posible una interacción con la misma. Finalmente, hay una serie de publicaciones centradas en la comparación del rendimiento de varios de los simuladores de red más populares, demostrando que `ns-3` es capaz de

Figura 2.4: Plataforma ORBIT (Fuente: https://www.orbit-lab.org/)

realizar simulaciones de escenarios a gran escala de una manera eficiente [Wein09]. A pesar de que no se han encontrado datos recientes que recojan el porcentaje de publicaciones realizadas por cada simulador, existen trabajos previos que destacan la gran popularidad de `ns-2`, siendo el entorno de simulación favorito por la comunidad científica, contando con un porcentaje de publicaciones muy superior al resto [Sark11; Kurk05]. Se puede aventurar que, aunque aún quede un número importante de desarrolladores que siguen usando `ns-2`, un gran porcentaje de ellos han podido migrar hacia la "nueva" versión, pudiendo intuir que `ns-3` se convertirá en el simulador de redes con mayor relevancia durante los próximos años.

2.2.3. Plataformas experimentales

Existe otra alternativa al uso de simuladores de red, basada en la utilización de plataformas experimentales, las cuales permiten utilizar de forma remota, en un entorno de laboratorio (o en un escenario exterior controlado), solucionadas, creadas y gestionadas por terceros. Estas plataformas ofrecen un servicio gratuito o con un precio muy inferior al que se requeriría invertir para construir una alternativa similar desde cero. Debido a que su uso es compartido por un gran número de usuarios, lo más habitual es que para poder acceder a ellas haya que reservar una ventana temporal con la suficiente antelación. A continuación se listan algunas de las que cuentan con mayor popularidad en la actualidad.

- *Roofnet* [Agua03]. El concepto de *roofnet* fue acuñado por primera vez por el *Massachusetts Institute of Technology (MIT)*, donde un grupo de investigadores desplegó más de 40 nodos *IEEE 802.11b* cubriendo un área urbano de unos 8 kilómetros cuadrados. En este *testbed* se han llevado a cabo estudios de diversa índole, desde la

caracterización del canal *IEEE 802.11* hasta el desarrollo de nuevos protocolos de encaminamiento que aprovechan las propiedades *broadcast* del medio inálámbrico. Todo el software desarrollado para este proyecto se encuentra disponible como código libre.

- *PlanetLab* [Planetlab]. Se trata de un grupo de equipos disponible como banco de pruebas para el análisis de sistemas distribuidos, que cuenta actualmente con 1090 nodos en 507 localizaciones diferentes de todo el mundo. Solamente aquellas corporaciones o universidades que contribuyan al proyecto con sus propios nodos pueden hacer uso del sistema. Dentro del conjunto de grupos que experimentan sobre esta plataforma, merece la pena destacar el proyecto *NITOS* [Kera12], que integra el uso de estaciones móviles para estudiar el rendimiento de la red dorsal (a gran escala) proporcionada por *PlanetLab* al combinarla con una red de acceso inalámbrica situada en la Universidad de Thessaly.

- *Emulab* [Emulab]. Más que una plataforma en sí, *Emulab* abarca el uso conjunto de instalaciones con el software desarrollado para ellas (*testbed* cerrado). El *cluster* principal de la red se encuentra en la Universidad de Utah, aunque existen otros en varias localizaciones alrededor del mundo. Con respecto al análisis en redes inalámbricas, se han realizado experimentos con robots móviles que transportaban diferentes tipos de sensores [John06], aunque esta línea de experimentación se ha abandonado.

- *ORBIT* [ORBIT]. En último lugar se presenta *Open-Access Research Testbed for Next-Generation Wireless Networks (ORBIT)*, en la que se ha desplegado un array de 20×20 dispositivos inalámbricos programables con interfaces *IEEE 802.11*, *Zigbee* y *Worldwide Interoperability for Microwave Access (WiMAX)*, situados en el interior de un laboratorio localizado en la Universidad de Rutgers (Estados Unidos), como se muestra en la Figura 2.4. En este caso, merece la pena destacar que es posible introducir software de `ns-3` en los nodos, para replicar las simulaciones utilizando un canal inalámbrico real, abstrayendo únicamente los niveles superiores. Sería interesante hacer uso en un futuro de esta propiedad para llevar a cabo una rápida comparación (o ajuste) entre los modelos de canal desarrollados en esta Tesis y los resultados obtenidos sobre esta plataforma.

2.2.4. Canal de comunicaciones

Uno de los elementos que tiende a ser menos considerado por la mayor parte de los simuladores de red más conocidos es la emulación apropiada del comportamiento de los niveles inferiores, factor que motiva, en el marco de esta Tesis, al desarrollo de nuevos modelos que hagan frente a estas limitaciones y proporcionen un marco realista sobre el que sustentar el resto del desarrollo. El primer paso antes de proceder con el diseño es el de comprender qué es exactamente un canal de comunicaciones y cuáles son sus componentes. Dentro de una de las referencias más ilustres, Claude E. Shannon describe en [Shan48] un canal de comunicaciones como el medio a través del cual se propaga una

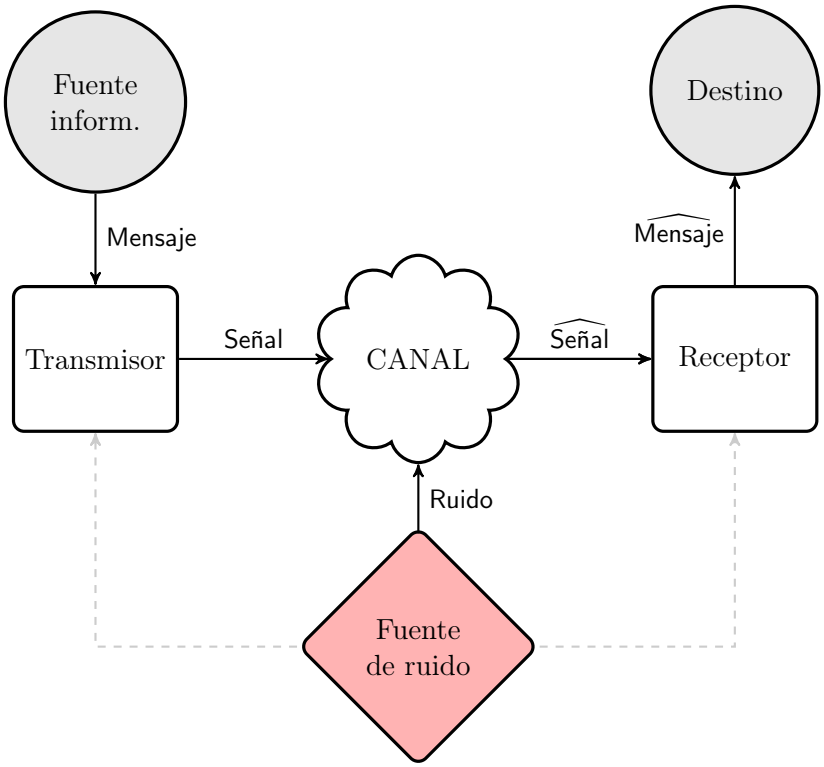


Figura 2.5: Esquema de un canal de comunicaciones genérico

señal desde un transmisor hacia un receptor. Pueden encontrarse multitud de canales, con comportamientos y propiedades físicas diferentes, como un par trenzado, un cable coaxial, una banda de radiofrecuencias o un haz de luz, entre otros. Tal y como muestra la Figura 2.5, la señal correspondiente a un mensaje generado en una fuente de información puede verse modificada al atravesar el canal de comunicaciones, haciéndola más o menos entendible para el receptor. Éste a su vez, debe diseñarse con el fin de poder sobreponerse a las alteraciones sufridas por la señal, para entregar el mensaje al destino con la menor distorsión posible. En el diagrama de bloques de la figura, los efectos nocivos del canal se modelan o representan a través de una fuente de ruido externa, sienda ésta la responsable del deterioro de la señal original.

Entrando ahora en la categoría de canales inalámbricos, en [Saun99] la fuente de ruido se divide en dos partes: *aditivo* y *multiplicativo*. El primero de ellos deriva de efectos como el ruido término y de disparo, procedentes de la utilización de componentes electrónicos no ideales en los terminales (transmisor y receptor), así como de diversas fuentes externas, como fenómenos atmosféricos, radiación cósmica y la interferencia producida por otras estaciones transmisoras o aparatos eléctricos. Desde el punto de vista del simulador utilizando en esta Tesis, `ns-3`, existe un módulo propio que se encarga de la cuantificación de este tipo concreto de ruido, por lo que los modelos presentados podrán abstraer el cálculo de esta fuente de interferencia. Por otra parte, el ruido *multiplicativo* aparece a lo largo

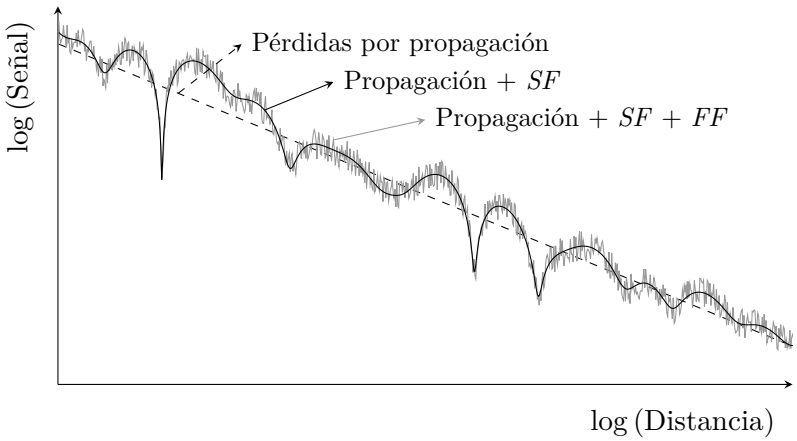


Figura 2.6: Atenuación de la señal en función de la distancia sobre un enlace inalámbrico

de la propagación de las ondas electromagnéticas a través del propio medio inalámbrico. La literatura suele clasificar los efectos producidos en tres contribuciones diferentes:

1. *Atenuación*, causada por la pérdida de energía de la señal a medida que atraviesa el canal. En el contexto de los simuladores, esta componente suele ser modelada a través de una función determinista, que calcula la cantidad de energía perdida en función de la distancia entre el transmisor y el receptor.

2. *Desvanecimientos lentos*, *Slow Fading (SF)* o *shadowing*, que causan variaciones lentas en amplitud y fase de la señal (en comparación con la longitud de onda), consecuencia de la presencia variable de obstáculos entre el transmisor y el receptor. Desde el punto de vista de la simulación, es habitual ver la utilización de distribuciones *log-normales* como solución estándar.

3. Los *desvanecimientos rápidos* o *Fast Fading (FF)* se generan a partir de las interferencias generadas por la recepción desfasada de la misma señal, causadas por los fenómenos de reflexión, difracción y refracción que aparecen por la presencia de elementos en el recorrido de la señal y modifican su trayectoria. Esto hace que lleguen múltiples "ecos" al receptor en tiempos diferentes. Para modelar este efecto, los esquemas más utilizados son variables aleatorias normales o, como solución más avanzada, la distribución de Nakagami [Naka60].

Como ejemplo ilustrativo, en la Figura 2.6 se muestra el efecto combinado de las tres contribuciones, utilizando una representación en escala logarítmica.

Otro de los fenómenos característicos de las comunicaciones inalámbricas es la correlación existente entre los procesos aleatorios que definen las recepciones de tramas consecutivas. Esta "dependencia" se traduce en que la aparición de un error durante la transmisión de una trama implica que las siguientes recepciones tendrán una mayor probabilidad de

resultar erróneas. Esta *memoria* observada en el canal inducirá la presencia de ráfagas de errores, factor que tiende a ser ignorado a la hora de diseñar los modelos de canal más habituales.

Tras descomponer la naturaleza de los canales inalámbricos, donde ha quedado patente la presencia de un gran número de efectos aleatorios nocivos que inducirán efectos no deseados en las señales que los atraviesan, es preciso centrarse ahora en la forma que tienen los simuladores en modelar su comportamiento. En el caso concreto de `ns-3`, la operación correspondiente a un modelo de canal se divide en tres etapas claramente diferenciadas: *(1)* un *modelo de propagación* se encarga de obtener la potencia de la señal recibida a través de la combinación de una serie de módulos, que tratan de replicar, de la manera más precisa posible, los efectos producidos desde el instante en que la señal es enviada por el transmisor hasta que es recibida en el destino. *(2)* Un *modelo de interferencia*, que calcula (de manera independiente) el nivel de ruido *aditivo* introducido en el canal; además, se tiene en cuenta la posible interferencia de transmisiones simultáneas en el medio (aunque sólo se consideran las pertenecientes al mismo canal). *(3)* Por último, un *modelo de error* se encargará de decidir, en base a la información proporcionada de las dos etapas anteriores, si la trama recibida es correcta o no.

En las próximas líneas se estudiarán con un mayor nivel de detalle aquéllas de las soluciones propuestas por el simulador que resulten más relevantes.

Modelos de propagación

El estudio de los fenómenos de propagación lleva décadas despertando el interés de la comunidad científica, que ha tratado de caracterizar, de manera numérica, el comportamiento de la señal cuando atraviesa el medio inalámbrico.

Dentro del simulador se pueden encontrar una gran variedad de modelos de diferente naturaleza para emular las pérdidas producidas en el canal. Es preciso remarcar de nuevo que uno de los grandes objetivos propuestos por un simulador de red es su escalabilidad, pudiendo tener que analizar escenarios con cientos o inclusos miles de nodos. Debido a esto, aunque pueden encontrarse en la literatura técnicas que ofrecen resultados más precisos, suelen ser descartadas debido a su complejidad, como por ejemplo la utilización de las conocidas ecuaciones de Maxwell [Lam03] o el trazado de rayos [Nits11]. Además, dado el carácter modular de la implementación, será extremadamente sencillo concatenar diferentes esquemas de propagación, de manera que la potencia de salida de uno será la de entrada del siguiente (y así sucesivamente), por lo que pueden agregarse varias etapas que, en su conjunto, generen una respuesta más realista.

- *Modelo "log distancia"*. Es la alternativa más sencilla para modelar, de manera determinista, las pérdidas producidas por la atenuación a lo largo del medio inalámbrico, independientemente de la banda de frecuencias sobre las que se está llevando a cabo la transmisión.

Tabla 2.5: Valores típicos del exponente de pérdidas según el tipo de entorno, extraídos de [Saun99]

Tipo de entorno	ν
Espacio libre	2
Tierra plana	4
Urbano	2.7 - 3.5
Urbano + *Shadowing*	3-5
Interior (*LOS*)	1.6 - 1.8
Interior (*NLOS*)	4-6

En la expresión mostrada en la Ecuación (2.6), L representa la atenuación total, L_0 es la atenuación medida en una distancia de referencia d_0, d es la distancia entre el emisor y receptor y, por último, al parámetro ν se le conoce como exponente de pérdidas por propagación, cuyo valor se establece habitualmente de manera empírica, ajustándose a las diferentes condiciones del escenario, tal y como se resume en la Tabla 2.5.

$$L = L_0 + 10\,\nu\,log_{10}\left(\frac{d}{d_0}\right) \tag{2.6}$$

- *Modelo "three log distance"*. Evolución del anterior que introduce una función a tramos definidos por tres distancias (d_0, d_1 y d_2), umbrales de áreas de atenuación diferentes, caracterizadas con un exponente de pérdidas ν_i diferente en cada caso, como recoge la Ecuación (2.7).

$$L = \begin{cases} 0, & d < d_0 \\ L_0 + 10\,\nu_0\,\frac{d}{d_0}, & d_0 \leq d < d_1 \\ L_0 + 10\,\nu_0\,\frac{d_1}{d_0} + 10\,\nu_1\,\frac{d}{d_1}, & d_1 \leq d < d_2 \\ L_0 + 10\,\nu_0\,\frac{d_1}{d_0} + 10\,\nu_1\,\frac{d_2}{d_1} + 10\,\nu_2\,\frac{d}{d_2}, & d_2 \leq d \end{cases} \tag{2.7}$$

- *Modelo de Friis o de espacio libre*. Basado en el trabajo de Friis [Frii46] en la década de los $1940's$, asume que la señal sigue un único camino entre el transmisor y el receptor. La potencia recibida se calcula según la Ecuación (2.8), donde P_{RX} y P_{TX} se corresponden con la potencias de recepción y transmisión, respectivamente; además, G_t y G_r son las ganancias de las antenas transmisora y receptora, λ es la longitud de onda (correspondiente a una frecuencia de trabajo aproximada de 2.4 GHz) y L ($L \geq 1$) representa las pérdidas adicionales del sistema. En las primeras versiones del simulador esta fórmula era válida para cualquier distancia; no obstante, en las últimas actualizaciones se limita su uso a distancias pertenecientes a la *región de campo lejano* ($d > 3\lambda$). La solución fuera de esta zona será la de asumir $P_{TX} = P_{RX}$.

$$P_{RX} = \frac{P_{TX}\,G_t\,G_r\lambda^2}{4\,\pi\,d^2\,L} \tag{2.8}$$

- *Modelo de tierra plana o "two ray ground"*. Este modelo nace a partir de las limitaciones derivadas del anterior (considerado demasiado simple), introduciendo una segunda contribución, asociada a la reflexión de las ondas electromagnéticas en el plano terrestre [Rapp01]. Su comportamiento se ajusta en base a la Ecuación (2.9), donde a los parámetros de la expresión de *Friis* se añaden las alturas de las antenas transmisora y receptora (h_t y h_r, respectivamente). Además, se vuelve a añadir el factor L de pérdidas, aunque en [Rapp01] se asume que es igual a 1.

$$P_{RX} = \frac{P_{TX}\, G_t\, G_r\, h_t^2\, h_r^2}{d^4\, L} \tag{2.9}$$

No obstante, este modelo no proporciona un comportamiento adecuado en distancias cortas, debido a la oscilación causada por las interferencias constructivas y destructivas producidas por la combinación de los dos rayos. Por lo tanto, se recomienda la utilización de la expresión de *Friis* en estos casos (teniendo en cuenta su limitación en el área $d < 3\lambda$), utilizando una distancia de corte (d_c) que se calcula a partir de la Ecuación (2.10).

$$P_{RX} = \frac{4\pi\, h_t\, h_r}{\lambda} \tag{2.10}$$

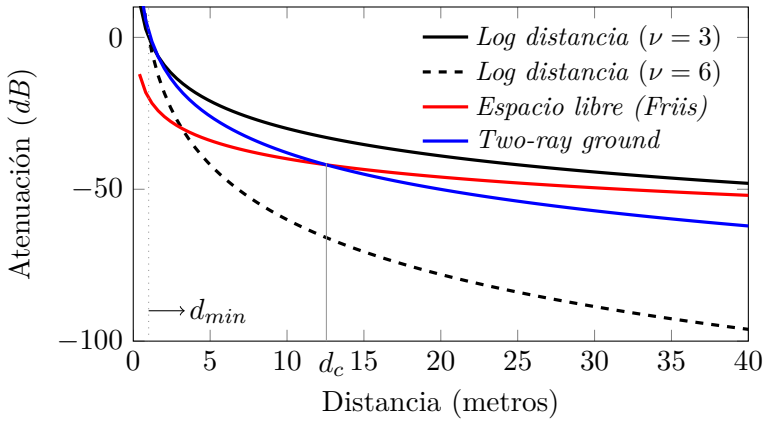


Figura 2.7: Modelos de propagación deterministas en función de la distancia para transmisiones en la banda de 2.4 *GHz* ($\lambda \approx 0.12$ metros). Los parámetros utilizados para los modelos de *Friis* y *Two-ray ground* son los siguientes: $G_t = G_r = 1$, $L = 1$, $h_t = h_r = 1$

En relación a las soluciones que utilizan funciones deterministas para modelar el comportamiento de un canal inalámbrico, en la Figura 2.7 se representa la atenuación sufrida en cada una de las posibles alternativas, a medida que los nodos se separan. Si no se tienen en cuenta las imprecisiones cuando los terminales se encuentran muy cercanos entre sí (distancia d_{min}), se ve cómo la aproximación "log distancia" está estrechamente ligada al factor ν, que damnifica los entornos más hostiles. Con respecto a los modelos *Friis* y *Two-ray ground*, se puede apreciar que el punto de intersección entre ambas curvas corresponde con

el parámetro d_c, poniendo de manifiesto que el factor d^4 del segundo penaliza en mayor medida a las distancias más largas.

- *Modelo aleatorio*. Más que un modelo en sí, determina el valor de potencia recibida en base a una variable aleatoria definida previamente. En el contexto de esta Tesis, se definirá una *variable aleatoria (v.a.)* normal para modelar el efecto de los desvanecimientos rápidos del canal.

- *Modelo basado en rango*. Otro de ejemplo de función a tramos, donde se define un atributo que marcará una distancia umbral, a partir de la cual toda trama recibida será errónea. Por el contrario, si el nodo receptor se encuentra dentro de la región delimitada por este valor, no sufrirá atenuación alguna ($P_{TX} = P_{RX}$). Este método suele ir concatenado con algún otro, sirviendo como herramienta de ayuda para controlar el escenario.

- *Modelo de Nakagami*. Otra forma de modelar la naturaleza de los desvanecimientos rápidos fue ideada por Nakagami en [Naka60] basándose en un nuevo proceso estocástico derivado de la distribución Γ: la distribución *Nakagami-m*, cuya *fdp* se define en la Ecuación (2.11), donde m es la profundidad del desvanecimiento o *fading* y Ω es la potencia media recibida.

$$f(x; m, \Omega) = \frac{2m^m}{\Gamma(m) \cdot \Omega^m} \cdot x^{2m-1} \cdot e^{-\frac{m}{\Omega}x^2} \tag{2.11}$$

- *Modelo con información de estructuras*. Existe un nuevo tipo de modelo (incluido muy recientemente en el simulador `ns-3`) que trata de emular los efectos de *shadowing* asociados a la presencia de muros (internos o externos) en edificios. Así, se podrá analizar de manera más realista un escenario con redes tanto en espacios exteriores como interiores. El precio a pagar será el de una configuración más compleja, con un gran número de parámetros.

Modelos de interferencia

El simulador incluye una única alternativa que calcula, por un lado, el ruido termal, expresado en la Ecuación (2.12), donde T es la temperatura en grados *Kelvin* (suele usarse un valor estándar de 290 K), k_B es la constante de Boltzmann ($k_B = 1.3803 \cdot 10^{-23}\ J/K$ con $T = 290$ K) y Δ_f es el ancho de banda de la señal utilizada.

$$N_t = k_B\, T\, \Delta_f \tag{2.12}$$

Una vez obtenido N_t, se estimará el nivel de ruido de fondo a partir de la expresión $N_{floor} = F\, N_t$, siendo F la figura de ruido, métrica que ajusta la diferencia entre un receptor estándar y uno ideal. Por otro lado, se calcula la interferencia (N_i) asociada a transmisiones simultáneas en el mismo canal, cuyo impacto dependerá del tiempo de transmisión y de la potencia percibida en el punto de estudio. Como resultado, la suma de

estas dos contribuciones dará lugar al ruido total, como se define en la Ecuación (2.13), donde I es el número de fuentes interferentes que han alterado la señal durante el tiempo en el que ha estado atravesando el canal.

$$N = N_{floor} + \sum_{i=0}^{I} N_i \tag{2.13}$$

El modelo devolverá la *SNR* como se muestra en la Ecuación (2.14).

$$SNR = \frac{P_{RX}}{N} \tag{2.14}$$

Modelos de error

Una vez recibida la señal distorsionada del canal inalámbrico, el receptor será el encargado de discernir entre aquellos casos en los que se puede recuperar el mensaje[10] o, por el contrario, será descartado. Hablando en términos de simulación, estas decisiones suelen tener lugar en una nueva etapa conocida como *modelo de error*, en la que, a partir de algún tipo de variable de entrada (normalmente la calidad de la señal percibida en el receptor), se llevará a cabo la decisión final con respecto al mensaje recibido. `ns-3` incorpora algunas soluciones para esta última fase, resumidas a continuación:

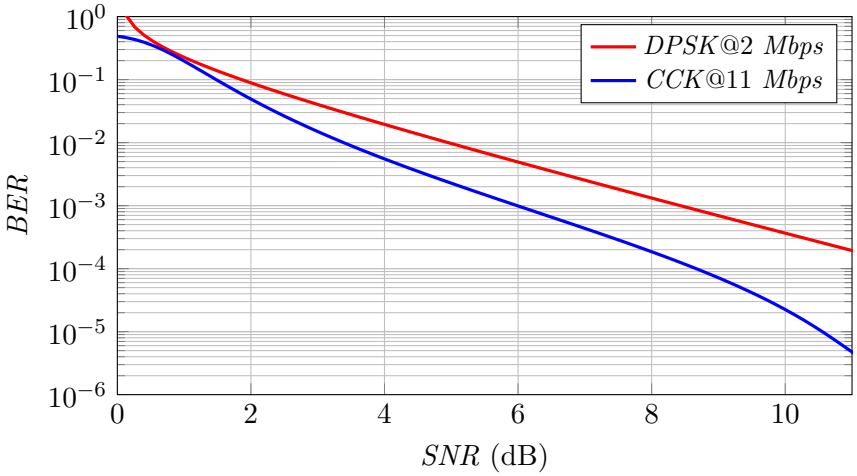


Figura 2.8: Ejemplo de curvas *BER* en *IEEE 802.11*

[10]Dentro del contexto de esta memoria sólo se hablará de comunicaciones digitales, por lo que es en los niveles inferiores donde se decidirá si las tramas se han recibido correctamente o no.

- En primer lugar se destaca un modelo básico, que presenta como atributo la tasa de error de trama[11], a partir del cual se tomará una decisión, comparándolo con un valor escogido arbitrariamente a través de una variable aleatoria uniforme. Los procesos entre recepciones consecutivas son completamente independientes, por lo que no se podrá imprimir un factor de memoria a las transmisiones.

- Posteriormente, como evolución del anterior, se ha incorporado al ámbito del simulador un nuevo modelo, que se centra en la creación de *ráfagas* de errores. Para ello se añade una cierta correlación a los eventos de recepción, junto con la tasa de error definida en el modelo anterior, definiendo otra variable aleatoria que ajusta la longitud de la ráfaga de tramas que van a perderse de manera consecutiva.

- Por último, aparecen otros modelos de error, basados en el cálculo de la probabilidad de error de *bit* o *Bit Error Rate (BER)*. Se trata de la solución más avanzada, que recibe, como parámetro de entrada, la calidad de la señal recibida, en términos de la relación señal a ruido o *SNR* (aunque es posible utilizar otros parámetros, como la energía de *bit* sobre el nivel de ruido, E_b/N_0). Con este dato, y a partir de estudios realizados en la literatura [Pei09a; Pei09b], se utilizarán diferentes funciones que devolverán el valor de la *BER* en base a la modulación empleada. A modo de ejemplo, en la Figura 2.8 se puede observar las curvas *BER* extraídas a partir del código implementado en el simulador para las dos modulaciones utilizadas a lo largo de esta Tesis: una modulación *Complementary Code Keying (CCK)* a 11 *Mbps* para el tráfico de datos y una *Differential Quadrature Phase Shift Keying (DQPSK)* para los mensajes de control (reconocimientos *IEEE 802.11*, principalmente), observando, como era de esperar, una mayor tasa de error asociada a la modulación más rápida (los símbolos se encuentran más cercanos entre sí, por lo se que necesitará una menor cantidad de ruido para inducir un símbolo erróneo).

  Una vez obtenida la probabilidad de pérdida para un *bit* individual, asumiendo independencia entre errores de *bits* consecutivos, se calcula la probabilidad de error de trama en función de su longitud, L, utilizando la Ecuación (2.15).

$$\mathcal{P}_{error}(\text{trama}) = 1 - (1 - \mathcal{P}_{error}(\text{bit}))^L \tag{2.15}$$

Tras analizar los modelos de propagación y error presentes en el repositorio original del simulador, es preciso destacar que la configuración por defecto, utilizará únicamente un modelo *log distancia* para calcular las pérdidas de atenuación, y uno basado en la obtención de las curvas *BER* para decidir el estado de las tramas recibidas (correctas o

[11] A lo largo de esta memoria será esencial distinguir entre tasa de error de trama o *FER* y tasa de error de paquete o *PER*. Hay que tener en cuenta el esquema de retransmisiones nativo de la norma *IEEE 802.11*, por el cual, tras la pérdida de una trama, se producirá una retransmisión inmediata, durante un número finito de intentos. Solamente en el caso de que no se logre recuperar el paquete tras todas las retransmisiones se producirá la pérdida del paquete correspondiente.

erróneas). Se realizan así una serie de simplificaciones con respecto al comportamiento real de un canal inalámbrico, en el que, además de las diferentes fuentes de ruido descritas anteriormente, se aprecia una clara componente aleatoria, donde la calidad del enlace puede variar enormemente en un corto intervalo de tiempo, o el efecto memoria del mismo, donde las tramas erróneas tienden a agruparse en forma de ráfagas.

2.3 Modelos de canal analizados

A lo largo de esta sección se procederá a presentar las tres alternativas propuestas a la hora de caracterizar el comportamiento de un canal inalámbrico: en primer lugar, se modelará un proceso oculto de Markov generado a partir de la caracterización empírica, descrita en la Sección 2.1. Partiendo de la misma base, se introducirá otro modelo, *BEAR*, cuya principal característica es la utilización de un filtro auto-regresivo (cuyos coeficientes estarán "calibrados" en base a los mencionados resultados experimentales) que complete el proceso de estimación de la calidad de la señal percibida en un nodo receptor, que, como se ha discutido en la Sección 2.2.4, presenta tres tipos de contribuciones diferentes. Finalmente, haciendo uso de las herramientas provistas por defecto en el simulador, se analizará una tercera alternativa, conceptualmente similar a *BEAR*, aunque con varias simplificaciones (e.g. la abstracción de parte de los efectos que caracterizan el medio inalámbrico) en pos de reducir su complejidad computacional y presentar un buen nivel de escalabilidad.

2.3.1. Modelo de canal basado en un proceso oculto de Markov (*HMP*)

La utilización de los conocidos procesos ocultos de Markov [Rabi86] ha ganado en popularidad desde mediados del siglo XX (década de los sesenta). Desde entonces, han sido innumerables los trabajos que han tratado de explotarlos para predecir o estimar el comportamiento de fenómenos producidos en la naturaleza. La aplicabilidad de este tipo de técnicas presenta un grado muy elevado de diversidad, dando lugar a múltiples líneas de investigación, que abarcan desde el uso de *HMPs* para modelar aplicaciones de reconocimiento de voz [Rabi89], predecir la localización de los individuos dentro de un edificio en función de sus hábitos (e.g. horario de trabajo, descanso, etc.), o incluso su utilización en el ámbito de novedosos estudios bio-informáticos, como análisis genéticos.

Centrándose ahora en su aplicación en redes inalámbricas, es preciso remarcar que no se trata de la primera vez que se utilizan los procesos de Markov para modelar un canal inalámbrico. Aunque no se trate de procesos ocultos, el modelo de *Gilbert-Elliott* [Gilb60; Elli63] ha tenido una gran relevancia en el mundo de la simulación, ya que ha estado presente en la mayor parte de las herramientas existentes. Tal y como muestra la Figura 2.9, se trata de un proceso de Markov de dos estados que establecen si una trama va a ser

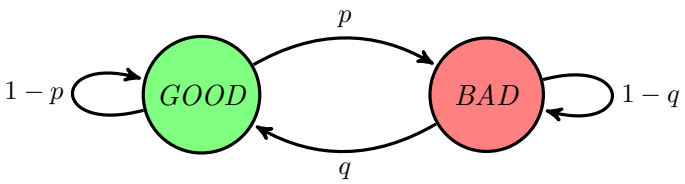
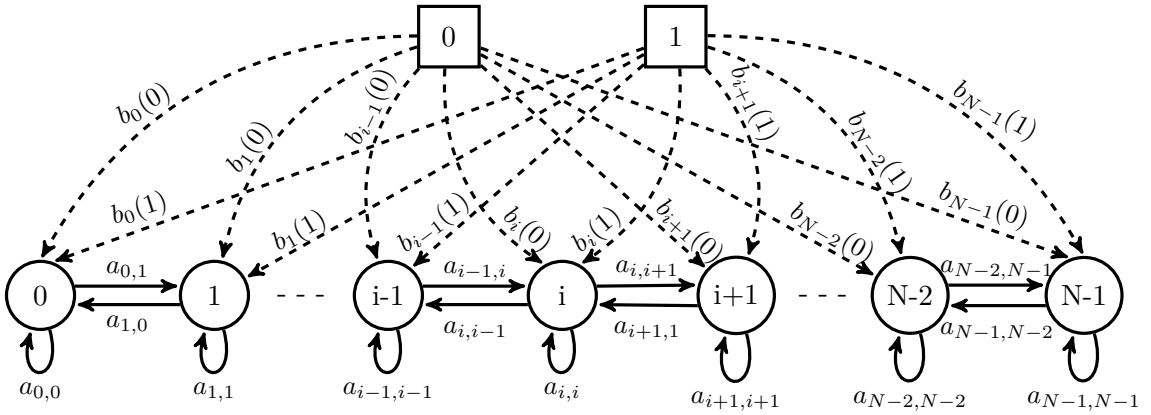


Figura 2.9: Modelo de Gilbert-Elliott (cadena de Markov de 2 estados)

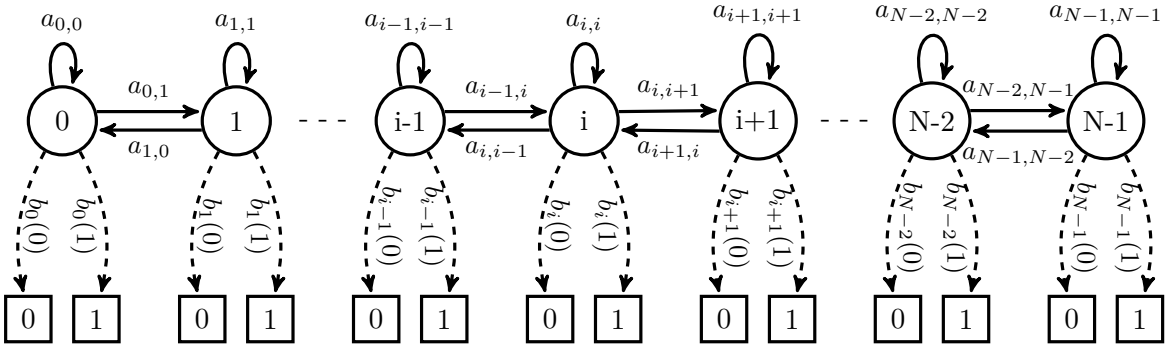
considerada como correcta (estado *GOOD*) o errónea (estado *BAD*). El comportamiento del modelo se centra en las probabilidades que definen la transición (o permanencia) entre los estados, como puede verse en la figura adjunta. A pesar de que las sucesivas evoluciones del modelo permiten parametrizar estos coeficientes en función de una *FER* y una longitud de ráfaga de errores o *EFB*, se ha demostrado que presenta una serie de limitaciones: en primer lugar, se encuentra fuertemente ligado a los parámetros que lo definen, dando lugar a un comportamiento muy previsible (opuesto a la naturaleza aleatoria observada en un canal real); además, teniendo en cuenta que su configuración más habitual [Agüe07] asocia los eventos de transición entre estados a la recepción de tramas, su comportamiento no se adapta a las posibles condiciones cambiantes de una red. Teniendo en cuenta estas deficiencias, la utilización de un *HMP* mejora las prestaciones de un modelo de *Gilbert-Elliott* en todos sus aspectos.

En [Rabi86] se describe un *HMP* como un doble proceso estocástico (discreto) en el que uno se encuentra oculto, pudiendo extraerse únicamente a partir de una secuencia de *observables* que definen el otro proceso "visible". O, desde un punto de vista más técnico, un proceso oculto de Markov equivale a una cadena de Markov discreta en el tiempo con un número finito de estados, observada a través de un canal discreto invariante en el tiempo y sin memoria [Ephr02].

Así, un proceso oculto de Markov está compuesto por un sistema discreto de N estados independientes, S_i, donde el subíndice i se asocia con cada uno de los estados que componen la cadena ($i = [0, 1...N-1]$). Las transiciones entre éstos se rigen a través de un conjunto de probabilidades estocásticas, conocidas como *probabilidades de transición* ($a_{i,j}$), representando la probabilidad de moverse de un estado i hacia otro j. Se caracteriza por ser de primer orden, ya que sólo se tiene en cuenta el estado predecesor, sin considerarse el efecto de los anteriores: $\mathcal{P}(q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, ...) = \mathcal{P}(q_t = S_j | q_{t-1} = S_i)$, donde q_t representa el estado actual en un instante t. Junto con este proceso estocástico se define otro, cuyo conjunto de probabilidades mapean las decisiones tomadas dentro de cada uno de los estados, $b_i(k)$, estableciendo las posibles salidas del sistema. En el marco concreto de un modelo de canal inalámbrico, tendrá una función similar a un modelo de error, donde a través de un valor aleatorio se decidirá si la recepción de una trama ha sido correcta o no, dependiendo del estado actual dentro de la cadena en un tiempo arbitrario, $q_t = S_i$. Es preciso destacar que, a diferencia de los clásicos procesos de Markov estándar, en este caso la cadena que define la matriz de transición se encuentra "oculta" tras una secuencia de elementos observables. Por último, para definir completamente el proceso, se necesita



(a) Esquema general



(b) Esquema desde el punto de vista del simulador

Figura 2.10: Interpretación gráfica de un proceso oculto de Markov de N estados

establecer el estado de la cadena en el momento en que se inicia el proceso, es decir $q_0 = S_i$. Para ello, el vector $\Pi = \{\pi_i\}$ (de longitud N) modela la probabilidad de pertenecer a cada uno de los estados de la cadena oculta de Markov en el momento en el que se inicializa el sistema.

Teniendo en cuenta la composición de los elementos clave en un *HMP*, a continuación se listarán los parámetros utilizados para generar un modelo de canal basado en un proceso oculto de Markov:

1. *Número de estados que definen el proceso*, N.

2. *Número de valores de salida*, M. Dentro del marco del modelo de canal que se ha diseñado, los valores de salida corresponden a un proceso binario, que determinará si la recepción de una trama ha sido errónea (0) o correcta (1). Por lo tanto, en el ámbito del modelo *HMP*, $M = 2$.

3. *Matriz de transición* (A). Con dimensión $N \times N$, contiene las probabilidades de cambio entre estados, $a_{i,j}$. Así, los elementos de cada fila suman 1: $\sum_{j=0}^{N-1} a_{i,j} = 1$.

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ a_{2,0} & a_{2,1} & \cdots & a_{2,N-1} \\ \vdots & & \ddots & \vdots \\ a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,N-1} \end{bmatrix}$$

4. *Matriz de emisión* (B). Con una dimensión $N \times M$, refleja la probabilidad de decisión[12] k en función del estado actual en un instante $q_t = S_i$, representado como $b_i(k)$. Al igual que para la matriz de transición, los elementos de cada fila deberán sumar 1, $\sum_{k=0}^{M-1} b_i(k) = 1$.

   Su dimensión es $N \times M$ y cada elemento refleja la probabilidad de decisión k en función del estado actual en un instante $q_t = S_i$, representado como $b_i(k)$.

$$B = \begin{bmatrix} b_0(0) & b_0(1) \\ b_1(0) & b_1(1) \\ b_2(0) & b_2(1) \\ \vdots & \vdots \\ b_{N-1}(0) & b_{N-1}(1) \end{bmatrix}$$

5. *Distribución inicial* ($t = 0$). Define las probabilidades de pertenecer a un estado i, $\Pi = \{\pi_i\}$ en un tiempo $t = 0$. Por motivos de simplicidad, se asumirá un valor equiprobable para todos los estados de la cadena, $\pi_i = \frac{1}{N}$, por lo que, el estado inicial de la cadena se elegirá aleatoriamente.

Con el fin de poder replicar de una manera fidedigna el comportamiento real de un canal inalámbrico, se ha seguido el siguiente proceso a la hora de parametrizar el proceso oculto de Markov: en primer lugar, se deberán analizar aquellas trazas que se obtuvieron durante la campaña de medidas realizada sobre un escenario real (Sección 2.1). Para los 15 experimentos (Tabla 2.3), se tiene, para cada una las tramas recibidas en el nodo destino, un valor igual a 0 si la trama se había considerado corrupta o un 1 en caso contrario. Este vector binario generará la secuencia de *observables* ($\mathcal{O} = \{o_1, o_2, ..., o_N\}$) que servirá como parámetro de entrada al algoritmo de *Baum-Weltch* [Baum72], el cual, empleando un procedimiento iterativo, estimará los parámetros del *HMP*, $\lambda = (A, B, \Pi)$ que maximizan la verosimilitud con la secuencia de observables dado el modelo, máx $P(\mathcal{O}|\lambda)$. Este proceso se describe en la Figura 2.10a, donde una secuencia de observables sirve como parámetro de entrada a un proceso estocástico *oculto*.

[12] En concreto, en el modelo de canal empleado en esta Tesis será de dimensión $N \times 2$, ya que sólo existen dos posibles eventos observables: recepción satisfactoria o errónea.

La herramienta utilizada para llevar a cabo este "entrenamiento" ha sido la función `hmmtrain` de Matlab, que utiliza, además de la secuencia de observables $\mathcal{O}$ obtenida de una traza real, una condición para limitar el número de iteraciones realizadas por el algoritmo, en función de un valor fijo, o a partir de un criterio de tolerancia. Por último, se introducirán unas matrices iniciales de transición (A_0) y emisión (B_0), que sirven como base a la hora de generar el modelo, definiendo algunas características básicas, como el número de estados o la naturaleza de los mismos. Se ha asegurado que se cumplan las siguientes condiciones:

1. Todas las matrices de transición corresponderán con procesos de nacimiento y muerte (las transiciones entre estados se limitan al anterior o posterior).
2. Los estados extremos de las matrices de emisión deberán tener un comportamiento dual, es decir: $b_0(0) = 1, b_0(1) = 0, b_{N-1}(0) = 0, b_{N-1}(1) = 1$.
3. Se mantendrá una tonicidad decreciente en la cadena, siendo el estado 0 el que presente una probabilidad de pérdida mayor, que decrece a medida que aumenta el número de estado: $b_0(0) > b_1(0) > ... > b_{N-1}(0)$.

Como resultado, la función devolverá las matrices de transición y emisión generadas a partir de las restricciones anteriores, que serán posteriormente utilizadas para definir el comportamiento del canal inalámbrico en `ns-3`.

Desde el punto de vista del simulador, como se muestra en la Figura 2.10b, la interpretación del *HMP* cambia radicalmente. En este caso, la parte "observable" del proceso es la propia cadena de estados, siendo la transición entre los mismos lo que define realmente el comportamiento del sistema. Así, será la etapa de decisión (matriz de emisión) la que se encuentre oculta, tras la secuencia que marca las transiciones entre estados. En otras palabras, durante el flujo de ejecución del proceso, el canal irá transitando aleatoriamente entre los diferentes estados de la cadena de Markov, de acuerdo con las probabilidades de transición definidas a través de los coeficientes $a_{i,j}$. El evento producido por la recepción de una trama será el desencadenante del uso del segundo proceso estocástico, encargado de decidir si la trama en cuestión va a ser considerada correcta (o no). Para ello, en función del estado en el que se encuentre el proceso primario en ese instante, $q_t = S_i$, se utilizará el correspondiente coeficiente de la matriz de emisión, $b_i(0)$, para tomar tal decisión (realizando una simple comparación con un valor escogido aleatoriamente).

Utilizando una técnica similar, Cardoso *et al.* [Card09] emplean un esquema basado en trazas para ajustar los diferentes parámetros que modelan el *HMP*, también a partir del algoritmo de *Baum-Welch*. Sin embargo, existe una clara diferencia entre su implementación y la que se propone en esta Tesis: mientras que para la propuesta aquí descrita las medidas fueron realizadas en condiciones de saturación (el canal es el cuello de botella del sistema), en [Card09] los autores fijaron un intervalo de 10 milisegundos entre dos paquetes consecutivos (tasa binaria constante del nivel de aplicación), valor sensiblemente inferior al retardo medio de un canal saturado, caracterizado en la Sección 2.1.1.

A partir de las trazas obtenidas experimentalmente, el modelo propuesto en dicho trabajo ligaba la duración de las ráfagas a los valores observados en las trazas. Como se verá más adelante, en el momento en que las condiciones del escenario sean distintas a las que se utilizaron para obtener las trazas, el comportamiento del modelo *HMP* no será el adecuado, demostrando que la configuración del proceso es incorrecta. Sin embargo, si se sigue una aproximación que caracterice el proceso según un *régimen temporal*, modelando el tiempo de permanencia en cada estado a partir del cálculo analítico del retardo medio entre tramas consecutivas, se conseguirá desacoplar el *HMP* de la secuencia de observables que se usó para su configuración, separando su comportamiento de las características concretas del tráfico. Gracias a esta modificación de concepto, se podrá utilizar el modelo de canal basado en un *HMP* en diferentes tipos de escenarios y aplicaciones (incluyendo las que utilizan *TCP* como nivel de transporte, en el que el tiempo de transmisión entre segmentos puede variar considerablemente en función de la dinámica de sus mecanismos de control de flujo y congestión).

Para poder llevar a cabo este nuevo acercamiento, será necesario analizar estadísticamente el intervalo de permanencia en cada uno de los estados de la cadena. En una configuración tradicional, donde los eventos de cambios de estado se encuentran ligados a la recepción de una trama ($\approx$ sistema discreto), el proceso estocástico que define el comportamiento del *HMP* se puede mapear como una *distribución geométrica*, donde el número medio de tramas consecutivos en un estado i, $\overline{F_i}$, se puede calcular a partir de la Ecuación (2.16), donde $p_i(j)$ se define como la probabilidad de estancia en el estado i sea de j tramas.

$$\overline{F_i} = \sum_{j=0}^{\infty} j \cdot p_i(j) = \sum_{j=0}^{\infty} j \cdot a_{i,i}^{j-1} \cdot (1 - a_{i,i}) = \frac{1}{1 - a_{i,i}} \tag{2.16}$$

Extrapolando el proceso de Markov a un sistema continuo (en el que se puede producir un cambio de estado en cualquier instante), la distribución que más se asemeja a la utilizada en el sistema discreto será una *exponencial negativa*, cuya *fdp* se modela como $f(t) = \lambda_i \cdot e^{-\lambda_i \cdot t}$, siendo $\overline{T_i} = \frac{1}{\lambda_i}$ el tiempo medio de permanencia en un estado arbitrario i. En un canal saturado, se puede asumir que el retardo medio entre dos tramas consecutivas es constante (ψ), por lo que utilizando la expresión anterior se puede obtener el tiempo medio de permanencia en un estado i, $\overline{T_i}$, como se define en la Ecuación (2.17).

$$\overline{T_i} = \psi \cdot \overline{F_i} = \frac{\psi}{1 - a_{i,i}} \tag{2.17}$$

Modelado dinámico del tiempo de permanencia en un estado i

Como ya se ha dicho, se propone una nueva aproximación basada en la *caracterización temporal* del tiempo medio de llegada entre tramas consecutivas. Como solución inicial, se

asumió que este tiempo permanecía inalterable para cualquier estado (representado por el parámetro ψ en la Ecuación 2.17), sin considerar la calidad del canal o la presencia de ráfagas de errores. Sin embargo, para obtener un modelo más avanzado, se necesitará calcular, con un mayor nivel de precisión, cuánto tarda una trama en alcanzar los niveles superiores del nodo receptor tras k intentos de retransmisión (utilizando el esquema de retransmisiones del estándar *IEEE 802.11*), en función del estado actual i del *HMP*. La Ecuación (2.18) define la expresión con la que se obtiene este retardo[13], representado como $\overline{\Delta_k}(L)$, siendo L la longitud de la trama.

$$\overline{\Delta_k}(L) = \overbrace{(k+1)\cdot\tau_{det}(L)}^{Determinista} + \underbrace{\left[\left(16\cdot\sum_{j=0}^{k}2^j\right)-\frac{k+1}{2}\right]\cdot\sigma}_{Aleatorio} \tag{2.18}$$

Donde (recordando los elementos que definen el mecanismo *Automatic Repeat Request (ARQ)* del *MAC* del estándar *IEEE 802.11*):

- k. Este parámetro representa el número máximo de retransmisiones *IEEE 802.11* que han sido enviadas de manera consecutiva para una trama en particular. Un valor $k = 0$ indica que la trama ha sido recibida satisfactoriamente en el primer intento. Dado que el número máximo de retransmisiones se ha fijado a *3* intentos, $k \leq 3$. Por otra parte, $\overline{\Delta_3}$ no implica necesariamente que la trama haya sido recibida correctamente tras la tercera retransmisión, aunque se habrá ocupado el canal la misma cantidad de tiempo.

- τ_{det}. Como se ha caracterizado analíticamente en la Sección 2.1.1, el esquema *DCF* del estándar *IEEE 802.11* define unos mecanismos de acceso al medio cuya operación puede dividirse en dos contribuciones: en un primer lugar se encuentran aquellas que pueden clasificarse como deterministas, entre las que se encuentran tiempos como el *DIFS*, el *SIFS*, así como la transmisión correspondiente a las diferentes cabeceras (*PLCP*, *MAC*, *LLC*, *IP*, *TCP/UDP*, etc.) o a la información procedente del nivel de aplicación. La suma de todos estos componentes, cuyo único valor variable es la longitud de las cabeceras y del *payload*, no depende del esquema de retransmisiones empleado, pudiendo modelarse como un elemento fijo en función de la longitud de la trama, representado por $\tau_{det}(L)$.

- Por otra parte, existe una componente aleatoria, asociada al mecanismo de *backoff* del estándar *IEEE 802.11*, que define el tamaño de la ventana de contención en función del número de retransmisión, ya que su tamaño se duplica tras cada intento

[13]No se ha tenido en cuenta la circunstancia en la que la ventana de contención deja de crecer tras haber alcanzado el valor de CW_{max}, lo que no sucedía en las medidas que generaron las trazas con las que se configura el modelo

(siguiendo un proceso de *backoff* exponencial binario), como se refleja en la segunda parte de la Ecuación (2.18).

- Δ. Con este parámetro se refleja el tiempo de *slot* utilizado en la ventana de contención definido por los mecanismos de control de acceso al medio *DCF* del estándar *IEEE 802.11*. Como puede verse en la Tabla 2.1, el valor por defecto de σ para una transmisión *IEEE 802.11b* será de 20 microsegundos.

Cada uno de los estados se caracteriza con la probabilidad subyacente que tiene una trama de ser correcta $(1 - p_i)$ o errónea (p_i), siendo i el estado actual en el que se encuentra el canal en la cadena de Markov en un instante dado, $q_t = S_i$. Dichos valores se pueden extraer directamente de la matriz de emisión del *HMP*, donde $p_i = b_i(0)$. A partir de este sistema se puede calcular fácilmente el número de retransmiones (definido por la variable k) que, en promedio, se llevarán a cabo en la transmisión de un paquete, en función del estado en el que se encuentra el proceso[14]. Para ello, la expresión tendrá que adaptarse al esquema de retransmisiones definido en el estándar *IEEE 802.11*, teniendo en cuenta el número máximo de reintentos antes de dar a un paquete como perdido. Con todo esto, la Ecuación (2.19) define el retardo medio generado por una trama cuando el proceso de Markov se encuentra en el estado i, donde R representa el número máximo de retransmisiones, fijado a 3 durante todo el análisis llevado a cabo. Aunque ya se ha mencionado con anterioridad, es preciso recalcar una vez más que el hecho de no poder recuperar el paquete original no implica que el canal no haya sido utilizado, por lo que su contribución deberá ser igualmente tenida en cuenta por dicha expresión (parte derecha de la ecuación, que corresponde con una ráfaga de $R + 1$ tramas erróneas).

$$E\left[\Delta_k(L)|S_i\right] = \psi_i = \overbrace{\sum_{k=0}^{R}(1-p_i)\cdot p_i^k \cdot \overline{\Delta_k}(L)}^{\text{Paquete correcto}} + \underbrace{p_i^{R+1}\cdot\overline{\Delta_R}(L)}_{\text{Paquete perdido}} \tag{2.19}$$

Finalmente, con el retardo medio entre tramas, ψ_i, se puede convertir la duración en un estado a unidades temporales, tal y como se recoge en la Ecuación 2.20. Así, en aquellos estados en los que la calidad del canal sea baja, la duración media de ocupación del canal por paquete será mayor, por lo que este incremento se reflejará en el tiempo de permanencia en el estado, con una consecuencia directa sobre las ráfagas de errores, que tendrán una mayor longitud que si se hubiera mantenido constante el valor de ψ.

$$\overline{T_i} = \frac{\psi_i}{1 - a_{i,i}} \tag{2.20}$$

[14] A modo de ejemplo, asumiendo que el estado actual en el proceso de Markov es S_i, la probabilidad de que un paquete requiera de dos retransmisiones antes de ser recibida correctamente se puede calcular como $p_i^2 \cdot (1 - p_i)$, donde hay dos transmisiones erróneas y una correcta.

Al contrario que en la configuración basada en tramas, en el que las transiciones entre estados se asociaban a los eventos producidos por la recepción de una trama, los cambios en esta configuración tendrán una naturaleza asíncrona, pudiendo suceder en cualquier instante. Para ello, y como ya se mencionó anteriormente, se asumirá que el proceso estocástico que define cada uno de los estados se rige por una distribución exponencial negativa de valor medio $\overline{T_i}$.

De entre todos los resultados obtenidos sobre el escenario real (Tabla 2.3 en la Sección 2.1.1), se han elegido *tres* de ellos para representar, de manera ilustrativa, el gran rango de comportamientos que se observan en un típico escenario en el interior de un edificio, cubriendo desde una situación con un canal *Malo* (Medida #5), caracterizado por unas condiciones muy adversas, hasta uno *Bueno* (Medida #12), cuyo rendimiento se encuentra cercano al que se obtendría sobre un canal ideal. Por último, en una situación intermedia se configurará un canal *Medio* (medida #9). La Tabla 2.6 resume los principales parámetros de estas tres calidades de enlace.

Tabla 2.6: Medidas *UDP* empleadas para configurar el modelo *HMP*

Canal	***FER***	***PER***	***Thput (Mbps)***	***Ráfaga de errores (EFB)***		
				Media	***Varianza***	***Máxima***
Bueno	0.163	0.025	4.79	2.63	57.63	144
Medio	0.298	0.127	3.80	4.836	301.49	219
Malo	0.517	0.179	2.33	6.22	983.66	821

Modelado (dinámico) basado en tiempo *Vs.* modelado basado en tramas

La contribución más relevante en relación a este modelo se encuentra ligada a la caracterización temporal del mismo, en contrapartida a las típicas configuraciones basadas en los procesos discretos de recepción de tramas, técnica más habitual a la hora de entrenar los parámetros asociados al proceso oculto de Markov. Como se ha discutido anteriormente, esta aproximación presenta un alto grado de correlación con las secuencias de observables que ajustaron los parámetros del *HMP*, haciéndolo muy preciso cuando las condiciones de simulación son *exactamente* las mismas que las utilizadas durante la caracterización experimental (como por ejemplo, cuando en [Card09] se entrena el modelo con un tiempo de 10 *mseg* entre paquetes consecutivos). Sin embargo, si las condiciones cambian, el modelo perderá toda su credibilidad. Por otro lado, si se utiliza una caracterización temporal, el proceso se desacopla de las secuencias de entrenamiento utilizadas para ajustar el *HMP*, adaptándose dinámicamente a las condiciones del escenario.

Para demostrar que la configuración del proceso *basado en tramas* no es capaz de recoger las variaciones producidas en el contexto de la simulación, se ha llevado a cabo un análisis complementario, en el que se varía la tasa binaria de los datos enviados desde el nivel de aplicación, utilizando tres variantes: en la primera de ellas se mantendrá una tasa

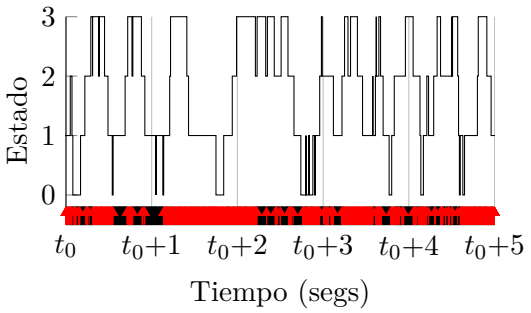
Tabla 2.7: Estadísticas obtenidas sobre un canal *HMP* no saturado ($R_b = 600$ *Kbps*)

Canal	***FER***	***Ráfagas de errores***		
		Media	***Varianza***	***Máxima***
Real (traza sintética)	0.5191	3.76	22.18	19
Modelo basado en tramas	0.4876	5.375	237.97	166
Modelo basado en tiempo	0.5398	3.774	16.045	28

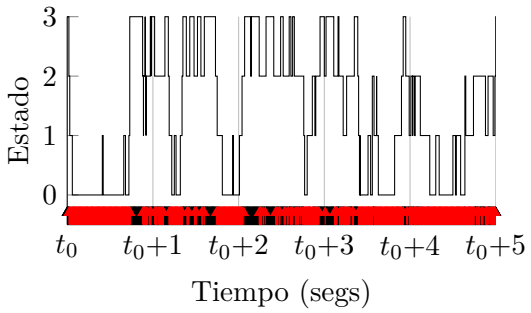
superior a la capacidad del canal. Bajo estas circunstancias, tal y como se ha calculado en la Sección 2.1.1, el nodo destino recibirá las tramas con un retardo medio de unos 2 *mseg*, con un *throughput* resultante de ≈ 6 *Mbps*. En los dos siguientes casos, las tasas de transmisión se han reducido en un factor de 10 (600 *Kbps* - 20 *mseg* entre tramas consecutivas) y 100 (60 *Kbps* - 200 *mseg*), por lo que la red ya no se encontrará en una situación de saturación. Además, para evitar la aparición de tiempos espurios entre paquetes (que puedan alterar los resultados), el esquema de retransmisiones *IEEE 802.11* ha sido deshabilitado.

Para evaluar este comportamiento anómalo, se ha reutilizado la traza correspondiente al canal *Malo*, variando artificialmente las condiciones de tráfico, para lo que se ha realizado un proceso de diezmado, extrayendo una de cada 10 tramas en el fichero traza, con lo que se generará un nuevo fichero. Con estas condiciones, el retardo medio entre dos recepciones consecutivas será de aproximadamente 20 milisegundos, correspondiendo con un valor de *throughput* (asumiendo paquetes de 1472 octetos) de unos 600 *Kbps*. De una forma similar a la mostrada en la etapa de caracterización del canal, en la Tabla 2.7 se representan los principales estadísticos que caracterizan las transmisiones realizadas, comparando los resultados de la traza sintética y de las dos configuraciones de modelo de canal basado en un *HMP*. En primer lugar, se destaca el comportamiento de la traza sintética, ya que, a pesar de haberla diezmado, sigue manteniendo las características de un canal de calidad baja, con una *FER* superior al 50 %, aunque se aprecia una reducción significativa en relación con las ráfagas de errores respecto a la traza original. Se puede decir que al aumentar el retardo en el canal se disminuye el efecto memoria del mismo. Con respecto al modelo *basado en tramas*, se observa una buena aproximación en términos de *FER*, aunque mantiene el mismo nivel de efecto memoria, con unas ráfagas ostensiblemente más largas que las observadas en el escenario real, conservando el comportamiento de la secuencia original. Por último, el *modelo temporal* sí es capaz de mantener un valor adecuado de *FER*, reduciendo la memoria del canal, ya que como la longitud de las ráfagas de errores disminuye significativamente, poniendo de manifiesto un comportamiento más flexible, que se adapta mejor a las condiciones cambiantes que puedan darse en una red.

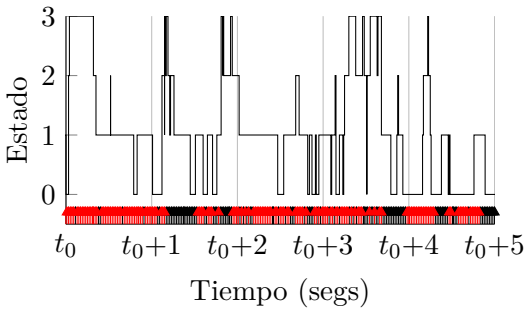
La Figura 2.11 muestra el seguimiento temporal de los estados de la cadena durante el tiempo en el que transcurre la simulación, durante un intervalo de cinco segundos. Además, en la parte inferior de la gráfica se representan los eventos discretos correspondientes a la recepción de tramas en el nodo destino (en color rojo se representan aquellas que han



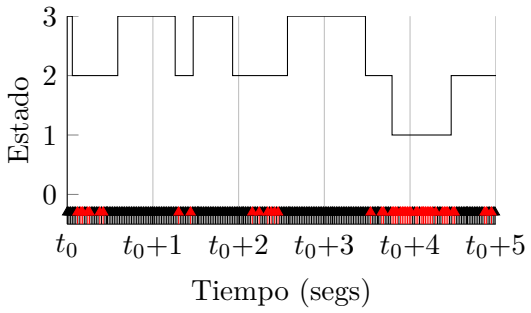
(a) Modelo basado en tiempo (canal saturado)



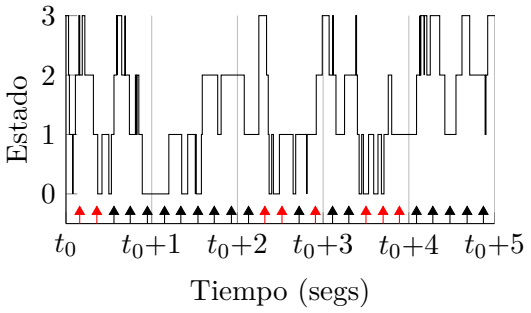
(b) Modelo basado en tramas (canal saturado)



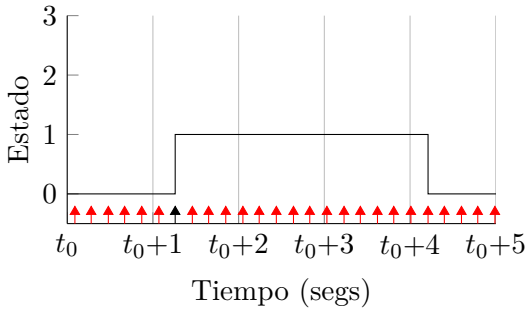
(c) Modelo basado en tiempo (600 *Kbps*)



(d) Modelo basado en tramas (600 *Kbps*)



(e) Modelo basado en tiempo (60 *Kbps*)



(f) Modelo basado en tramas (60 *Kbps*)

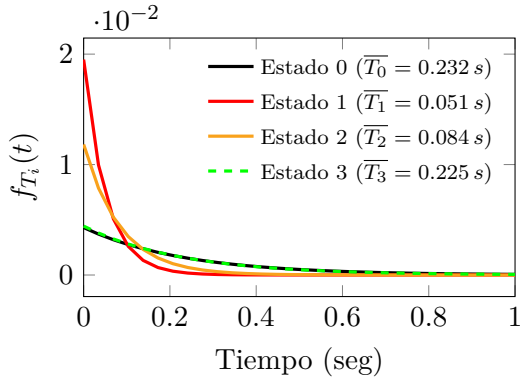
Figura 2.11: Pérdida de precisión inherente del modelo *HMP* basado en tramas sobre canales no saturados

sido consideradas erróneas, mientras que en negro se marcan las correctas). Como puede apreciarse (observando por ejemplo la secuencia de Figuras 2.11a → 2.11c → 2.11e, que ilustran el comportamiento del modelo *basado en tiempo* ante los diferentes flujos de tráfico), a medida que la tasa de envío desde el nivel de aplicación disminuye, la separación entre las recepciones aumenta. Bajo estas condiciones, la caracterización temporal mantiene el tiempo medio de permanencia en cada estado, independientemente del patrón del tráfico, demostrando que los eventos que generan la transición entre éstos están desacoplados de la recepción de tramas, pudiendo apreciarse situaciones en las que se producen varios cambios de estado en el intervalo entre recepciones de dos tramas consecutivas. De este modo, el efecto *memoria* observado en un canal real (que induce la aparición de ráfagas de errores) se verá reducido, ya que la correlación entre los procesos que definen los eventos de recepción disminuirá significativamente. Sin embargo, el modelo *basado en tramas* presenta una respuesta completamente distina, en la que se observa cómo estos tiempos de permanencia se van "estirando", reflejando las nuevas condiciones del tráfico. Este comportamiento encuentra una explicación en la Ecuación (2.17), en la que el parámetro ψ refleja el retardo medio entre tramas consecutivas, con lo que si éste aumenta (en un factor $\times 10$, Figura 2.11d y $\times 100$, Figura 2.11f), el tiempo medio de permanencia en cada estado crecerá, para mantener la misma escala. Desde otro punto de vista, al estar el cambio de estado ligado a los eventos de recepción, y requiriendo (por promedio) el mismo número de intentos para que se produzca una transición, el modelo se comportará del mismo modo, independientemente del tiempo entre llegadas, ya que en este caso, la variable ψ está asociada al tiempo medio de ocupación del canal de una trama (individual) y no al retardo medio entre dos llegadas consecutivas. Como puede verse en la representación de las recepciones correctas/erróneas, el hecho de incrementar la separación entre dos tramas consecutivas no implica la reducción de la correlación entre las mismas, por lo que, como ya se comprobó en la Tabla 2.7, el comportamiento a ráfagas va a presentar los mismos resultados.

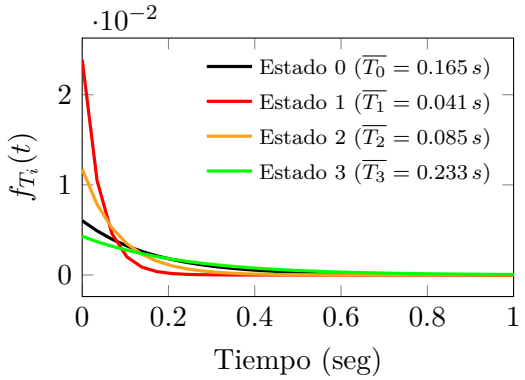
Para extender el análisis realizado hasta este punto se analizará el tiempo de permanencia en los estados (al igual que antes, se empleará el canal *Malo* como traza de referencia). Para ello, se ha repetido el experimento hasta un total de 500 realizaciones por escenario, para representar (Figura 2.12) la *fdp* del tiempo de permanencia en cada uno de los estados del proceso de Markov. Las Figuras 2.12a, 2.12c y 2.12e) muestran la independencia entre la caracterización temporal del sistema y las características del tráfico. Por otra parte, las Figuras 2.12b, 2.12d y 2.12f exhiben una respuesta opuesta, ya que la caracterización estadística del tiempo en cada estado presenta una gran correlación con las condiciones del escenario utilizado para entrenar al modelo *basado en tramas*.

Por último, se ilustra en la Figura 2.13 el comportamiento combinado de los dos procesos estocásticos que componen el modelo, para cada una de las calidades de canal estudiadas (*buena*, *media* y *mala*). Tal y como se representa en la figura[15], se asume que

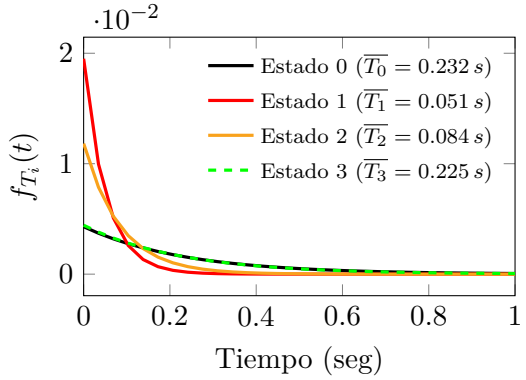
[15]Realizada a partir de 500 realizaciones para cada configuración, manteniendo el mismo escenario que el presentado en la Sección 2.1.



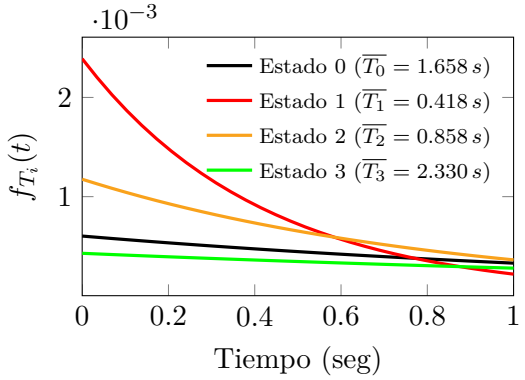
(a) Modelo basado en tiempo (canal saturado)



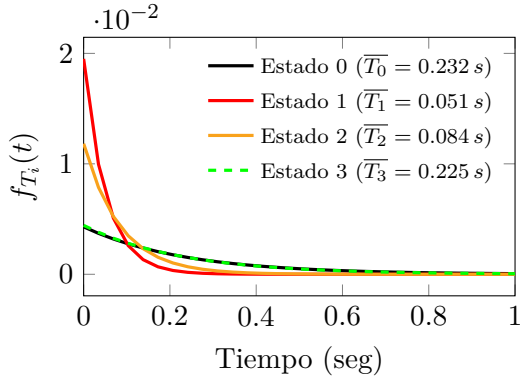
(b) Modelo basado en tramas (canal saturado)



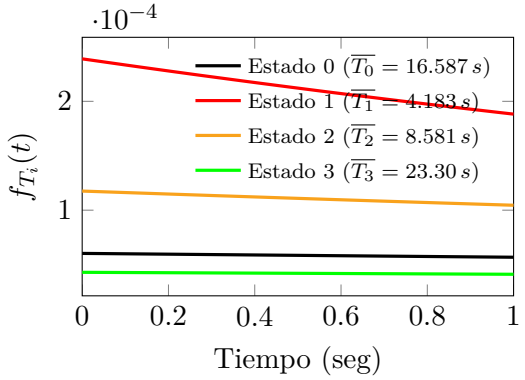
(c) Modelo basado en tiempo ($R_b = 600\ Kbps$)



(d) Modelo basado en tramas ($R_b = 600\ Kbps$)



(e) Modelo basado en tiempo ($R_b = 60\ Kbps$)



(f) Modelo basado en tramas ($R_b = 60\ Kbps$)

Figura 2.12: Función densidad de probabilidad del tiempo de permanencia en los estados de la cadena de Markov

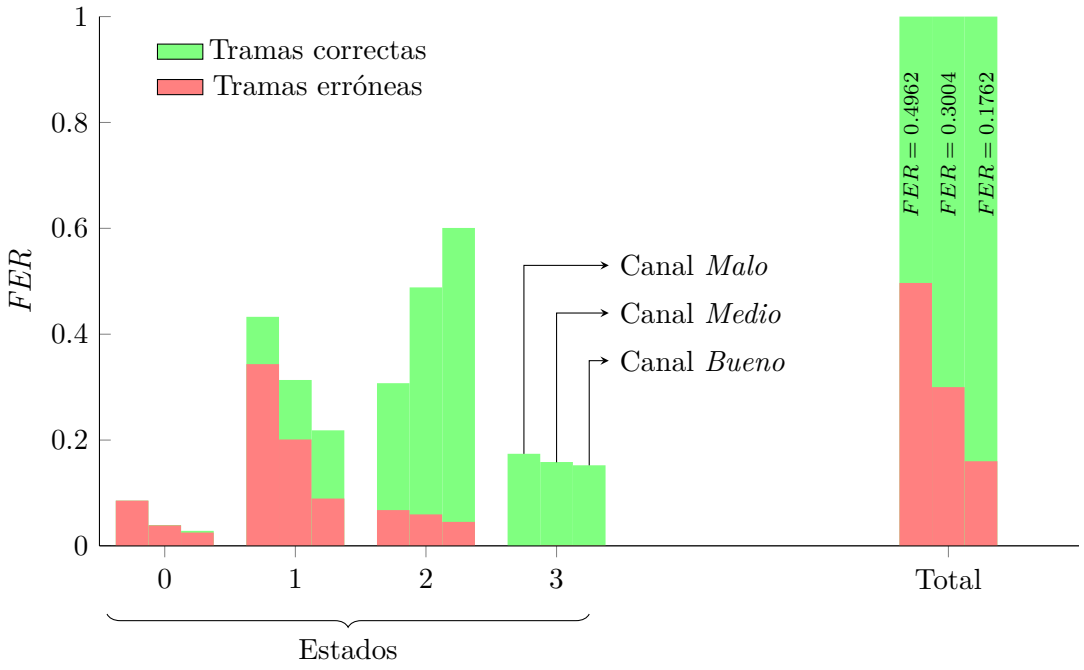


Figura 2.13: Comportamiento del canal en función del estado de la cadena de Markov

ambos procesos son ortogonales, por lo que, dentro de un estado i, la probabilidad de decisión es independiente de la definida en otro estado j, con $j \neq i$. De hecho, se podría afirmar que un *HMP* de N estados se corresponde con un modelo de *Gilbert-Elliott* [Gilb60; Elli63] *N-dimensional*. Asimismo, se puede apreciar otro efecto que no ha sido tenido en cuenta hasta este momento: el porcentaje de tiempo en el que la cadena estará, *en total*, dentro de cada estado. Se observa el impacto que tendrá sobre la tasa de error: en el canal *Malo* el proceso permanece más de un 60 % del tiempo en los estados más hostiles (0 y 1); por otra parte, el canal bueno se mantendrá casi un 75 % del tiempo en los estados 2 y 3, dando lugar a un mejor comportamiento.

2.3.2. Modelo de errores a ráfagas basado en un filtro AR

Otra de las aproximaciones para modelar un canal inalámbrico es la propuesta de Agüero *et al.* [Agüe10]: *BEAR*, un modelo cuyo rasgo principal se basa en la estimación de la señal percibida en el receptor a través de un filtro auto-regresivo, cuyos coeficientes se ajustan en base a los resultados obtenidos en medidas reales. Este filtro añadirá memoria al canal, causando cierta correlación entre recepciones consecutivas.

Dentro del marco concreto de esta Tesis, se ha portado la funcionalidad de *BEAR* al entorno del simulador `ns-3`[16], llevando a cabo una profunda caracterización del mismo, como se describe a lo largo de la Sección 2.4. Adaptándose a la arquitectura que define

[16]La implementación original del modelo fue desarrollada en el simulador `ns-2`.

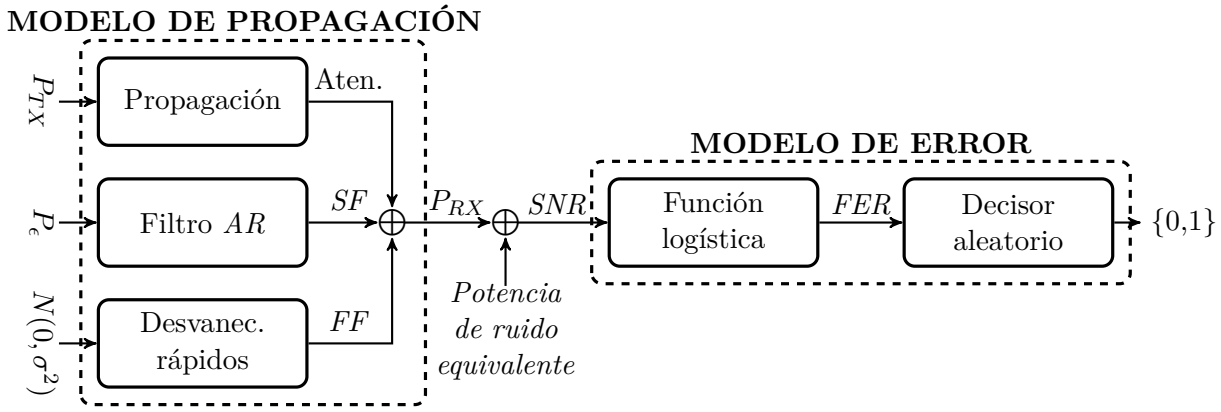


Figura 2.14: Implementación del modelo *BEAR*

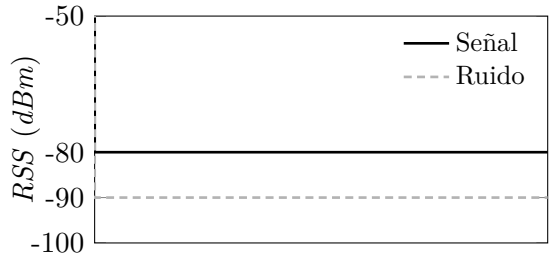
a la capa física en `ns-3`, el modelo de canal se estructura en tres etapas diferentes, como muestra la Figura 2.14: en primer lugar se calcula la potencia de señal recibida de la trama tras su propagación por el medio radio; posteriormente, este valor se combina con un proceso independiente que modela el ruido o nivel de interferencia asociado a la transmisión, estimando así la relación señal a ruido o *SNR* que sirve de parámetro de entrada de la tercera etapa, que determina si una trama se ha recibido de manera correcta o no. A continuación se detalla la operación correspondiente a cada una de las tres fases.

La piedra angular de *BEAR* es la estimación de la calidad de la señal recibida en una primera etapa de propagación, que puede verse como la suma de tres contribuciones diferentes, como se ilustra en la Figura 2.15:

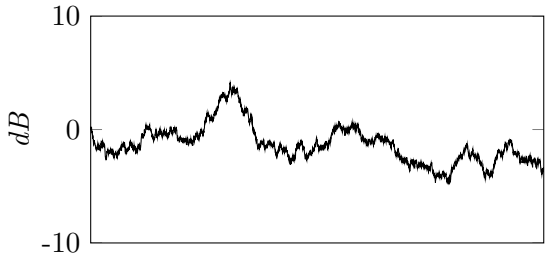
- La primera de ellas se encuentra estrechamente ligada a la *distancia física* entre los nodos transmisor y receptor, d, y se calcula de manera proporcional a $d^{-\nu}$, donde ν es un parámetro cuyo valor depende del entorno de propagación (ver Tabla 2.5). A pesar de la amplia gama de modelos disponibles en el marco del simulador `ns-3` (e.g. *Friis*, *Nakagami*, etc.), en este trabajo se utilizará una aproximación sencilla, basada en el cálculo determinista de la atenuación de la señal en base a una pérdida logarítmica en función de la distancia (ver Figura 2.15a), haciendo uso de la Ecuación 2.6.

- Una segunda contribución refleja los desvanecimientos lentos producidos en el canal, también conocidos como *shadowing* (Figura 2.15b), típicamente asociados a la presencia de obstáculos físicos entre transmisor y receptor, cuando las variaciones de amplitud y fase son más lentas que el *tiempo de coherencia* del canal[17]. Estos fenómenos tienden a ser modelados a través de una variable aleatoria *log-normal* con una desviación típica acorde con la atenuación sufrida [Stüb01]. *BEAR*, por el contrario,

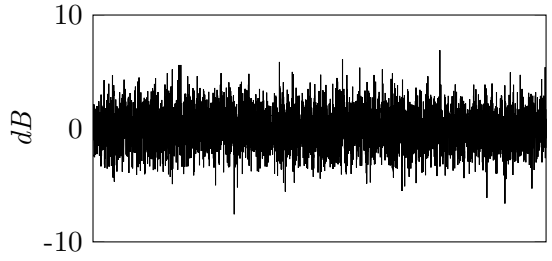
[17]Se define el tiempo de coherencia como el intervalo en el que la respuesta al impulso del canal puede ser considerada como constante.



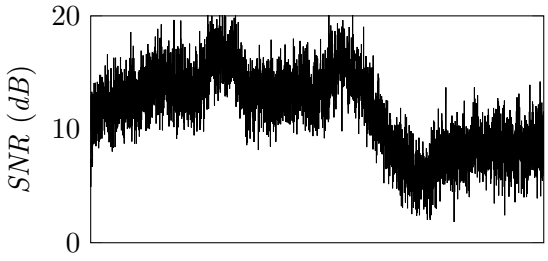
(a) Propagación (modelo determinista)



(b) Contribución debida a los desvanecimientos lentos (*Slow Fading - SF*)



(c) Contribución debida a los desvanecimientos rápidos (*Fast Fading - FF*)



(d) Relación señal a ruido global de la trama recibida

Figura 2.15: Ejemplo ilustrativo del cálculo de la señal recibida en *BEAR*

hace uso de un filtro AR, por lo que el desvanecimiento lento o SF en un instante temporal será calculado teniendo en cuenta los producidos anteriormente, tal y como se muestra en la Ecuación (2.21), de manera que el sistema adquiera un grado de memoria. Dada la naturaleza empírica del modelo, los coeficientes del filtro ($a[j]$) serán obtenidos utilizando el método de *Yule-Walker* [Yule27; Walk31] a partir de los valores obtenidos experimentalmente. El valor actual de esta componente, $SV[i]$, se calculará a partir de las muestras anteriores, $SV[i-j]$, limitando la ventana de muestreo al orden del filtro AR, F. Por su parte, $\varepsilon[i]$ introduce aleatoriedad al sistema y será configurado como una fuente de ruido blanco con una densidad espectral de potencia ($P_\epsilon\ mW/Hz$).

$$SV[i] = \sum_{j=0}^{F} a[j] \cdot SV[i-j] + \varepsilon[i] \tag{2.21}$$

Por último, con el fin de desacoplar el modelo de canal del patrón de tráfico (por ejemplo, en escenarios en los que el canal no se encuentre saturado), se ha incluido un temporizador, denominado T_c, que limita la validez de las muestras previamente calculadas; en otras palabras, una trama sólo afectará a las siguientes durante un intervalo finito de tiempo. Este parámetro tendrá una influencia especial en transmisiones que utilicen *TCP* como nivel de transporte, ya que un nodo fuente puede mantenerse inactivo durante periodos de tiempo elevados.

- La tercera de las contribuciones refleja los efectos producidos por la propia naturaleza multi-camino de un medio inalámbrico, donde la recepción de múltiples "ecos" de la señal provoca grandes cambios en términos de amplitud y fase, cortos en relación con el tiempo de coherencia del canal (Figura 2.15c). El modelado de este tipo de desvanecimientos, conocidos como *rápidos* o *FF*, se realiza a través de una fuente de ruido blanco *Gaussiano*, de media nula y varianza σ^2, $N(0, \sigma^2)$.

La suma de estas tres contribuciones conformará el valor de potencia recibida, P_{RX}. Posteriormente, y antes de pasar a la siguiente etapa, se descartan aquellas tramas cuya potencia sea inferior a un umbral $P_{umbral_{RX}}$, cuyo valor depende del dispositivo y de la modulación empleada. En el caso de que la potencia sea superior, el esquema de interferencia incorporado por defecto en el simulador se encargará de calcular el nivel de ruido equivalente del canal (representado en la Figura 2.15a), combinando este resultado con el valor P_{RX} y estableciendo finalmente la relación señal o ruido que actuará como entrada para el modelo de error. Es importante mencionar que el cálculo de la potencia de ruido se ejecuta de una manera completamente independiente [NS3Int], utilizando parámetros como el ancho de banda del canal, la figura de ruido del dispositivo o los niveles de interferencia en el momento del análisis.

Una vez obtenida la *SNR*, el modelo de error será el encargado de tomar una decisión en función de la misma (cuya representación gráfica global aparece en la Figura2.15d). En *BEAR*, a diferencia del modelo de canal estándar, donde se suele calcular la probabilidad de error de *bit* (*BER*), se ha ajustado, a partir de los resultados obtenidos en la campaña experimental descrita en la Sección 2.1, una *función logística* que establece una relación entre la *SNR* y la probabilidad de error de trama, según la Ecuación (2.22), que permite configurar el comportamiento del modelo en base a los umbrales *LT* y *HT* y los parámetros *a*, *b* y *c*.

$$\widehat{FER} = \begin{cases} 1, & SNR < LT \\ \dfrac{a}{1 + e^{b \cdot (SNR - c)}}, & SNR \, \epsilon \, [LT, HT] \\ 0, & SNR > HT \end{cases} \tag{2.22}$$

Tal y como se describe en [Agüe07], la relación entre la *FER* y la *SNR* no es única, sino que depende principalmente de dos factores: la longitud de la trama y la modulación. En cuanto al primero de ellos, será especialmente relevante modelar adecuadamente el comportamiento frente a los segmentos de reconocimiento utilizados por el protocolo *TCP*, pues serán importantes en el desarrollo de esta Tesis. Estos mensajes se verán afectados por una *FER* notablemente menor que la correspondiente a las tramas de longitud máxima. Por otra parte, cuanto menor sea la tasa binaria, menor será también el valor de *FER* para una misma *SNR*. El protocolo *IEEE 802.11* establece que tanto las tramas *broadcast* como los mensajes de control (e.g. reconocimientos) se transmitan a una tasa binaria de 2 *Mbps*, por lo que la respuesta del modelo del canal frente a la calidad de la señal recibida será

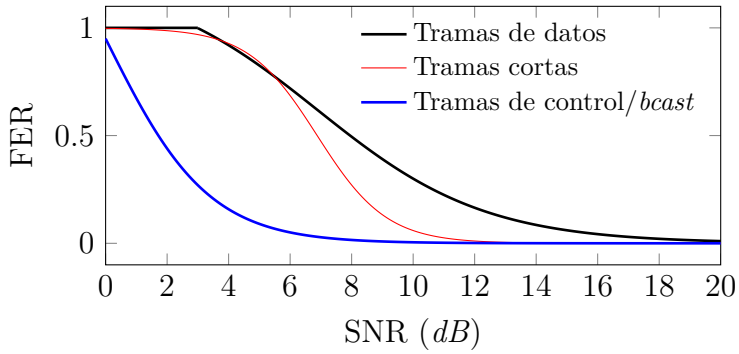


Figura 2.16: Funciones logísticas para tres tipos de tramas

diferente. En base a esto, se configura una función logística para cada uno de los puntos de interés definidos (tramas cortas, largas y mensajes de control/*broadcast*). La Tabla 2.8 recoge los parámetros utilizados para cada uno de los casos, representando las funciones correspondientes en la Figura 2.16, donde se aprecia claramente como las transmisiones más lentas (o de tramas más cortas) serán más robustas frente a una *SNR* menor.

Tabla 2.8: Parámetros de ajuste de la función logística (*BEAR*)

Parámetro	**Tramas de datos** ($L > 1300$ *bytes*)	**Tramas cortas** ($L > 40$ *bytes*)	**Tramas *broadcast*/control** ($R_B = 2$ *Mbps*)
a	1.24	1	1.9
b	0.366	0.886	0.6
c	6.88	6.88	0
LT	3	0	0
HT	16	13	10

Ilustrando el comportamiento general del modelo, en la Figura 2.17 se observa la fdp que caracteriza estadísticamente la *SNR* observada en la transmisión de 100000 paquetes en un enlace directo entre dos nodos. Se pueden aprecian también las diferentes distribuciones de las *SNRs* correspondientes a las tramas correctas y erróneas, situándose el valor medio de las primeras unos $3-4$ *dB* por debajo del de las segundas, reflejando un comportamiento similar al exhibido por un canal real [Agüe10].

2.3.3. Modelo *Nativo* del simulador

Tras haber descrito los dos modelos anteriores se presenta en esta sección una tercera alternativa, ofrecida por defecto por el simulador, siendo por tanto una solución representativa de los esquemas utilizados por la comunidad de usuarios de `ns-3`. Así, será utilizado como marco de referencia para ser comparado con *HMP* y *BEAR*.

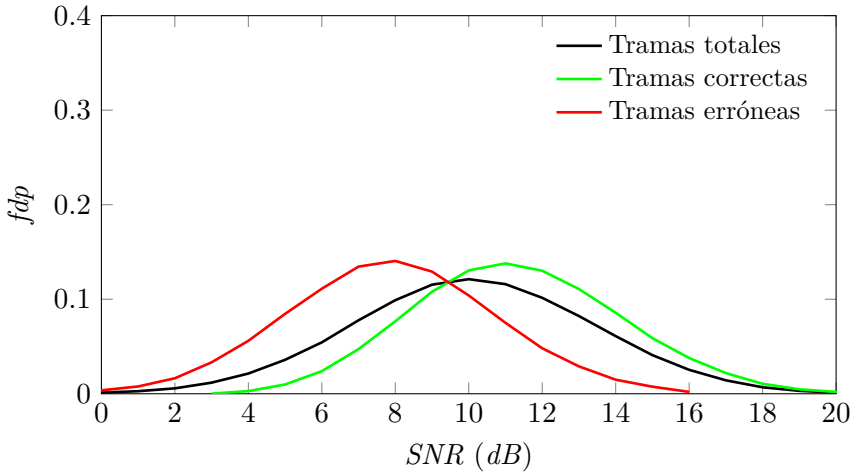


Figura 2.17: Función densidad de probabilidad de la *SNR* obtenida con *BEAR*. Parámetros: atenuación *log distance* $\nu = 3$, entrada de ruido del filtro AR $P_\epsilon = 5\ mW/Hz$, desvanecimientos rápidos $N(0, 2.8\ \mathrm{dB}^2)$

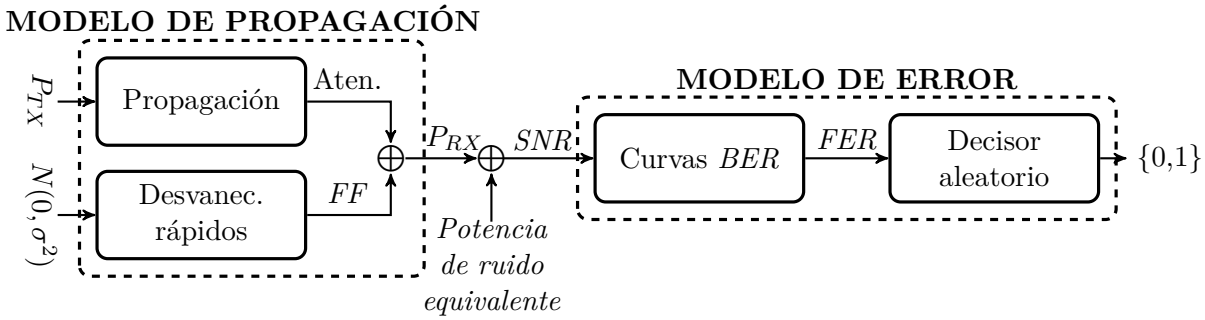


Figura 2.18: Implementación del modelo *Nativo*

Como puede verse en la Figura 2.18, este modelo presenta un cierto grado de semejanza con *BEAR*, manteniendo dos de sus tres etapas en el módulo de propagación, aunque prescinde de su elemento diferencial: el *filtro auto-regresivo*, lo que elimina la memoria del canal, dificultando la presencia de ráfagas significativas de errores durante una transmisión. Con respecto a los modelos de propagación que sí se han introducido, se encuentran configurados del mismo modo que en *BEAR*: primero, el cálculo de la atenuación utiliza un esquema "*log distancia*" cuyo factor de atenuación se modela a través de su exponente de pérdidas, ν; posteriormente, la contribución derivada de los desvanecimientos rápidos, se modela mediante una *distribución aleatoria normal* de media nula y varianza σ^2, $N(0, \sigma^2)$. Es preciso remarcar que en las configuraciones más típicas del simulador, solamente se tiene en cuenta el proceso de atenuación determinista, habiéndose añadido la componente aleatoria con fines exclusivamente comparativos.

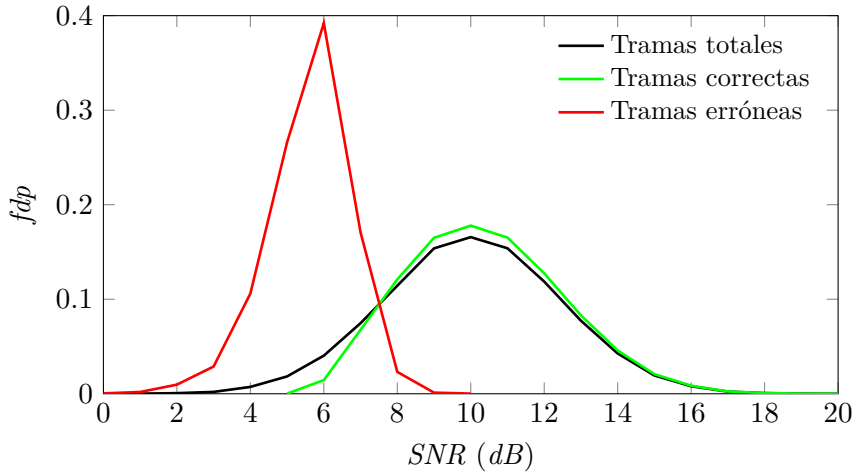


Figura 2.19: Función densidad de probabilidad de la *SNR* obtenida con el modelo *Nativo*. Parámetros: atenuación *log distance* $\nu = 3$, desvanecimientos rápidos $N(0, 2.8\ \mathrm{dB}^2)$

Una vez calculada la suma de las dos contribuciones se establece el valor del nivel de señal de entrada al nodo[18], P_{RX}. El siguiente paso, al igual que en *BEAR*, será el de combinar este parámetro con el nivel de ruido presente a la entrada del receptor, proporcionado por el modelo de interferencia que por defecto incluye el simulador; así se determina la *relación señal a ruido* o *SNR*, parámetro de entrada al modelo de error.

En una tercera etapa, se calculará la probabilidad de error de *bit*, en función de la calidad de la señal recibida y de la modulación empleada, utilizando para ello las curvas *BER* definidas en [Pei09a]. Con el valor resultante, y a partir de la longitud de la trama, determina la probabilidad de error de trama, tal y como se define en la Ecuación (2.15).

A modo de ejemplo ilustrativo, en la Figura 2.19 se muestra la fdp de la *SNR* recogida durante una transmisión de 100000 paquetes en un enlace directo transmisor-receptor, observando las curvas correspondientes a las tramas correctas y erróneas, así como la combinación de ambas. Resulta interesante observar, comparando con lo visto para *BEAR* (Figura 2.17), la menor variabilidad obtenida con este modelo *Nativo*, así como un resultado bastante más optimista en cuanto a la tasa de error obtenida, ya que la mayor parte de las tramas recibidas será interpretadas como correctas (la *fdp* de la *SNR* de las tramas erróneas puede resultar engañosa, ya que el porcentaje de errores es pequeño).

2.4 Resultados

Una vez descritas las tres aproximaciones que se han utilizado para replicar el comportamiento de un canal inalámbrico en escenarios interiores, se procederá a presentar los

[18]La señal resultante sería la combinación de las mostradas en las Figuras 2.15a y 2.15c.

principales resultados obtenidos, comparándolos con los valores observados durante la etapa de caracterización sobre un escenario real. Respetando en la mayor medida posible el procedimiento *experimental* (descrito en la Sección 2.1), el escenario en cuestión consiste en un enlace punto a punto inalámbrico entre dos nodos, uno de ellos transmitiendo al otro un total de 10000 paquetes de datos, con una longitud de trama de 1500 *bytes* (correspondiéndose con 1472 octetos de información cuando se utilice *UDP* y 1460 en el caso de *TCP*, debido a la sobrecargada asociada a sus cabeceras). La tasa de inyección de paquetes desde el nivel de aplicación asegura que siempre haya al menos uno esperando ser transmitido en el *buffer* de salida del nodo, de manera que el canal esté saturado, siendo el cuello de botella del sistema. De este modo, se obtendrán los umbrales máximos de rendimiento soportados por un enlace inalámbrico basado en el estándar *IEEE 802.11b*, que, en el caso de utilizar *UDP*, serán similares a los calculados en la Sección 2.1.1. Antes de continuar con la descripción del escenario, es preciso destacar que la potencia de transmisión se ha reducido de $P_{TX} = 16.0206\ dBm$ (valor por defecto en el simulador) a $0\ dBm$, para adaptar las distancias a aquellas vistas sobre el canal real.

A continuación se describen las configuraciones empleadas por cada uno de los modelos de canal estudiados:

- En lo que respecta al modelo *HMP*, se utilizarán cadenas de Markov de cuatro estados[19]. Como se mencionó anteriormente, se han seleccionado tres de los resultados obtenidos en la campaña de medidas (Tabla 2.3), tratando de ilustrar otras tantas calidades de enlace: un canal *Bueno* (medida #12), que presenta una tasa de error de trama de $\approx 16\,\%$, un canal *Medio* (medida #9), cuya $FER \approx 29\,\%$, y un canal *Malo* (medida #5), en el que más de la mitad de las tramas transmitidas no serán recibidas correctamente por el nodo receptor. En la Tabla 2.6 se recogió el comportamiento asociado a cada una de las calidades del canal empleadas para entrenar al módulo *HMP*.

- En el caso de *BEAR*, el exponente de pérdidas ν que define el comportamiento del modelo de atenuación "*log distancia*" tendrá un valor de 3; además, el filtro AR será de orden 3 ($T = 3$), con una entrada de ruido blanco de potencia $P_\epsilon = 5 \cdot 10^{-3}\ W/Hz$; finalmente, el componente de desvanecimiento rápido (FF) se modela con una variable aleatoria normal $N(0, 2.8\ dB^2)$. Por otro lado, el temporizador T_c, que establece la validez temporal de las muestras, se ha fijado a 2 segundos.

- Por último, el modelo *Nativo* mantendrá la misma fuente de ruido normalizado que *BEAR*, $N(0, 2.8\ dB^2)$, para definir la componente de variación rápida (FF), así como un valor de $\nu = 3$ para el modelo de atenuación "*log distancia*". Además, para poder comparar los resultados con los del modelo *HMP*, se buscarán tres configuraciones que presenten unas condiciones similares a éste (es decir, canal *Bueno*, *Medio* y *Malo*).

[19]Se ha estudiado el rendimiento con cadenas de 3, 4, 8 y 16 estados, obteniendo resultados similares, por lo que de aquí en adelante se trabajará exclusivamente con un proceso oculto de Markov de 4 estados.

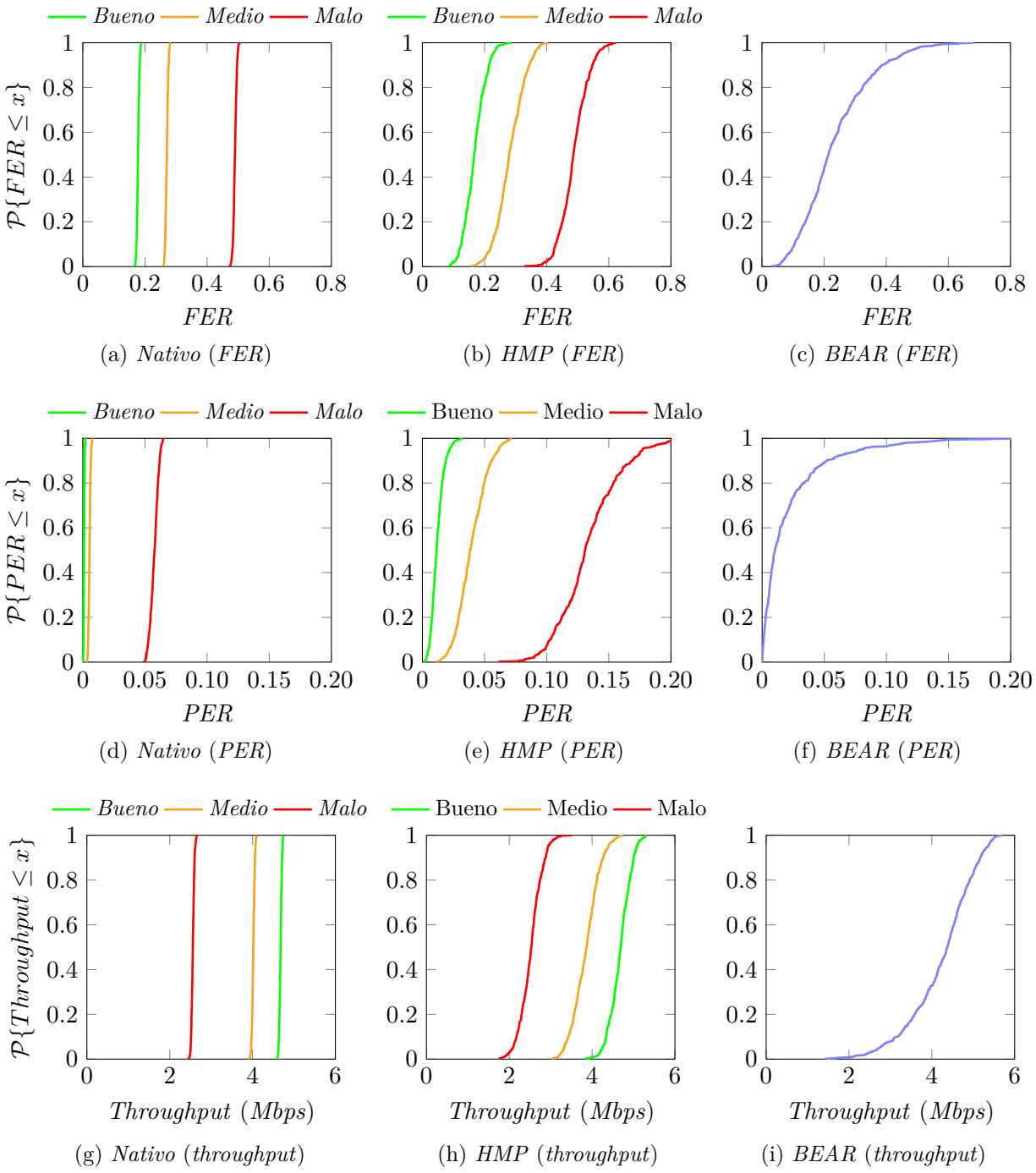
Para ello, se modificará la distancia entre los terminales hasta conseguir unos valores similares, en función de la tasa de error de trama.

El proceso a seguir se divide en cuatro etapas diferentes: en primer lugar, se caracterizará el comportamiento de las tres soluciones presentadas, utilizando *UDP* como nivel de transporte; posteriormente, se estudiará el impacto de este tipo de canales sobre el protocolo *TCP*. También se analizará la dependencia de cada uno de los modelos con la distancia entre los nodos, aspecto fundamental para facilitar el uso de estos canales en escenarios diferentes. Finalmente se compararán las prestaciones de los tres modelos en términos de su coste computacional, analizando su relación *complejidad-rendimiento*.

2.4.1. Caracterización del canal con tráfico *UDP*

Como ya se ha comentado anteriormente, a la hora de caracterizar el rendimiento neto de un canal inalámbrico *IEEE 802.11*, *UDP* aparece como el protocolo de nivel de transporte más apropiado. Su principal característica es que no introduce ningún mecanismo adicional que pudiese interferir en el comportamiento nativo de los niveles inferiores. De este modo, se consigue que la capacidad del canal inalámbrico y los protocolos de los niveles físico y de enlace sean el cuello de botella del sistema.

A modo de presentación de los principales resultados obtenidos en esta primera fase, la Figura 2.20 representa las *Función de Distribución (FD)* de un conjunto de métricas que caracterizan el rendimiento de este tipo de transmisiones: tasa de error de trama (*FER*), de paquete (*PER*) y *throughput*, obtenidas a partir de la ejecución de 500 realizaciones independientes. En primer lugar, se aprecia claramente que el modelo *Nativo* se comporta de manera muy previsible, estando sus resultados cercanos a un proceso determinista. Como podrá observarse a lo largo de toda esta sección, la respuesta de este modelo a todos los estadísticos estudiados es siempre similar, caracterizándose por una clara falta de aletoriedad. Por otra parte, a pesar de replicar de una forma bastante fiable tanto la *FER* como el *throughput* (comparando con los valores de la Tabla 2.6), no es capaz de proporcionar una tasa de error de paquete apropiada, quedándose siempre muy por debajo de lo observado en un canal real. Respecto al modelo *HMP*, se aprecia como sus diferentes configuraciones presentan un comportamiento similar en los tres casos (*bueno*, *medio* y *malo*), evidenciando una variabilidad mucho mayor que el canal *Nativo*. Todas las curvas presentan pendientes paralelas, a excepción de la correspondiente a la *PER* del canal *malo* (Figura 2.20e), la cual cubre un mayor rango de posibilidades. Si se comparan con los valores de la Tabla 2.6, se ve que en este caso el modelo *sí* es capaz de comportarse de una manera similar al canal real tomado como referencia. El tercer modelo, *BEAR*, abarca, con una única configuración, prácticamente todo el rango de resultados observados experimentalmente (Tabla 2.3), salvo los casos más extremos (medidas #1, #2, #14 y #15), mostrando un elevado grado de aleatoridad.



(a) *Nativo* (*FER*) (b) *HMP* (*FER*) (c) *BEAR* (*FER*)

(d) *Nativo* (*PER*) (e) *HMP* (*PER*) (f) *BEAR* (*PER*)

(g) *Nativo* (*throughput*) (h) *HMP* (*throughput*) (i) *BEAR* (*throughput*)

Figura 2.20: Función de distribución de la tasa de error de trama, de paquete y *throughput* para los diferentes modelos de canal

Otra forma de analizar las tasas de error es a partir de la relación entre la *FER* y la *PER*. Esta representación pone de manifiesto el efecto *memoria* del canal inalámbrico, que causa la aparición de ráfagas de tramas erróneas. Para paliar los efectos de estas ráfagas, existen diferentes mecanismos, como en el caso del estándar *IEEE 802.11* [IEEE802.11], que realiza un número de retransmisiones antes de dar por perdido el paquete, eliminándolo de la cola de salida. No obstante, si el número de errores consecutivos es mayor que el límite de reintentos, se producirá la pérdida definitiva del paquete, correspondiendo a los protocolos de niveles superiores el recuperarlo si fuese necesario. En el caso concreto de *UDP*, al tratarse de un protocolo *best-effort*, todo datagrama que los mecanismos de retransmisión a nivel *MAC IEEE 802.11* no sean capaces de recuperar se perderá definitivamente.

Para el caso particular que se está analizando, donde el escenario se encuentra libre de interferencias, pudiendo descartarse las colisiones con otras estaciones transmisoras, un canal sin memoria sería aquel en el que los procesos aleatorios asociados a las recepciones de tramas de manera consecutiva son totalmente independientes de los anteriores. Ante esta afirmación, la tasa de error de paquetes se define en función de la *FER* como se muestra en la Ecuación (2.23):

$$\begin{aligned}\mathcal{P}_{error}(\text{paquete}) &= \prod_{i=0}^{R+1} \mathcal{P}_{error}(\text{trama}_i) = \\ &= \{\mathcal{P}_{error}(\text{trama}_i) \equiv \mathcal{P}_{error}(\text{trama})\ \forall i \in [0...3]\} = [\mathcal{P}_{error}(\text{trama})]^{R+1}\end{aligned} \tag{2.23}$$

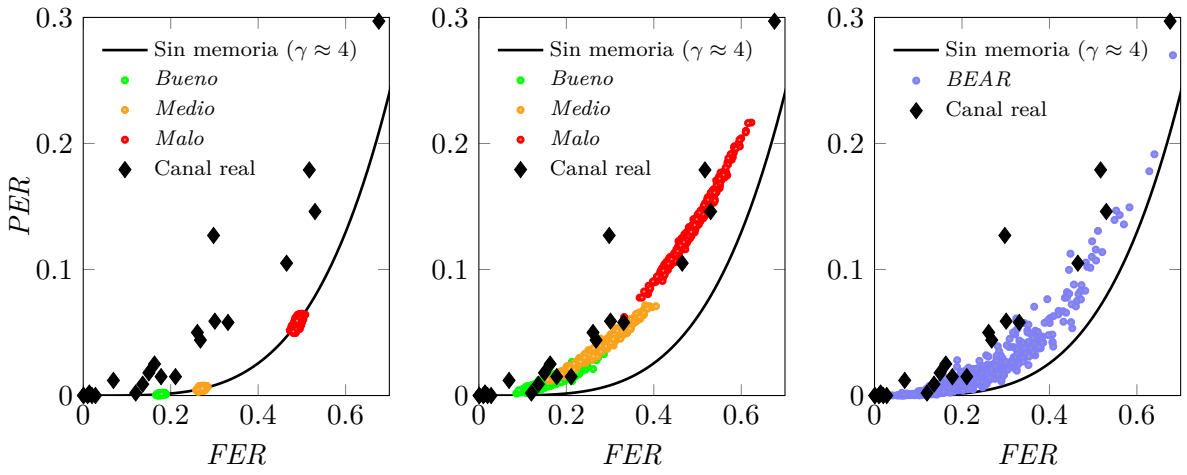
Donde un paquete se pierde tras la llegada de $R+1$ tramas erróneas consecutivas, siendo R el número de reintentos *IEEE 802.11* (3 en esta Tesis).

Para modelar un caso más general, donde el proceso de recepción de una trama *sí* que presenta un cierto nivel de correlación con los siguientes, la relación entre ambas tasas de error es la que se muestra en la Ecuación (2.24), donde γ representa el impacto de la memoria del canal (cuanto menor sea el valor de γ, mayor será el grado de memoria correspondiente).

$$\mathcal{P}_{error}(\text{paquete}) = \mathcal{P}_{error}(\text{trama})^{\gamma} \tag{2.24}$$

La Figura 2.21 muestra los valores observados para cada realización individual. Se representan los valores que se obtuvieron sobre el canal real, así como una curva que refleja el comportamiento de un canal sin memoria. Se pone nuevamente de manifiesto la alta previsibilidad del modelo *Nativo* (Figura 2.21a), cuyos valores se caracterizan por no mostrar ningún grado de memoria ($\gamma \approx 4$). Por otro lado, el modelo *HMP* (Figura 2.21b) ofrece, considerando las tres configuraciones de manera conjunta, un comportamiento bastante cercano al observado en el canal real, tanto en términos de memoria como en la variabilidad de resultados. Finalmente, *BEAR* presenta un conjunto de resultados más amplio,

cubriendo, con una simple configuración, el rango completo de valores obtenidos durante la caracterización empírica. Sin embargo, se puede ver que el valor de γ sería menor al correspondiente con el canal real, lo que podría compensarse utilizando una mayor potencia de ruido blanco en la entrada del filtro AR [Agüe10].



(a) *Nativo* (b) *HMP* (c) *BEAR* ($P_\epsilon = 5mW/Hz$)

Figura 2.21: Relación entre la tasa de error de trama, de paquete y del *throughput* para los modelos de canal estudiados

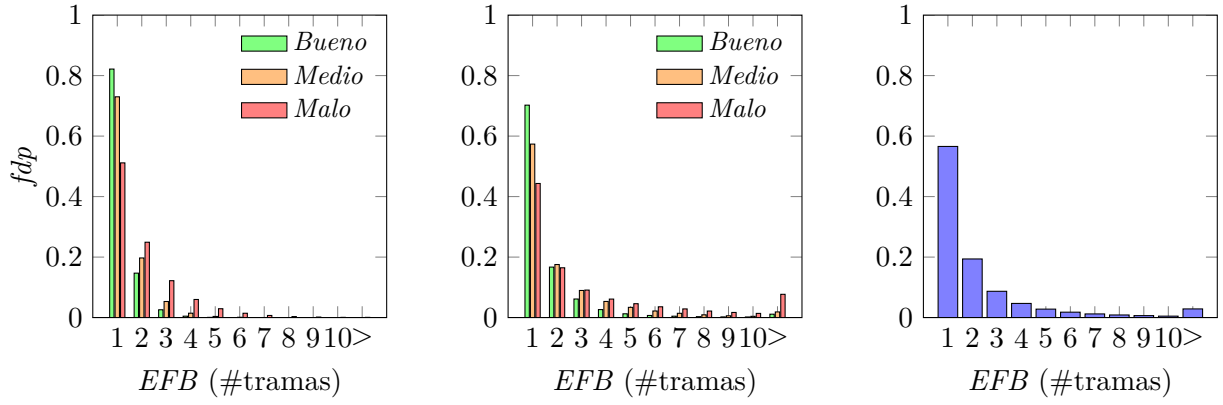
La Tabla 2.9 resume el comportamiento global de todas las configuraciones estudiadas, tanto en términos de valores medios como de varianzas. El primer aspecto a tener en cuenta es que ninguno de los modelos cubre la gran variabilidad de las medidas reales; en este sentido, hay que tener en cuenta que hay una medida en particular (#9) que se encuentra muy alejada de la tendencia seguida por las demás, penalizando notablemente el valor medio y la varianza.

Tabla 2.9: Promediado estadístico del efecto memoria del canal inalámbrico (modelado a través del parámetro γ)

Modelo	$\overline{\gamma}$	**Var** (γ)
Real	2.4467	0.3313
Nativo bueno	4.0371	0.0383
Nativo medio	4.0124	0.0090
Nativo malo	4.0023	0.0014
HMP bueno	2.5126	0.0125
HMP medio	2.5757	0.0101
HMP malo	2.8269	0.0119
BEAR	2.9782	0.1001

La Figura 2.22 muestra la función densidad de probabilidad de las longitudes de las ráfagas de errores para los tres modelos de canal, teniendo en cuenta todas las que se

produjeron a lo largo de las 500 realizaciones por configuración. Nuevamente, el modelo *Nativo* es el que tiene los resultados más alejados de la realidad, ya que prácticamente no existen ráfagas superiores a cuatro tramas erróneas; es importante destacar este valor concreto porque a partir del mismo comienzan a producirse pérdidas *IP*. Esto implica que, para un canal sin memoria, se necesitarán valores elevados de *FER* para que haya descartes a nivel *IP*.



(a) *Nativo* (b) *HMP* (c) *BEAR*

Figura 2.22: Función densidad de probabilidad de la longitud de las ráfagas de errores para los tres modelos de canal analizados

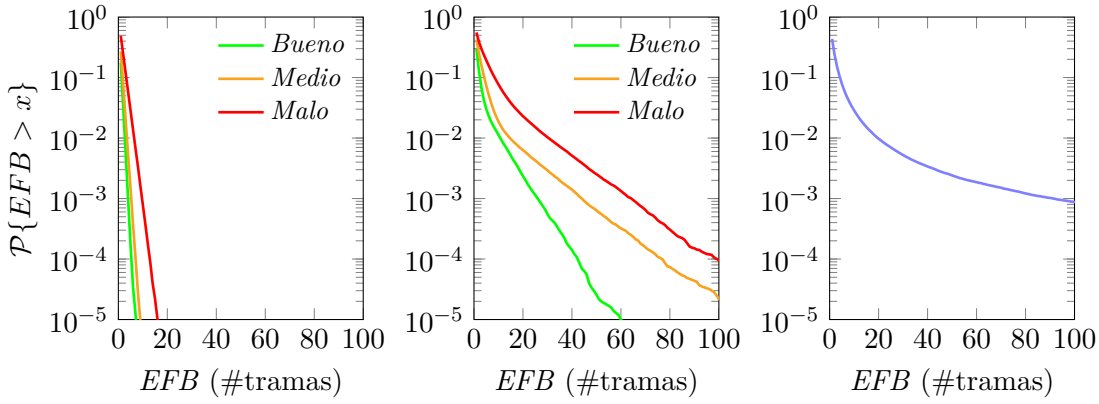
Teniendo en cuenta la *carencia de memoria*, se puede calcular la longitud media de las ráfagas como se muestra en la Ecuación (2.25).

$$\overline{EFB} = \sum_{i=1}^{\infty} i \cdot \mathcal{P}_{error}(\text{trama})^i \cdot (1 - \mathcal{P}_{error}(\text{trama})) = \frac{1}{1 - \mathcal{P}_{error}(\text{trama})} \tag{2.25}$$

En el modelo *HMP*, en cambio, se observa que predominan los errores en ráfaga, disminuyendo el porcentaje de errores aislados (ráfagas unitarias). Otra característica a tener en cuenta es la probabilidad no despreciable de ráfagas mayores a 10 tramas en cualquiera de las configuraciones, llegando a alcanzar valores bastante elevados en el canal *malo* (en torno a un 8 %). Finalmente, *BEAR* evidencia una gran variabilidad, generando un amplio rango de longitudes de ráfaga, primando las cortas (< 4) sobre el resto, con aproximadamente un 85 % del total. Además, la probabilidad de ráfagas largas (> 10) se sitúa alrededor del 5 %.

Por otra parte, la aparición de una ráfaga de más de 100 tramas erróneas es un fenómeno bastante improbable en un entorno real, produciéndose en menos de un 0.7 % de las veces [Agüe10]. Para visualizar este aspecto se representa la *función de distribución complementaria* de las ráfagas de errores en la Figura 2.23. Puede afirmarse que tanto el

modelo *HMP* como *BEAR* reflejan esta característica de una manera bastante fidedigna; a pesar de que *sí* se aprecia la presencia de *EFBs* mayores a 100 tramas, su probabilidad es lo bastante baja para mantenerse por debajo del valor observado en un canal real, cumpliéndose siempre que: $\mathcal{P}\{EFB > 100\} \leq 10^{-3}$. En el otro extremo se encuentra el modelo *Nativo*, ya que las ráfagas son demasiado cortas, no llegando a alcanzar ni las 30 tramas en el peor de los casos (canal *Malo*).



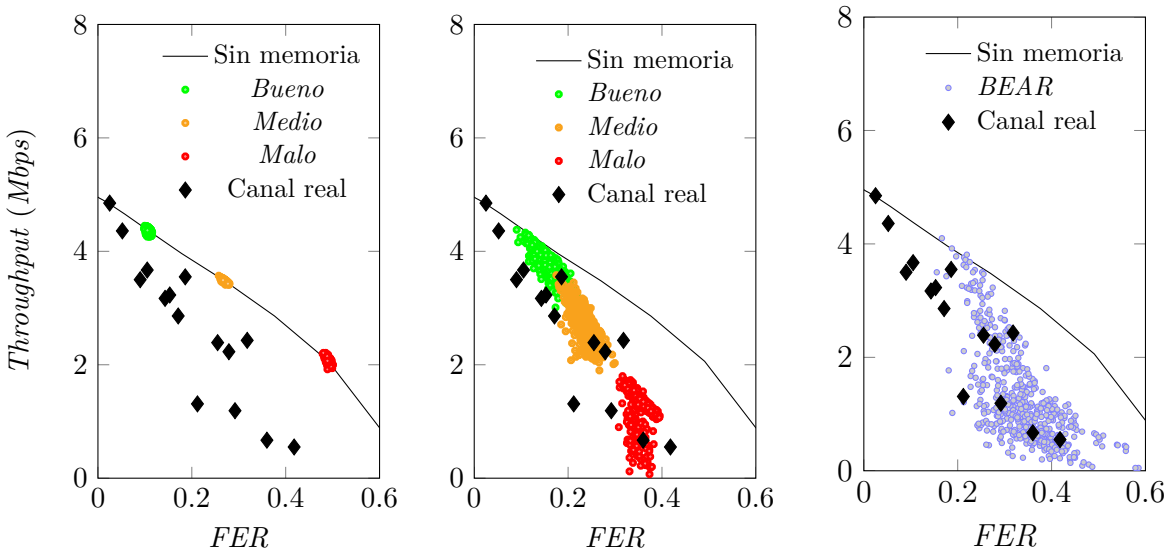
(a) *Nativo* (b) *HMP* (c) *BEAR*

Figura 2.23: Función de distribución complementaria de la longitud de las ráfagas de errores para los tres modelos de canal analizados

2.4.2. Impacto de los errores sobre el rendimiento de *TCP*

Una vez estudiado el comportamiento de los modelos de canal, se procede ahora al análisis de la relación de *TCP* sobre ellos, debido a que el principal objetivo de la presente Tesis radica en la mejora del pobre rendimiento ofrecido por este protocolo cuando se utiliza sobre redes inalámbricas.

Como ya se ha comentado anteriormente, *TCP*, protocolo orientado a la conexión por excelencia, presenta una serie de carencias cuando se usa sobre canales inalámbricos, debido principalmente a que *no es capaz* de diferenciar la causa de la pérdida de un segmento: congestión en los *routers* a lo largo de la red o, por el contrario, las condiciones del propio enlace inalámbrico, que deteriora la señal hasta el punto que el receptor no es capaz de reconstruir el mensaje original. Cuando la entidad *TCP* transmisora detecta la pérdida de un segmento, lo interpretará como consecuencia de una situación de congestión, por lo que los mecanismos de control de flujo actuarán en consecuencia, reduciendo el tamaño de la ventana de congestión (*Congestion Window (CWND)*). Si la pérdida se debiera a las condiciones del canal, esta reacción disminuirá el rendimiento de la transmisión, ya que se estará produciendo, de manera innecesaria, la *infrautilización* del canal inalámbrico.



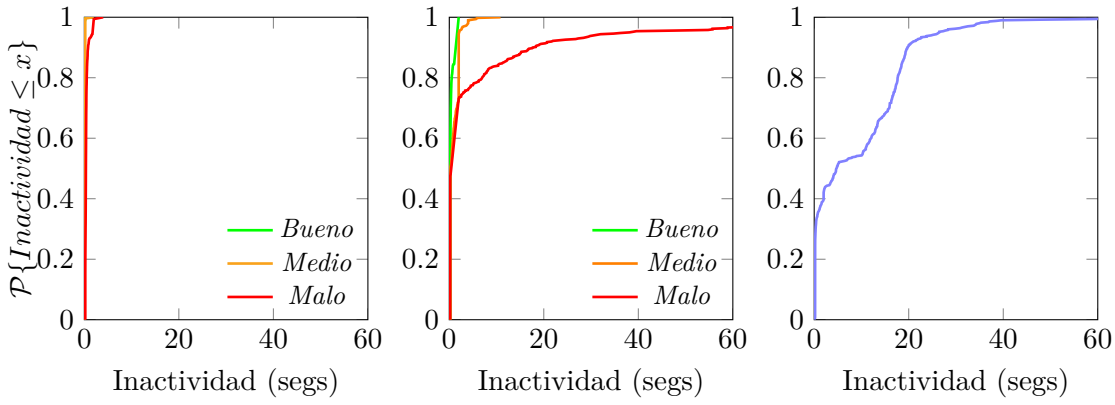
(a) *Nativo* (b) *HMP* (c) *BEAR* ($P_\epsilon = 5\ mW/Hz$)

Figura 2.24: Relación entre la tasa de error de trama y el *throughput TCP* para los diferentes modelos de canal analizados

Como se ha visto anteriormente, hay dos posibles eventos que dan lugar a una retransmisión: *(1)* Tras la recepción de un *triple ACK duplicado* (cuatro recepciones fuera de orden), el algoritmo *Fast Retransmit* [Allm09] establece que el segmento correspondiente a dicho número de secuencia se retransmita con carácter inmediato. *(2)* Tras la expiración del temporizador *RTO*, el nodo procederá a la retransmisión del primer segmento sin confirmar. A medida que retransmite el mismo segmento por la expiración de este tiempo de guarda de manera continuada, éste irá creciendo gradualmente (siguiendo un proceso de *backoff* binario), pudiendo llegar a alcanzar valores mayores de 100 segundos.

La presencia de largas ráfagas de tramas erróneas (fenómeno característico de un canal inalámbrico real) durante una sesión *TCP* tiene por tanto un enorme impacto en el rendimiento de la transmisión. Por este motivo, se hace indispensable que los modelos de canal utilizados sean capaces de replicar este comportamiento de la forma más precisa posible.

La Figura 2.24 refleja la relación entre la *FER* y el caudal eficaz, representando los resultados de cada una de las realizaciones. También se añaden los puntos correspondientes a lo observado en un escenario real, así como el comportamiento al emplear un modelo de canal sin memoria. A simple vista vuelven a verse las pobres prestaciones del modelo *Nativo*, con un comportamiento que se corresponde con el de un canal "sin memoria". En segundo lugar, *HMP* presenta unos niveles aceptables de variabilidad, emulando razonablemente las medidas reales. Por último, *BEAR* vuelve a ofrecer el rango más amplio de posibles salidas, reflejando en este caso con bastante fidelidad lo observado en el canal real.



(a) *Nativo* (b) *HMP* (c) *BEAR*

Figura 2.25: Función de distribución del tiempo de inactividad para los modelos de canal analizados

La Figura 2.25 representa la *FD* del tiempo máximo de inactividad experimentado en el nivel *TCP* del nodo transmisor. Como consecuencia directa de la falta de memoria del modelo *Nativo*, la longitud de las ráfagas de errores no son lo suficientemente largas como para producir unas pérdida significativa de segmentos de manera consecutiva. Por lo tanto, se observa que los valores de inactividad máximos se sitúan siempre por debajo de los 2 segundos (incluso para la configuración *mala*, que presentaba una tasa de error cercana al 50 %). Por otra parte, el modelo *HMP* es capaz de reflejar, para cada una de las condiciones bajo las que fue entrenado, resultados similares a los observados durante la fase de caracterización experimental, aunque en el canal *Malo* aparezcan, en menor proporción (menos de un 2 % de las veces), situaciones de inactividad de más de 100 segundos, circunstancia no observada en la Tabla 2.4 (página 2.4). En referencia a *BEAR*, puede apreciarse un amplio rango de valores, desde intervalos de inactividad cortos en canales de calidad aceptable, a situaciones realmente hostiles, en los que la fuente puede llegar a mantener hasta 60 segundos de inactividad para reiniciar el proceso de transmisión.

Junto con todas las figuras presentadas hasta el momento, resulta especialmente interesante analizar el efecto de los errores aleatorios sobre algunas de las simulaciones en particular. Para poder interpretar correctamente los resultados, las gráficas representan la evolución de tres métricas a lo largo del tiempo de transmisión, descritas a continuación:

1. En primer lugar, la evolución del número de secuencia en función del tiempo da una idea de la calidad de la transmisión, pudiendo relacionarla con el *throughput* instantáneo.

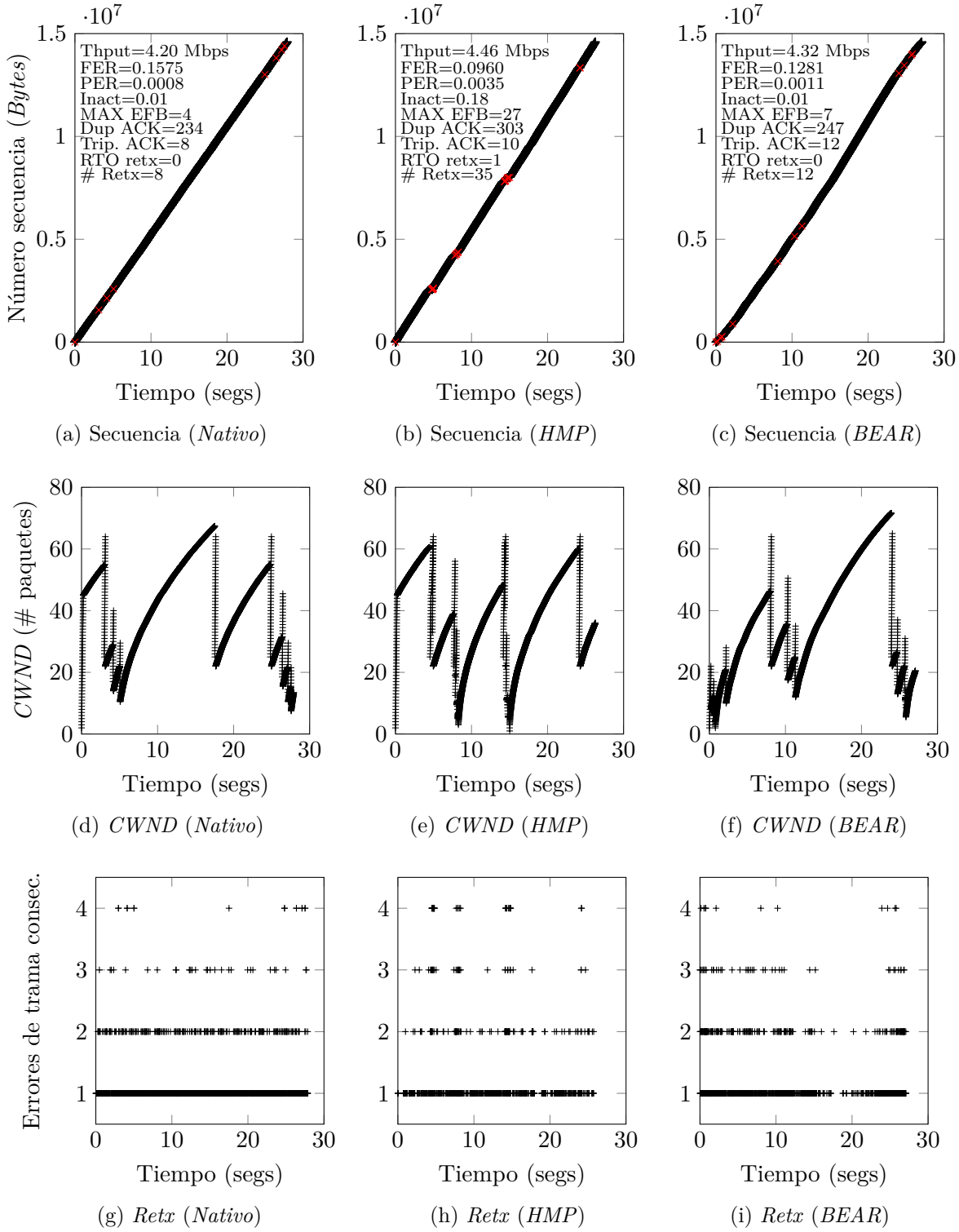
2. Como un elemento intermedio, y estrechamente ligado a la métrica anterior, el tamaño de la ventana de congestión permite percibir los efectos de las condiciones cambiantes del canal inalámbrico. Su tendencia será creciente en ausencia de pérdida

de segmentos, mientras que la aparición de largas ráfagas de errores tendrá como consecuencia la disminución del tamaño de la ventana según imponen los algoritmos de control de congestión del protocolo *TCP*.

3. Finalmente, aunque se trate de una métrica más vinculada al nivel del enlace, se analizarán las ráfagas de errores observadas a nivel de segmento. Tomando como referencia el número de retransmisiones utilizado por el nivel *MAC* (recuérdese que mantiene un valor de 4 intentos en todo momento), las figuras sirven para inferir la capacidad de estos mecanismos para recuperarse de los errores. En concreto, se representará el número de tramas consecutivas que son recibidas de manera errónea. En aquellos casos en los que la ráfaga sea igual a cuatro tramas, será el propio *TCP* el que se encargue de recuperar dicho segmento.

Las Figuras 2.26 y 2.27 se basan, de entre las 500 simulaciones realizadas, en aquellas correspondientes al mejor y peor de los casos (en términos de *throughput*) para cada uno de los modelos. En relación a los resultados más optimistas (Figura 2.26), no se aprecia una gran diferencia entre las tres alternativas. El elemento más destacable es la ausencia de ráfagas largas de errores para los modelos *Nativo* y *BEAR* (4 y 7 tramas, respectivamente), mientras que para el caso de *HMP* este valor se encuentra claramente por encima (se perdieron hasta 27 tramas de manera consecutiva). Lo curioso es que, a pesar de que el número de retransmisiones sigue el orden $Nativo > BEAR > HMP$, el throughput presenta el orden inverso ($HMP > BEAR > Nativo$). La explicación radica en que en el modelo *Nativo* la *FER* es la más elevada de las tres, aunque los errores se producen de una manera más aislada, presentando ráfagas de longitudes medias inferiores a la capacidad de recuperaración del nivel *MAC*. Bajo estas características, estas retransmisiones serán capaces de recuperar prácticamente todos los segmentos, aunque, como contrapartida, causarán una disminución en el rendimiento final de la transmisión.

En el otro extremo (Figura 2.27) se encuentran las realizaciones que obtuvieron un rendimiento más pobre, marcadas por la hostilidad del canal inalámbrico. En este caso, el modelo *Nativo* presenta un valor de *FER* mucho más alto que el de las otras alternativas, llegando prácticamente a duplicar al obtenido con *BEAR*. No obstante, la nula correlación entre recepciones consecutivas causa que la longitud de las ráfagas no sea lo suficientemente larga como para perder varios segmentos de manera consecutiva. Por lo tanto, y a pesar de que el número total de retransmisiones sea ostensiblemente mayor que en el resto de los casos (casi cuatro veces más que las producidas en *BEAR*, por ejemplo), el tiempo máximo de inactividad es mucho menor que el observado en los otros dos modelos, en los que la aparición de largas ráfagas de errores conduce a la expiración de múltiples temporizadores *RTO* consecutivos, llevando al nodo fuente a mantenerse inactivo durante decenas de segundos, lo que dará lugar a una gran degradación del rendimiento.

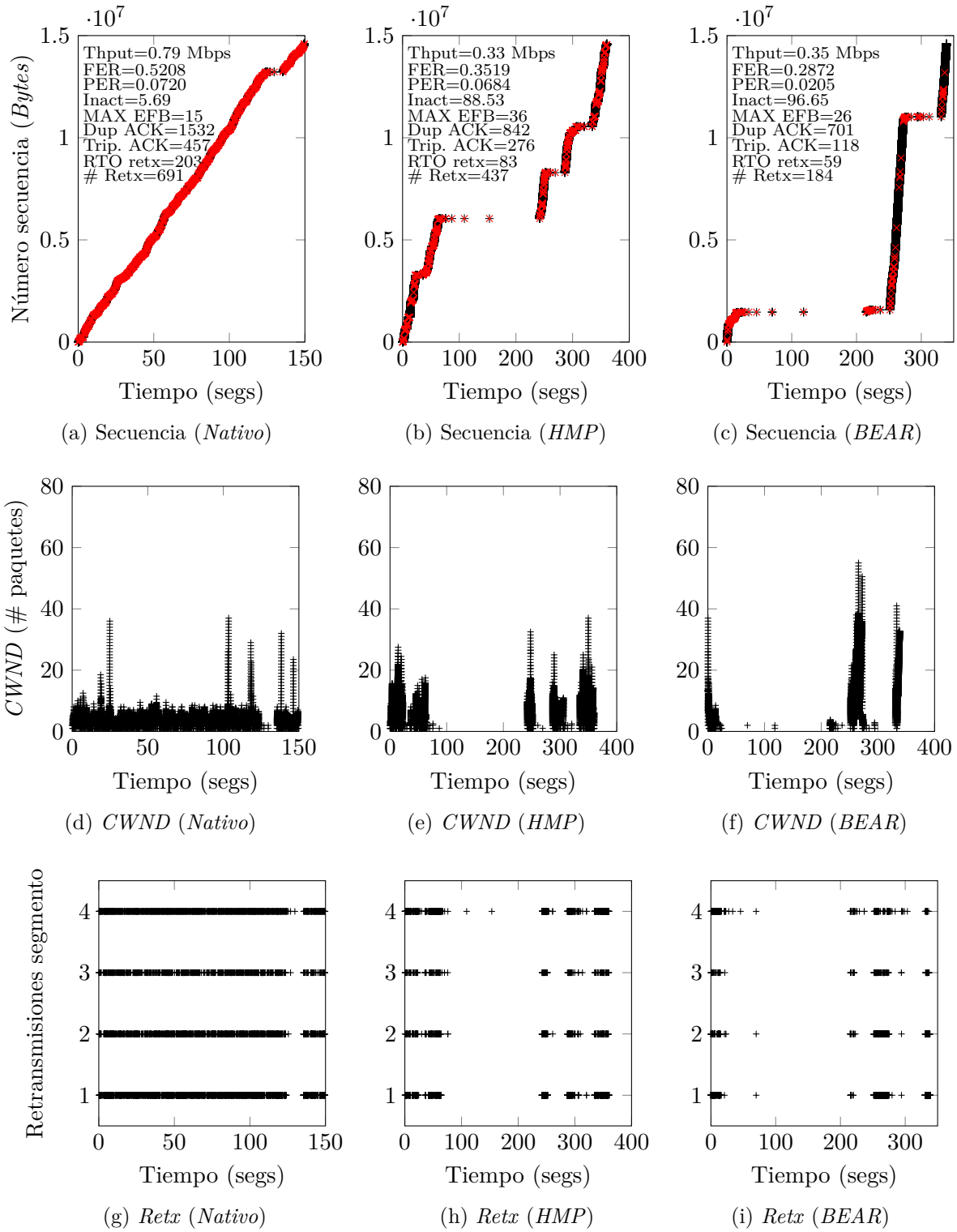


(a) Secuencia (*Nativo*) (b) Secuencia (*HMP*) (c) Secuencia (*BEAR*)

(d) *CWND* (*Nativo*) (e) *CWND* (*HMP*) (f) *CWND* (*BEAR*)

(g) *Retx* (*Nativo*) (h) *Retx* (*HMP*) (i) *Retx* (*BEAR*)

Figura 2.26: Caracterización de las medidas *TCP* con mejor comportamiento

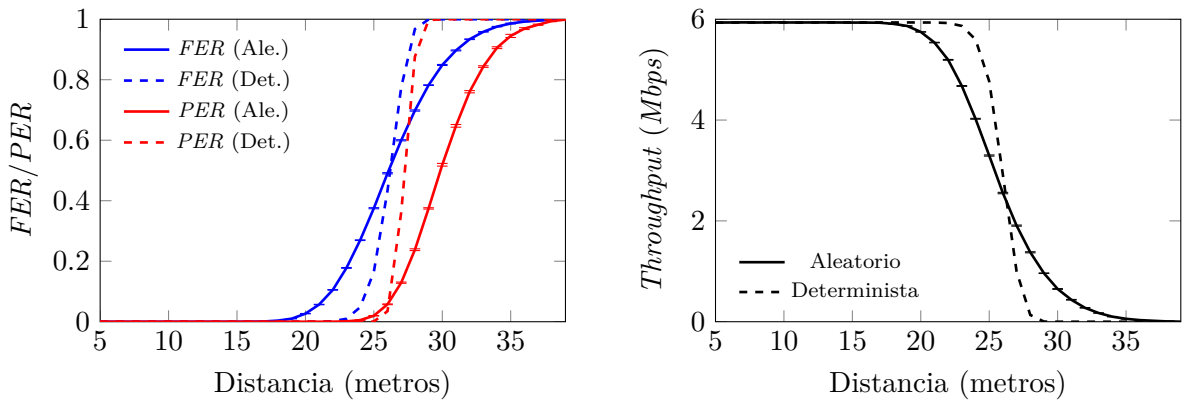


(a) Secuencia (*Nativo*) (b) Secuencia (*HMP*) (c) Secuencia (*BEAR*)

(d) *CWND* (*Nativo*) (e) *CWND* (*HMP*) (f) *CWND* (*BEAR*)

(g) *Retx* (*Nativo*) (h) *Retx* (*HMP*) (i) *Retx* (*BEAR*)

Figura 2.27: Caracterización de las medidas *TCP* con peor comportamiento



(a) *FER/PER Vs.* distancia

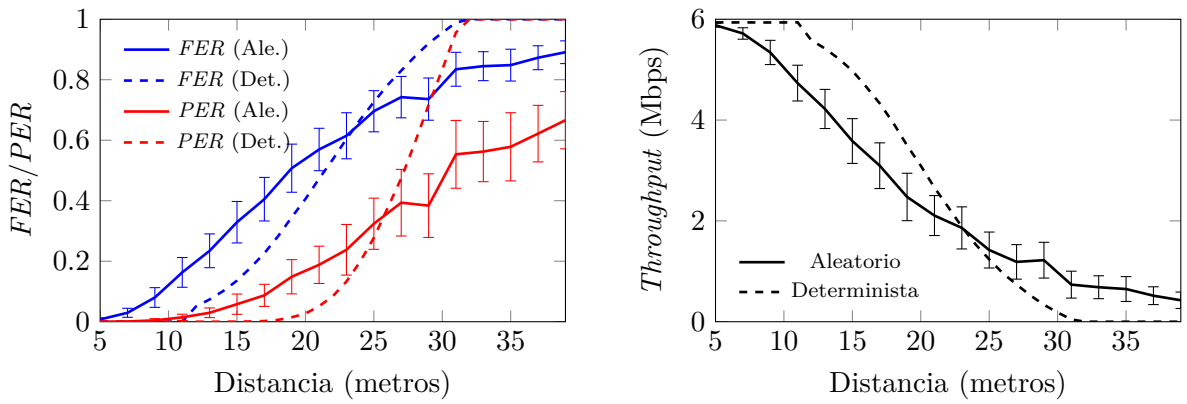
(b) *Throughput Vs.* distancia

Figura 2.28: Relación de la calidad del canal y la distancia entre nodos para el modelo *Nativo*

2.4.3. Calidad del canal en función de la distancia

Otro aspecto que parece interesante es que los modelos de canal reflejen una correcta degradación de la potencia en función de la separación de los nodos. Para ello se analizará la relación de diferentes métricas, como la tasa de error de trama (*FER*), de paquete (*PER*) y el *throughput*, con la distancia entre los terminales. Se mantienen las condiciones del escenario descritas en las campañas de medidas anteriores, utilizando *UDP* como nivel de transporte, variando la distancia entre los nodos de 5 a 39 metros. Además, cada uno de los modelos se configurará a través de dos formas: la primera de ellas (*Determinista*) deshabilita cualquier contribución aleatoria en la propagación, obteniéndose siempre un valor de *SNR* fijo. De este modo, será el modelo de error el responsable de elegir si una trama es correcta o no. En segundo lugar, se ilustrará el comportamiento por defecto de los modelos, con todas las contribuciones *aleatorias*. Cada experimento se ha repetido 50 veces y se ha representado el intervalo de confianza del 95 % para dar muestra de la variabilidad de los resultados.

La Figura 2.28 muestra la respuesta del modelo *Nativo* al incremento paulatino de la distancia entre los terminales. Se hace patente la escasa aleatoriedad en los resultados obtenidos, con un intervalo de confianza prácticamente nulo. Se observa además que el canal pasa de un comportamiento bueno ($FER \approx 0$) a uno hostil ($FER \approx 1$) en una distancia muy corta (entre 19 y 34 metros). Este rango disminuye al eliminar las componentes aleatorias de la señal, pasando a ser de aproximadamente 7 metros. Además, resulta de especial interés analizar la relación entre las tasas de error de trama y de paquete, representadas de manera conjunta en la Figura 2.28a. En concreto, se observa que los paquetes empiezan a perderse a partir de un valor aproximado de $FER \approx 0.4$, poniendo nuevamente de manifiesto el comportamiento *sin memoria* que caracteriza a este modelo de canal (se necesita un alto porcentaje de pérdida de tramas para que las ráfagas de errores sean lo



(a) *FER/PER Vs.* distancia

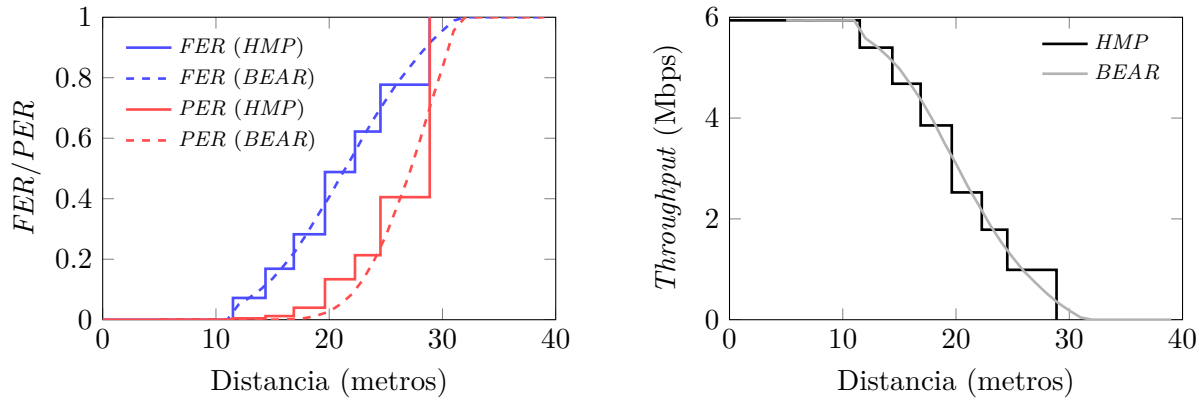
(b) *Throughput Vs.* distancia

Figura 2.29: Relación de la calidad del canal y la distancia entre nodos para el modelo *BEAR*

suficientemente largas como para superar la capacidad de recuperación del esquema de retransmisión del nivel *MAC*). Con respecto a los valores de *throughput* observados (Figura 2.28b, se aprecia claramente la correlación con las métricas anteriores, mostrando una tendencia inversa a la de la tasa de error de trama.

El comportamiento de *BEAR* se recoge en la Figura2.29. En este caso se aprecia claramente el alto componente aleatorio de este modelo, a la vista del intervalo de confianza. Se ve que pueden producirse errores de trama en todas las distancias mostradas en el eje de abscisas (con las componentes aleatorias habilitadas). Incluso al utilizar únicamente la etapa de atenuación determinista para calcular la potencia recibida, la función logística definida en el modelo de error (ver Ecuación 2.22) permite que el citado rango de distancias sea mucho mayor que el obtenido con el modelo *Nativo*. Por último, merece la pena destacar el factor de memoria que se extrapola de la relación entre las dos tasas de error, viendo como los paquetes empiezan a perderse incluso con valores bajos de *FER* (aproximadamente 0.1).

Al contrario de todos los modelos de canal estándar, una de las mayores carencias que pueden ser achacadas a uno basado en procesos de Markov ocultos (*HMP*) es su falta de capacidad para poder determinar, de manera dinámica, la calidad de la señal recibida en función de la distancia entre los nodos (por defecto, no hay una relación directa entre dicha separación y la probabilidad de que las tramas lleguen erróneas a su destino). El motivo es que la elección de la dupla de matrices que define el proceso oculto de Markov (transición y emisión) son pre-cargadas en el escenario antes de ejecutar las simulaciones, manteniendo sus valores constantes durante todo el tiempo. Para superar esta limitación se utilizarán, a modo de referencia, la curva de tasa de error de trama con el modelo *BEAR* determinista y las medidas obtenidas sobre el escenario real con *UDP*. Debido a que el número de capturas obtenidas en la campaña experimental tiene un valor finito, la respuesta presentará una forma escalonada. Se han elegido siete medidas ilustrativas (i.e. #1, #2, #5, #7, #9, #12



(a) Función de distribución de las tasas de error de trama y de paquete

(b) Función de distribución del *throughput*

Figura 2.30: Comportamiento del modelo *HMP* en función de la distancia entre nodos

y #13) para efectuar el ajuste de las instancias del modelo, minimizando el error cuadrático medio entre la función base y la buscada. La Figura 2.30 resume sus principales métricas de rendimiento (*FER*, *PER* y *throughput*) a medida que se aumenta la separación entre el transmisor y el receptor. Indirectamente, la estrecha dependencia entre este parámetro y el *throughput* hará que este comportamiento se vea reflejado en éste (Figura 2.30b). Sin embargo, en lo que respecta a la tasa de error de paquete, se observa un problema de ajuste. Hay que tener en cuenta, como se estudió en la Sección 2.4.1, que el modelo *HMP* presenta un factor de memoria (γ) más elevado que *BEAR*, dando lugar a valores de *PER* superiores, sobre todo en la región donde la *FER* es baja ($FER \leq 0.4$); por otra parte, la escasez de realizaciones con una calidad de canal hostil (e.g. $FER \geq 0.7$) hace que la solución propuesta no sea capaz de presentar configuraciones con altos valores de tasa de error de paquete ($PER \geq 0.4$).

2.4.4. Análisis del coste computacional

Como último paso para concluir con la comparación entre los tres modelos de canal estudiados en el marco de esta Tesis, se estudiará el coste operacional de cada uno de ellos. Con esto se pretende analizar la relación entre la capacidad de replicar de la manera más precisa el comportamiento real de un canal inalámbrico y el complejidad computacional que ello conlleva. Para esta nueva tanda de simulaciones se ha realizado una serie de cambios en relación a las configuraciones previas, tal y como se describen a continuación:

1. La modificación más sustancial con respecto a las etapas de simulación anteriores es que el número de nodos (N) presentes en el escenario oscila entre 2 (enlace directo transmisor-receptor) y 32 (donde cada paquete deberá ser reenviado por 30 nodos intermedios antes de llegar a su destino). Como resultado, se creará una red

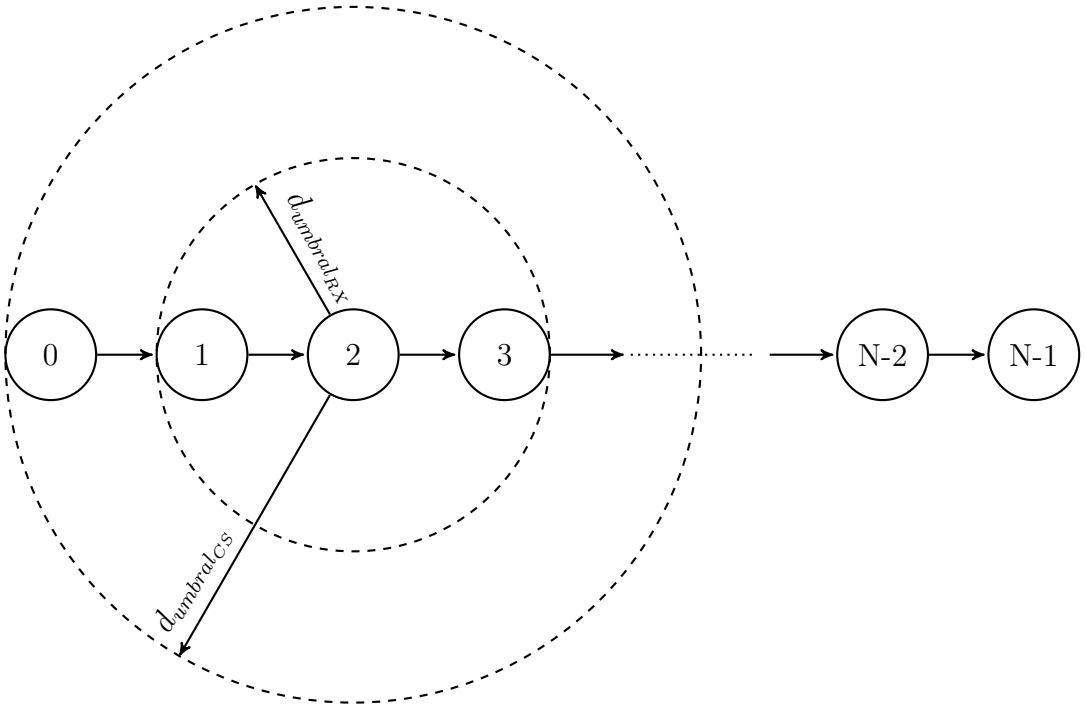


Figura 2.31: Topología empleada para el análisis del coste computacional de los modelos de canal

inalámbrica multi-salto en una única dimensión, donde el incremento en el número de nodos tendrá asociado un crecimiento en el tiempo requerido para llevar a cabo las simulaciones.

2. Todas las nuevas configuraciones presentan un denominador común: junto con la operación estándar de los tres modelos de canal, se ha añadido una nueva etapa previa al cálculo de las pérdidas producidas por la propagación, a través de un esquema basado en "rangos". Así, se definen dos umbrales (distancias) diferentes: el primero de ellos (d_1) limita el radio dentro del cual toda trama recibida mantendrá una atenuación de 0 dB; de este modo, dependerá de la siguiente etapa el valor final de la potencia recibida P_{RX}. En segundo lugar, se delimita la región en la que los nodos son capaces de detectar que se ha producido una transmisión, pero la energía percibida no es lo suficientemente elevada como para recuperar con éxito la información transmitida. En la literatura, éste se denomina *umbral de detección de portadora* (*Carrier Sense threshold - CS threshold*), teniendo una fuerte dependencia con la modulación empleada. En concreto, el valor de la potencia recibida en esta zona podrá ser definida como $P_{umbral_{RX}} < P_{RX} < P_{umbral_{CS}}$. Fuera de las mencionadas regiones, cualquier trama percibida por los nodos será automáticamente descartada, puesto que el valor de la potencia en recepción P_{RX} será siempre inferior al umbral mínimo de detección de portadora del dispositivo receptor, tal que $P_{RX} < P_{umbral_{CS}}$. De esta manera se asegura que los paquetes sean reenviados por los $N-2$ nodos intermedios desplega-

dos en la topología en línea antes de alcanzar el destino. Además, con esto se evita la aparición del efecto conocido como *teminal oculto*, ya que el nodo *i-ésimo* detecta las transmisiones efectuadas por sus $i \pm 2$ entidades vecinas, evitando transmisiones simultáneas y las consecuentes colisiones. De este modo, no es precisa la utilización de la técnica *RTS/CTS*. La Figura 2.31 muestra la topología genérica que va a ser utilizada en esta etapa, representando los dos umbrales mencionados, que delimitan las diferentes zonas de operación.

3. Durante las simulaciones, el nodo origen (el primero de la cadena) envía 5000 datagramas *UDP* hacia el nodo destino (el último), utilizando para ello un aplicación que envía tráfico hacia los niveles inferiores a una tasa binaria constante de 100 *Kbps*. De este modo, el canal inalámbrico no se encontrará en condiciones de saturación.

4. Los parámetros restantes mantienen los valores de las configuraciones previas.

5. Por último, se han realizado 25 simulaciones independientes para cada una de las configuraciones.

Para comprender mejor los resultados, se pueden destacar cuatro factores como aquellos que marcan el coste computacional de los modelos:

- El número de nodos desplegados en el escenario. Es el elemento clave que marcará el tiempo de simulación requerido para cada uno de los modelos. Analizando el código fuente del simulador, tras la transmisión de una trama, se calculará, mediante un bucle, la potencia recibida en todos los interfaces físicos asociados al mismo canal inalámbrico (excepto el correspondiente al nodo transmisor); es decir, en la configuración utilizada, cada transmisión de una trama será procesada por los $N - 1$ nodos restantes.

- El cálculo de la atenuación producida por la transmisión a través del espectro radioeléctrico. Mientras que tanto el modelo *Nativo* como *BEAR* realizan cálculos a partir de la combinación de las diferentes contribuciones que conforman la señal, el módulo *HMP* no tiene una etapa de propagación propia. En este caso en particular, su comportamiento se basa en las transiciones entre estados de la cadena de Markov que lo define, utilizando la búsqueda indexada como técnica para poder acceder a los valores de la matriz de emisión adecuados.

- La decisión final de si una trama es correctamente recibida en el receptor o no, presenta una operación diferente en cada uno de los modelos estudiados, aunque su impacto en el rendimiento es bastante inferior a los dos puntos anteriores. Mientras que el modelo *Nativo* recurre a las curvas *BER* para obtener la tasa de error de trama en función de su longitud y la modulación empleada, *BEAR* recurre a la utilización de una función logística. Por su parte, el modelo *HMP* basará su decisión en la búsqueda del valor de salida $b_i(0)$ en función del enlace utilizado y del estado actual de la cadena de Markov que lo define, i.

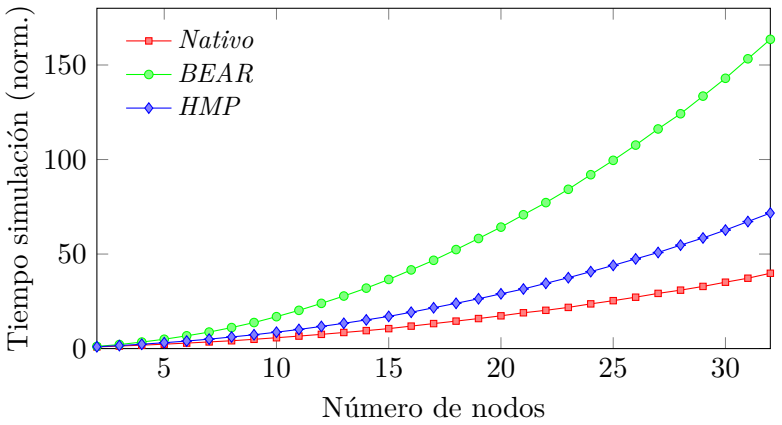


Figura 2.32: Comparación del coste computacional en función del número de nodos

- El resto de parámetros, entre los que se incluye el número de paquetes, la tasa binaria del nivel de aplicación, el encaminamiento, la operación de los protocolos de niveles superiores o la generación del escenario, entre otros, afectan exactamente del mismo modo a los tres modelos de canal, por lo que no tendrán influencia en los resultados mostrados[20], formando parte del proceso general de la simulación.

La Figura 2.32 representa el tiempo de simulación promedio normalizado (con respecto al menor de los resultados obtenidos, que se corresponde con el escenario con el canal *Nativo* y $N = 2$), así como el intervalo de confianza del 95 % (aunque la -prácticamente nula- variabilidad de los resultados hace que sea difícilmente apreciable), como función del número de nodos (N) deplegados a lo largo de la topología descrita anteriormente (ver Figura 2.31). De acuerdo con los resultados obtenidos, puede apreciarse claramente como el modelo *Nativo* es el más ligero de los tres, de manera más apreciable a medida que se incrementa el valor de N. Su simplicidad radica en el hecho de que su operación consiste, básicamente, en la utilización de tres operaciones básicas: en primer lugar, la potencia recibida será calculada a través de un modelo de propagación[21] en el espacio libre, cuyo resultado se obtiene a través de la Ecuación (2.6). Posteriormente, el modelo de interferencia obtiene el nivel de ruido en función de la modulación empleada, la duración de la transmisión y la presencia de transmisiones co-canal[22]. Por último, el modelo de error se basa en el cálculo de la probabilidad de error de trama en función de la relación señal a

[20]Debido a que la representación de los tiempos de simulación se encuentran normalizados respecto al valor más pequeño.

[21]El modelo basado en rangos de distancia, añadido exclusivamente para este apartado, no tendrá relevancia alguna en los resultados, ya que afecta a las tres alternativas analizadas.

[22]Es importante destacar en este punto que, en el momento de escribir esta Tesis, los módulos *Wifi* del simulador no son capaces de: *(1)* modelar el efecto captura de una transmisión, donde si la energía perteneciente a un paquete es muy superior a la del resto (en caso de una colisión), puede darse el caso de ser recibido correctamente; *(2)* calcular la interferencia producida por canales adyacentes o por técnologías que comparten la misma banda de *2.4 GHz*, como por ejemplo, la reciente inclusión del protocolo *IEEE 802.15.4*.

ruido, la longitud de la misma y la probabilidad de error de *bit* (cuya expresión puede encontrarse en [Laca06; Pei09a]), como se definió en la Ecuación (2.15).

A partir de este punto, la decisión de si una trama es correcta o no se llevará a cabo con la elección de un valor aleatorio que será comparado con el obtenido a través de todas las etapas anteriores. Este último paso será común para todos los modelos.

Con un coste intermedio se encuentra el rendimiento ofrecido por el módulo *HMP*, cuya complejidad, aunque mayor que la del modelo *Nativo*, presenta una tendencia mejor que la mostrada por *BEAR*, manteniendo una curva de crecimiento bastante aceptable. En este caso, se mantienen algunas de las operaciones comunes (e.g. definición de la potencia en función del rango de distancias, cálculo de interferencias, etc.), pero el peso del modelo recae sobre la búsqueda de los coeficientes apropiados (en función del estado actual en la cadena de Markov) en las matrices de transición (A) y emisión (B). No obstante, cada uno de los enlaces presentes en el escenario ($l_{i,j}$, siendo i el índice del nodo que ha transmitido la trama y j el que la recibe) mantiene una tabla que contiene ambas matrices, por lo que cada vez que se produzca un evento de recepción deberá realizarse una doble búsqueda antes de poder acceder al coeficiente $b_i(0)$ con el que se decidirá si la trama es correcta o no.

En el otro extremo, *BEAR* aparece como el modelo con mayor carga computacional, viéndose penalizado a medida que el número de nodos desplegados en el escenario crece. Aunque comparte gran parte de la estructura del modelo *Nativo*, *BEAR* requiere el cálculo de una nueva contribución a la hora de estimar la calidad de la señal recibida, obtenida a través de un filtro *AR*, que almacenará las últimas F muestras de la *SNR* de las tramas previas, produciendo como salida el resultado de la Ecuación (2.21). Al igual que con el modelo *HMP*, la ubicación de estas "ventanas" se encuentra individualizada para cada uno de los enlaces $l_{i,j}$, teniendo que llevarse a cabo, tras cada recepción, una nueva búsqueda. Una vez localizada la entrada deseada, se calculará el valor (en dB) de esta contribución a través de la expresión definida en la Ecuación (2.21), donde se puede apreciar la presencia de otro bucle adicional dependiente del número de muestras almacenadas. Tras la obtención del valor de *SNR* como suma de las tres señales independientes (propagación + desvanecimiento lento + desvanecimiento rápido), se producirá el mapeo $SNR \rightarrow FER$ a través de la Ecuación (2.22).

2.4.5. Discusión

En primer lugar, el modelo *Nativo* replica adecuadamente algunos de los parámetros fundamentales de comportamiento del canal inalámbrico, como son la tasa de error de trama o el *throughput*. No obstante, no es capaz de reflejar la memoria que caracteriza este tipo de transmisiones, dando lugar a unos resultados muy predecibles. Esto hace que, aunque el coste computacional de este modelo sea muy reducido, su utilización no sea recomendable.

Tabla 2.10: Resumen comparativo de los modelos de canal analizados

Parámetro	***Nativo***	***HMP***	***BEAR***
Precisión (*FER* y *throughput*)	*Alto*	*Alto*	*Medio*
Memoria del canal	*Muy baja*	*Media*	*Alta*
Variabilidad	*Muy baja*	*Media*	*Alta*
Complejidad	*Baja*	*Media*	*Alta*
Reproducibilidad	*Muy alta*	*Baja*	*Media*

Por otra parte, el modelo *BEAR* proporciona una alta variabilidad en sus resultados, reflejando el comportamiento que se aprecia en un canal real. De hecho, para una única configuración, manteniendo la potencia de entrada de ruido del filtro *AR* (P_ϵ) y la varianza de la variable aleatoria que define la componente de *Shadowing*, $N(0, \sigma^2)$, es capaz de cubrir prácticamente todo el rango observado en los resultados experimentales (Sección 2.1). No obstante, el proceso es complejo, y su coste computacional crece muy rápido con el número de nodos en comparación con los modelos *Nativo* y *HMP*.

Por último, el canal basado en un proceso oculto de Markov (*HMP*) se sitúa en un término medio entre los dos anteriores, ya que es capaz de ofrecer un rango de resultados razonablemente amplio. Además, su ritmo de crecimiento en términos de tiempo de simulaciones se mantiene dentro de unos márgenes aceptables, cercanos a los ofrecidos por el modelo *Nativo*. Con todo esto, la opción *HMP* parece la idónea en aquellos escenarios donde se necesite replicar correctamente el comportamiento a ráfagas del canal inalámbrico, especialmente si el número de nodos desplegados desaconseja la utilización de *BEAR*.

Otro aspecto que merece tenerse en consideración es, debido a la naturaleza *empírica* de los modelos presentados (*HMP* y *BEAR*), la *reproducibilidad* de los mismos. Este término engloba tanto a la complejidad a la hora de obtener los ficheros traza (mediante una campaña experimental) como a la dificultad de configurar los parámetros de los modelos a partir de estos archivos. Una de las mayores limitaciones de este tipo de enfoques, es su fuerte dependencia con las condiciones bajo las que se obtuvieron los ficheros traza. En concreto, son muchas las variables que pueden tener un gran impacto en el comportamiento final del canal inalámbrico, como la versión del estándar *IEEE 802.11* elegida, el escenario sobre el que se realiza el banco de pruebas, la longitud de las tramas, la presencia de fuentes interferentes, etc. Todos ellos influyen de manera directa en la relación entre la calidad de la señal recibida, cuantificada habitualmente a través del parámetro *SNR* y la probabilidad de error de trama. Podría decirse que la *SNR* es independiente de la longitud de las tramas y de la modulación empleadas, ya que su valor se estima durante la recepción de la cabecera *PLCP* [Vlav08], pero sí tendrán gran relevancia en el cálculo de la *FER*. En este contexto, el módulo *HMP* parece ser el más perjudicado, dado que al carecer de un modelo de propagación, requiere la realización de múltiples campañas de medidas en función de la longitud de las tramas y de la modulación utilizada. No obstante, gracias a la configuración basada en el tiempo presentada en esta Tesis, ha sido posible desacoplar el comportamiento de las matrices del proceso de las características del tráfico que lo definen.

Finalmente, la Tabla 2.10 recoge, a modo de resumen, las diferentes prestaciones ofrecidas por cada uno de los tres modelos de canal analizados.

2.5 Conclusiones y líneas futuras

Como contribución más destacada, en este capítulo se han presentado dos aproximaciones diferentes a la hora de replicar el rendimiento *real* de un canal inalámbrico en escenarios interiores. El principio fundamental de ambos modelos es su naturaleza empírica, ya que los resultados obtenidos en una exhaustiva campaña de medidas serán utilizados para su posterior configuración y ajuste. En primer lugar, el modelo basado en un proceso oculto de Markov (*HMP*) "discretiza" la respuesta del canal en un número finito de estados, construyendo una cadena oculta de Markov, generada a partir de las secuencias de observables obtenidas en trazas de transmisiones reales. En concreto, para generar las matrices de transición y emisión se utilizan trazas de recepciones, correctas o erróneas por parte del nodo destino. Además, se ha propuesto un nuevo esquema que consigue *desacoplar* el modelo de las condiciones de entrenamiento de los parámetros, permitiendo adaptarse dinámicamente a diferentes patrones de tráfico. Por otra parte, *BEAR* estima la calidad de la señal recibida a través de la combinación de tres tipos de contribuciones diferentes (atenuación + *shadowing* + desvanecimientos rápidos); su factor diferencial se encuentra en la representación de la segunda de las contribuciones a través de la utilización de un filtro *auto-regresivo*. Otro rasgo común es que son capaces de emular dos de los fenómenos más característicos de las transmisiones realizadas sobre este tipo de de escenarios: su gran aleatoriedad y la presencia de ráfagas de tramas erróneas, consecuencia de la memoria que presenta el medio inalámbrico.

Junto con las dos propuestas anteriores se ha incluido un tercer modelo, disponible en las versiones por defecto del simulador. En concreto, combina un esquema de propagación compuesto por un cálculo determinista de las pérdidas de propación más una entrada de ruido normalizado para replicar el efecto producido por los desvanecimientos rápidos (*FF*). Comparando con *BEAR*, este modelo *Nativo* prescinde del filtro para representar la memoria del canal (efecto asociado al *shadowing*); por otro lado, el modelo de error se basa en el cálculo de la probabilidad de error de trama (*FER*) en función de la modulación, la calidad de la señal recibida y la longitud de la trama.

Se ha llevado a cabo un exhaustivo análisis de las diferentes propuestas en el marco del simulador `ns-3`, dividiendo el proceso de medidas en cuatro fases diferentes:

1. En primer lugar se ha usado el protocolo *UDP* para obtener los umbrales máximos de rendimiento sobre un enlace directo entre dos nodos sobre un canal *IEEE 802.11b*. Se ha observado que, aunque el modelo *Nativo* proporciona unos resultados aceptables en algunas de las métricas de rendimiento más básicas, como el *throughput* o la tasa de error de trama, no es capaz de replicar los dos fenómenos más importantes en

una transmisión inalámbrica: la alta variabilidad y el efecto memoria. Todo esto viene a demostrar que este tipo de modelos de canal, típicamente ofrecidos en los entornos de simulación de red más populares, no reflejan algunas de las características más determinantes observadas en escenarios reales, como consecuencia de poseer un alto grado de simplicidad. Por otro lado, los dos modelos propuestos, *HMP* y *BEAR*, además de generar unos valores adecuados de *FER* y *throughput*, muestran un alto rango de aleatoriedad entre las realizaciones, replicando en gran medida el comportamiento de canal real. Al mismo tiempo, se ha observado como ambos reflejan el factor de memoria de una manera adecuada, capturando del mismo modo un rendimiento razonable en términos de ráfagas de errores.

2. Posteriormente, y tras la caracterización de los niveles inferiores (y del propio canal inalámbrico), se ha evaluado el efecto que este tipo de canales *a ráfagas* producen sobre el protocolo de nivel de transporte más utilizado en la actualidad, *TCP*. Al igual que con *UDP*, se ha vuelto a observar la escasa variabilidad debida a la falta de memoria en el modelo *Nativo*. Por el contrario, tanto *BEAR* como *HMP* muestran un amplio rango de resultados, así como un comportamiento similar al visto sobre el canal real, siendo ambos capaces de reflejar los efectos nocivos producidos por la presencia de largas ráfagas de tramas corruptas, que ocasionan largos tiempos de inactividad en las estaciones transmisoras.

3. En una tercera etapa, se ha evaluado la evolución del rendimiento del canal en función de la distancia entre los nodos. Mientras que en los modelos *Nativo* y *BEAR* la atenuación producida por la pérdida de energía a lo largo del canal se calcula a través de esquemas de propagación "log distancia", una de las limitaciones más obvias de los modelos basados en procesos ocultos de Markov es su falta de dependencia con la calidad del enlace inalámbrico, ya que su comportamiento se encuentra estrechamente ligado a las matrices que los definen, cuyos valores son configurados *offline*, antes de generar el escenario. Para poder solventar esta deficiencia, se ha propuesto la modificación dinámica de los coeficientes del *HMP* en función de la distancia existente entre el nodo transmisor y su receptor. Se han establecido los diferentes umbrales entre los cuales se ha llevado a cabo el mapeo *distancia/coeficientes HMP*. A tenor de los resultados puede afirmarse que, a pesar de presentar una respuesta discretizada (limitada por el número finito de medidas utilizadas para entrenar al modelo), la variedad de resultados obtenidos en el canal real permite al modelo *HMP* ajustarse dinámicamente a la distancia entre los nodos.

4. En último lugar, se ha analizado la complejidad computacional de cada uno de los tres modelos estudiados, en función del número de nodos desplegados en el escenario. En términos estrictos, el modelo *Nativo* presenta el menor coste, mientras que *BEAR* se trata de la alternativa más "pesada", ya que las búsquedas que realiza de forma recursiva (i.e. enlace + ventana del filtro) cada vez que se recibe una trama castigan en gran medida al tiempo total de la simulación. En un punto intermedio se encuentra el modelo *HMP*, cuyo coste computacional es claramente inferior al de

BEAR, manteniendo una tendencia de crecimiento a medida que aumenta el número de nodos en el escenario bastante cercano al observado en el modelo *Nativo*.

A tenor de lo visto en este capítulo, si se tuviese que responder a la pregunta"*¿Cuál de los dos modelos es la mejor opción?*", no sería posible una respuesta categórica. Por un lado, si lo que se desea es centrarse en emular canales muy impredecibles o que requieran un alto nivel de dependencia en cuanto a la separación entre los nodos, la opción más razonable sería *BEAR*; por otro lado, si se necesita un modelo que proporcione un adecuado nivel de precisión, que refleje el efecto *memoria* y que, además, necesite de una menor cantidad de tiempo para proporcionar los resultados, la elección más interesante sería el modelo *HMP*. Puede afirmarse, por lo tanto, que ambos modelos se complementan, ya que los puntos más débiles de uno se ven compensados con mejores prestaciones del otro, y viceversa.

En relación a las posibles líneas de investigación abiertas a partir del contenido expuesto en este capítulo están destinadas, en su mayor parte, a mejorar y extender las funcionalidades de los modelos de canal propuestos, los cuales también pueden aprovecharse para analizar diferentes algoritmos y protocolos, entre los que se pueden incluir técnicas *cross-layer*. De entre todas las posibles actuaciones, se citan a continuación algunas de las más relevantes:

- El paso más inmediato a realizar, ya que no implicaría cambio alguno en la implementación de los modelos, sería el adaptar nuevas y diferentes recomendaciones de capa física. En concreto, el simulador cuenta con una versión estable para la extensión *IEEE 802.11g* [Pei09b], mientras que la correspondiente a la norma *IEEE 802.11n* ha sido adaptada recientemente (Agosto 2013) y aún no ha sido verificada. Por otro lado, a pesar de que el estándar *IEEE 802.11ac* ha sido aprobado hace escasos meses (Enero 2014), todavía no parece que esté dentro de los planes del simulador. El procedimiento a seguir para sintetizar los parámetros de configuración de los modelos serían los mismos que los descritos en la Sección 2.1.2, donde los resultados obtenidos durante las campañas de medida servirían como elementos de entrada, definiendo las matrices que componen los procesos ocultos de Markov o los coeficientes de los filtros auto-regresivos utilizados en *BEAR*, así como las funciones que mapean la calidad de la señal recibida en la tasa de error de trama.

- Junto con la incorporación de nuevas versiones del estándar, sería preciso también extender el número de configuraciones disponibles para la recomendación *IEEE 802.11b*, cubriendo un mayor rango de modulaciones (se definen 4 opciones, con tasas binarias de 1, 2, 5.5 y 11 *Mbps*) y de longitudes de trama (solamente se ha estudiado el comportamiento asociado a tramas de longitud máxima).

- Otro aspecto interesante sería la ampliación del análisis a nuevas situaciones en la red, modificando por ejemplo los patrones de tráfico, el número de flujos presentes en el sistema o la cantidad de dispositivos desplegados (pudiendo habilitar la movilidad

como parámetro adicional del sistema). Este punto en concreto debería verse asimismo reforzado con la búsqueda de caracterizaciones analíticas que permitan modelar los nuevos efectos que aparecen a raíz de la inclusión de diferentes fuentes de transmisión, como son las colisiones o el comportamiento de los mecanismos *DCF* de control de acceso al medio del estándar *IEEE 802.11*. Además, los modelos propuestos *no tienen en cuenta* la contribución asociada a posibles fuentes de interferencia en el canal inalámbrico, como por ejemplo posibles celdas *Wireless Local Area Network (WLAN)* situadas en canales adyancentes o de tecnologías que comparten la misma región del espectro. En cualquier caso, serán los mecanismos estándar del simulador, cuya implementación se encuentra aún en desarrollo [NS3Int], los que llevarían a cabo tales cálculos. Esta interacción debería ser modificada en un futuro con el fin de proporcionar una solución integral que abarque toda la operación asociada a la capa física.

- En el canal *Nativo*, sería interesante utilizar otro modelo de error que permita emular longitudes mayores de ráfagas de errores, dotando al sistema de la memoria necesaria. Para el modelo *HMP*, no se ha tenido en cuenta una posible correlación entre las cadenas de Markov asociadas a nodos cercanos, manteniéndose como procesos estocásticos independientes. Asimismo, sería interesante realizar una caracterización inversa que cuantifique las ráfagas de tramas correctas, para comprobar si los modelos propuestos son capaces de aproximarse al canal real en este aspecto.

Para dar por finalizado este bloque es preciso hacer hincapié en que durante la realización de esta Tesis se ha tratado de fomentar la estrategia *open-source* del propio simulador. Así, cualquier tipo de documentación, modelo, diseño, implementación, *testbed* o resultado de los que se han descrito a lo largo de este Capítulo puede ser replicado. En concreto, los desarrollos aquí expuestos pueden localizarse en [Góme14]. Además, los módulos *HMP* y *BEAR* se encuentran en *proceso de revisión* por parte de los responsables del simulador, por lo que podrían incorporarse en un futuro a la distribución oficial del simulador.

Referencias

[Agua03] Daniel Aguayo, John Bicket, Sanjit Biswas y Douglas De Couto. *MIT Roofnet: Construction of a Production Quality Ad-Hoc Network*. 2003.

[Agüe07] Ramón Agüero. «Contribución a la mejora de las prestaciones en redes de acceso inalámbricas no convencionales». Tesis doct. The address of the publisher: Universidad de Cantabria, nov. de 2007.

[Agüe10] Ramón Agüero, Marta García-Arranz y Luis Muñoz. «Accurate simulation of 802.11 indoor links: a Bursty channel model based on real measurements». En: *EURASIP J. Wirel. Commun. Netw.* 2010 (abr. de 2010), 16:1-16:12. ISSN: 1687-1472.

[AlBa12] Mustafa Al-Bado, Cigdem Sengul y Ruben Merz. «What details are needed for wireless simulations? - A study of a site-specific indoor wireless model.» En: *INFOCOM*. IEEE, 2012, págs. 289-297. ISBN: 978-1-4673-0773-4.

[Allm09] M. Allman, V. Paxson y E. Blanton. *TCP Congestion Control*. RFC 5681 (Draft Standard). Internet Engineering Task Force, sep. de 2009. URL: http://www.ietf.org/rfc/rfc5681.txt.

[Amir95] E. Amir, H. Balakrishnan, S. Seshan y R.H. Katz. «Efficient TCP over networks with wireless links». En: *Hot Topics in Operating Systems, 1995. (HotOS-V), Proceedings., Fifth Workshop on*. Mayo de 1995, págs. 35-40. DOI: 10.1109/HOTOS.1995.513451.

[Bakr95] A Bakre y B. R. Badrinath. «I-TCP: indirect TCP for mobile hosts». En: *Distributed Computing Systems, 1995., Proceedings of the 15th International Conference on*. Mayo de 1995, págs. 136-143. DOI: 10.1109/ICDCS.1995.500012.

[Bars09] P. Barsocchi, G. Oligeri y F. Potorti. «Measurement-based frame error model for simulating outdoor Wi-Fi networks». En: *IEEE Transactions on Wireless Communications* 8.3 (mar. de 2009), págs. 1154-1158. ISSN: 1536-1276. DOI: 10.1109/TWC.2009.060475.

[Baum72] Leonard E. Baum. «An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes». En: *Inequalities III: Proceedings of the Third Symposium on Inequalities*. Ed. por Oved Shisha. University of California, Los Angeles: Academic Press, 1972, págs. 1-8.

[Berk05] Berkeley University of California. *A Comparative Analysis of TCP Tahoe, Reno, New-Reno, SACK and Vegas*. 2005.

[Card09] K.V. Cardoso y J.F. de Rezende. «Accurate hidden markov modeling of packet losses in indoor 802.11 networks». En: *Communications Letters, IEEE* 13.6 (jun. de 2009), págs. 417-419. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2009.082008.

[Conv06] G. Convertino, S. Oliva, F. Sigona y L. Anchora. «MMC05-1: An Adaptive FEC Scheme to Reduce Bursty Losses in a 802.11 Network». En: *IEEE Global Telecommunications Conference, 2006. GLOBECOM '06.* 2006, págs. 1-6. DOI: 10.1109/GLOCOM.2006.221.

[Eckh96] David Eckhardt y Peter Steenkiste. «Measurement and analysis of the error characteristics of an in-building wireless network». En: *SIGCOMM Comput. Commun. Rev.* 26.4 (ago. de 1996), págs. 243-254. ISSN: 0146-4833. DOI: 10.1145/248157.248178. URL: http://doi.acm.org/10.1145/248157.248178.

[Elli63] E. O. Elliott. «Estimates of Error Rates for Codes on Burst-Noise Channels». En: *Bell System Technical Journal* 42 (sep. de 1963), págs. 1977-1997.

[Emulab] *Emulab. Total network testbed.* http://www.emulab.net/.

[Ephr02] Y. Ephraim y N. Merhav. «Hidden Markov processes». En: *IEEE Transactions on Information Theory* 48.6 (jun. de 2002), págs. 1518-1569. ISSN: 0018-9448. DOI: 10.1109/TIT.2002.1003838.

[Ford13] A. Ford, C. Raiciu, M. Handley y O. Bonaventure. *TCP Extensions for Multipath Operation with Multiple Addresses*. RFC 6824 (Experimental). Internet Engineering Task Force, ene. de 2013. URL: http://www.ietf.org/rfc/rfc6824.txt.

[Frii46] H.T. Friis. «A Note on a Simple Transmission Formula». En: *Proceedings of the IRE* 34.5 (mayo de 1946), págs. 254-256. ISSN: 0096-8390. DOI: 10.1109/JRPROC.1946.234568.

[Gand08] Venkat Raju Gandikota, Bheemajun Reddy Tamma y C. Siva Ram Murthy. «Adaptive FEC-Based Packet Loss Resilience Scheme for Supporting Voice Communication over Ad hoc Wireless Networks». En: *IEEE Transactions on Mobile Computing* 7.10 (oct. de 2008), págs. 1184-1199. ISSN: 1536-1233. DOI: 10.1109/TMC.2008.42. URL: http://dx.doi.org/10.1109/TMC.2008.42.

[Garc04] Marta García-Arranz, Ramón Agüero y Luis Muñoz. «On the Unsuitability of TCP RTO Estimation over Bursty Error Channels». English. En: *Personal Wireless Communications*. Ed. por Ignas Niemegeers y SoniaHeemstra de Groot. Vol. 3260. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, págs. 343-348. ISBN: 978-3-540-23162-2. DOI: 10.1007/978-3-540-30199-8_28. URL: http://dx.doi.org/10.1007/978-3-540-30199-8_28.

[Gilb60] E. N. Gilbert. «Capacity of a Burst-Noise Channel». En: *Bell System Technical Journal* 39.5 (1960), págs. 1253-1265. ISSN: 1538-7305. DOI: 10.1002/j.1538-7305.1960.tb03959.x. URL: http://dx.doi.org/10.1002/j.1538-7305.1960.tb03959.x.

[Góme14] David Gómez y Ramón Agüero. *GitHub's repository for HMP and BEAR IEEE 802.11b indoor channel models (source code and documentation)*. https://github.com/dgomezunican/ns3-wifi-memory-channel. 2014.

[Ha08] Sangtae Ha, Injong Rhee y Lisong Xu. «CUBIC: A New TCP-friendly High-speed TCP Variant». En: *SIGOPS Oper. Syst. Rev.* 42.5 (jul. de 2008), págs. 64-74. ISSN: 0163-5980. DOI: 10.1145/1400097.1400105. URL: http://doi.acm.org/10.1145/1400097.1400105.

[Hass11] G. Hasslinger y O. Hohlfeld. «Analysis of packet errors in Gilbert-Elliott channels». En: *Signal Processing Advances in Wireless Communications (SPAWC), 2011 IEEE 12th International Workshop on*. Jun. de 2011, págs. 216-220. DOI: 10.1109/SPAWC.2011.5990398.

[Hend12] T. Henderson, S. Floyd, A. Gurtov e Y. Nishida. *The NewReno Modification to TCP's Fast Recovery Algorithm*. RFC 6582 (Proposed Standard). Internet Engineering Task Force, abr. de 2012. URL: http://www.ietf.org/rfc/rfc6582.txt.

[Hoch99] B.M. Hochwald y P.R. Jelenkovic. «State learning and mixing in entropy of hidden Markov processes and the Gilbert-Elliott channel». En: *Information Theory, IEEE Transactions on* 45.1 (ene. de 1999), págs. 128-138. ISSN: 0018-9448. DOI: 10.1109/18.746777.

[IEEE802.11] «IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications». En: *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)* (29 de 2012), págs. 1-2793. DOI: 10.1109/IEEESTD.2012.6178212.

[Ikku02] P. Ikkurthy y M.A. Labrador. «Characterization of MPEG-4 traffic over IEEE 802.11b wireless LANs». En: *Local Computer Networks, 2002. Proceedings. LCN 2002. 27th Annual IEEE Conference on*. Nov. de 2002, págs. 421-427. DOI: 10.1109/LCN.2002.1181814.

[Iperf] *Iperf - The TCP/UDP Bandwidth Measurement Tool*. http://iperf.fr/.

[John06] D. Johnson, T. Stack, R. Fish, D.M. Flickinger, L. Stoller, R. Ricci y J. Lepreau. «Mobile Emulab: A Robotic Wireless and Sensor Network Testbed». En: *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*. Abr. de 2006, págs. 1-12. DOI: 10.1109/INFOCOM.2006.182.

[JSim] *J-Sim*. http://j-sim.cs.uiuc.edu/. 2012.

[Kera12] Stratos Keranidis, Dimitris Giatsios, Thanasis Korakis, Iordanis Koutsopoulos, Leandros Tassiulas, Thierry Rakotoarivelo y Thierry Parmentelat. «Experimentation in Heterogeneous European Testbeds through the Onelab Facility: The Case of PlanetLab Federation with the Wireless NITOS Testbed.» En: *TRIDENTCOM*. Ed. por Thanasis Korakis, Michael Zink y Maximilian Ott. Vol. 44. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, 2012, págs. 338-354. ISBN: 978-3-642-35575-2. URL: http://dblp.uni-trier.de/db/conf/tridentcom/tridentcom2012.html#KeranidisGKKTRP12.

[Kurk05] Stuart Kurkowski, Tracy Camp y Michael Colagrosso. «MANET Simulation Studies: The Incredibles». En: *SIGMOBILE Mob. Comput. Commun. Rev.* 9.4 (oct. de 2005), págs. 50-61. ISSN: 1559-1662. DOI: 10.1145/1096166.1096174. URL: http://doi.acm.org/10.1145/1096166.1096174.

[Laca06] Mathieu Lacage y Thomas R. Henderson. «Yet another network simulator». En: *Proceeding from the 2006 workshop on ns-2: the IP network simulator*. WNS2 '06. ACM, 2006. ISBN: 1-59593-508-8. DOI: 10.1145/1190455.1190467. URL: http://doi.acm.org/10.1145/1190455.1190467.

[Lam03] King-Wai Lam, Xu Peng, Leung Tsang, Qin Li y K.L. Lai. «Wireless communications channel modeling based on numerical solutions of Maxwell equations». En: *Wireless Communication Technology, 2003. IEEE Topical Conference on*. Oct. de 2003, págs. 261-262. DOI: 10.1109/WCT.2003.1321515.

[Lert14] Daniel Lertpratchta, George F. Riley y Douglas M. Blough, eds. *Simulating Frame-Level Bursty Links in Wireless Networks*. ACM, 2014.

[Math96] M. Mathis, J. Mahdavi, S. Floyd y A. Romanow. *TCP Selective Acknowledgment Options*. RFC 2018 (Proposed Standard). Internet Engineering Task Force, oct. de 1996. URL: http://www.ietf.org/rfc/rfc2018.txt.

[Matlab] *Matlab. El lenguaje del cálculo técnico*. http://www.mathworks.es/products/matlab/. 1984.

[MatWi] *Matlab wireless network simulator*. http://wireless-matlab.sourceforge.net/. 2006.

[Mitt12] Jens Mittag y Stylianos Papanastasiou. *PhySimWiFi for NS-3*. https://dsn.tm.kit.edu/english/ns3-physim.php. 2012.

[Mush89] M. Mushkin e I. Bar-David. «Capacity and coding for the Gilbert-Elliott channels». En: *Information Theory, IEEE Transactions on* 35.6 (nov. de 1989), págs. 1277-1290. ISSN: 0018-9448. DOI: 10.1109/18.45284.

[Naka60] M. Nakagami. «The m-distribution – A general formula of intensity distribution of rapid fading». En: *Statistical Methods in Radio Wave Propagation*. Ed. por W. C. Hoffmann. Elmsford, NY, 1960.

[Nguy96] G.T. Nguyen, B. Noble, R.H. Katz y M. Satyanarayanan. «A trace-based approach for modeling wireless channel behavior». En: *Simulation Conference, 1996. Proceedings. Winter*. 1996, págs. 597-604. DOI: 10.1109/WSC.1996.873340.

[Nits11] Thomas Nitsche y Thomas Fuhrmann. «A Tool for Raytracing Based Radio Channel Simulation». En: *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*. SIMUTools '11. ICST (Institute for Computer Sciences, Social-Informatics y Telecommunications Engineering), 2011, págs. 69-74. ISBN: 978-1-936968-00-8. URL: http://dl.acm.org/citation.cfm?id=2151054.2151067.

[NS2] *The ns-2 network simulator*. http://www.isi.edu/nsnam/ns/. 1995.

[NS3] *The ns-3 network simulator*. http://www.nsnam.org/. 2008.

[NS3Int] *802.11b Interference Modeling in NS-3 Simulator*. http://www.ee.washington.edu/research/funlab/802_11_b_intf_model/index.html.

[Octave] *GNU Octave*. https://www.gnu.org/software/octave/.

[OMNET] *OMNet++*. http://www.omnetpp.org/.

[ORBIT] *Open-Access Research Testbed for Next-Generation Wireless Networks (ORBIT)*. https://www.orbit-lab.org/. 2003.

[Papa10] S. Papanastasiou, J. Mittag, E.G. Strom y H. Hartenstein. «Bridging the Gap between Physical Layer Emulation and Network Simulation». En: *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*. 2010, págs. 1-6. DOI: 10.1109/WCNC.2010.5506341.

[Pei09a] Guangyu Pei y Tom Henderson. *Validation of ns-3 802.11b PHY model*. http://www.nsnam.org/p̃ei/80211b.pdf. Mayo de 2009.

[Pei09b] Guangyu Pei y Tom Henderson. *Validation of OFDM error rate model in ns-3*. Mayo de 2009. URL: %5Curl%7Bhttp://www.nsnam.org/~pei/80211ofdm.pdf%7D.

[Planetlab] *Planetlab. An open platform for developing, deploying and accessing planetary-scale services*. https://www.planet-lab.org/. 2002.

[Post81] J. Postel. *Transmission Control Protocol*. RFC 793 (INTERNET STANDARD). Updated by RFCs 1122, 3168, 6093, 6528. Internet Engineering Task Force, sep. de 1981. URL: http://www.ietf.org/rfc/rfc793.txt.

[QualNet] *Qualnet Network Simulator*. http://web.scalable-networks.com/content/qualnet. 2012.

[Rabi86] L. R. Rabiner y B. H. Juang. «An introduction to hidden Markov models». En: *IEEE ASSp Magazine* (1986).

[Rabi89] L. Rabiner. «A tutorial on hidden Markov models and selected applications in speech recognition». En: *Proceedings of the IEEE* 77.2 (1989), págs. 257-286. ISSN: 0018-9219. DOI: 10.1109/5.18626.

[Rapp01] Theodore Rappaport. *Wireless Communications: Principles and Practice*. 2nd. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN: 0130422320.

[Riverbed] *Riverbed Application Performance Platform*. http://www.riverbed.com/products/performance-management-control/opnet.html. 2012.

[Sark11] Nurul I. Sarkar, Senior Member y Syafnidar A. Halim. «A review of simulation of telecommunication networks: simulators, classification, comparison, methodologies, and recommendations». En: *Journal of Selected Areas in Telecommunications (JSAT* (2011), págs. 10-17.

[Saun99] Simon R. Saunders y Saunders R Simon. *Antennas and Propagation for Wireless Communication Systems*. 1st. New York, NY, USA: John Wiley & Sons, Inc., 1999. ISBN: 0471986097.

[Shan48] Claude Shannon. «A Mathematical Theory of Communication». En: *Bell System Technical Journal* 27 (jul. de 1948), págs. 379-423, 623-656. URL: http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf.

[Srid08] M. Sridharan, K. Tan, D. Bansal y D. Thaler. *Compound TCP: A New TCP Congestion Control for High-Speed and Long Distance Networks*. Working Draft. Internet-Draft. 2008.

[Stev97] W. Stevens. *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*. RFC 2001 (Proposed Standard). Obsoleted by RFC 2581. Internet Engineering Task Force, ene. de 1997. URL: http://www.ietf.org/rfc/rfc2001.txt.

[Stew07] R. Stewart. *Stream Control Transmission Protocol*. RFC 4960 (Proposed Standard). Updated by RFCs 6096, 6335, 7053. Internet Engineering Task Force, sep. de 2007. URL: http://www.ietf.org/rfc/rfc4960.txt.

[Stüb01] Gordon L. Stüber. *Principles of Mobile Communication (2Nd Ed.)* Norwell, MA, USA: Kluwer Academic Publishers, 2001. ISBN: 0-7923-7998-5.

[Vasu03] V. Vasudevan, M. Parikh, K. Ch y C. Thompson. *TCP and IEEE 802.11b Protocol Performance in Indoor Wireless Channels*. 2003.

[Vlav08] A. Vlavianos, L.K. Law, I. Broustis, S.V. Krishnamurthy y M. Faloutsos. «Assessing link quality in IEEE 802.11 Wireless Networks: Which is the right metric?» En: *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*. Sep. de 2008, págs. 1-6. DOI: 10.1109/PIMRC.2008.4699837.

[Walk31] G. Walker. «On Periodicity in Series of Related Terms». En: *Proc. R. Soc. ser. A* 313 (1931), págs. 518-532.

[Wein09] E. Weingartner, H. vom Lehn y K. Wehrle. «A Performance Comparison of Recent Network Simulators». En: *Communications, 2009. ICC '09. IEEE International Conference on*. Jun. de 2009, págs. 1-5. DOI: 10.1109/ICC.2009.5198657.

[Yule27] Udny G. Yule. «On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers». En: (1927).

[Zeng98] Xiang Zeng, Rajive Bagrodia y Mario Gerla. «GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks». En: *Proceedings of the Twelfth Workshop on Parallel and Distributed Simulation*. PADS '98. IEEE Computer Society, 1998, págs. 154-161. ISBN: 0-8186-8457-7. DOI: 10.1145/278008.278027. URL: http://dx.doi.org/10.1145/278008.278027.

Bloque II

El protocolo *MPTCP*

3

Transmisión simultánea a través de múltiples interfaces. El protocolo *MPTCP*

Puede afirmarse que, desde los primeros pasos de la conmutación de paquetes, el tráfico de datos ha estado principalmente dominado por dos protocolos: en primer lugar, el protocolo *IP* [Post81a], que proporciona un servicio de red no confiable; sobre éste se sitúa la solución de nivel de transporte por excelencia, *TCP* [Post81b], que cubre las limitaciones inherentes a *IP*, que ofrece un servicio *best effort*, no orientado a la conexión y que no garantiza la correcta recepción de la información en los extremos. A pesar de que con el paso del tiempo han ido apareciendo numerosas alternativas al binomio *TCP/IP*, entre las que se encuentran diferentes soluciones en el nivel de transporte, como *UDP*, *SCTP* o *Data Congestion Control Protocol (DCCP)*, entre otros, *TCP* se mantiene a día de hoy como el protocolo más utilizado. Cuando se sentaron las bases del protocolo, la tecnología estaba marcada por el dominio de redes y dispositivos cableados. Sin embargo, con el paso del tiempo, esta tendencia ha ido evolucionando gradualmente y actualmente las tecnologías inalámbricas han ganado un gran peso en las redes de acceso.

Gran parte de la culpa de este cambio la tiene el enorme crecimiento, tanto en términos de tecnologías disponibles como de despliegue y popularidad, de los dispositivos inalámbricos. Junto con los terminales más tradicionales (e.g. teléfonos móviles, ordenadores portátiles, etc.), ha aparecido un elenco de nuevos equipos, como *tablets* o *smartphones*, que vienen a demostrar el enorme potencial de este tipo de comunicaciones. Además, la inmensa mayoría cuenta a día de hoy con múltiples interfaces, incorporando diferentes tecnologías de acceso radio, como *IEEE 802.11*, *Bluetooth*, *Universal Mobile Telecommunications System (UMTS)* o *LTE*, entre otra muchas, posibilitando un uso simultáneo.

Hay que tener en cuenta, además, las limitaciones que el binomio *TCP/IP* presenta a la hora de gestionar la movilidad de los usuarios. En su modo de operación tradicional, un cambio de tecnología obligaría a romper la conexión a nivel *TCP*, teniendo que restablecer una nueva.

Todo esto ha suscitado un elevado interés dentro de la comunidad científica, que está aunando esfuerzos en la búsqueda de alternativas que aprovechen las posibilidades que tienen los nuevos dispositivos y las tecnologías que incorporan. Una de las propuetas más destacables es el protocolo MPTCP [Ford13], originado en el marco de de un grupo de trabajo en el *IETF*. Se puede definir como una evolución natural de *TCP*, que permite la transmisión de la información de manera *simultánea* a través de múltiples caminos (o subflujos) dentro de una única conexión *TCP*, permitiendo además la movilidad de los usuarios sin necesidad de reiniciar una sesión a nivel de transporte. A través de esta idea "aparentemente" sencilla, las aplicaciones podrán beneficiarse de un mejor rendimiento y de una mayor robustez frente a los posibles fallos producidos durante la transmisión.

Tradicionalmente, la mayor parte de los análisis de este tipo de técnicas estaban ligados a la utilización de topologías cableadas; sin embargo, es cada vez más habitual encontrar trabajos que centran sus esfuerzos en el estudio de estas soluciones multi-camino sobre redes inalámbricas, entre los que destacan [Maga01; Chen13; Paas13], entre otros. Este tipo de escenarios presenta un interés adicional, ya que es sobradamente conocido que la naturaleza intrínseca del medio radio deteriora notablemente el rendimiento de los protocolos de nivel de transporte orientados a la conexión, como se puso de manifiesto en el Capítulo 2.

A modo de ejemplo, la Figura 3.1 representa un escenario ilustrativo en el que se pone de manifiesto el beneficio adicional del protocolo *MPTCP*. En el extremo izquierdo del diagrama se reflejan dos diferentes tendencias respecto a la conexión de servidores (por ejemplo en *data centers*). Hoy en día, la mayor parte de los equipos utilizados en grandes granjas de servidores cuentan con varios interfaces de red de alta velocidad que pueden ser combinados para lograr un mejor rendimiento (varias conexiones en el mismo *switch*, como el *Servidor 1* de la figura) o una mayor resistencia a fallos (gracias al uso de diferentes alternativas de acceso a la red dorsal, como es el caso del *Servidor 2*). En la primera configuración, para permitir que el protocolo *TCP* haga un uso más eficiente de la propia red, suele ser habitual "virtualizar" todas estas conexiones bajo una misma dupla de direcciones *IP-MAC*, repartiendo los paquetes entre las diferentes conexiones[1]. No obstante, puede darse el caso en el que el tráfico se descompense en las diferentes rutas, dando lugar a la recepción (en los correspondientes destinos) de la información en forma desordenada, lo que, como ya se describió en el Capítulo 2, puede llegar a perjudicar el rendimiento de la transmisión (sobre todo en *TCP*), ya que se producirán retransmitisiones derivadas de la recepción de múltiples *ACKs* duplicados. Por el contrario, en la segunda configuración, los interfaces de red se conectan a *switches* diferentes, siendo lo más habitual mantener el binomio *IP-MAC* para todos ellos (igual que en el caso anterior). En este caso concreto, una de las conexiones haría las veces de enlace primario, encargado de transportar todo el tráfico, manteniéndose el resto como soluciones alternativas, que solamente actuarían en

[1]Normalmente, un controlador será el encargado de distribuir la carga entre los diferentes interfaces, utilizando alguna de las técnicas de balanceo de carga más comunes (aunque en muchos casos se suele optar por la solución más simple posible, *Round Robin*).

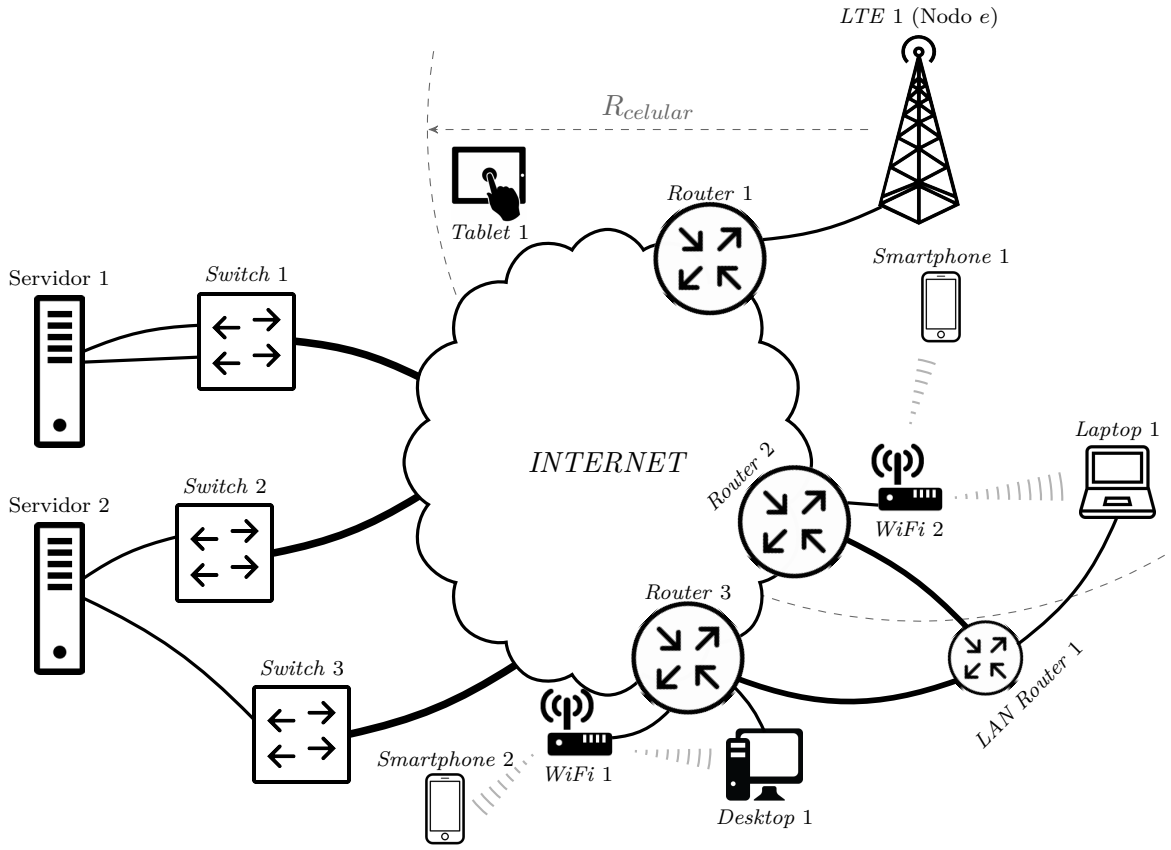


Figura 3.1: Escenario ilustrativo de utilización del protocolo *MPTCP*

caso de emergencia (e.g. ruptura o avería). Como puede deducirse, esta aproximación no contempla la transmisión simultánea a través de múltiples interfaces, por lo que no será una solución óptima cuando se desee utilizar el protocolo *MPTCP*.

La topología correspondiente a la *red dorsal* (zona central de la figura) ha sido suprimida por motivos de simplicidad. En el caso concreto de una sesión *MPTCP*, se necesitaría establecer, al menos, una ruta para cada uno de los subflujos definidos durante la etapa de negociación de la conexión (a través de la utilización de diferentes protocolos de encaminamiento o *routing*). En un esquema más tradicional (como es el caso de una transmisión típica con *TCP* como nivel de transporte) es suficiente con encontrar un único camino entre los terminales, recurriendo a rutas alternativas solamente en caso de necesidad, por lo que la utilización de protocolos como *Routing Information Protocol (RIP)* [Hedr88], *Open Shortest Path First (OSPF)* [Moy98] o *BGP* [Rekh06] será lo más habitual.

Finalmente, la parte conocida como *red de acceso* (parte derecha de la figura) presenta un gran número de tecnologías diferentes con las que poder acceder a la información generada por los servidores, pudiendo hacer uso de conexiones cableadas (*Ethernet*), celulares o *WLANs*. En el diagrama se muestra una serie de terminales de usuario (*PCs* fijos y

portátiles, *smartphones* y *tablets*), viéndose como, dependiendo de las tecnologías de acceso disponibles en su localización física, podrían recibir la información de manera simultánea por cada uno de sus interfaces. En el caso concreto del escenario mostrado, el equipo más beneficiado con el uso del protocolo *MPTCP* sería el correspondiente al *Laptop 1*, ya que se podría establecer una sesión que hiciera uso de hasta tres subflujos diferentes, como son la red cableada, la celular y la conexión *WLAN* a través de una celda *IEEE 802.11*.

A la vista de esta arquitectura y de lo comentado hasta este punto, se infiere que la utilización de diferentes niveles físicos puede producir grandes diferencias en términos de rendimiento (e.g. *throughput*, latencia, *jitter*, etc.). Con todo ello, supone un gran reto la implementación de mecanismos capaces de aprovechar todo el potencial que trae consigo el uso *eficiente* de varios flujos de manera simultánea. De entre todo el rango de funcionalidades que deben ser analizadas, hay tres que presentan una mayor importancia:

1. En primer lugar, será esencial *distribuir los paquetes* que van a ser enviados por cada uno de los interfaces en un nodo transmisor de una manera eficiente, otorgando un mayor peso a aquellas conexiones que presenten un mejor rendimiento.

2. Los *mecanismos de control de flujo y congestión* distribuyen el tráfico disponible entre los subflujos presentes y activos, que estarán a su vez estrechamente ligados a las direcciones *IP* definidas en los nodos. Ante situaciones en las que el rendimiento de alguno de los subflujos decaiga de manera puntual, estos algoritmos deberán balancear la carga hacia las conexiones que ofrezcan un mejor comportamiento.

3. Por último, la *reordenación de los paquetes* en el destino tendrá una especial relevancia, teniendo en cuenta la sensibilidad de los procedimientos utilizados en *TCP* ante la recepción de segmentos desordenados. Se buscarán diferentes alternativas para minimizar el impacto generado por retransmisiones espurias.

A pesar de que las tecnologías inalámbricas que predominan actualmente se basan mayoritariamente en el paradigma del último salto de la red de acceso, las comunicaciones *multi-salto* basadas en redes malladas inalámbricas o *Wireless Mesh Networks (WMNs)* están adquiriendo una atención cada vez más relevante. Por norma general, las tecnologías subyacentes en este tipo de despliegues suelen ser de corto alcance, primando el uso de canales basados en los estándares *IEEE 802.11* o *IEEE 802.15.4*. Sin embargo, es preciso mencionar que también existe la posibilidad de utilizar estas técnicas en redes celulares (de cuarta generación o *LTE*) para conseguir incrementar el radio de cobertura o la capacidad de una celda [Wei13].

En estos despliegues de red, será necesario el empleo de varios saltos para alcanzar el destino, apoyándose para ello en los nodos intermedios[2] que se encuentran desplegados en

[2]En este tipo de redes, los propios terminales podrán interpretar el papel de *routers* y redirigir paquetes hacia el siguiente salto.

la propia red. Para poder establecer de manera eficiente una (o varias) rutas, es imprescindible la utilización de protocolos de encaminamiento. Dentro de este tipo de soluciones, se distinguen dos grandes grupos o filosofías: por un lado se encuentran los protocolos *reactivos*, donde el intercambio de mensajes de descubrimiento o mantenimiento de rutas se lleva a cabo en caso de necesidad (esto es, de manera asíncrona), mientras que las alternativas *proactivas* utilizan unas tablas de rutas que se actualizan de manera periódica, añadiendo una sobrecarga que podría ser considerable, sobre todo en canales compartidos o basados en contienda, como es el caso del estándar *IEEE 802.11*, objeto de estudio a lo largo de todo este trabajo.

Sin embargo, la utilización de estas soluciones de encaminamiento *single-path* se antoja insuficiente en el momento en el que se necesite utilizar más de un camino de manera simultánea, aspecto fundamental en el protocolo *MPTCP*. Valga como ejemplo el caso cada vez más habitual de que los terminales sean *multi-homing*, término utilizado para describir a los dispositivos que pueden comunicarse a través de diferentes tecnologías, pudiendo recibir la información de manera simultánea por todos de sus interfaces de red. Ante esta nueva corriente, han ido surgiendo nuevos protocolos *multi-path* que son capaces de obtener el conjunto óptimo de caminos potencialmente disjuntos que conectan dos nodos en una red mallada inalámbrica. Tal y como se afirma en [Megh10], la difusión del tráfico a través de diferentes caminos puede aliviar situaciones de congestión en la red. Además, si se emplea un protocolo de nivel de transporte (*MPTCP*) que controle la gestión de los recursos disponibles de una forma optimizada, se conseguirá un mayor ancho de banda agregado y un balanceo de carga adecuado, distribuyéndola entre todos los subflujos disponibles en la sesión.

Con todo esto, este Capítulo se centrará en el estudio del rendimiento del protocolo *MPTCP* sobre redes malladas inalámbricas, buscando mejorar el comportamiento observado al utilizar *TCP* sobre este tipo de escenarios. Para ello, se han llevado a cabo los siguientes pasos:

- La primera etapa ha consistido en migrar una implementación del protocolo *MPTCP* en el marco del simulador `ns-3`, llevada a cabo originalmente por Chihani *et al.* [Chih11]. Para comprobar su correcto comportamiento, se realizará un extenso proceso de simulación utilizando un escenario canónico, en el que cuatro nodos se sitúan en los vértices de una topología en forma de *diamante*. Con estas condiciones, se caracterizarán algunos de los algoritmos de control de congestión incluidos en el módulo *MPTCP* (concretamente, *uncoupled subflows*, *fully coupled*, *linked increases* y *RTT compensator*). Además, se demostrará que el rendimiento obtenido a través del protocolo *MPTCP* en redes inalámbricas multi-interfaz es netamente superior al ofrecido por el esquema tradicional de un único camino y *TCP*.
- En segundo lugar, se presentarán algunas de las soluciones para la búsqueda de múltiples rutas en redes malladas inalámbricas, haciendo hincapié en el estudio de las prestaciones de tres algoritmos *Link Disjoint (LD)*, *Node Disjoint (ND)* y *Zone*

Disjoint (ZD) [Megh10] para encontrar la combinación óptima de caminos disjuntos. Se hara uso de la teoría de grafos para la obtención de todas las rutas disponibles entre dos nodos, origen y destino, recurriendo para ello a un proceso externo que genera como resultado un fichero que será utilizado posteriormente en el ámbito del simulador `ns-3`.

- Finalmente, combinando las dos fases anteriores se ampliará la comparativa entre ambas soluciones de nivel de transporte a un escenario más realista y genérico, con un despliegue aleatorio de los nodos. A partir de los resultados obtenidos por los algoritmos mencionados anteriormente, se generarán las tablas de rutas estáticas que emplea el simulador, comprobando si la mejora del protocolo *MPTCP* respecto a *TCP* se sigue dando sobre topologías más complejas.

3.1 Estado del arte

La tendencia a ofrecer cada vez más alternativas de conexión en los dispositivos da lugar al planteamiento de un nuevo paradigma de comunicación, según el cual el tráfico puede dividirse y transportarse al mismo tiempo por caminos potencialmente disjuntos. Muestra de este creciente interés, el *IETF* pone en marcha en el año 2009 un grupo de trabajo (llamado explícitamente *Multipath TCP Working Group*) que se centra en la creación de una solución que permita la división del tráfico de una única sesión *TCP* a través de diferentes rutas. La base de este protocolo se encuentra en el *Request For Comments (RFC)* 6824 [Ford13] que, a su vez, se apoya en una serie de extensiones para conformar el protocolo completo. Además, el impacto que han generado las comunicaciones inalámbricas ha originado una nueva extensión [Hamp11], aún no aprobada, que propone una serie de modificaciones para adaptar correctamente la operación de *MPTCP* sobre este tipo de enlaces/redes.

MPTCP no es la única solución que busca romper las barreras impuestas por *TCP*, donde las conexiones estaban restringidas a un único flujo. El *IETF*, previamente a la aparición de *MPTCP* y a través del grupo de trabajo *SIGnalling TRANsport (SIGTRAN)*, sentó las bases de las comunicaciones multi-camino definiendo el protocolo *SCTP* [Stew07]. Ambas soluciones guardan ciertas similitudes, como la posibilidad de utilizar múltiples direcciones en los nodos terminales, lo que permite utilizar conexiones *multi-path*; no obstante, las diferencias entre ambas también son notables: en *MPTCP* el objetivo principal es el de mejorar las prestaciones (*throughput* y robustez) de *TCP*, mediante la transmisión simultánea; por su parte, *SCTP* trata de proporcionar un factor de redundancia y movilidad a nodos multi-interfaz, pero no permite la utilización de más de un camino al mismo tiempo (a pesar de que existen extensiones que habilitan esta posibilidad [Iyen06], aún no se han incorporado al protocolo principal). Otra diferencia importante es el enorme esfuerzo que se realizó desde un principio para que el protocolo *MPTCP* fuera compatible con *TCP*, facilitando la migración de aplicaciones y siendo, por tanto, la opción más

cómoda [Scha13] para desarrolladores. Por otra parte, en el nivel de red, algunos nodos intermedios o pasarelas, como los *Network Address Translations (NATs)*, podrían no ser conocedores del protocolo *SCTP*, bloqueando los paquetes al procesarlo [Steg10].

Existen trabajos que analizan el comportamiento de *MPTCP* sobre enlaces inalámbricos. Basándose en la implementación disponible para el *Kernel* de *Linux* [MPLin], Maxine Lim y Josh Valdez [Lim] demuestran la mejora introducida con una transmisión *MPTCP*, comparándola con el rendimiento de *TCP* en una combinación de canales *Ethernet*, *IEEE 802.11* y *3G*; además, ponen de manifiesto la capacidad de mantener una sesión durante la realización de un traspaso vertical sin interrumpirla. La campaña de medidas que llevan a cabo se basa en canales emulados, configurados con tasas de error de paquete fijas (0 % en *Ethernet*, 2 % en 3G y 3 % en *IEEE 802.11*); posteriormente, Constin Raiciu *et al.* [Raic12] comparan sus resultados con medidas sobre canales reales, concluyendo que *MPTCP* permite mejorar significativamente el rendimiento del sistema, aunque no analizan el impacto de reducir la capacidad de los enlaces inalámbricos. Por otro lado, en [Nguy11] se utiliza otra implementación del protocolo para evaluar el rendimiento instantáneo y la distribución de la carga entre subflujos en escenarios que mezclan tecnologías *Ethernet*, *IEEE 802.11* y *3G*. En este caso, los resultados obtenidos no son tan alentadores, ya que cuando se mezclan dos canales físicos con características muy dispares (e.g. tasas de error, *Round Trip Time (RTT)*), el rendimiento obtenido es inferior al que obtiene *TCP* utilizando el mejor camino posible. Los autores lo justifican con el impacto negativo de una reordenación poco eficiente sobre subflujos de diferentes características, pero no indican qué tipo de mecanismo utilizan para sus análisis. En cuanto a contribuciones más recientes, destacan algunas como la de Chen et al. [Chen13], donde los autores evalúan el rendimiento de diferentes combinaciones de subflujos *IEEE 802.11* y celulares (*LTE* y *3G*) en una plataforma de medidas real, en la que utilizan un servicio de descarga de ficheros. Sus resultados muestran que la utilización del protocolo *MPTCP* suele mejorar a *TCP*, aunque advierten también que el tamaño de los ficheros tiene un impacto notable en su comportamiento. En concreto, afirman que la elección de un algoritmo de control de congestión u otro no tiene una especial relevancia en la descarga de ficheros pequeños (hasta 4 MB), proporcionando resultados similares; sin embargo, observan que para archivos más grandes ($>$ 4 MB), la utilización de los algoritmos "clásicos" (los utilizados en el marco de esta Tesis) no presenta las mismas prestaciones que otras alternativas más recientes, como el algoritmo *Opportunistic Linked Increases Algorithm (OLIA)* [Khal14a]. Por otra parte, Paasch et al. [Paas13] utilizan el módulo *MPTCP* del *kernel* de *Linux*, que ellos mismos implementaron, para llevar a cabo una plataforma experimental que pone de manifiesto algunas de las principales deficiencias del protocolo, sobre todo en lo relativo a los algoritmos de control de la congestión y el *scheduler* encargado de multiplexar los paquetes entre los diferentes subflujos [Khal13].

Junto a una solución de nivel de transporte que divida el tráfico por diferentes caminos, se requiere un protocolo de encaminamiento que encuentre el conjunto óptimo de rutas entre cualquier par de nodos dentro de una red. Como ya se ha comentado, este tipo de protocolos puede dividirse en dos grandes grupos que, partiendo de bases opues-

tas, establecen los mecanismos para descubrir y mantener las rutas en redes multi-salto. Por un lado se encuentran los protocolos **proactivos** (siendo sus principales representantes *Optimized Link State Routing - OLSR* [Clau03] y *OLSRv2* [Clau14]), que mantienen actualizada la información topológica inundando periódicamente la red, lo que introduce una notable sobrecarga, lo que habitualmente desaconseja su utilización en redes *ad hoc* inalámbricas [Tari09]. Por otra parte, los protocolos **reactivos** o bajo demanda (con *Ad hoc On-demand Distance Vector routing - AODV* [Perk03] como principal exponente) tratan de reducir al máximo el intercambio de mensajes de control, recurriendo a ellos sólo cuando sea imprescindible, actuando de manera asíncrona.

La mayoría de los protocolos de encaminamiento se basan en estrategias "single-path", en las que una avería/sobrecarga en la ruta establecida provoca que un nodo transmisor tenga que volver a iniciar un mecanismo de descubrimiento para encontrar caminos alternativos con los que alcanzar al destino. Debido a la reciente aparición de soluciones multi-camino (y al crecimiento de dispositivos multi-interfaz), se hace necesaria la implementación de nuevos protocolos que hagan frente a los retos impuestos por esta tendencia.

La mayor parte de las soluciones existentes son modificaciones de los algoritmos *single-path*, pudiendo clasificarse según el criterio utilizado para obtener las alternativas al camino más corto tras una primera iteración común, basada en el algoritmo de *Dijkstra* [Dijk59]: *LD*, que excluye únicamente los enlaces de las rutas calculadas previamente (e.g. *Ad hoc On-demand Distance Vector Multi-path- AODVM* [Ye03]), *ND*, que establece que los nodos intermedios no puedan formar parte de dos rutas diferentes (e.g. *Geographic Multipath routing Protocol - GMP* [Losc06]) y, por último, *ZD*, que restringe la participación de los propios nodos y de sus vecinos (e.g. *Zone Disjoint Multipath extension of the Dynamic Source Routing - ZD-MPDSR* [Java07]).

Entre los trabajos más destacables en esta línea se encuentran los realizados por Meghanathan [Megh07; Megh10] que, apoyándose en la teoría de grafos, elabora un completo análisis de las prestaciones de los algoritmos *Link*, *Node* y *Zone Disjoint* en esquemas multi-camino sobre una red mallada inalámbrica; así, evalúa exhaustivamente, a través de simulaciones, las diferentes métricas de rendimiento que caracterizan el comportamiento de los algoritmos analizados (e.g. número medio de rutas encontradas, número medio de saltos, tiempo medio entre solicitudes de descubrimiento de rutas, etc.). Por otra parte, Waharte *et al.* [Waha06] llevan a cabo un estudio alternativo, centrado en los algoritmos *LD* y *ND*, prestando especial atención a las posibles interferencias producidas entre los diferentes subflujos (que comparten el mismo canal inalámbrico), estimando el *throughput* en función del rango de cobertura de los nodos y su posición en el escenario. A diferencia de las contribuciones de Meghanathan, que se centra exclusivamente en los algoritmos de encaminamiento, en este caso compara el rendimiento de los algoritmos de encaminamiento multi-camino con el de un esquema *single-path*, utilizando en ambos casos el protocolo *UDP*.

Es importante mencionar también las implementaciones disponibles en la actualidad para la plataforma de simulación `ns-3`. En primer lugar, cabe destacar el trabajo realizado por Chihani *et al.* [Chih11], responsables de la primera implementación de *MPTCP* en la arquitectura del simulador, concretamente para su versión `ns-3.6`. Describen el funcionamiento de los cuatro algoritmos para el control de congestión básicos del protocolo (en [Raic09] se estudian con un mayor nivel de detalle) y comparan el rendimiento de dos algoritmos de reordenación, haciendo uso de una transmisión *FTP* entre un cliente y un servidor conectados a través de dos enlaces cableados punto a punto, analizando el comportamiento de las ventanas de congestión asociadas a cada uno de los subflujos. Esta implementación, que dejó de actualizarse en Mayo de 2011, tiene una serie de carencias, tal y como describe Kheirkah *et al.* [Khei14]. Por ejemplo, tiene la limitación de que solamente un cliente podrá conectarse a un servidor *MPTCP*, impidiendo el acceso a nuevas peticiones. Además, no permite la coexistencia en un mismo nodo de instancias *TCP* y *MPTCP*, lo que imposibilita llevar a cabo estudios para evaluar la interacción entre ambos protocolos. Por lo tanto, Kheirkah *et al.* deciden desarrollar una nueva aproximación al protocolo *MPTCP*, utilizando como referencia el *RFC* 6824, el control de los subflujos del módulo el *kernel* de *Linux*, así como los mecanismos de recuperación de errores definidos en *TCP NewReno*. A pesar de que el trabajo aún no está concluido, aparece listado como la implementación oficial del protocolo en futuras versiones del simulador. Como tercera opción, la herramienta *Direct Code Execution (DCE)* [DCE] permite la ejecución directa de implementaciones del espacio de usuario o del *kernel* dentro del simulador, pudiendo combinar las ventajas asociadas a la simulación, sin el coste de tener que desarrollar el correspondiente módulo *MPTCP* [Arza14], aprovechando la versión del *kernel* existente. La parte negativa en este caso es la falta de libertad a la hora de realizar modificaciones o añadir funcionalidades nuevas, ya que implicarían la recompilación del *kernel*.

Por último, y como prueba más tangible de la relevancia del protocolo *MPTCP*, resulta interesante analizar algunas de las implementaciones prácticas que se pueden encontrar en el mercado hoy en día. El ejemplo más destacado sería el asistente de voz de Apple, Siri [MPiOS7], aunque el tráfico no se transmite de manera simultánea por las interfaces *IEEE 802.11* y celular, sino que se utiliza la última como respaldo en caso de que falle la primera (de un modo similar a como actuaría el protocolo *SCTP*). Otro ejemplo menos popular sería el de la empresa *Multipath Networks* [MPNet], la cual, utilizando un *router* inicialmente financiado a través de una campaña de financiación pública [MPRout], permite combinar varias tecnologías (cableadas, celulares, *WLAN*, etc.) en una única conexión, garantizando un rendimiento y robustez muy superiores a los observados en conexiones tradicionales. Además, ya se ha mencionado la existencia de una versión casi definitiva del protocolo en el *kernel* de *Linux* [MPLin], cuyo código fuente se encuentra disponible en [MPKern], permitiendo su incorporación a otros sistemas que comparten el mismo *kernel* (e.g. *Mac OS X*, *Android*, etc.).

3.2 El protocolo *MPTCP*

Uno de los aspectos que en mayor medida limita las comunicaciones basadas en *TCP/IP* es que están restringidas a un único flujo por conexión en un instante dado, a pesar de que lo más habitual es que exista más de un camino posible entre cualquier par de nodos. Por contra, el protocolo *MPTCP* [Ford13], que se postula como una modificación de *TCP* [Post81b], propone una serie de extensiones que permiten dividir una sesión de nivel de transporte en múltiples *subconexiones*, de manera que la información viaje a través de diferentes rutas de manera simultánea. Gracias a esta estrategia multi-camino, se incrementa el rendimiento global del sistema (el *throughput* agregado podría ser notablemente mayor que si la información fluyese únicamente por una ruta), produciéndose además una mejora de la robustez de la comunicación (si una de las rutas presentase peores condiciones, los mecanismos de control de flujo y congestión desviarían la carga hacia otras alternativas con mejores prestaciones).

Tabla 3.1: *RFCs* e *Internet Drafts* propuestos por el *IETF*

Estado	**Título**	**Fecha**
[Ford13]	*TCP Extensions for Multipath Operation with Multiple Addresses*	Enero *2013*
[Ford11]	*Architectural Guidelines for Multipath TCP Development*	Marzo *2011*
[Raic11]	*Coupled Congestion Control for Multipath Transport Protocols*	Octubre *2011*
[Scha13]	*Multipath TCP (MPTCP) Application Interface Considerations*	Marzo *2013*
[Bagn11]	*Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses*	Marzo *2011*
[Hamp11]	*Enhancements to Improve the Applicability of Multipath TCP to Wireless Access Networks*	Diciembre *2011*
[Khal14a]	*Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP*	Julio *2014*
[Bona14]	*Experience with Multipath TCP*	Septiembre *2014*
[Khal14b]	*Performance issues with MPTCP*	Septiembre *2014*

La importancia que ha adquirido el protocolo durante estos últimos años se refleja en la existencia de una serie de especificaciones o *RFCs*, que establecen las principales líneas de actuación a seguir para la implementación del protocolo, tomando como base un documento inicial [Ford13], en el que se enumeran los cambios requeridos para dotar a *TCP* de capacidad *multi-path*. Así, describe aquellos mecanismos encargados de la señalización y el establecimiento de múltiples subflujos (primitivas de servicio), la gestión de los mismos, el reensamblado de los datos en recepción o la finalización de las sesiones. Sin embargo, como puede deducirse a la vista de la Tabla 3.1, han ido apareciendo paulatinamente otras

especificaciones que complementan la descripción del protocolo, definiendo con un mayor nivel de detalle aspectos como su arquitectura [Ford11], los mecanismos de control de la congestión [Raic11; Khal14a], la interacción con las aplicaciones [Scha13; Bona14] u otros puntos, como su aplicabilidad en redes de acceso inalámbricas [Hamp11], o los posibles problemas que puedan producirse durante las transmisiones [Bagn11; Khal14b].

Así, una de las tareas más críticas del protocolo *MPTCP* será la de distribuir la información a través de las distintas "subconexiones", que hacen uso de caminos potencialmente disjuntos, asociados a medios físicos que pueden a su vez pertenecer a diferentes tecnologías. En primera instancia, el protocolo ha sido diseñado para trabajar en esquemas multi-camino extremo a extremo, donde (al menos) uno de los equipos terminales debería disponer de más de una dirección *IP*. De hecho, el protocolo identifica la presencia de múltiples caminos en función de la existencia de múltiples direcciones en los nodos. Por lo tanto, para que sea posible establecer una conexión *MPTCP* entre dos terminales, ambos (o al menos uno de ellos) deberán contar con más de un interfaz de red.

Tabla 3.2: Clasificación de los diferentes subtipos de la cabecera *MPTCP*

Valor	Símbolo	Nombre
0x0	MP_CAPABLE	Compatible con el protocolo *MPTCP*
0x1	MP_JOIN	Unirse a la conexión *MPTCP*
0x2	DSS	Relacionado con el secuenciamiento *DSS*
0x3	ADD_ADDR	Añadir dirección a la conexión
0x4	REMOVE_ADDR	Eliminar dirección de la conexión
0x5	MP_PRIO	Cambio de prioridad en el subflujo
0x6	MP_FAIL	Fallo en la transmisión
0x7	MP_FASTCLOSE	Conexión abortada

Otra de las características fundamentales de *MPTCP* es el requisito de mantener la compatibilidad hacia atrás con su protocolo raíz, *TCP*, factor que facilitaría la migración, ya que al reutilizar la infraestructura existente el coste asociado sería mucho menor. De este modo, para cualquier aplicación no compatible, una conexión *MPTCP* sería, a todos los efectos, indistinguible de una realizada con *TCP*, con el establecimiento de una única sesión a nivel de transporte, a través de la creación de un único *socket*. Para ello, toda la señalización derivada del protocolo *MPTCP* irá encapsulada dentro del campo *"Opciones"* de la cabecera *TCP*, habiendo reservado la *Internet Assigned Numbers Authority (IANA)* el valor 30 para referirse a los mensajes pertenecientes a *MPTCP*. Por su parte, los diferentes tipos de mensajes *MPTCP* irán determinados por un *subtipo*, utilizando para su codificación un esquema *Tipo-Longitud-Valor (TLV)*. En la Tabla 3.2 se recogen todos los subtipos actualmente reservados; quedando abierta la posibilidad de incorporar nuevos elementos en un futuro, tal y como marcan las pautas definidas en [Nart08].

La Figura 3.2 muestra la arquitectura del protocolo, que actúa en el nivel de transporte, llevando a cabo todas las tareas tradicionalmente realizadas por *TCP*. Puede apreciarse

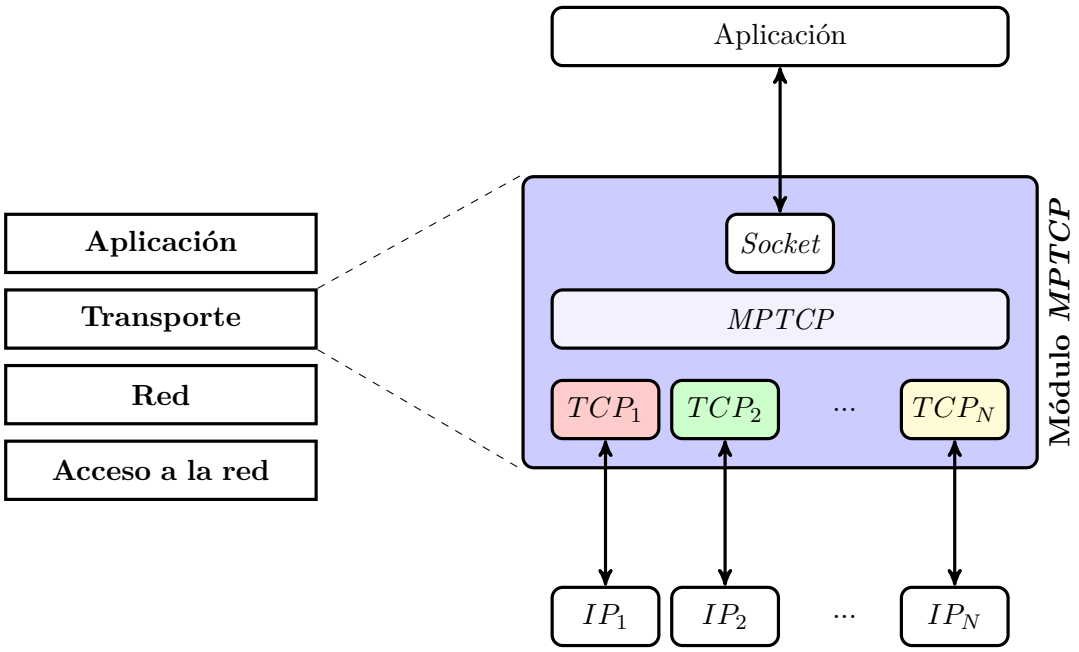


Figura 3.2: Arquitectura del protocolo *MPTCP*

como *MPTCP* hace uso de múltiples sesiones *TCP* estándar, denominadas *subflujos*, manteniendo así la compatibilidad con la red, pues aparenta un comportamiento equivalente al de cualquier flujo *TCP* tradicional (con lo que se evitan problemas al atravesar *firewalls*, *NATs* u otros *middleboxes*, en los que existe la posibilidad de que el intercambio de mensajes *MPTCP* sea alterado [Paas14]).

Asimismo, la operación del protocolo se ha dividido en dos partes con funcionalidades claramente diferenciadas:

1. El subnivel superior, también conocido como *capa semántica*, será el encargado de realizar las funciones *orientadas a la aplicación*: inicialización/finalización de las sesiones, establecimiento de los subflujos, detección y uso de múltiples caminos, etc.

2. La subcapa inferior, enfocada a las tareas *orientadas hacia la red*, se encargará de gestionar cada uno de los subflujos creados durante la fase de inicialización de la conexión. Para ello, se dividirá la operación de control en tantas entidades como subflujos haya en la conexión.

En el nivel de red, cada uno de los subflujos estará asociado a un interfaz *IP* diferente. Como ya se ha comentado, esta arquitectura permite una operación transparente, tanto hacia las capa superiores como a las inferiores, ofreciendo un comportamiento equivalente al de una conexión *TCP*.

3.2.1. Funciones específicas de la capa semántica

A continuación se describen las principales funcionalidades llevadas a cabo por el subnivel superior, orientadas a la interacción con las aplicaciones.

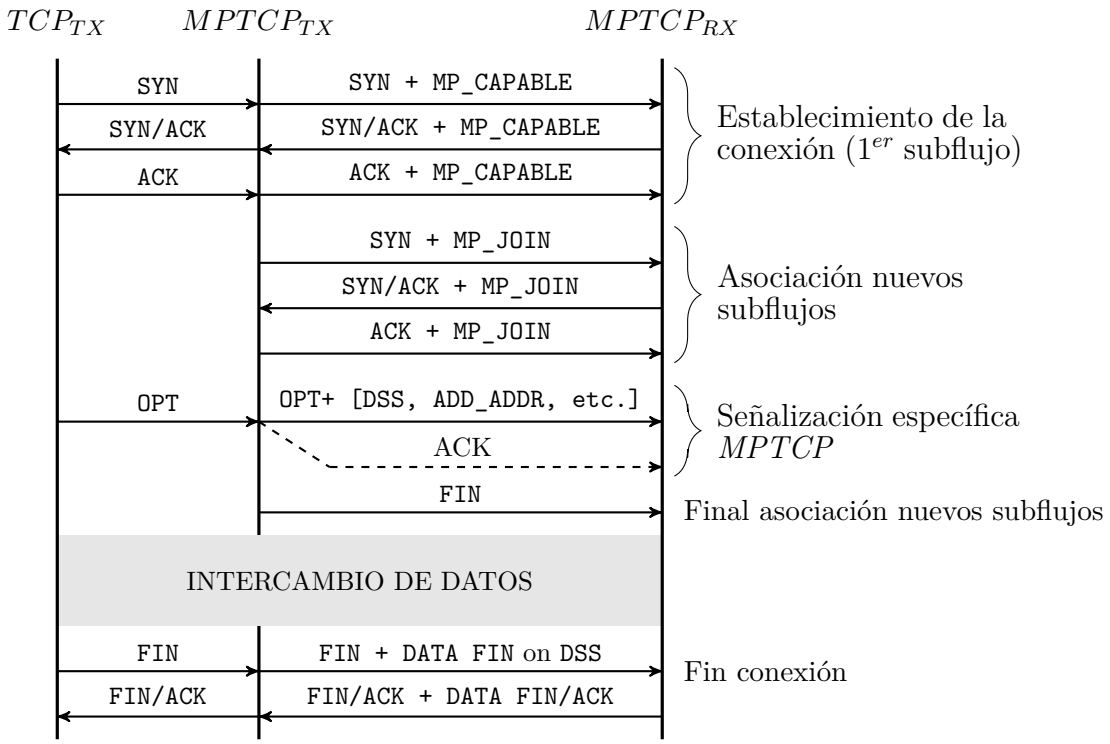


Figura 3.3: Primitivas para el establecimiento/finalización de una conexión

Establecimiento/finalización de la conexión. Con el objetivo de respetar las primitivas de servicio que definen las etapas de una conexión *TCP*, la Figura 3.3 muestra la correspondencia definida en el *RFC* 6824 para establecer sesiones *MPTCP* en equipos compatibles. En una primera etapa, similar al saludo a tres vías de *TCP*, los nodos origen y destino deberán notificar su intención de utilizar el protocolo (a través del subtipo `MP_CAPABLE`). Así, la primera de las conexiones se dará por establecida, permaneciendo a la espera de la creación de nuevos subflujos. Posteriormente, para cada una de las direcciones adicionales que se desee añadir a la conexión (formando un nuevo subflujo) se deberá repetir el intercambio de mensajes `SYN + SYN/ACK + ACK`, activando el subtipo `MP_JOIN`. Además, existe la posibilidad de añadir nuevas opciones a la conexión (e.g. establecer prioridades entre los subflujos, añadir/eliminar direcciones, etc.), para lo que se definen nuevos tipos de mensaje. Por último, se especifirá de manera explícita el instante en el que la fase de inicialización de la conexión se da por finalizada, dando paso a la fase de intercambio de información. Finalmente, cuando el transmisor no tenga más datos que transmitir, se procederá a finalizar la conexión, utilizando para ello el subtipo `DATA_FIN`. Junto a esta opción, se mantiene la posibilidad de utilizar las primitivas `FIN` (propias de *TCP*), que

afectarán exclusivamente al subflujo por el que se ha enviado el mensaje. Esto confiere un elevado grado de libertad a la hora de elegir los caminos que se van a utilizar en cada instante.

Números de secuencia. Como principio básico de su operación, los datos generados por los terminales en una sesión *MPTCP* pueden ser enviados indistintamente por cualquiera de los subflujos que componen la conexión de nivel de transporte. Para ello, en primer lugar, cada subflujo por separado equivale a una sesión *TCP* tradicional e independiente. Por otra parte, el subnivel superior de *MPTCP* mantiene un espacio de secuenciamiento adicional al empleado por *TCP*, recogido dentro del subtipo *Data Sequence Signal (DSS)*, que se representa en la Figura 3.4. Como puede apreciarse, el esquema se apoya principalmente en dos variables: el número de secuencia de datos (Data Sequence Number - DSN), mapeado a través del campo *DSN_MAP*, y el del reconocimiento, utilizando el campo *DATA_ACK*. Cuando un nodo envía un segmento *TCP* sobre un subflujo, indica, utilizando la opción *DSN_MAP*, la correspondencia entre el número de secuencia de 32 bits utilizado por el subflujo (*TCP*) y el número de secuencia de datos o *Data Sequence Number (DSN)* (nivel *MPTCP*). Como puede verse en la figura, este valor puede tener una longitud de 4 u 8 *bits*. De esta manera, la entidad receptora podrá ordenar los paquetes recibidos, y enviarlos al nivel de aplicación. Es preciso destacar también la presencia en la cabecera del número de secuencia del subflujo o *Subflow Sequence Number (SSN)*, que utiliza un secuenciamiento relativo a cada uno de los subflujos, mientras que el valor del *DSN* es absoluto. Además, en el protocolo *MPTCP* los reconocimientos de los segmentos recibidos se realizan en dos niveles. Primeramente, el reconocimiento *TCP* confirma la correcta recepción de los segmentos en cada uno de los subflujos; en un nivel superior, el campo *DATA_ACK* es utilizado por el receptor para confirmar (de manera acumulativa) a nivel *DSN*. Cuando se pierde un segmento, el receptor detecta un "hueco" en la secuencia y el esquema de retransmisión clásico de *TCP* actúa tal y como se describió en el Capítulo 2, mediante la recepción de *triples ACKs duplicados* o tras la expiración del *RTO*. En este momento, cuando uno de los subflujos falla, la entidad *MPTCP* detecta el error y retransmite los datos no confirmados por otro de los subflujos activos.

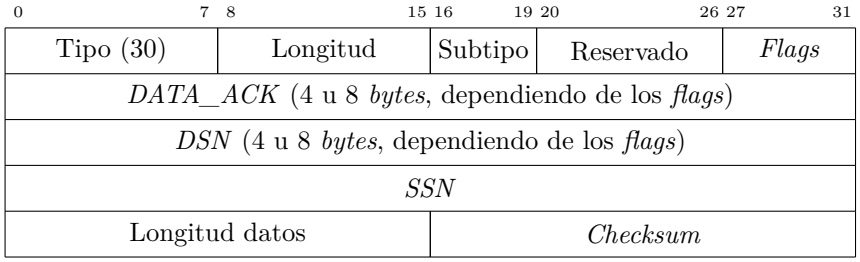


Figura 3.4: Formato del campo opciones de la cabecera *TCP* correspondiente al secuenciamiento *DSS*

Como se ha podido ver, existe la posibilidad de acortar la longitud del número de secuencia (tanto para los datos como para los reconocimientos). Junto con esta opción, se

habilita un conjunto de *flags* que, cuando se activan, definen el contenido y el tamaño del resto de campos del mensaje, tal y como se describe a continuación.

- `A` = Campo `DATA_ACK` presente.
- `a` = El campo `DATA_ACK` ocupa un total de 8 *bytes*.
- `M` = Indica la presencia de los campos `DSN`, `SSN`, la longitud de los datos y un *checksum*, si su utilización ha sido acordada durante el establecimiento de la conexión (dentro del mensaje con subtipo `MP_CAPABLE`).
- `m` = El campo `DSN` ocupa 8 bytes.
- `F` = Este flag (`DATA_FIN`) indica que el transmisor ya no tiene más información para ser enviada, por lo que notifica al nodo destino su intención de finalizar la conexión.

Como puede deducirse, los *flags* "`a`" y "`m`" sólo tienen sentido si "`A`" y "`M`" están activadas. En total, el tamaño máximo de la cabecera *MPTCP*, con todos los *flags* activados, será de 28 *bytes*.

Reordenación de los paquetes. El hecho de dividir una conexión en varios subflujos que, a su vez, recorren diferentes caminos (pudiendo utilizar tecnologías diferentes con características dispares), tiene como consecuencia la probable llegada desordenada de los segmentos de información al destino. Por ello es imprescindible disponer de esquemas que permitan recuperar la información de una forma eficiente, introduciendo la menor cantidad posible de retransmisiones innecesarias. Aunque existen numerosas alternativas, en el módulo desarrollado originalmente por Chihani et al. [Chih11] se pueden encontrar dos soluciones, que almacenan el estado de la conexión (e.g. el tamaño de la ventana de congestión, el *ssthreshold*[3], etc.) en el instante anterior a una retransmisión.

En el primero de los casos se define el esquema de reordenación *Eifel* [Ludw03], cuya operación se resume en la Figura 3.5. El transmisor introduce la opción *Timestamp* de *TCP* para marcar el instante temporal en el que cada segmento abandona el *buffer* de transmisión de la capa de transporte. En el otro extremo, el receptor inserta el *timestamp* al recibir el segmento, incluyéndolo en el *ACK* que envía a la fuente. En caso de pérdida, el transmisor mantiene los valores de la ventana de congestión y el *ssthreshold* y, de manera inmediata, retransmitirá el segmento solicitado, incluyendo el nuevo valor del *timestamp* (que también almacena). En el momento en el que el transmisor recibe un *ACK* que confirme la recepción de un segmento retransmitido, compara el valor del *timestamp* con el que tiene almacenado (correspondiente al instante en el que se produjo la retransmisión). Si se da el caso de que el *timestamp* del *ACK* sea anterior al valor almacenado, se asume que se ha producido una retransmisión espuria o innecesaria (se está confirmando el segmento original y no la retransmisión), restaurando los valores de ventana de congestión y *ssthreshold* que se habían guardado previamente.

[3]Se define como el umbral que delimita las fases de *slow start* y *congestion avoidance*.

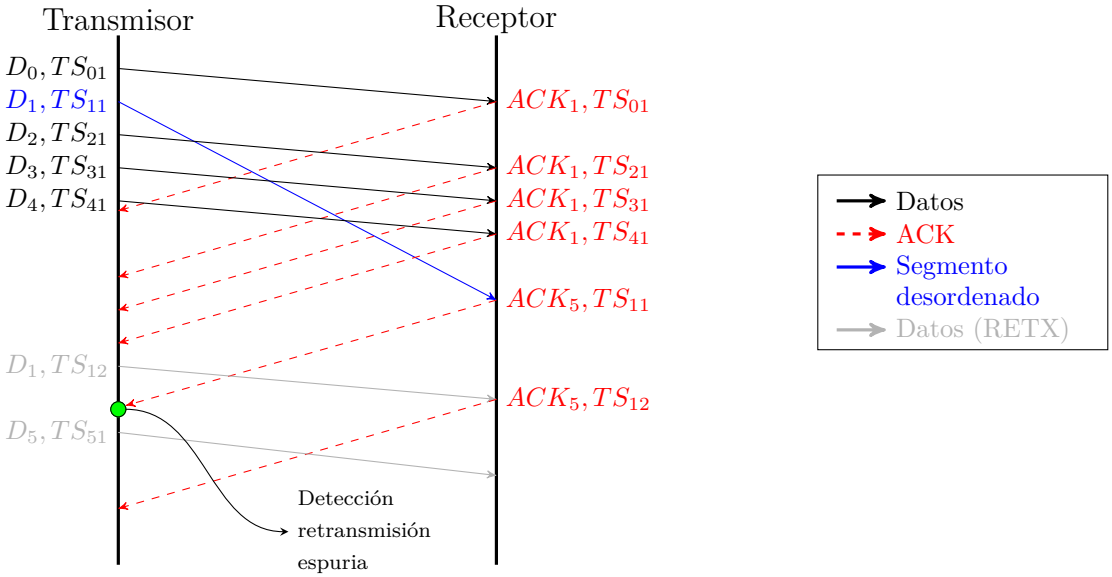


Figura 3.5: Esquema de reordenación de paquetes *Eifel*

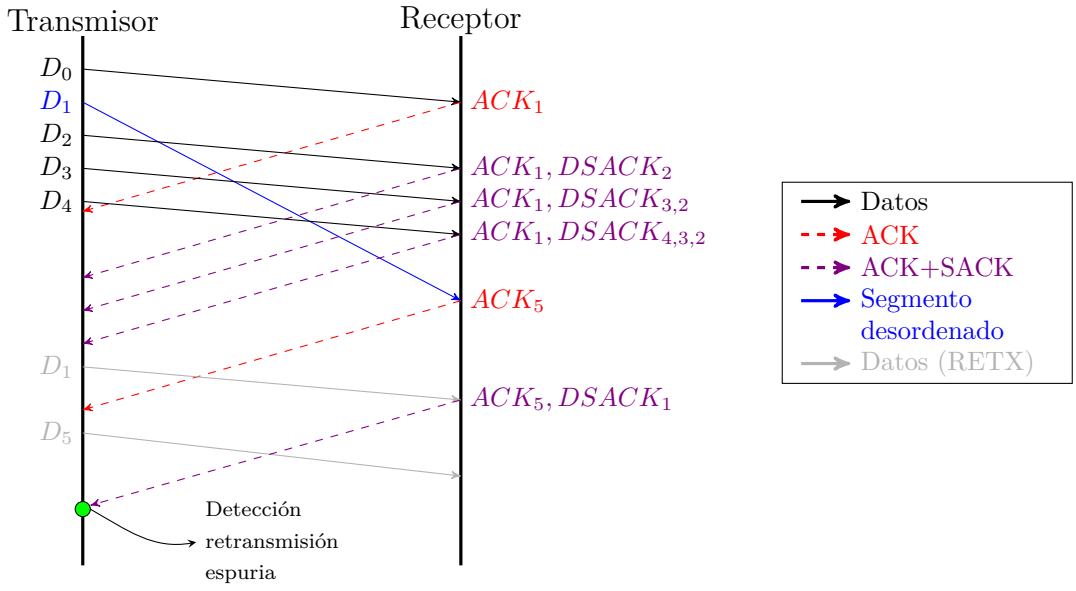


Figura 3.6: Esquema de reordenación de paquetes *DSACK*

La otra de las opciones implementadas se basa en la utilización del mecanismo *Duplicated Selective ACKnowledgement (DSACK)* [Floy00], extensión del algoritmo Selective Acknowledgement (SACK), que permite el reconocimiento y retransmisión selectiva de aquellos fragmentos de información necesarios, evitando la retransmisión de bloques enteros de datos (como sería el caso del esquema *Go-Back-N* tradicional de *TCP*). La Figura 3.6 refleja su operación, observándose la reacción de la estación receptora tras la pérdida de un segmento; se envía hacia atrás un *ACK* duplicado que incluye (en el campo *SACK*), los números de secuencia posteriores que han sido recibidos correctamente. Tras la recepción de tres *ACKs duplicados* en el nodo transmisor, se procede a retransmitir el segmento perdido y se guarda el valor de la ventana de congestión actual, cambiando al estado *congestion avoidance*. Posteriormente, si el receptor recibe dos veces el segmento retransmitido, se generaría una doble confirmación, por la que el nodo fuente detectará que se ha producido una retransmisión innecesaria, pasando a un estado *slow start* propio del esquema *DSACK*, en el que se restaura el valor de la ventana de congestión previamente almacenado, en lugar del valor propio de la fase *slow start* tradicional.

Distribución de paquetes (*scheduling*). Finalmente, se necesita un mecanismo que distribuya los paquetes entre los diferentes subflujos *MPTCP*. La solución utilizada en la versión original del módulo [Chih11] se basa en un sencillo esquema *Round Robin (RR)*, que reparte equitativamente los segmentos de datos a través de las subconexiones habilitadas en la conexión. Así, el subnivel superior enviará alternativamente un paquete a cada uno de los subflujos, repitiéndose el proceso mientras su *buffer* de transmisión disponga de paquetes esperando a ser enviados. La Figura 3.7 ilustra el modo de operación del mecanismo *RR* implementado en el subnivel superior de la capa *MPTCP*.

3.2.2. Mecanismos de control de la congestión

Una vez descritas las funciones relacionadas con la conectividad y su interacción con el nivel de aplicación, se presentan seguidamente los mecanismos que controlan el tráfico a través de los diferentes subflujos habilitados durante el establecimiento de la conexión. Para ello, un interfaz situado entre el nivel *MPTCP* y cada una de las instancias *TCP* asociadas a los subflujos deberá gestionar el paso de información entre ambos, distribuyendo los recursos entre las diferentes conexiones establecidas. En este punto es importante seleccionar aquellos mecanismos de control de congestión que sean capaces de asegurar el correcto cumplimiento de tres objetivos clave, planteados durante la etapa de diseño del protocolo, que se enumeran a continuación:

1. ***Aumentar el throughput***: El rendimiento obtenido a través de un esquema MPTCP debe ser, como mínimo, no inferior al ofrecido por *TCP* cuando emplea la mejor ruta disponible.
2. ***No perjudicar***: Un subflujo *MPTCP* no debe consumir más recursos que los que *TCP* necesitaría utilizando únicamente uno de los caminos.

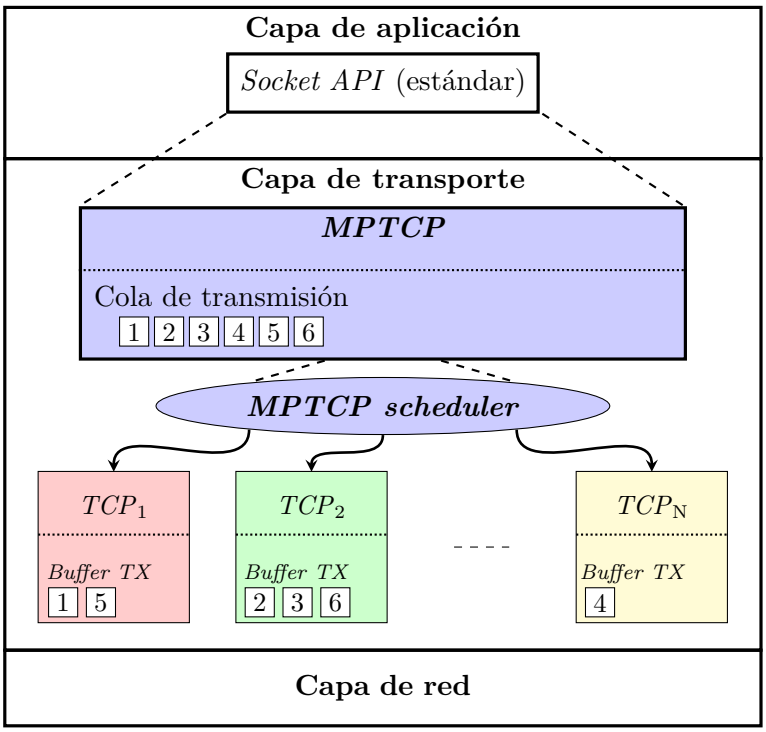


Figura 3.7: Distribución de los paquetes entre los diferentes subflujos

3. ***Balancear la congestión***: Ante una situación de congestión, los algoritmos de control de flujo deberán dedicar la mayor cantidad de recursos posible a aquellos subflujos que se encuentren menos congestionados (respetando los dos primeros objetivos).

Para poder cumplir con estos objetivos, es necesario llevar a cabo una correcta gestión de los recursos; *resource pooling*, como se define en [Wisc08], donde se propone gestionar todos los recursos disponibles en la red como si se trataran de un único recurso global. En concreto, el criterio a seguir es el de dotar a cada uno de los subflujos una ventana de congestión diferente. No obstante, para lograr un correcto *resource pooling*, es necesario que la evolución de las ventanas de congestión no sea completamente independiente, presentando un cierto factor de acoplamiento.

A continuación, se enumeran los cuatro algoritmos para el control de la congestión soportados por el esquema de *MPTCP* implementado en el simulador. Es preciso destacar que la formulación empleada solamente se aplica en la fase creciente del estado *congestion avoidance*, especificando el factor de crecimiento de la ventana de congestión tras la recepción de un *ACK*. El resto de elementos (los algoritmos *slow start*, *fast retransmit*, *fast recovery*), así como la disminución multiplicativa correspondiente al estado *congestion avoidance* se mantienen inalterados con respecto a la especificación original [Allm09].

De aquí en adelante, se utilizará la notación w_i para referirse al tamaño de la ventana de congestión del subflujo i-ésimo, definiendo w como el tamaño acumulado de todas las ventanas: $w = \sum_i w_i$, donde $i = 1, \ldots, N$, siendo N el número de subflujos.

- ***Uncoupled subflows***: Como primera opción se incorpora un algoritmo básico, donde la ventana de congestión de cada subflujo se controla de manera completamente aislada, comportándose como conexiones *TCP* independientes, como se expresa en la Ecuación (3.1). En esta versión la entidad *MPTCP* no tendría que efectuar ningún tipo de control sobre los subflujos, que actúan por tanto de manera autónoma.

$$\widetilde{w}_i = w_i + \frac{1}{w_i} \tag{3.1}$$

- ***Fully coupled***: A través de este algoritmo, en el que el comportamiento de las ventanas de congestión está estrechamente ligado entre ellas, cuando uno de los subflujos experimenta una tasa de error ligeramente superior al otro, el protocolo *MPTCP* interpreta una situación de congestión y trata de aliviarla asignando una carga mayor al nodo menos saturado (**Objetivo 3**). Su principal problema surge cuando se pierden tramas en un subflujo de manera continuada, llegando a un valor de ventana mínimo ($w_i \approx 1$), situación que se podría mantener durante un intervalo de tiempo. Para volver a una situación estable, el otro subflujo deberá sufrir un número mucho mayor de pérdidas hasta que las tasas vuelvan a nivelarse. A este efecto, que puede resultar nocivo, se le conoce como "*flappiness*". La expresión que modela el crecimiento de la ventana se recoge en la Ecuación (3.2).

$$\widetilde{w}_i = w_i + \frac{1}{w} \tag{3.2}$$

- ***Linked increases***: Para reducir el efecto "flappiness" mencionado en el procedimiento anterior, Raiciu *et al.* proponen en [Raic09] una solución que, a costa de sacrificar la eficiencia en la gestión de recursos, evita el reparto poco equitativo entre los subflujos. Como se aprecia en la Ecuación (3.3), se introduce un nuevo parámetro nuevo, α, que será descrito posteriormente.

$$\widetilde{w}_i = w_i + \frac{\alpha}{w} \tag{3.3}$$

- ***RTT compensator***: El último de los algoritmos se trata de una ligera modificación del anterior, pensado para ser utilizado en escenarios cuyos caminos estén caracterizados por valores de *RTT* muy dispares (e.g. diferentes tecnologías, cuellos de botella en algún enlace concreto, mayor carga de tráfico, etc.). Al igual que para el mecanis-

mo *Linked increases*, la tasa de crecimiento de la ventana de congestión está también controlada por el parámetro α, tal y como se ve en la Ecuación (3.4).

$$\widetilde{w}_i = w_i + min\left(\frac{\alpha}{w}, \frac{1}{w_i}\right) \tag{3.4}$$

Como se ha visto en los dos últimos algoritmos, el factor de crecimiento de la ventana incorpora un nuevo factor, α, conocido como *factor de agresividad*, y que responde a la expresión mostrada en la Ecuación (3.5).

$$\alpha = w \cdot \frac{\text{máx}_i\left(\frac{w_i}{RTT_i^2}\right)}{\left(\sum_{i=1}^{N}\frac{w_i}{RTT_i}\right)^2} \tag{3.5}$$

A través de este parámetro se busca mejorar el ancho de banda agregado del sistema, de tal manera que se cumplan los **Objetivos 1** y **2**. No obstante, no se consigue satisfacer completamente el **Objetivo 3**, ya que no permite reducir la carga de tráfico en aquellos subflujos que se encuentren más congestionados.

Hay que destacar que estudios posteriores han desvelado que el rendimiento ofrecido por el protocolo *MPTCP* está lejos de ser óptimo, identificando una serie de problemas cuando comparte recursos con otros protocolos. A modo de ejemplo, Khalili *et al.* describen en [Khal13] dos principales problemas: *(1)* El reparto de recursos en los enlaces cuello de botella en los que coincida tráfico *MPTCP* y *TCP* perjudicará en gran medida al segundo grupo. *(2)* Los usuarios del protocolo *MPTCP* pueden resultar muy agresivos para los que utilizan *TCP*.

Estas limitaciones están causadas por el algoritmo de control de congestión (en la literatura se señala exclusivamente al algoritmo *linked increases*, que es además el único que se encuentra en las especificaciones oficiales del protocolo [Raic11]), donde se redirigiría una excesiva carga de tráfico hacia las rutas más congestionadas. Como solución, se ha propuesto una modificación de este algoritmo, conocida como *OLIA* [Khal14a], que trata de corregir estos problemas, ofreciendo una solución óptima a la gestión de los recursos, cumpliendo los tres objetivos en cualquier circunstancia.

Gracias a este algoritmo, cuya especificación aún se encuentra en estado de *draft*, el tráfico se redigirá automáticamente desde aquellos enlaces que presenten una mayor ocupación ($\approx$ cuellos de botella) a otros que dispongan de la capacidad suficiente para dar soporte a esta carga adicional, sin llegar a una situación de congestión. De este modo, la coexistencia entre flujos *TCP* y *MPTCP* se verá favorecida enormemente.

3.3 Algoritmos de encaminamiento multi-camino

El objetivo que persiguen los diferentes algoritmos de encaminamiento multi-camino es el de encontrar una configuración óptima de rutas disjuntas, de tal manera que aseguren las mejores prestaciones a la hora de transportar tráfico de manera simultánea a través de los múltiples subflujos, en un escenario basado en una red mallada inalámbrica.

Para describir el comportamiento de cada uno de los algoritmos utilizados (LD, ND y ZD) se empleará la notación característica de la teoría de grafos, tal y como se resume a continuación. Sea $\mathcal{G}(V, E)$ el grafo que representa el escenario sobre el que se requiere obtener el conjunto de rutas entre los nodos origen S y destino D. El conjunto V representa al grupo de nodos desplegados sobre el escenario, mientras que E es el conjunto de enlaces entre ellos. Se considera que existe un enlace entre dos nodos si la distancia entre ellos es inferior al rango de transmisión de los dispositivos, que se asume es el mismo para todos los nodos.

El primer paso para obtener el conjunto de caminos es común para todos los mecanismos: utilizar el algoritmo de *Dijkstra* [Dijk59] para determinar el camino con menor número de saltos desde S a D. Si existe tal ruta en el grafo $\mathcal{G}$, será incluida en el conjunto correspondiente (P_L, P_N o P_Z, para los algoritmos LD, ND o ZD, respectivamente), procediendo posteriormente a actualizar el grafo según un criterio que determina el algoritmo empleado. A continuación se muestra el proceso seguido por cada una de las soluciones analizadas:

- ***Link Disjoint (LD)***. Después de obtener el camino más corto (Dijkstra), se eliminarán del grafo $\mathcal{G}$ todos los *enlaces* que conforman la ruta obtenida, dando lugar a un nuevo grafo $\mathcal{G}'(V, E^L)$. El proceso se repetirá mientras haya rutas entre S y D (que volverán a ser calculadas a través del algoritmo de Dijkstra), que se irán añadiendo al conjunto P_L que, al finalizar el procedimiento, devolverá el conjunto de caminos *link-disjoint* del grafo original.

Algoritmo 3.1: Algoritmo para determinar el conjunto de rutas *link-disjoint*

Entrada: $\mathcal{G}(V, E)$ y nodos origen (S) y destino (D)
Variables auxiliares: Grafo $\mathcal{G}^L(V, E^L)$, conjunto de rutas P_L
Inicialización: $\mathcal{G}^L(V, E^L) \leftarrow \mathcal{G}(V, E)$, $P_L \leftarrow \emptyset$
while $\exists$ *al menos un camino entre* $S \rightarrow D$ *en* $\mathcal{G}^L$ **do**
  $p \leftarrow$ Ruta (*path*) más corta $S - D$ en $\mathcal{G}^L$
  $P_L \leftarrow P_L \bigcup \{p\}$
  $\forall_{\text{Enlaces: } e \in p} \; \mathcal{G}^L(V, E^L - \{e\})$
Salida: Conjunto de rutas de enlaces disjuntos P_L

- ***Node Disjoint (ND)***. En este caso, el criterio de modificación del grafo es el borrado de todos aquellos *nodos* que han pertenecido a la ruta seleccionada. Por lo

tanto, tras la conclusión de un ciclo, se obtendrá una nueva ruta a añadir al conjunto *node-disjoint* P_N y un grafo modificado $\mathcal{G}'(V^N, E^N)$. De nuevo, el proceso se repetirá mientras exista una ruta entre S y D, obteniendo al final el conjunto P_N.

Algoritmo 3.2: Algoritmo para determinar el conjunto de rutas *node-disjoint*

Entrada: $\mathcal{G}(V, E)$ y nodos origen (S) y destino (D)
Variables auxiliares: Grafo $\mathcal{G}^N(V^N, E^N)$, conjunto de rutas P_N
Inicialización: $\mathcal{G}^N(V^N, E^N) \leftarrow \mathcal{G}(V, E)$, $P_N \leftarrow \emptyset$
while $\exists$ *al menos un camino entre* $S \rightarrow D$ *en* $\mathcal{G}^N$ **do**
  $p \leftarrow$ Ruta (*path*) más corta $S - D$ en $\mathcal{G}^N$
  $P_N \leftarrow P_N \bigcup \{p\}$
  $\underset{\substack{\text{Vértices: } u \in p, u \neq S, D \\ \text{Enlaces: } e \in Adj - \{u\}}}{\forall} \mathcal{G}^N(V^N - \{u\}, E^N - \{e\})$
Salida: Conjunto de rutas de enlaces disjuntos P_N

- ***Zone Disjoint (ZD)***. Es la alternativa más restrictiva, ya que es la que más penaliza el grafo entre sucesivas iteraciones, eliminándose todos los *nodos* pertenecientes a la ruta calculada por Dijkstra, así como sus *vecinos adyacentes*, pasando del grafo original $\mathcal{G}$ al modificado $\mathcal{G}'(V^Z, E^Z)$. Al finalizar, el algoritmo devolverá el conjunto de rutas P_Z.

Algoritmo 3.3: Algoritmo para determinar el conjunto de rutas *zone-disjoint*

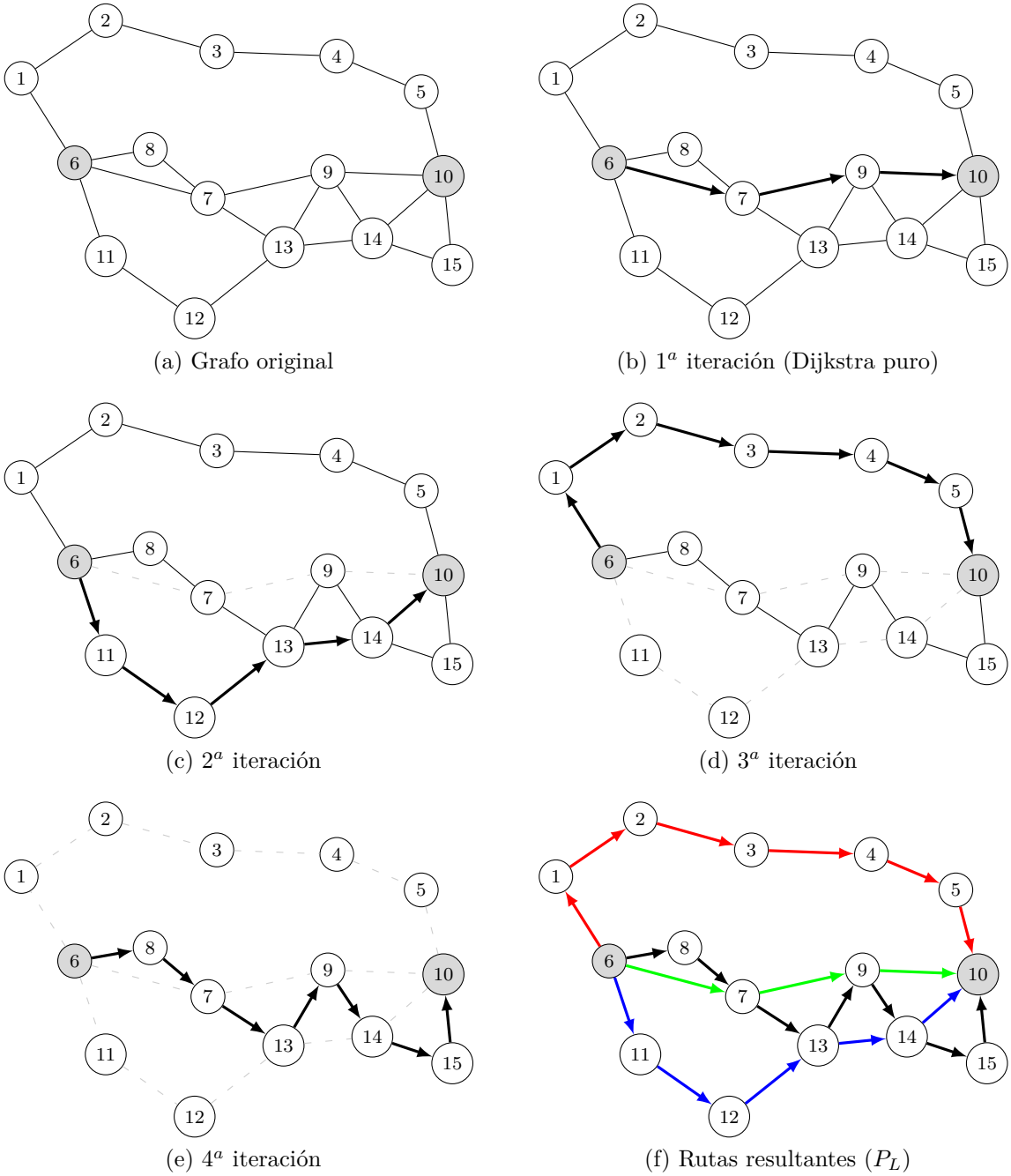
Entrada: $\mathcal{G}(V, E)$ y nodos origen (S) y destino (D)
Variables auxiliares: Grafo $\mathcal{G}^Z(V^Z, E^Z)$, conjunto de rutas P_Z
Inicialización: $\mathcal{G}^Z(V^Z, E^Z) \leftarrow \mathcal{G}(V, E)$, $P_Z \leftarrow \emptyset$
while $\exists$ *al menos un camino entre* $S \rightarrow D$ *en* $\mathcal{G}^Z$ **do**
  $p \leftarrow$ Ruta (*path*) más corta $S - D$ en $\mathcal{G}^Z$
  $P_Z \leftarrow P_Z \bigcup \{p\}$
  $\underset{\substack{\text{Vértices: } u \in p, u \neq S, D \\ \text{Enlaces: } e \in Adj - \{u\}}}{\forall} \mathcal{G}^Z(V^Z - \{u\}, E^Z - \{e\})$
  $\underset{\substack{u \in p, u \neq S, D \\ v \in Vecino(u), v \neq S, D \\ e' \in Adj - \{v\}}}{\forall} \mathcal{G}^Z(V^Z - \{v\}, E^Z - \{e'\})$
Salida: Conjunto de rutas de enlaces disjuntos P_Z

En este trabajo las tareas de elección de rutas se han llevado a cabo de manera ortogonal a la implementación del protocolo *MPTCP* (y la simulación correspondiente), empleando para ello un código propietario en `C++`; en primera instancia, se genera un despliegue aleatorio de nodos, dando lugar al grafo $\mathcal{G}(V, E)$ para, a continuación, obtener las diferentes rutas.

Dado que el objetivo es utilizar *MPTCP* para transmitir información a través de múltiples caminos, solamente se considerarán aceptables aquellos conjuntos de rutas (P_L, P_N o P_Z) que contengan más de un camino entre S y D, descartando los escenarios restantes.

Como ejemplo ilustrativo de la operación de cada uno de los tres algoritmos empleados, en las Figuras 3.8, 3.9 y 3.10 se representa el procedimiento seguido en un escenario formado por el despliegue aleatorio de 16 nodos, donde los nodos #6 y #10 hacen las veces de origen y destino, respectivamente. Como se ha mencionado anteriormente, la primera iteración será común en los tres casos, resolviendo el camino más corto a través de una llamada al algoritmo de Dijkstra con el grafo original $\mathcal{G}(V, E)$ como entrada, devolviendo la ruta $6 \rightarrow 7 \rightarrow 9 \rightarrow 10$. A partir de este punto es donde las diferentes aproximaciones comienzan a divergir, llevando a cabo los cambios pertinentes antes de buscar de nuevo la ruta óptima en el grafo modificado. En concreto, *LD* eliminará los enlaces que componen el camino inicial, *ND* lo hará con todos los nodos involucrados (y sus respectivos enlaces); por su parte, *ZD* seguirá el criterio más restrictivo de las tres alternativas y borrará el área de influencia de cada uno de los nodos presentes en la ruta más corta (es decir, nodos + vecinos + enlaces). Continuando con el proceso, se irán sucediendo las iteraciones, hasta que en el grafo resultante no exista un camino directo entre el origen (6) y el destino (10), momento en el que se dará por concluido el algoritmo. En este escenario en concreto se comprueba que el algoritmo *LD* devolverá un conjunto P_L con cuatro rutas diferentes, con longitudes entre los 3 y los 7 saltos. Por su parte, el conjunto P_N contará con un total de 3 rutas (de entre 3 y 6 saltos), mientras que el conjunto P_Z, generado por el algoritmo ZD, tendrá tan sólo dos caminos (de 3 y 6 saltos). Hay que tener en cuenta que la existencia de un mayor número de rutas no implicará necesariamente un mejor rendimiento, sino que deberán tenerse en cuenta otros factores, como el número de saltos de las rutas[4] o su área de interferencia para poder establecer un criterio de decisión adecuado. De hecho, existe en la literatura un conjunto de trabajos que defiende maximizar la diversidad espacial entre las rutas, minimizando los "dominios de colisión", lo que tendrá un impacto positivo en el rendimiento [Waha06; Lee01].

[4]En medios de transmisión basados en contienda, como el estándar *IEEE 802.11*, un mayor número de saltos perjudicará notablemente el rendimiento global del sistema, ya que la contención por el acceso al canal inalámbrico será mayor, así como la probabilidad de que se produzcan colisiones producidas por transmisiones simultáneas.



(a) Grafo original

(b) 1^a iteración (Dijkstra puro)

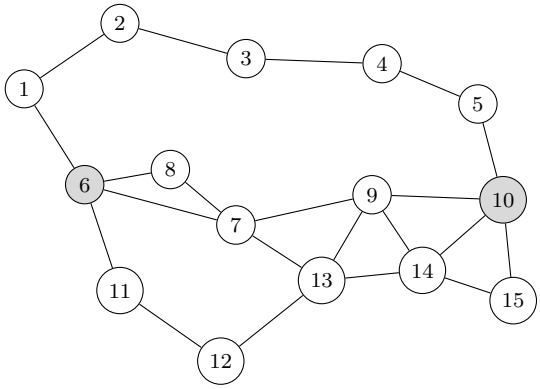
(c) 2^a iteración

(d) 3^a iteración

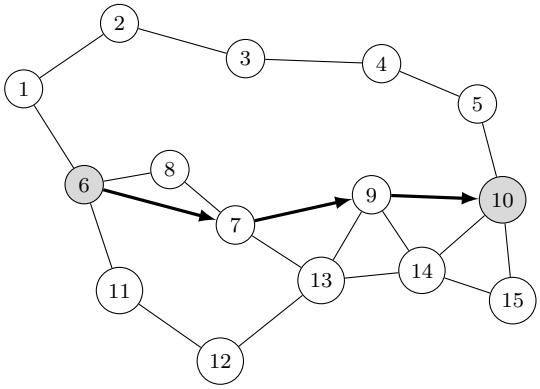
(e) 4^a iteración

(f) Rutas resultantes (P_L)

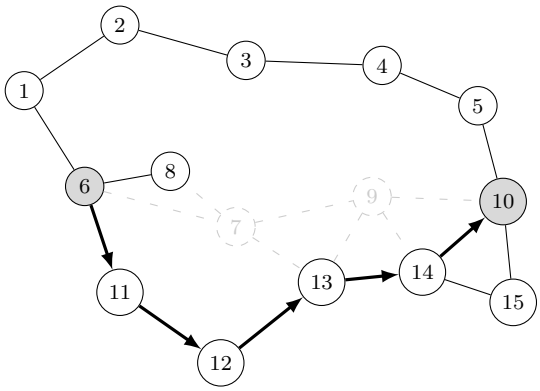
Figura 3.8: Ejemplo ilustrativo del algoritmo *LD*



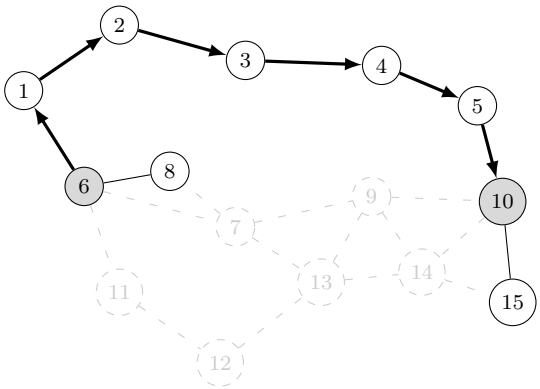
(a) Grafo original



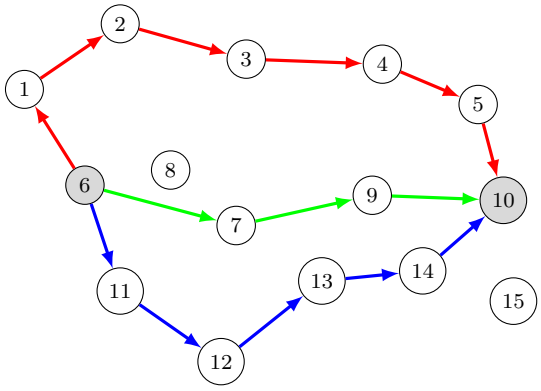
(b) 1^a iteración (Dijkstra puro)



(c) 2^a iteración



(d) 3^a iteración



(e) Rutas resultantes (P_N)

Figura 3.9: Ejemplo ilustrativo del algoritmo *ND*

3.4 Simulación y resultados

Con el fin de validar los beneficios de transmitir simultáneamente información a través de múltiples interfaces, en este apartado se llevará a cabo una profunda caracterización del rendimiento del protocolo *MPTCP* sobre redes inalámbricas multi-salto, comparándolo con *TCP*. Para ello, se han seguido tres etapas diferentes: en la primera de ellas, se utilizará un escenario sencillo (donde la obtención de las rutas entre los nodos origen y destino es trivial) para estudiar, con un mayor nivel de detalle, alguno de los mecanismos que definen el comportamiento del protocolo *MPTCP* (i.e. control de la congestión, capacidad de balancear la carga en condiciones de canal asimétricas). Posteriormente, y antes de proceder a extender el análisis a escenarios más genéricos, se evalúan las prestaciones de los tres esquemas de encaminamiento presentados anteriormente (*LD*, *ND* y *ZD*). Estas soluciones se utilizarán para obtener el conjunto óptimo de caminos entre cualquier par de nodos en una red mallada inalámbrica. A partir de estos resultados, se extenderá el análisis del rendimiento de ambos protocolos a escenarios genéricos, donde el despliegue de los dispositivos se realiza de manera aleatoria.

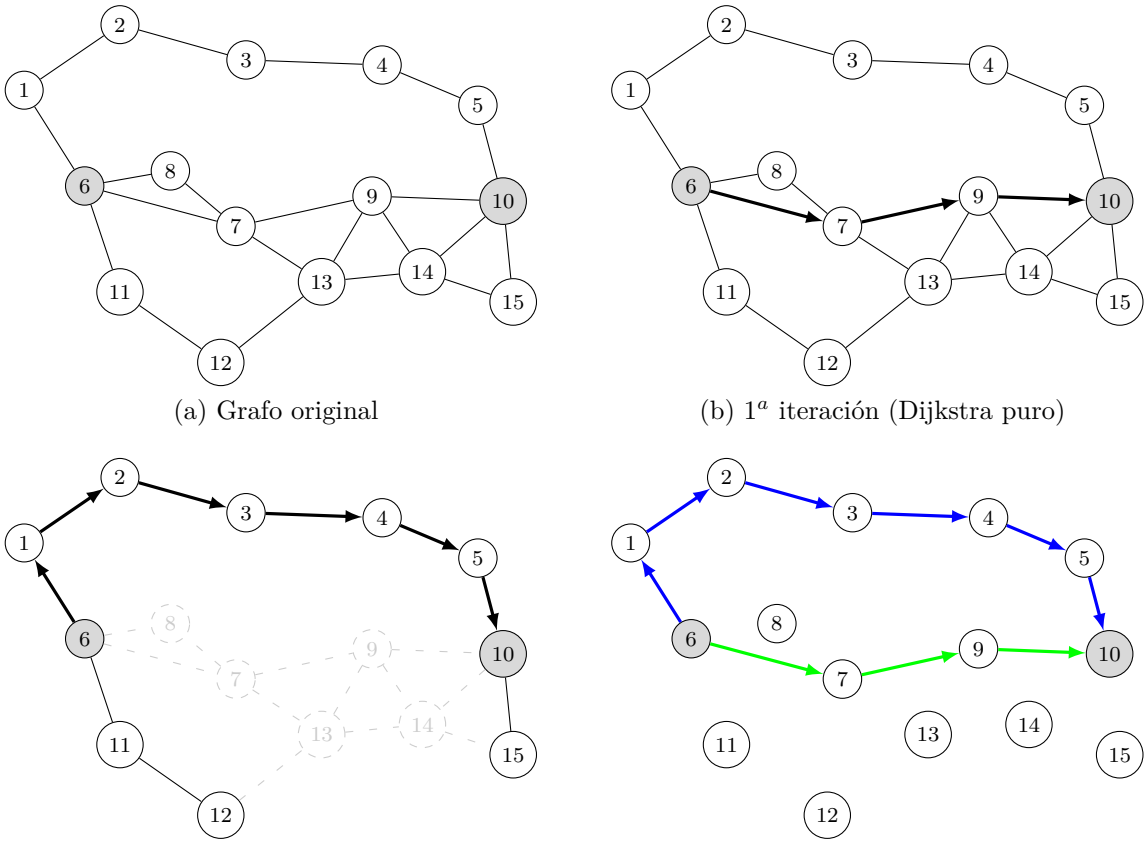
3.4.1. Rendimiento de *MPTCP* sobre topologías sencillas

En esta primera etapa se estudiará el rendimiento de la implementación del protocolo *MPTCP* adaptada a la plataforma `ns-3.13`[5], que sufrió actualizaciones importantes en el módulo *TCP*. Para ello se dividirá el análisis en tres campañas diferentes, centradas en el estudio de características particulares del protocolo: en la primera de ellas se comparará el rendimiento de tres configuraciones básicas bajo unas condiciones ideales. Además, caracterizará el comportamiento más representativo de los algoritmos de control de congestión. En una segunda campaña, se introducirán errores en el canal, provocando la pérdida de segmentos y su retransmisión en el nivel de transporte, con la consecuente disminución del rendimiento global del sistema. Para concluir con este análisis en un escenario canónico, se pondrá a prueba la capacidad de los algoritmos de control de la congestión para balancear la carga entre subflujos en situaciones en las que alguno de los enlaces presente peores condiciones (**Objetivo 3**).

Tal y como muestra la Figura 3.11, el escenario utilizado presenta una forma de "diamante", donde S_1 transmite la información hacia el destino, D_1. Dado que la distancia entre ambos es elevada, no pueden comunicarse a través de un enlace directo, por lo que R_1 y R_2 harán las veces de *routers* inalámbricos, redirigiendo el flujo de información entre los nodos extremos.

En lo que respecta a la configuración del simulador, los principales parámetros se describen a continuación:

[5]Recuérdese que durante el desarrollo de la Tesis se adoptó el módulo *MPTCP* desarrollado inicialmente por Chihani *et al.* [Chih11] para una versión anterior del simulador (`ns-3.6`).



(a) Grafo original

(b) 1^a iteración (Dijkstra puro)

(c) 2^a iteración

(d) Rutas resultantes (P_Z)

Figura 3.10: Ejemplo ilustrativo del algoritmo *ZD*

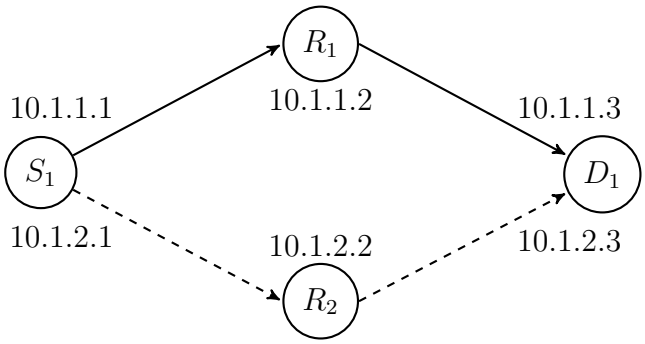


Figura 3.11: Escenario canónico *MPTCP*

- Como se define en el *RFC* que define el estándar [Ford13], un requerimiento básico para el correcto funcionamiento del protocolo *MPTCP* consiste en que los terminales (al menos uno de ellos) dispongan de más de una dirección *IP* activa durante el establecimiento de la conexión. En concreto, para la realización de este experimento, cada nodo contará con dos interfaces *IEEE 802.11*, cada uno de ellos asociado a una subred *IPv4* diferente, como se muestra en la Figura 3.11.

- Desde el punto de vista del nivel de aplicación, el nodo S_1 enviará un fichero de 20 MB de longitud a D_1. Al utilizar *TCP* la información solamente podrá ser transmitida por un interfaz, fijado durante el proceso de establecimiento de la conexión, mientras que el protocolo *MPTCP* será capaz de dividir el tráfico, de manera simultánea, a través de los dos caminos que unen los nodos origen y destino: $S_1 \rightarrow D_1$ y $S_1 \dashrightarrow D_1$.

- Manteniendo el mismo criterio de simulación que en el Capítulo 2, el objetivo de la campaña de medidas será el de establecer los umbrales superiores de rendimiento (reflejado a través del *throughput*) que es capaz de soportar la red con la configuración utilizada. Por lo tanto se asumirán condiciones de saturación, de manera que el propio medio inalámbrico constituya el cuello de botella del sistema (esto es, la tasa binaria con la que la aplicación envía información hacia los niveles inferiores es superior a la capacidad de la red, por lo que siempre habrá, al menos, una trama esperando a ser transmitida en S_1).

- Los enlaces se han configurado según las especificaciones del estándar *IEEE 802.11*, limitando el número total de transmisiones por trama a cuatro intentos, tras los cuales el segmento se descartará.

- Además, para analizar el impacto de los errores de transmisión, cada enlace podrá estar caracterizado individualmente por una tasa de error o *FER* (utilizando el modelo *Nativo* del simulador, estudiado con profundidad en el Capítulo 2), permitiendo analizar el rendimiento de la capa de transporte bajo condiciones adversas.

El objetivo principal de esta fase es demostrar que la utilización del protocolo *MPTCP* realmente mejora al *TCP* tradicional, beneficiándose de la existencia de dispositivos con más de un interfaz inalámbrico. Para ello, se compararán las prestaciones obtenidas a través de tres configuraciones diferentes:

1. La primera de ellas corresponde al caso de una transmisión *TCP* estándar (concretamente, con la versión *New Reno*), en la que únicamente habrá un flujo de datos. Por defecto (ya que el esquema de encaminamiento empleado se basa en tablas estáticas), la ruta seguida por los segmentos de datos es: $S_1 \rightarrow R_1 \rightarrow D_1$.

2. El segundo caso implementa *MPTCP* como solución de transporte, creando para ello dos subflujos independientes sobre el escenario ($S_1 \rightarrow R_1 \rightarrow D_1$ y $S_1 \dashrightarrow R_2 \dashrightarrow D_1$,

configurados sobre un único interfaz inalámbrico, que tendrá asignadas dos direcciones *IP*), por lo que las transmisiones utilizarán el mismo canal. Dada la naturaleza *broadcast* del medio inalámbrico, los envíos pertenecientes a ambos subflujos interferirán entre sí, incrementando el número de estaciones que contienden por el acceso al canal, así como las potenciales colisiones, factores que afectarán negativamente al rendimiento global del sistema.

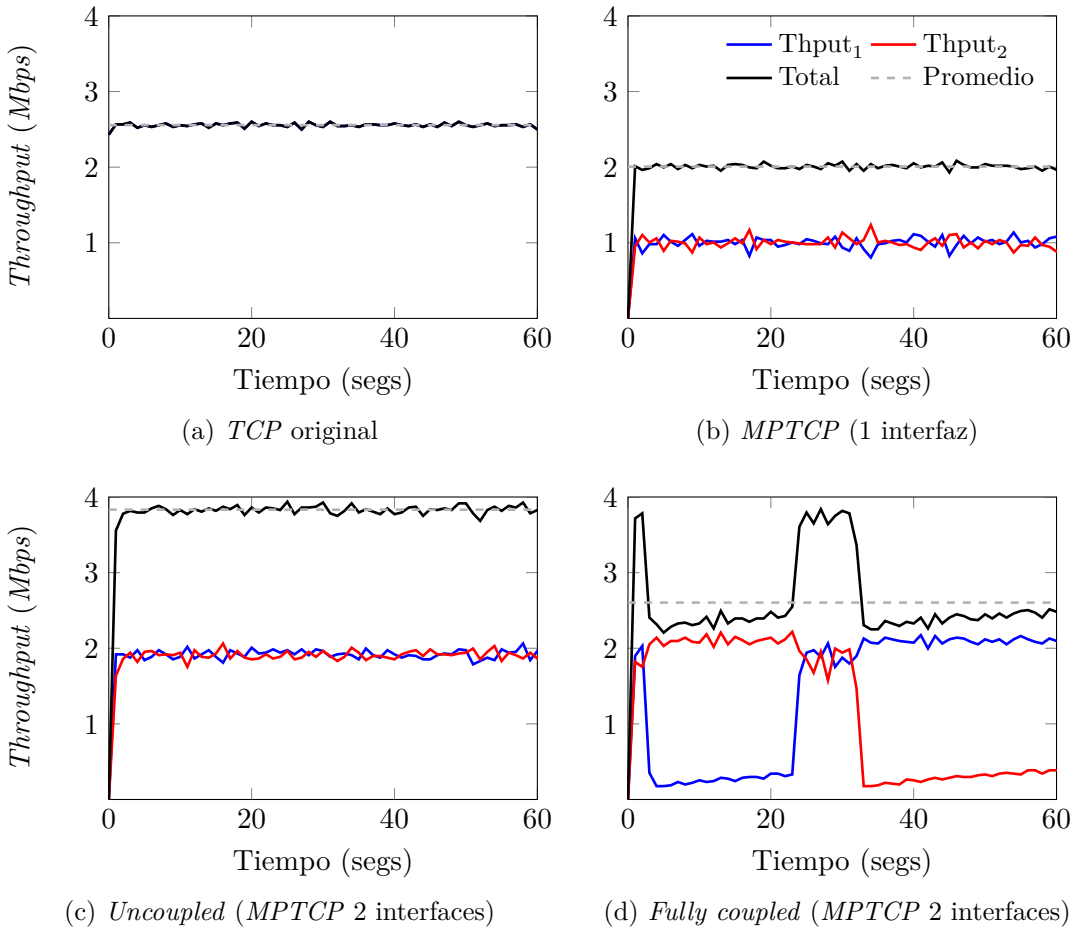
3. Finalmente, y partiendo de la configuración anterior, se emulará la posibilidad de utilizar interfaces *IEEE 802.11* independientes, estando asociados a canales diferentes (y ortogonales entre sí), por lo que, presumiblemente, el rendimiento final se verá incrementado notablemente.

Rendimiento en condiciones ideales

El primer grupo de resultados se obtiene sobre un escenario ideal, en el que la pérdida de tramas, debido a los efectos adversos de la propagación inalámbrica, puede considerarse despreciable; por lo tanto, en este caso, la única fuente de errores estará asociada a las colisiones producidas durante la comunicación. En la Figura 3.12 se puede observar el *throughput* instantáneo de una medida arbitraria para las diferentes configuraciones estudiadas con anterioridad: la Figura 3.12a muestra el rendimiento al utilizar el protocolo *TCP* con un único flujo, manteniendo una tasa estable de ≈2.5 *Mbps*. Este valor cobra una gran relevancia, ya que el **Objetivo 2** de los mecanismos de control de congestión de MPTCP pretende que ninguno de los subflujos emplee más recursos de los que utilizaría *TCP* en las mismas condiciones, por lo que se podrá tomar como referencia (margen superior). Por otra parte, la Figura 3.12b refleja el comportamiento de una configuración *MPTCP* cuando sólo se emplea un único interfaz inalámbrico, representando el *throughput* instantáneo individual de cada uno de los dos subflujos, así como la suma de ambos y el rendimiento medio de la simulación. Puede concluirse que, a pesar de que la carga está repartida equitativamente entre cada uno de ellos, el rendimiento agregado es notablemente inferior al obtenido con *TCP*, debido principalmente a la penalización que supone que ambas subredes estén compartiendo el mismo canal de transmisión (lo que causa tiempos de contención más largos, aumento del número de colisiones, etc.).

En lo que se refiere a la caracterización de *MPTCP* sobre dos interfaces, la Figura 3.12c refleja el comportamiento del algoritmo *Uncoupled*[6]; tras una breve fase inicial transitoria correspondiente al mecanismo *Slow Start* de *TCP*, el *throughput* de ambos subflujos muestra un reparto equitativo y estable, cuyo valor agregado supera ampliamente (en torno a un 48 %) el obtenido con *TCP*. Es importante destacar en que ninguno de los subflujos se observan rendimientos superiores a *TCP*, cumpliendo de esta manera el **Objetivo 2**.

[6]Este algoritmo muestra una gran similitud con los mecanismos *Linked increases* y *RTT compensator* cuando la tasa de errores producidos durante la propagación es despreciable.



(a) *TCP* original (b) *MPTCP* (1 interfaz)

(c) *Uncoupled* (*MPTCP* 2 interfaces) (d) *Fully coupled* (*MPTCP* 2 interfaces)

Figura 3.12: *Throughput* instantáneo sobre un escenario ideal para los diferentes algoritmos

Finalmente, la Figura 3.12d muestra el impacto del efecto conocido como *flappiness*, observado principalmente en el algoritmo *Fully Coupled* cuando los errores[7] de ambos subflujos son similares y las ventanas de congestión muestran comportamientos parejos ($w_1 \sim w_2$). Puede comprobarse el comportamiento opuesto en el *throughput* de ambas conexiones, ya que uno de los subflujos transmite la mayor parte del tráfico, reduciendo considerablemente la carga del otro. En el momento en el que el primero acumule una cierta cantidad de errores, la tendencia se invertirá y será el segundo subflujo el que pase a ser el "dominante". Esta inestabilidad se debe a que, cuando el tamaño de una de las ventanas de congestión se reduce hasta alcanzar un valor prácticamente nulo ($w_i \approx 0$), hace que el otro flujo se apodere del canal casi por completo. Para que el tamaño de su ventana recupere unos valores aceptables, será necesario que se produzca un número significativo de pérdidas en un corto espacio de tiempo en el camino del segundo subflujo. Como puede apreciarse en la figura, este efecto "rebote" aparece dos veces durante la transmisión del fichero. Todo

[7]Debe recordarse que los errores solamente se producirán por colisiones en el canal.

esto repercutirá en gran medida al ancho de banda agregado, que es notablemente inferior al obtenido a través de un algoritmo que no presente este comportamiento (concretamente, el *throughput* total es un $\approx 47\,\%$ más bajo).

En lo que respecta a la técnica elegida para la reordenación de los segmentos en el destino, se ha utilizado en todo momento el esquema *DSACK*, debido a que es capaz de detectar una retransmisión innecesaria más rápidamente que el algoritmo *Eifel*.

Rendimiento en un escenario con errores

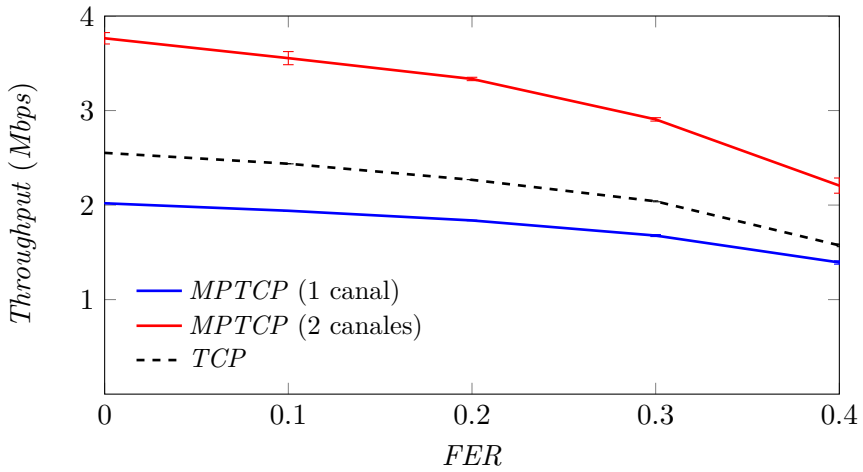


Figura 3.13: *Throughput* sobre un canal con errores

En este segundo caso, la probabilidad de perder un paquete en los enlaces entre los nodos ya no es despreciable, con lo que el rendimiento global del sistema se verá afectado en gran medida. Se asume que sólo los enlaces dirigidos $S_1 \rightarrow R_1$ y $S_1 \rightarrow R_2$ generarán errores de transmisión[8]. La Figura 3.13 muestra la evolución del *throughput* global percibido en el nivel de aplicación a medida que las condiciones del canal van empeorando. Después de analizar el comportamiento de los diferentes algoritmos de control de congestión, en este punto se utilizará el *Linked increases* (aunque con la solución *RTT compensator* se obtendría una respuesta similar), ya que es capaz de repartir eficientemente la carga entre los dos subflujos, sin provocar el efecto "flappiness", manteniendo una evolución similar en las ventanas de congestión.

En la figura se representa el valor medio y el intervalo de confianza del 95 % de un experimento de 50 simulaciones para cada una de las configuraciones analizadas. Resulta evidente que el valor de rendimiento más pobre corresponde a una transmisión *MPTCP* basada en un único canal inalámbrico (a través de un sólo interfaz), debido a las razones

[8]Por razones de simplicidad, el flujo inverso de reconocimientos *TCP* no se verá afectado por las pérdidas del canal.

comentadas con anterioridad: en primer lugar, el control de flujo de cada una de las conexiones limita la tasa de envío ($\approx$ el tamaño máximo de la suma de las dos ventanas de congestión) con el fin de no sobrepasar al que se obtendría con *TCP*, respetando el **Objetivo 2**; por otra parte, ya que todas las transmisiones están compartiendo el mismo canal inalámbrico, cada nodo que intente enviar una trama deberá contender con el resto de estaciones por el acceso; cuantos más nodos traten de enviar información, mayor será el tiempo de espera promedio[9]. En un punto intermedio se encuentra el rendimiento alcanzado por las transmisiones *TCP* a través de un único flujo. Como era de esperar, se observa que el uso de *MPTCP* sobre dos interfaces independientes (pertenecientes a canales ortogonales) mejora significativamente el rendimiento mostrado por *TCP*.

Prueba del balanceo de carga

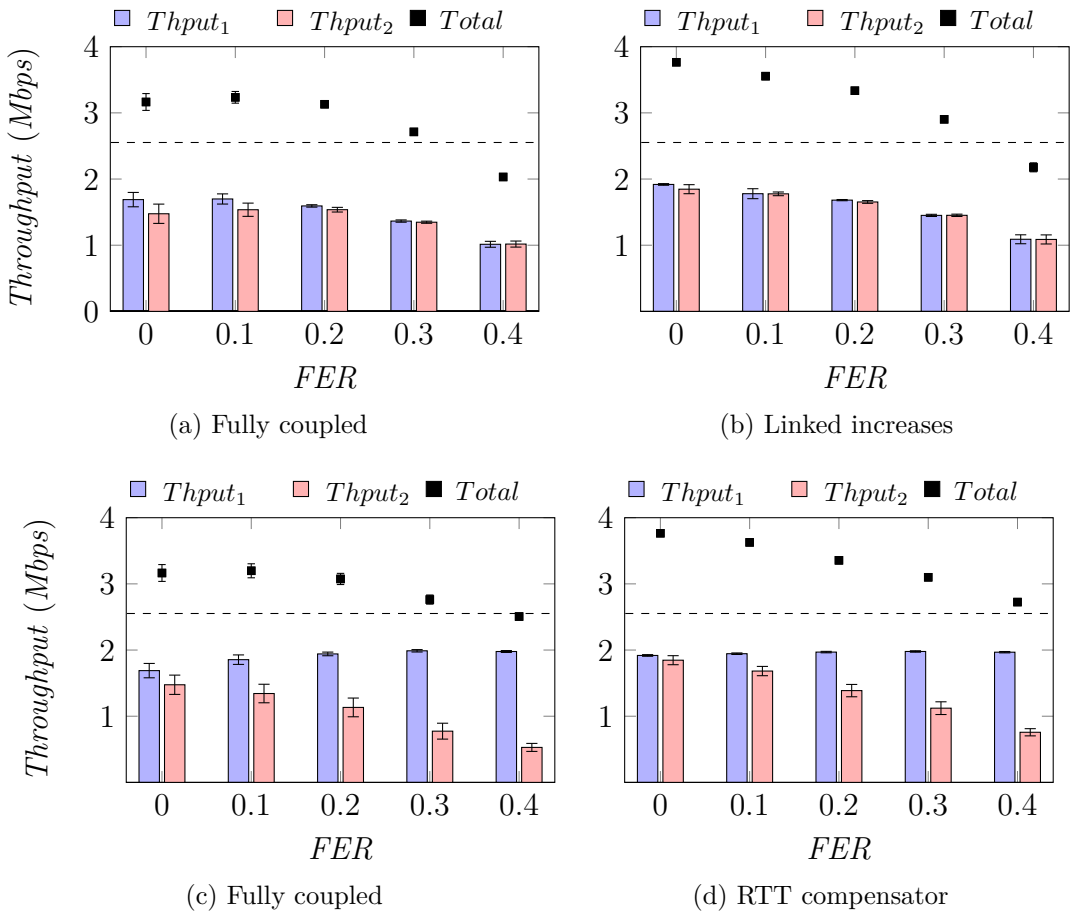
En esta última parte se analizará la capacidad de *MPTCP* para balancear la carga entre los subflujos bajo diversas configuraciones del canal, con el fin de comprobar si los algoritmos correspondientes son capaces de compensar el deterioro de rendimiento producido cuando alguno de ellos presenta unas condiciones más hostiles (**Objetivo 3**).

La Figura 3.14 muestra el *throughput* medio de cada uno de los subflujos y el intervalo de confianza del 95 % de un total de 50 realizaciones por cada configuración. Además, se puede observar el ancho de banda agregado (puntos cuadrados) y su correspondiente intervalo de confianza. Por último, una línea horizontal discontinua refleja el *throughput* medio obtenido a través de las transmisiones de *TCP* en un escenario sin errores, umbral que será utilizado como margen superior de rendimiento, y que no deberá ser sobrepasado por ningún subflujo MPTCP (**Objetivo 2**).

Las Figuras 3.14a y 3.14b muestran el reparto de ancho de banda ofrecido por los algoritmos *Fully Coupled* y *Linked increases*, respectivamente, cuando las pérdidas en el canal afectan de igual modo a ambos caminos (configuración *simétrica*). Si bien ambos mecanismos evidencian un reparto equitativo entre los dos subflujos, en el primero de los casos se observa una mayor variabilidad en los resultados, consecuencia de la falta de estabilidad asociada al propio algoritmo (como ya se observó en la Figura 3.12d), que podría derivar en el efecto *flappiness*. Se puede concluir que este algoritmo es inestable cuando las ventanas de congestión de ambos subflujos son similares ($w_1 \sim w_2$), ya que un ligero cambio en una de ellas inducirá una descompensación evidente entre las cargas de ambos subflujos.

Posteriormente, se ha modificado ligeramente la configuración de los errores de propagación, creando en el escenario una situación *asimétrica*; mientras que la ruta superior ($S_1 \rightarrow R_1$) no presenta pérdidas en el canal, la inferior ($S_1 \rightarrow R_2$) presentará una *FER* mayor de 0. A través de este simple cambio, se evaluará si la implementación realizada del

[9] Además, mayor será la probabilidad de colisión o transmisiones simultáneas.



(a) Fully coupled

(b) Linked increases

(c) Fully coupled

(d) RTT compensator

Figura 3.14: Reparto del ancho de banda entre subflujos sobre un canal con errores. (a) y (b) configuración simétrica. (c) y (d) configuración asimétrica

protocolo *MPTCP* es capaz de compensar la congestión[10] producida en el segundo subflujo, incrementando la carga de tráfico en el primero. La Figura 3.14c muestra los resultados obtenidos utilizando el *Fully Coupled* como algoritmo de control de la congestión, poniendo de manifiesto una variabilidad más elevada y un rendimiento global inferior que el *RTT compensator* (ver Figura 3.14d). Esta última será la alternativa más adecuada, ya que adaptará el crecimiento de las ventanas de contención al valor de *RTT* estimado en cada uno de los subflujos. La principal diferencia radica en el *flappiness* característico del primero de los mecanismos, lo que causa la infrautilización de los recursos, lo que conlleva a un *throughput* agregado más bajo. Sin embargo, puede apreciarse en ambos mecanismos que, a medida que la tasa de error en el enlace $S_1 \rightarrow R_2$ sube, la carga soportada por el primero de los subflujos crece gradualmente (sin sobrepasar al valor de *TCP*), demostrando que una parte del tráfico que supuestamente iría destinado al segundo de los caminos es tras-

[10]Debe tenerse en cuenta que *MPTCP*, al igual que *TCP*, no es capaz de distinguir una pérdida ocasionada por congestión frente a otra causada por la propagación a través del medio inalámbrico.

pasado satisfactoriamente al subflujo menos congestionado, compensando (parcialmente) la pérdida que provocaría el canal más hostil.

3.4.2. Prestaciones de los algoritmos de encaminamiento

Antes de proceder a extender el análisis del protocolo *MPTCP* a escenarios más genéricos, es preciso analizar los problemas de encaminamiento que surgen cuando un nodo decide comunicarse con otro en una topología aleatoria. En esta etapa intermedia se caracterizará el rendimiento de los tres algoritmos de encaminamiento presentados en la Sección 3.3: *LD*, *ND* y *ZD*.

Es importante recalcar de nuevo que el método utilizado para la obtención de las rutas entre los nodos es *independiente* del simulador `ns-3`, habiendo implementado un módulo externo (desarrollado en `C++`) que, tras la introducción de los parámetros de entrada (área del escenario, número de nodos y rango de cobertura), será capaz de: *(1)* desplegar los nodos sobre la superficie, *(2)* ejecutar los tres algoritmos de búsqueda de rutas y *(3)* generar los ficheros que serán utilizados en `ns-3` para llevar a cabo las simulaciones pertinentes.

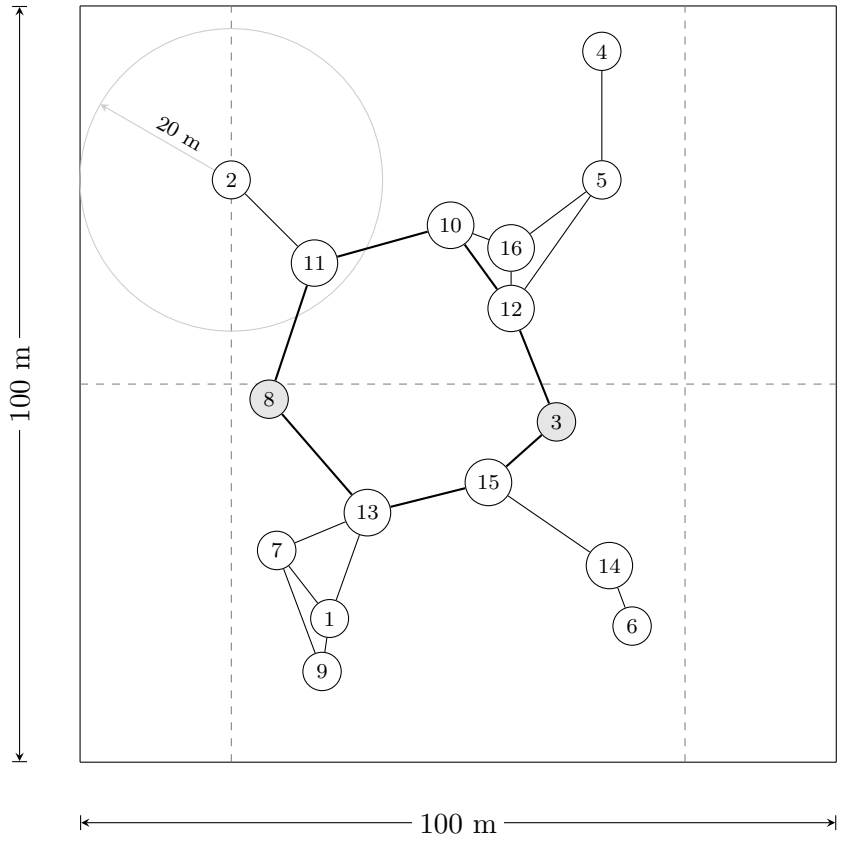


Figura 3.15: Despliegue aleatorio de nodos

Concretamente, para la realización de este experimento se ha optado por establecer el siguiente conjunto de condiciones y/o restricciones:

- Los nodos estarán ubicados dentro de un área cuadrada de 100×100 metros.
- Inicialmente se descartarán aquellos despliegues que originen un grafo no conexo.
- Se asumirá que los nodos permanecerán estáticos durante todo el tiempo de simulación. Uno de los efectos más inmediatos si se considerara la movilidad de los nodos sería la necesidad de actualizar dinámicamente las tablas de rutas durante la simulación, a medida que se va modificando la topología de la red, dotando de una mayor relevancia a los protocolos de encaminamiento multi-camino.
- El rango de cobertura de los nodos estará delimitado por una circunferencia de 20 metros de radio.
- Para fijar una cierta consistencia a la hora de elegir cuál será la pareja de nodos origen-destino, se ha establecido la siguiente restricción: el nodo más cercano al punto de coordenadas $(20, 50)$ será elegido como transmisor, mientras que el más próximo al $(80, 50)$ ejercerá el papel de receptor, tal y como puede verse en la Figura 3.15, en un ejemplo ilustrativo.

La Figura 3.15 muestra un despliegue completamente aleatorio de 16 nodos. En esta topología en concreto se observa que el nodo 8 es el elegido para hacer las veces de transmisor, por ser el más cercano al punto $(20, 50)$,mientras que el nodo 3 será el destino. En lo relativo a las rutas elegidas, los tres algoritmos proporcionarán el mismo resultado: mientras que el camino óptimo estará formado por la secuencia $8 \rightarrow 13 \rightarrow 15 \rightarrow 3$, la segunda opción sigue el orden $8 \rightarrow 11 \rightarrow 10 \rightarrow 12 \rightarrow 3$.

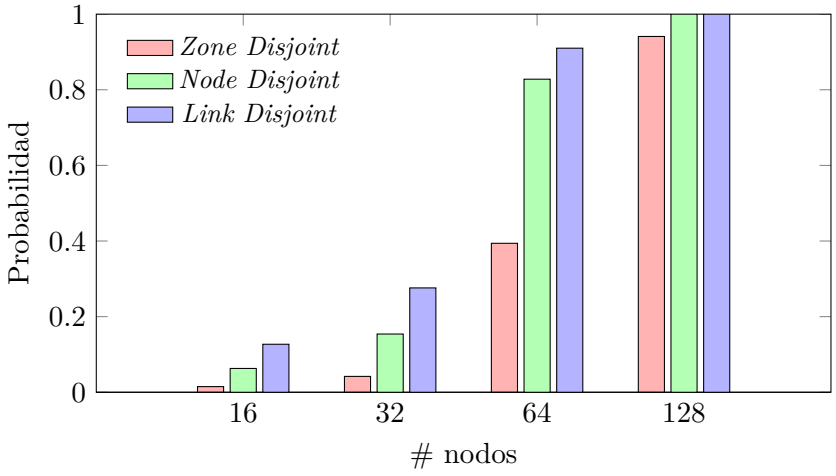


Figura 3.16: Probabilidad de encontrar una solución multi-camino

La Figura 3.16 muestra el porcentaje de escenarios factibles para emplear un esquema multi-camino en función del número total de nodos desplegados en el escenario[11]. Para ello será necesario que el algoritmo obtenga, al menos, dos rutas diferentes que permitan dividir el tráfico empleando el protocolo *MPTCP* (en caso negativo, no se podría sacar partido a la transmisión simultánea por múltiples subflujos). A tenor de los resultados, se observa una superioridad generalizada del algoritmo *LD*, seguido por *ND*, mostrando ambos unos porcentajes muy superiores a los ofrecidos por *ZD*, que aparece como la alternativa más restrictiva.

Para la obtención de los siguientes estadísticos se ha añadido una nueva restricción, que permite que los nuevos escenarios generados puedan ser empleados en las simulaciones de `ns-3` de manera equitativa: sólo se darán por válidas aquellas topologías que presenten, al menos, dos rutas para los tres algoritmos estudiados. Además, se ha fijado el número de nodos desplegados a 32. Para disponer de un número lo suficientemente elevado de escenarios, se han obtenido un total de 1000 despliegues que cumplen este requisito. Merece la pena mencionar que para conseguirlo se han tenido que desechar un número elevado de topologías, ya que, como se observó en la Figura 3.16, sólo un 4.2 % de los escenarios de 32 nodos generados presenta las condiciones necesarias para encontrar (al menos) dos caminos con *ZD*.

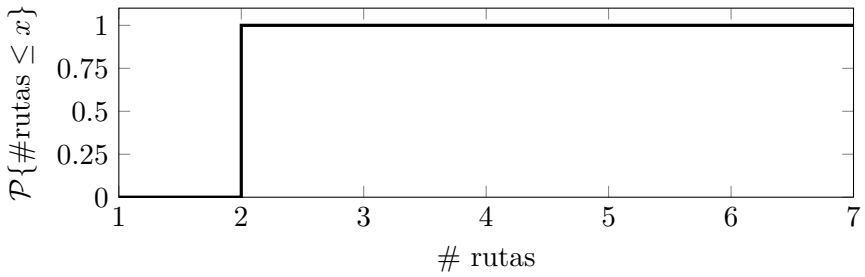
Con estas condiciones se representa, en la Figura 3.17, la función de distribución del número total de rutas encontradas por cada uno de los algoritmos. Tal y como era de esperar, *LD* es el que más alternativas proporciona, debido a que es el esquema que menos penaliza a los grafos entre las sucesivas iteraciones. Por su parte, las restricciones impuestas por *ZD* hacen que no sea posible encontrar despliegues con más de dos caminos alternativos en ningún caso, dejando al algoritmo *ND* como una solución intermedia.

Otro parámetro de gran interés es la función de distribución del número de saltos encontrados en las dos primeras rutas, que representa la Figura 3.18. Como era esperable, la primera iteración es igual para todos, por lo que la longitud del camino más corto no depende del algoritmo empleado. Para el segundo, se vuelve a observar la tendencia anterior: como *LD* es el algoritmo que menos modifica el grafo entre iteraciones, encontrará caminos más cortos; por contra, *ZD* vuelve a ser otra vez el esquema que presenta peores resultados, dejando a *ND* de nuevo en un plano intermedio. El número de saltos tendrá una gran influencia en el rendimiento agregado del sistema, ya que cuanto mayor sea la longitud de las rutas más se penalizará el *throughput* del subflujo correspondiente.

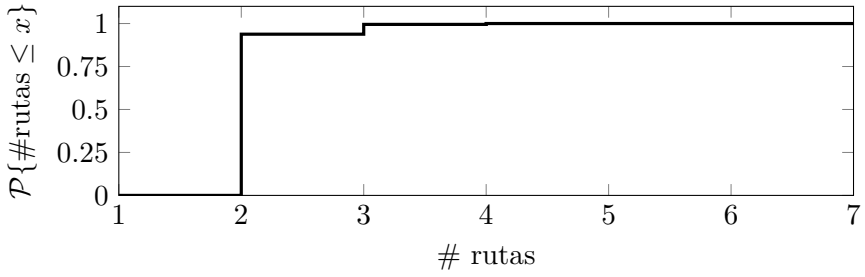
3.4.3. Rendimiento de *MPTCP* en redes multi-salto

Tras el análisis efectuado para caracterizar y comparar los algoritmos para obtener múltiples caminos en despliegues aleatorios de 32 nodos, se procederá a la simulación

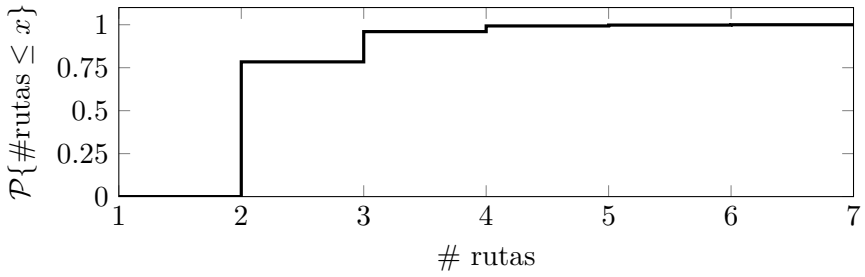
[11] El experimento realizado ha constado de 1000 realizaciones.



(a) *Zone disjoint*



(b) *Node disjoint*



(c) *Link disjoint*

Figura 3.17: Función de distribución del número de rutas encontradas

de los escenarios resultantes, manteniendo la misma configuración que la utilizada en la Sección 3.4.1 (con canales basados en el estándar *IEEE 802.11* y 3 retransmisiones a nivel *MAC*). En lo que respecta al protocolo *MPTCP*, se presentarán los resultados obtenidos utilizando los esquemas *linked increases* y *DSACK* para el control de la congestión y la reordenación, respectivamente. Además, la única fuente de pérdidas en la transmisión estará ligada a las posibles colisiones, configurando los canales inalámbricos como ideales, con una $FER = 0$.

Todos los despliegues aquí analizados serán los obtenidos en el análisis anterior, garantizando la existencia de al menos dos rutas para los tres algoritmos. Además, se asume que los nodos permanecerán estáticos durante toda la simulación, por lo que las rutas se cargarán al comienzo de la misma, conformando tablas estáticas.

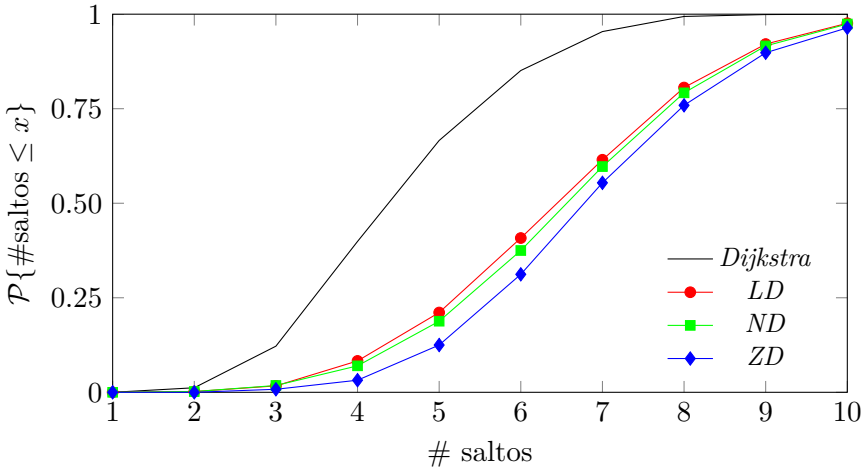


Figura 3.18: Función de distribución del número de saltos de las dos mejores rutas

Una vez generados los escenarios, se procederá a simular el comportamiento de cada una de las configuraciones estudiadas con anterioridad (*TCP* con un único flujo y *MPTCP* con uno o dos interfaces inalámbricos), para estudiar su comportamiento.

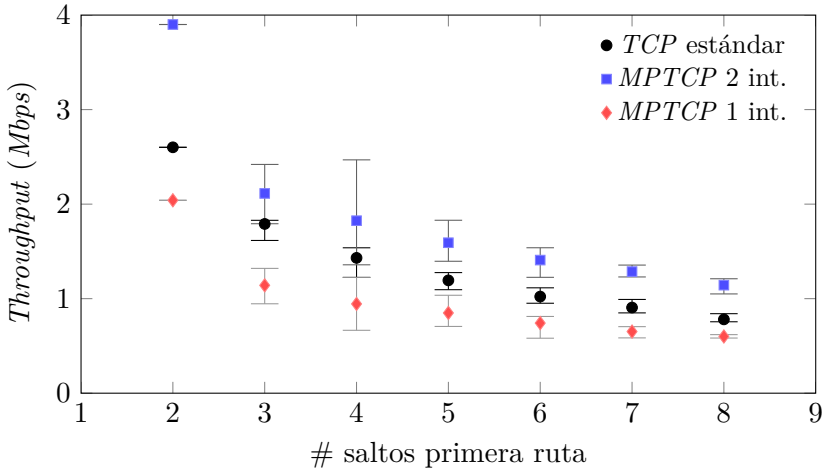


Figura 3.19: *Throughput* medio frente al número de saltos de la primera ruta

La Figura 3.19[12] muestra los resultados obtenidos con las tres configuraciones, representando el *throughput* medio, así como los valores máximo y mínimo observados en función del número de saltos requeridos en la ruta más corta. Puede observarse la gran mejora que introduce la utilización del protocolo *MPTCP* sobre dos interfaces, consiguiendo en la mayoría de las situaciones un rendimiento agregado superior a *TCP*, llegando a alcanzar tasas hasta un 50 % superiores en escenarios de dos saltos. Por otro lado, se demuestra nuevamente el efecto negativo de utilizar la solución multi-camino sobre un único

[12]Debido a la similitud de los resultados, solamente se representan los obtenidos con el algoritmo *LD*, que se corresponden con los que presentan un menor número de saltos.

interfaz, ya que la contención causada por el elevado número de nodos accediendo al canal produce unos largos tiempos de espera y un incremento de la probabilidad de colisión, con la consecuencia de ofrecer unos rendimientos muy inferiores a los vistos en *TCP*.

3.4.4. Configuración óptima de rutas

Los resultados vistos hasta este momento han puesto de manifiesto una clara inferioridad del algoritmo de *ZD* con respecto a *LD* y *ND* en todos los aspectos estudiados: debido al carácter restrictivo que presenta a la hora de modificar el grafo entre iteraciones, no es posible, en un porcentaje muy elevado de los escenarios, encontrar dos rutas diferentes que permitieran utilizar técnicas multi-camino. Además, en aquellas topologías en las que sí se obtienen dos caminos, el número de saltos necesarios en la segunda ruta es significativamente mayor, factor que penaliza, a priori, el ancho de banda agregado en una conexión *MPTCP*.

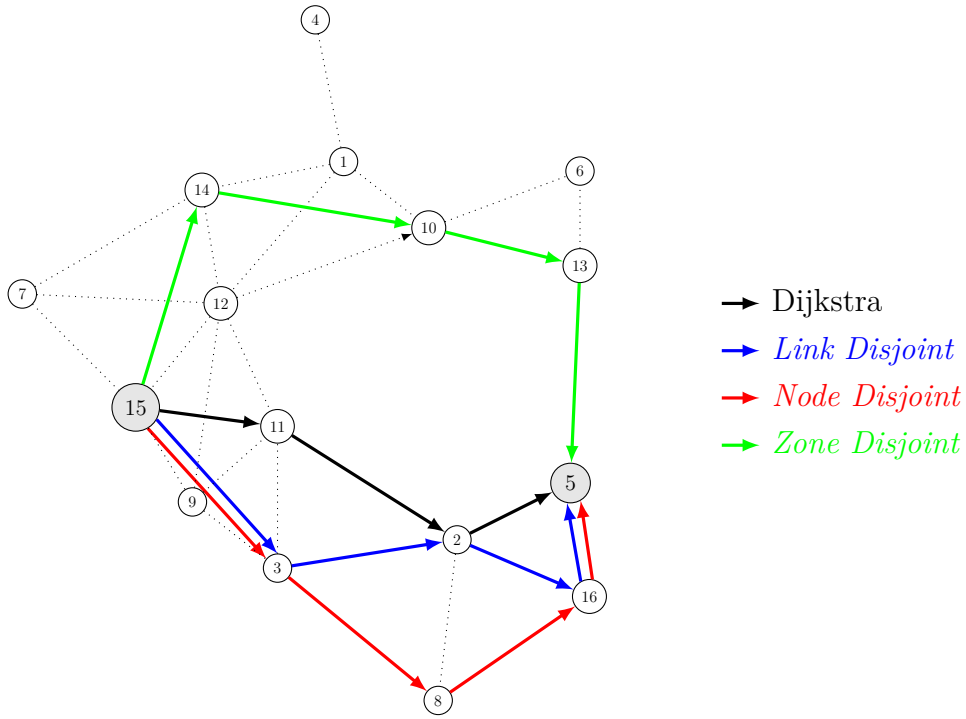


Figura 3.20: Ejemplo de mejora con el algoritmo *ZD*

No obstante, tal y como se dejó entrever en la Sección 3.3, existe un supuesto en la que *ZD* puede presentar unas prestaciones superiores a las alternativas *LD* y *ND*: en escenarios en los que los dispositivos cuenten únicamente con un interfaz inalámbrico, ya que podría reducir la interferencia producida entre los subflujos.

Como ejemplo ilustrativo del potencial beneficio de "separar" las rutas siempre que sea posible, la Figura 3.20 muestra un despliegue aleatorio de 16 nodos en un área de

100×100 metros. El primero de los caminos se obtiene a través del algoritmo de *Dijkstra*, y es idéntico para los tres algoritmos, correspondiéndose con la secuencia $15 \rightarrow 11 \rightarrow 2 \rightarrow 5$. A continuación se presentan las diferentes posibilidades para la segunda ruta: $15 \rightarrow 3 \rightarrow 2 \rightarrow 16 \rightarrow 5$ para *LD*, $15 \rightarrow 3 \rightarrow 8 \rightarrow 16 \rightarrow 5$ para *ND* y por último, $15 \rightarrow 14 \rightarrow 10 \rightarrow 13 \rightarrow 5$ para *ZD*. En la figura se aprecia que los caminos obtenidos con *LD* y *ND* se encuentran próximos, especialmente con el camino más corto, por lo que parece lógico que el efecto de la contención y las colisiones sea elevado, ya que el número de nodos transmitiendo será mayor, incrementando la saturación del medio inalámbrico. Por el contrario, gracias a que *ZD* separa "espacialmente" los caminos, conseguirá reducir el esta contención. Tras los resultados obtenidos realizando en un total de 10 simulaciones con cada uno de los algoritmos, se ha observado un incremento de más de un 5 % en términos de *throughput*, valor que demuestra que el esquema *ZD* es capaz de mejorar las prestaciones obtenidas con los otros dos algoritmos en circunstancias particulares.

3.5 Conclusiones y líneas futuras

En este Capítulo se ha portado el protocolo *MPTCP* (inicialmente implementado por Chihani et al. [Chih11]) a una versión más completa y estable del simulador `ns-3` (concretamente, `ns-3.13`). El módulo desarrollado sigue las especificaciones definidas en las recomendaciones del protocolo, siendo la principal la mostrada en [Ford13]. Sus principales características se basan en una operación transparente con el nivel de aplicación, que interactúa con una única conexión *TCP* a nivel de tranporte. De manera subyacente, se crean múltiples subflujos independientes que transmiten el tráfico de manera simultánea.

Durante la realización de este trabajo se ha llevado a cabo una completa caracterización del protocolo *MPTCP* sobre redes malladas inalámbricas sobre el simulador `ns-3`, demostrando que es capaz de mejorar sustancialmente el pobre rendimiento mostrado por *TCP* en este tipo de escenarios. En primer lugar, se ha estudiado el comportamiento de algunos de los mecanismos más importantes del protocolo sobre una topología sencilla de 4 nodos, para analizar exhaustivamente las diferencias entre algunas de sus opciones más básicas. Sobre un canal sin errores, se han comparado cuatro mecanismos de control de congestión: mientras que los algoritmos *Uncoupled*, *Linked increases* y *RTT compensator* presentan unos resultados muy similares, repartiendo equitativamente el ancho de banda entre los dos subflujos, el esquema *Fully coupled* evidencia el efecto conocido como "flappiness", en el que uno de los flujos (el que presenta la tasa de pérdidas más reducida) "captura" la mayor parte de los recursos. En consecuencia, el *throughput* agregado del sistema se ve severamente dañado.

Posteriormente, se han introducido pérdidas de paquetes entre los nodos, comparando el rendimiento de una versión de *TCP* clásica (*NewReno*) con dos configuraciones diferentes del protocolo *MPTCP*, empleando el mismo interfaz inalámbrico para transmitir dos subflujos (que, por tanto, comparten el medio) y emulando una red multicanal, sin inter-

ferencias entre las diferentes rutas. Tras analizar los resultados obtenidos, se ha observado que *MPTCP* introduce una mejora de *throughput* respecto a *TCP* del 48 % en el caso en el que los flujos vayan por canales ortogonales ideales.

Por último, se ha estudiado la capacidad de *MPTCP* para balancear la carga entre los subflujos. Para ello, se ha analizado el comportamiento de diferentes algoritmos de control de congestión bajo diferentes condiciones. Mientras el tráfico de los dos subflujos se reparte de manera equitativa cuando las pérdidas afectan por igual a los dos caminos, en el momento en el que uno de ellos sufre una *FER* superior, se verá cómo la carga del subflujo menos congestionado aumenta, paliando el efecto negativo producido por los errores de propagación en el otro camino.

En una segunda etapa, se han caracterizado tres algoritmos diferentes (*Link*, *Node* y *Zone Disjoint*) para la obtención de múltiples rutas en despliegues genéricos, aunque en este caso han sido aplicados a la utilización de protocolos multi-camino en redes malladas inalámbricas. Se han comparado en términos de utilidad, calculando el porcentaje de realizaciones en los que se obtienen dos o más rutas, el número de caminos encontrados y las longitudes de las mismas. A tenor de los resultados obtenidos, el algoritmo *ZD* parece una técnica poco conveniente en la búsqueda de rutas para soluciones *multi-path*, aunque es posible que, cuando encuentra múltiples caminos, pueda lograr un mejor rendimiento en la transmisión, ya que sus condiciones restrictivas separan "físicamente" las rutas en la mayor medida posible, reduciendo el impacto producido por las interferencias entre diferentes subflujos (e.g. colisiones, contención del canal, etc.).

Posteriormente, se ha extendido el análisis del comportamiento del protocolo *MPTCP* a topologías más complejas, en las que el despliegue de los terminales se ha llevado a cabo de una manera aleatoria. En estos escenarios, se ha vuelto a demostrar que la utilización de múltiples interfaces puede producir mejoras significativas en el rendimiento, de hasta el 50 % en algunos casos.

Tras el estudio realizado, todavía quedan pendientes una serie de cuestiones y líneas abiertas que deberán ser tratadas con profundidad en un futuro. De entre todas ellas, se citan a continuación algunas de las más importantes:

- En el momento de llevar a cabo la campaña de simulación, la falta de modelos maduros y estables en `ns-3` en relación a las múltiples tecnologías de acceso inalámbricas existentes en la actualidad (e.g. *LTE*, *UMTS*, *WiMAX* e incluso versiones más recientes del estándar *IEEE 802.11*) obligó a la utilización de interfaces *IEEE 802.11*, emulando la presencia de tecnologías diferentes usando canales ortogonales. Sería interesante combinar en un futuro medios físicos diferentes, donde métricas como el *RTT* o las tasas de error dejen de ser equivalentes.
- Por otra parte, dada la sencillez del algoritmo utilizado para repartir los segmentos procedentes de los niveles superiores (*Round Robin*), sería posible plantear la implementación de técnicas más complejas, como *Weighted Round Robin (WRR)*, que

permitan redistribuir la carga entre los subflujos en función de la capacidad del canal en un instante dado. De este modo, el cumplimiento del **Objetivo 3** se vería mejorado en canales donde los subflujos presenten rendimientos dispares. Existen además otras opciones que utilizan mejor la información disponible en la capa de transporte, como el valor del *RTT* o el tamaño de la ventana de congestión de cada uno de los subflujos en un instante dado, como se propone en [Yang05].

- También sería interesante añadir movilidad a los nodos para estudiar el rendimiento en una red mallada dinámica, incluyendo nuevas métricas al análisis (e.g. durabilidad de las rutas, estabilidad de las mismas, etc.). En este caso, ya no sería posible la obtención de rutas mediante un proceso independiente, sino que tendría que utilizarse un protocolo de encaminamiento *multi-path* que sea capaz de descubir nuevos caminos tras cambios en la topología de la red.

- Otro paso a analizar sería cuando el número de conexiones de datos no se encuentre limitada a una, pudiendo mezclarse el tráfico *MPTCP* con otros protocolos, como *TCP* o *UDP*, compartiendo los recursos disponibles en la red. Como ya se ha mencionado, la implementación utilizada no permite mantener instancias simultáneas de *MPTCP* y TCP en los nodos, así que la opción más lógica sería la utilización del módulo desarrollado por Kheirkah [Khei14] cuando se integre en el repositorio oficial del simulador. Así, se podrían comprobar la limitaciones de los algoritmos de control de congestión estudiados, analizando nuevas soluciones, como el algoritmo *OLIA* [Khal14a].

Antes de terminar, es conveniente destacar que la implementación del protocolo MPTCP realizada a lo largo de este trabajo se ha regido respetando la filosofía *Generic Public License (GPL)* y puede encontrarse en [MPNS3], estando a total disposición para su utilización y modificación. Se puede destacar el gran interés mostrado en esta implementación del protocolo.

Referencias

[Allm09] M. Allman, V. Paxson y E. Blanton. *TCP Congestion Control*. RFC 5681 (Draft Standard). Internet Engineering Task Force, sep. de 2009. URL: http://www.ietf.org/rfc/rfc5681.txt.

[Arza14] B. Arzani, A. Gurney, Shuotian Cheng, R. Guerin y Boon Thau Loo. «Impact of Path Characteristics and Scheduling Policies on MPTCP Performance». En: *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*. Mayo de 2014, págs. 743-748. DOI: 10.1109/WAINA.2014.121.

[Bagn11] M. Bagnulo. *Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses*. RFC 6181 (Informational). Internet Engineering Task Force, mar. de 2011. URL: http://www.ietf.org/rfc/rfc6181.txt.

[Bona14] Olivier Bonaventure, Christoph Paasch y Gregory Detal. *Experience with Multipath TCP*. Inglés. Internet Draft. 16 de sep. de 2014. URL: https://tools.ietf.org/id/draft-ietf-mptcp-experience-00.txt.

[Chen13] Yung-Chih Chen, Yeon-sup Lim, Richard J. Gibbens, Erich M. Nahum, Ramin Khalili y Don Towsley. «A Measurement-based Study of Multipath TCP Performance over Wireless Networks». En: *Proceedings of the 2013 Conference on Internet Measurement Conference*. IMC '13. ACM, 2013, págs. 455-468. ISBN: 978-1-4503-1953-9. DOI: 10.1145/2504730.2504751. URL: http://doi.acm.org/10.1145/2504730.2504751.

[Chih11] Bachir Chihani y Denis Collange. «A Multipath TCP model for ns-3 simulator». En: *CoRR* abs/1112.1932 (2011).

[Clau03] T. Clausen y P. Jacquet. *Optimized Link State Routing Protocol (OLSR)*. RFC 3626 (Experimental). Internet Engineering Task Force, oct. de 2003. URL: http://www.ietf.org/rfc/rfc3626.txt.

[Clau14] T. Clausen, C. Dearlove, P. Jacquet y U. Herberg. *The Optimized Link State Routing Protocol Version 2*. RFC 7181 (Proposed Standard). Updated by RFCs 7183, 7187, 7188. Internet Engineering Task Force, abr. de 2014. URL: http://www.ietf.org/rfc/rfc7181.txt.

[DCE] *Ns-3 Direct Code Execution*. http://www.nsnam.org/overview/projects/direct-code-execution/.

[Dijk59] E. W. Dijkstra. «A Note on Two Problems in Connexion with Graphs». En: *NUMERISCHE MATHEMATIK* 1.1 (1959), págs. 269-271.

[Floy00] S. Floyd, J. Mahdavi, M. Mathis y M. Podolsky. *An Extension to the Selective Acknowledgement (SACK) Option for TCP*. RFC 2883 (Proposed Standard). Internet Engineering Task Force, jul. de 2000. URL: http://www.ietf.org/rfc/rfc2883.txt.

[Ford11] A. Ford, C. Raiciu, M. Handley, S. Barre y J. Iyengar. *Architectural Guidelines for Multipath TCP Development*. RFC 6182 (Informational). Internet Engineering Task Force, mar. de 2011. URL: http://www.ietf.org/rfc/rfc6182.txt.

[Ford13] A. Ford, C. Raiciu, M. Handley y O. Bonaventure. *TCP Extensions for Multipath Operation with Multiple Addresses*. RFC 6824 (Experimental). Internet Engineering Task Force, ene. de 2013. URL: http://www.ietf.org/rfc/rfc6824.txt.

[Hamp11] G. Hampel y T. Klein. *Enhancements to Improve the Applicability of Multipath TCP to Wireless Access Networks*. IETF Internet Draft – work in progress. Jun. de 2011. URL: http://tools.ietf.org/html/draft-hampel-mptcp-applicability-wireless-networks-00.

[Hedr88] C.L. Hedrick. *Routing Information Protocol*. RFC 1058 (Historic). Updated by RFCs 1388, 1723. Internet Engineering Task Force, jun. de 1988. URL: http://www.ietf.org/rfc/rfc1058.txt.

[Iyen06] J.R. Iyengar, P.D. Amer y R. Stewart. «Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths». En: *Networking, IEEE/ACM Transactions on* 14.5 (oct. de 2006), págs. 951-964. ISSN: 1063-6692. DOI: 10.1109/TNET.2006.882843.

[Java07] Nastooh Taheri Javan y Mehdi Dehghan. «Reducing end-to-end delay in multi-path routing algorithms for mobile ad hoc networks». En: *Proceedings of the 3rd international conference on Mobile ad-hoc and sensor networks*. MSN'07. Springer-Verlag, 2007, págs. 715-724. ISBN: 3-540-77023-2, 978-3-540-77023-7. URL: http://dl.acm.org/citation.cfm?id=1781974.1782046.

[Khal13] R. Khalili, N. Gast, M. Popovic y J.-Y. Le Boudec. «MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution». En: *Networking, IEEE/ACM Transactions on* 21.5 (oct. de 2013), págs. 1651-1665. ISSN: 1063-6692. DOI: 10.1109/TNET.2013.2274462.

[Khal14a] Ramin Khalili, Nicolas Garbiel Gast, Miroslav Popovic y Jean-Yves Le Boudec. *Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP*. Inglés. Internet Draft. 4 de jul. de 2014. URL: https://tools.ietf.org/id/draft-khalili-mptcp-congestion-control-05.txt.

[Khal14b] Ramin Khalili, Nicolas Garbiel Gast, Miroslav Popovic y Jean-Yves Le Boudec. *Performance Issues with MPTCP*. Inglés. Internet Draft. 4 de jul. de 2014. URL: https://tools.ietf.org/id/draft-khalili-mptcp-performance-issues-06.txt.

[Khei14] Morteza Kheirkah, Ian Wakeman y George Parisis. «Multipath TCP in ns-3». En: *Proceedings of the 2014 Workshop on Ns-3 (Short paper)*. WNS3 '14. 2014.

[Lee01] Sung-Ju Lee y M. Gerla. «Split multipath routing with maximally disjoint paths in ad hoc networks». En: *Communications, 2001. ICC 2001. IEEE International Conference on*. Vol. 10. 2001, 3201-3205 vol.10. DOI: 10.1109/ICC.2001.937262.

[Lim] Maxine Lim y Josh Valdez. *MPTCP Wireless performance*. http://reproducingnetworkresearch.wordpress.com/2012/06/06/mptcp-wireless-performance-draft/.

[Losc06] V. Loscri y S. Marano. «A new Geographic Multipath Protocol for Ad hoc Networks to reduce the Route Coupling phenomenon». En: *Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd.* Vol. 3. 2006, págs. 1102-1106. DOI: 10.1109/VETECS.2006.1683005.

[Ludw03] R. Ludwig y M. Meyer. *The Eifel Detection Algorithm for TCP*. RFC 3522 (Experimental). Internet Engineering Task Force, abr. de 2003. URL: http://www.ietf.org/rfc/rfc3522.txt.

[Maga01] Luiz Magalhaes y Robin H. Kravets. «Transport level mechanisms for bandwidth aggregation on mobile hosts». En: *Network Protocols, 2001. Ninth International Conference on*. Nov. de 2001, págs. 165-171.

[Megh07] Natarajan Meghanathan. «Stability and Hop Count of Node-Disjoint and Link-Disjoint Multi-path Routes in Ad Hoc Networks». En: *Wireless and Mobile Computing, Networking and Communications, 2007. WiMOB 2007. Third IEEE International Conference on*. Oct. de 2007, págs. 42-42. DOI: 10.1109/WIMOB.2007.4390836.

[Megh10] Natarajan Meghanathan. «Performance Comparison of Link, Node and Zone Disjoint Multi-path Routing Strategies and Minimum Hop Single Path Routing for Mobile Ad Hoc Networks». En: *CoRR* abs/1011.5021 (2010).

[Moy98] J. Moy. *OSPF Version 2*. RFC 2328 (INTERNET STANDARD). Updated by RFCs 5709, 6549, 6845, 6860. Internet Engineering Task Force, abr. de 1998. URL: http://www.ietf.org/rfc/rfc2328.txt.

[MPiOS7] *Multipath TCP Support in iOS 7*. http://support.apple.com/kb/HT5977.

[MPKern] *Código fuente del protocolo MPTCP (GitHub)*. https://github.com/multipath-tcp/mptcp.

[MPLin] *MPTCP - Linux Kernel implementation*. http://multipath-tcp.org/.

[MPNet] *Multipath Networks*. http://www.multipathnetworks.com/.

[MPNS3] *Source code and documentation of the MPTCP implementation (ns-3.13)*. https://github.com/dgomezunican/multipath-ns3.13.

[MPRout] *Multipath Router - Super reliable Skype, Super HD Netflix*. https://www.indiegogo.com/projects/multipath-router-super-reliable-skype-super-hd-netflix.

[Nart08] T. Narten y H. Alvestrand. *Guidelines for Writing an IANA Considerations Section in RFCs*. RFC 5226 (Best Current Practice). Internet Engineering Task Force, mayo de 2008. URL: http://www.ietf.org/rfc/rfc5226.txt.

[Nguy11] Sinh Chung Nguyen y Thi Mai Trang Nguyen. «Evaluation of multipath TCP load sharing with coupled congestion control option in heterogeneous networks». En: *Global Information Infrastructure Symposium (GIIS), 2011*. 2011, págs. 1-5. DOI: 10.1109/GIIS.2011.6026698.

[Paas13] Christoph Paasch, Ramin Khalili y Olivier Bonaventure. «On the Benefits of Applying Experimental Design to Improve Multipath TCP». En: *Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. http://dl.acm.org/authorize?6960036. ACM, dic. de 2013.

[Paas14] Christoph Paasch y Olivier Bonaventure. «Multipath TCP». En: *Communications of the ACM* 57.4 (abr. de 2014). See http://queue.acm.org/detail.cfm?id=2591369, págs. 51-57.

[Perk03] C. Perkins, E. Belding-Royer y S. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing*. RFC 3561 (Experimental). Internet Engineering Task Force, jul. de 2003. URL: http://www.ietf.org/rfc/rfc3561.txt.

[Post81a] J. Postel. *Internet Protocol*. RFC 791 (INTERNET STANDARD). Updated by RFCs 1349, 2474, 6864. Internet Engineering Task Force, sep. de 1981. URL: http://www.ietf.org/rfc/rfc791.txt.

[Post81b] J. Postel. *Transmission Control Protocol*. RFC 793 (INTERNET STANDARD). Updated by RFCs 1122, 3168, 6093, 6528. Internet Engineering Task Force, sep. de 1981. URL: http://www.ietf.org/rfc/rfc793.txt.

[Raic09] Costin Raiciu, Damon Wischik y Mark Handley. «Practical Congestion Control for Multipath Transport Protocols». En: *UCL Technical Report* 6824 (ene. de 2009).

[Raic11] C. Raiciu, M. Handley y D. Wischik. *Coupled Congestion Control for Multipath Transport Protocols*. RFC 6356 (Experimental). Internet Engineering Task Force, oct. de 2011. URL: http://www.ietf.org/rfc/rfc6356.txt.

[Raic12] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure y Mark Handley. «How hard can it be? designing and implementing a deployable multipath TCP». En: *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. NSDI'12. USENIX Association, 2012, págs. 29-29. URL: http://dl.acm.org/citation.cfm?id=2228298.2228338.

[Rekh06] Y. Rekhter, T. Li y S. Hares. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271 (Draft Standard). Updated by RFCs 6286, 6608, 6793. Internet Engineering Task Force, ene. de 2006. URL: http://www.ietf.org/rfc/rfc4271.txt.

[Scha13] M. Scharf y A. Ford. *Multipath TCP (MPTCP) Application Interface Considerations*. RFC 6897 (Informational). Internet Engineering Task Force, mar. de 2013. URL: http://www.ietf.org/rfc/rfc6897.txt.

[Steg10] Tine Stegel, Janez Sterle, Urban Sedlar, Janez Bešter y Andrej Kos. «SCTP multihoming provisioning in converged IP-based multimedia environment». En: *Computer Communications* 33.14 (2010). Special Issue on Multimedia Networking and Security in Convergent Networking, págs. 1725-1735. ISSN: 0140-3664. DOI: http://dx.doi.org/10.1016/j.comcom.2010.03.036. URL: http://www.sciencedirect.com/science/article/pii/S0140366410001684.

[Stew07] R. Stewart. *Stream Control Transmission Protocol*. RFC 4960 (Proposed Standard). Updated by RFCs 6096, 6335, 7053. Internet Engineering Task Force, sep. de 2007. URL: http://www.ietf.org/rfc/rfc4960.txt.

[Tari09] Mohammed Tarique, Kemal E. Tepe, Sasan Adibi y Shervin Erfani. «Survey of multipath routing protocols for mobile ad hoc networks». En: *Journal of Network and Computer Applications* 32.6 (2009), págs. 1125-1143. ISSN: 1084-8045. DOI: http://dx.doi.org/10.1016/j.jnca.2009.07.002. URL: http://www.sciencedirect.com/science/article/pii/S1084804509001027.

[Waha06] S. Waharte y R. Boutaba. «Totally Disjoint Multipath Routing in Multihop Wireless Networks». En: *Communications, 2006. ICC '06. IEEE International Conference on*. Vol. 12. 2006, págs. 5576-5581. DOI: 10.1109/ICC.2006.255550.

[Wei13] Hung-Yu Wei, Jarogniew Rykowski y Sudhir Dixit. *WiFi, WiMAX and LTE Multi-hop Mesh Networks: Basic Communication Protocols and Application Areas*. 1st. Wiley Publishing, 2013. ISBN: 0470481676, 9780470481677.

[Wisc08] Damon Wischik, Mark Handley y Marcelo Bagnulo Braun. «The resource pooling principle». En: *SIGCOMM Comput. Commun. Rev.* 38.5 (sep. de 2008), págs. 47-52. ISSN: 0146-4833. DOI: 10.1145/1452335.1452342. URL: http://doi.acm.org/10.1145/1452335.1452342.

[Yang05] Yaling Yang, Jun Wang y Robin H. Kravets. «Interference-aware Load Balancing for Multihop Wireless Networks». En: UIUCDCS-R-2005-2526 (2005). URL: http://www.ideals.uiuc.edu/handle/2142/10974.

[Ye03] Z. Ye, S.V. Krishnamurthy y S.K. Tripathi. «A framework for reliable routing in mobile ad hoc networks». En: *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*. Vol. 1. 2003, 270-280 vol.1. DOI: 10.1109/INFCOM.2003.1208679.

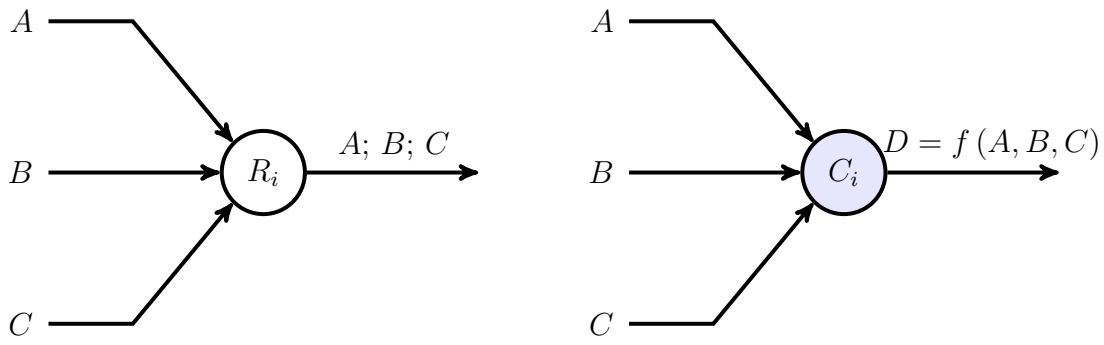
Bloque III

Network Coding

4

Introducción

Durante las últimas décadas se ha producido un crecimiento sin precedentes en la capacidad de los dispositivos de comunicación inalámbrica. tanto computacional como tecnológica. Con el paso del tiempo se ha venido observando una *miniaturización* de la tecnología, de manera que un equipo actual posee la potencia de dispositivos de hace tres-cuatro años con un tamaño mucho menor y con un consumo energético mucho más reducido. Gracias a esta evolución, el concepto clásico de red comienza a dar síntomas de obsolescencia. Esta premisa ha dado lugar a la búsqueda de alternativas que se adapten mejor a los cambios que han ido produciéndose con el paso del tiempo, aprovechando las posibilidades que surgen con los nuevos dispositivos.



(a) Esquema tradicional *store-and-forward*

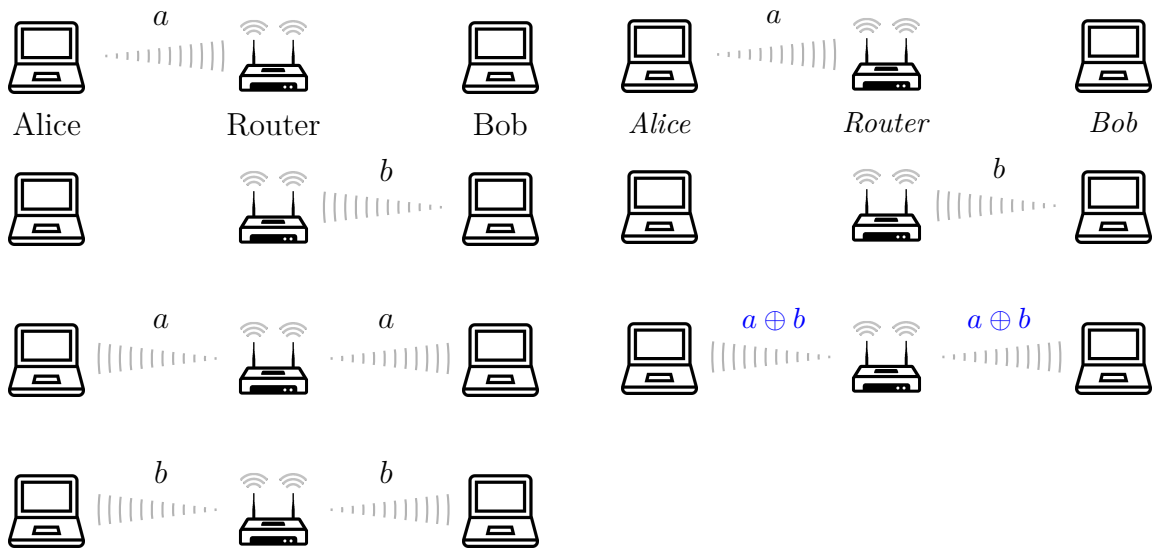
(b) Esquema *Network Coding*

Figura 4.1: Funcionalidad nodos intermedios esquema tradicional *Vs. Network Coding*

De entre todas ellas, una de las soluciones que más auge ha tenido en estos últimos años es la conocida como *Network Coding*. Su idea básica, resumida en una única frase, es aparentemente sencilla: *utilizar a los nodos intermedios como entidades activas que puedan procesar y transformar el contenido de los paquetes que circulan en una red*. Esta definición contradice a los esquemas de encaminamiento clásicos, en los que la función de estos dispositivos se limita a recibir un mensaje, buscar su siguiente destino en la red y reenviarlo. Para poder entender mejor la diferencia entre ambos esquemas, en [Fitz10] se

propone un ejemplo ilustrativo, resumido en la Figura 4.1, que explica el modo básico de operación seguido por las dos filosofías/paradigmas.

En la parte de la izquierda (Figura 4.1a) se representa el comportamiento de un encaminamiento basado en *almacenamiento y reenvío*; los tres paquetes entrantes "A", "B" y "C" son retransmitidos por R_i tras haberlos recibido previamente. En la solución *NC* (Figura 4.1b), el nodo C_i genera una salida "nueva" a partir de los paquetes que le llegan. Para ello, utilizará una serie de operaciones algebraicas (por norma general, combinaciones lineales) a partir de las que se construirá un mensaje "D".



(a) Encaminamiento tradicional (b) *Network Coding*

Figura 4.2: Ejemplo clásico *routing* tradicional *Vs. NC*

El ejemplo más clásico para ilustrar el comportamiento canónico del *NC* se representa en la Figura 4.2. En este escenario, tanto *Alice* como *Bob* desean intercambiar un mensaje entre ellos. Como no pueden comunicarse directamente, tienen que recurrir a un nodo intermedio que haga las veces de *router* y reenvíe los mensajes a su correspondiente destino. Según el esquema más tradicional (Figura 4.2a), cada uno de los nodos deberá contender por el acceso al canal para transmitir sus respectivos paquetes (i.e. 'a' y 'b'). Por su parte, el *router* almacenará los paquetes recibidos, comprobará en su tabla de rutas la dirección del siguiente salto y los reenviará. En total, serán 4 las transmisiones "físicas" necesarias para completar el proceso. Por otra parte, haciendo uso de las técnicas de *NC* (Figura 4.2b), el *router* podrá combinar la información de los paquetes recibidos, generando uno nuevo a partir de la operación *XOR* ($a \oplus b$) de ambos[1]. Al recibirlo, los nodos finales utilizarán los paquetes que tenían almacenados localmente ('a' en el caso de *Alice* y 'b' en el de *Bob*) para

[1]Como se describirá más adelante, existen múltiples alternativas a la hora de llevar a cabo la codificación/decodificación de la información.

recuperar la información dirigida a ellos. Así, si un nodo dispone del paquete *nativo*[2] 'a', puede realizar la misma operación *XOR* con el codificado y recuperar 'b': $a \oplus (a \oplus b) = b$. Se deduce que los nodos deberán almacenar una copia local de los paquetes transmitidos, ya que podrán ser utilizados en un futuro para las tareas de decodificación.

En definitiva, habrán sido necesarias 3 transmisiones, en lugar de las 4 del esquema clásico, con lo que, en condiciones ideales, se consigue un ahorro del 25 % del total de transmisiones necesarias. Además, esta ganancia dará lugar a varias ventajas adicionales, como un incremento en el *throughput*, una reducción de la energía consumida por los dispositivos, así como un incremento en la seguridad de la transmisión, ya que la información viaja "cifrada" por la red.

Los ejemplos previos han puesto de manifiesto la grandes ventajas asociadas al uso de este tipo de mecanismos de codificación, que en [Ho08] se resumen en cuatro puntos.

1. *Throughput*. Al reducir el número de transmisiones necesarias, el rendimiento o *throughput* se incrementará correspondientemente.
2. *Robustez*. Una de las principales desventajas de las comunicaciones inalámbricas es la presencia de errores causados por la transmisión sobre enlaces radio. En el esquema *NC*, los nodos intermedios adquieren un papel más protagonista, lo que les permitirá mitigar el impacto de estos errores aleatorios.
3. *Complejidad*. El uso de ciertas soluciones de *NC* sobre topologías de red complejas permitirá mejorar significativamente el comportamiento del sistema global, recurriendo a soluciones subóptimas.
4. *Seguridad*. Como se ha mencionado anteriormente, las técnicas de *NC* pueden aportar, de manera indirecta, un beneficio desde el punto de vista de la seguridad de las comunicaciones.

Como puede verse, el potencial ofrecido por este esquema es notable, dando lugar a un número de beneficios simultáneos. Así, las cuatro grandes ventajas mencionadas derivan en otras, como la reducción del consumo energético [Para13], factor que tiene una importancia capital en dispositivos móviles o portátiles.

Como prueba de su potencial se debe destacar el creciente interés de la comunidad científica en las técnicas de *NC*. Desde su propuesta inicial, fechada en el año 2000, el número de publicaciones relacionadas[3] ha ido aumentando año tras año, asentándose en los últimos 5 años en una cifra bastante significativa (cercana a las 1000 publicaciones), tal y como se muestra en la Figura 4.3.

[2] De ahora en adelante, se utilizará la notación de paquete *nativo* para referirse a los que no se encuentran codificados.

[3] Estas cifras se han obtenido en *Google Scholar*, buscando referencias con "Network Coding" y "Network Coded". No son, por tanto, cifras exactas, pero dan una idea de la popularidad de este tipo de técnicas.

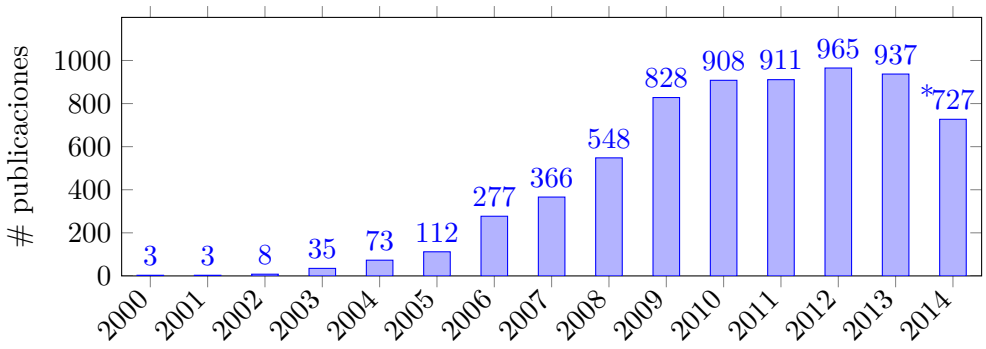


Figura 4.3: Número de publicaciones que hacen relación al *NC*. El * hace relación a que no todas las actas (*proceedings*) del año pasado (2014) han sido indexadas en el momento de escribir estas líneas (Febrero 2015)

Por otro lado y tras una etapa inicial más enfocada a una caracterización teórica [Ahls00; Koet01; Li03], la investigación más reciente tiene una naturaleza más práctica, tratando de encontrar los nichos en los que se pueda sacar mayor partido al uso de *NC*, pasando de resultados analíticos a implementaciones reales. En la Figura 4.4 se recogen algunas de las principales líneas de trabajo más importantes en el momento de escribir esta memoria.

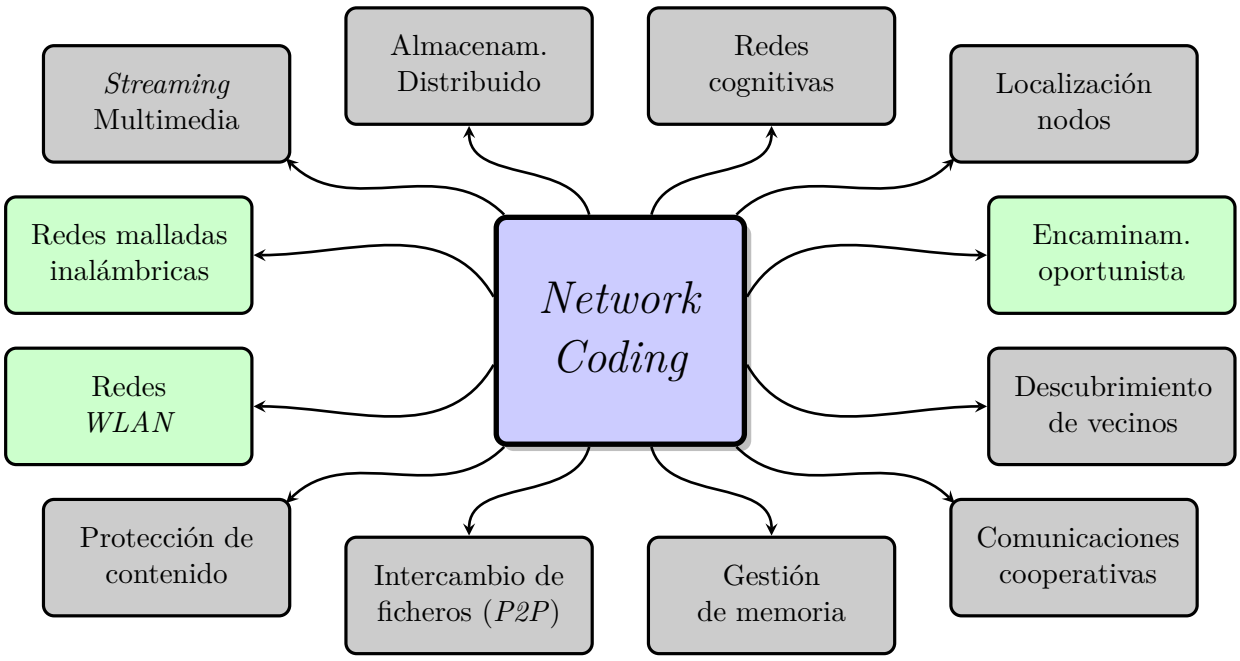


Figura 4.4: Aplicaciones con soluciones basadas en *NC*. En verde se encuentran aquéllas que presentan una mayor relación con el contenido que se va a tratar en esta Tesis

Como puede verse en el diagrama, hay una gran heterogeneidad. Por citar algunos ejemplos de aplicación de *NC*, se encuentran esquemas para la transmisión *multicast* de tráfico multimedia [Wu13], optimizando la utilización del ancho de banda en redes inalámbricas y mejorando la calidad del servicio sobre canales hostiles. También tiene una gran relevancia en técnicas de almacenamiento "en la nube" (*Cloud Computing*) [Dima10]; así, aprovechando la redundancia distribuida, se incrementa la fiabilidad del servicio. Otro de los grupos a destacar son los esquemas de intercambio de ficheros a través de redes *Peer-to-Peer (P2P)* [Li11a], en los que se pretende demostrar la validez de una de las premisas iniciales del uso de *NC*: *puede maximizar la tasa de los flujos en sesiones* multicast *sobre grafos dirigidos acíclicos*.

De entre todos estas aplicaciones prácticas, hay tres que tienen una mayor relación con el contenido que va a ser tratado con mayor detenimiento a lo largo de este bloque. Todas se basan en transmisiones sobre enlaces inalámbricos en los que las técnicas de *NC*, gracias a su habilidad para reducir el número de transmisiones y su potencial para simplificar la operación de los protocolos de encaminamiento, ofrece un mayor rendimiento con un coste energético inferior. Además de las típicas redes de área local inalámbricas, uno de los casos de uso más populares en los últimos años se basa en la utilización de redes malladas inalámbricas. Su uso permite, por ejemplo, proporcionar una vía alternativa económica para extender la cobertura de una red convencional. Estas topologías multi-salto también ofrecen beneficios a las comunicaciones *peer-to-peer* o colaborativas, reduciendo la congestión en puntos de conexión con redes externas, por ejemplo. Pero sin duda alguna, el elemento con mayor impacto en este tipo de técnicas es la capacidad de los nodos de recibir todas las transmisiones realizadas en su región de cobertura; esta "escucha oportunista" permitirá establecer múltiples rutas entre un nodo origen y su destino, dando lugar toda una línea de investigación con gran relevancia en la actualidad: el *encaminamiento oportunista*. Como se verá con más detalle (en el Capítulo 6), la búsqueda de las mejores combinaciones a la hora de transmitir la información desde múltiples dispositivos es un reto importante, teniendo que mantener un equilibrio entre la redundancia (para enmascarar los errores en canales hostiles) y la utilización del canal (intentando maximizar el *throughput*).

4.1 Estado del arte

El término *Network Coding* fue originalmente postulado por Rudolf Ahlswede *et al.* en [Ahls00]. El grupo liderado por este matemático alemán fue pionero al proponer modificar la noción de comunicación de paquetes que se había mantenido hasta el momento. En este trabajo, los autores demostraron matemáticamente que, si una red se modela como un grafo dirigido $\mathcal{G}(V, E)$, una fuente $s \in V$ puede comunicarse con un conjunto de receptores $T \subseteq V$ alcanzando la tasa *broadcast* h (siendo h el valor del mínimo corte entre s y $t \in T$) siempre y cuando se permita la recodificación de los paquetes en los nodos intermedios, según el teorema conocido como "*Max Flow-Min Cut*" [Papa82].

Después de este primer artículo, se dio paso a una etapa inicial marcada por el estudio analítico de las prestaciones de este tipo de técnicas. Así, Li, Yeung y Cai [Li03] demostraron que no es necesario recurrir a complejas operaciones para conseguir un rendimiento óptimo, sino que es suficiente con utilizar sencillas combinaciones lineales. Posteriormente, Koetter y Médard muestran en [Koet01] cómo encontrar los coeficientes apropiados para las operaciones de codificación y decodificación, a través de una combinación entre ciertos sistemas de ecuaciones polinómicas y la solución a problemas concretos. Al mismo tiempo, establecen una serie de teoremas para determinar la estrategia a seguir sobre diferentes tipos de escenarios. Tras este trabajo, surgieron otros [Sand03; Jagg05] que demuestran que, para tráfico *multicast*, los códigos lineales alcanzan las cotas superiores de rendimiento, pudiendo llevar a cabo las tareas de codificación y decodificación en tiempo polinomial. Dando un paso más allá, Ho *et al.* mostraron en [Ho03a] que la elección aleatoria de estos coeficientes lineales dentro de un cuerpo finito de Galois proporciona también la máxima capacidad en esquemas *multicast* con múltiples fuentes. A este nuevo enfoque se la conocerá como *Random Linear Network Coding (RLNC)*. Más tarde, otros trabajos cuantificaron analíticamente el impacto del uso de este tipo de cuerpos matemáticos en el rendimiento [Luca09; Trul11].

Tras esta fase inicial, se empiezan a conformar los dos grandes grupos en cuanto al criterio seguido para la codificación[4]: en el primero de ellos, conocido como *inter-flujo* o *inter-sesión*, se combina la información procedente de diferentes flujos en los nodos intermedios. La propiedad fundamental de este tipo de combinaciones es que se aumenta el *throughput* gracias a la reducción del número de transmisiones necesarias para intercambiar la misma cantidad de información. Por otra parte, aparece el *NC intra-flujo* o *intra-sesión*, donde cada una de los flujos de datos será tratado de forma independiente. En este caso, el primer proceso de codificación lo lleva a cabo el nodo fuente, mientras que los elementos intermedios *recombinan* los mensajes a medida que atraviesan la red.

Existe un tercera alternativa, más reciente, que trata de explotar las virtudes de las anteriores, combinándolas en una única solución *NC*. Este tipo de esquemas, definidos aquí como *híbridos*, mejoran las prestaciones de las anteriores. Más adelante se describirán algunos ejemplos con un mayor nivel de profundidad.

En la Figura 4.5 se resumen las principales categorías de *NC*, así como una serie de ejemplos para cada uno de los grupos.

El denominador común de todas las aproximaciones presentadas en la figura anterior es que todas ellas están adaptadas para trabajar sobre redes (malladas) inalámbricas. En el diseño de estas soluciones se tienen en cuenta las propiedades del canal radio, principalmente su naturaleza *broadcast*, que proporciona una redundancia adicional que puede ser utilizada para mejorar el rendimiento de las comunicaciones, incluso transmisiones *unicast* [Ho08]. De hecho, y como ya se ha discutido anteriormente, la aplicación de las

[4]No obstante, no es la única; por ejemplo, en [Qure14] la clasificación sigue otros parámetros diferentes, diviendo al *NC* en: aleatorio, vectorial y lineal.

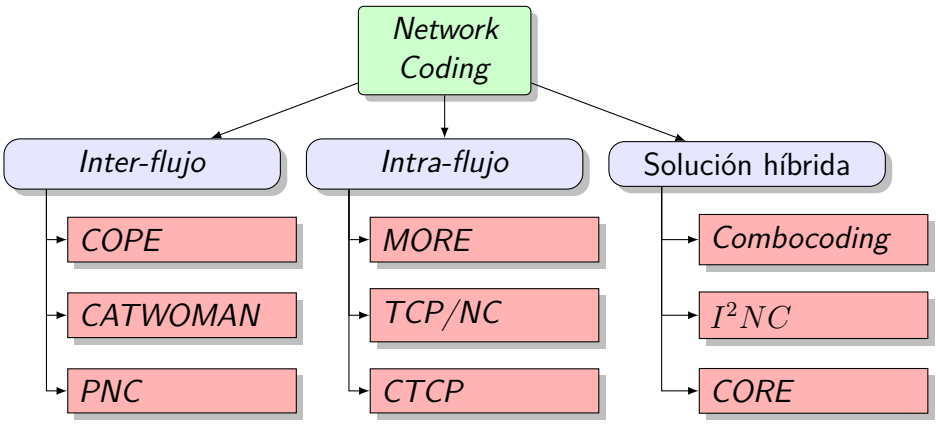


Figura 4.5: Clasificación de los protocolos de *NC*

técnicas de *NC* a este tipo de redes supone una de las ramas de investigación más activas en la actualidad, viéndose como un potencial elemento central de las futuras comunicaciones *5G* [Luca14b].

Dentro del primero de los grupos, uno de los trabajos más relevantes ha sido el de Katti *et al.* [Katt08], en el que los autores presentaron el protocolo *COPE*, la primera implementación de *NC* evaluada en escenarios reales (utilizando enlaces IEEE *802.11*). Añadiendo una capa intermedia entre los niveles de enlace y de red, los autores utilizan operaciones *XOR* para codificar los paquetes en los nodos intermedios de una red mallada. A través de tres conceptos clave (i.e. escucha oportunista, codificación oportunista y conocimiento de la información del vecino), los resultados ponen de manifiesto una mejora notable en el *throughput* sobre flujos *UDP*, incrementándolo hasta $3-4$ veces frente a los valores vistos con esquemas de transmisión clásicos *store-and-forward* para tráficos *unicast*. No obstante, cuando el protocolo de nivel de transporte es *TCP*, las prestaciones son menos prometedoras, con una ganancia del $2-3\,\%$ (los autores achacan esta disminución al efecto *terminal oculto* entre las estaciones). En estos casos, las colisiones no pueden ser enmascaradas por el esquema de retransmisiones del estándar *IEEE 802.11*, por lo que los mecanismos de control de congestión de *TCP* hacen que las fuentes no envíen la información a una tasa que garantice un uso óptimo del canal. Se puede decir que los flujos activos van a envían datos de una forma "desincronizada", impidiendo que los nodos intermedios "codificadores" encuentren *oportunidades de codificación* para combinar el contenido de diferentes paquetes.

Como puede verse, la interacción entre el *NC* y *TCP* puede generar problemas, por lo que existe un gran número de trabajos que tratan de mejorar el comportamiento de esta combinación sobre entornos inalámbricos [Nage10; Liu10; Liu12]. Con un enfoque similar al de la solución *inter-flujo* que será descrita en el Capítulo 5, Huang *et al.* presentan

en [Huan08] un estudio empírico con una versión simplificada[5] de *COPE*, en la que solamente los paquetes que viajan en sentidos contrarios se codifican entre sí. A diferencia de [Katt08], donde no se analizan los parámetros operacionales propios de los procesos de codificación/decodificación (e.g. configuración de los *buffers*, etc.), en este trabajo se evalúa la influencia del tiempo máximo de permanencia de un paquete *nativo* en un nodo codificador, a la espera de la aparición de una *oportunidad de codificación*; además, la información utilizada para decodificar los mensajes no se recibe a través de la *escucha oportunista*, sino que usa las copias locales que permanecen en el nodo tras su propia transmisión. Los resultados confirman que el comportamiento de *TCP* muestra una cierta dependencia con estos tiempos, ya que un valor alto asegura un mayor número de *oportunidades de codificación*, pero causa también un incremento del *RTT*. Los autores concluyen que, ajustando los valores de estos temporizadores dentro de unos márgenes adecuados, la ganancia de *throughput* se sitúa entre el 20 % y el 70 %, dependiendo de las condiciones del experimento.

La principal desventaja que se les puede achacar a estas dos referencias es que no dan importancia al impacto de los errores aleatorios en las transmisiones, en especial sobre los mecanismos de control de flujo de *TCP*. En [Hass10], los autores llevan a cabo un análisis analítico/heurístico (con el simulador `ns-2`) en el que modelan el comportamiento de *TCP* sobre escenarios sencillos; demuestran que los errores producidos en los enlaces promiscuos perjudican la ganancia de la codificación, llegando a afirmar que *en algunos casos la mejor solución es no utilizar NC*. También demuestran que la pérdida de sincronización por parte de los diferentes flujos *TCP* contribuye a la degradación del rendimiento total.

Una de las limitaciones restricciones de este primer grupo de técnicas es que deberán existir, al menos, dos conexiones activas en la red para obtener beneficios. También, existen otras alternativas que proponen combinar un flujo de datos *TCP* con aquel que transporta los reconocimientos correspondientes. En primer lugar, Scalia et al. presentan en [Scal07] *Piggycode*, donde los *ACKs* de *TCP* son forzados a recorrer el mismo camino que los segmentos de datos (en sentido inverso), consiguiendo: *(1)* reducir la complejidad a la hora de seleccionar las combinaciones que deben realizarse en los nodos codificadores (simplemente hay que detectar que pertenecen a la misma sesión de *TCP*); *(2)* disminuir la contención causada por los propios *ACKs*. Los autores, utilizando una topología matricial *sin errores en los enlaces*, consiguen mejorar el rendimiento en un 16 %, reduciendo en un factor de 10 el tiempo en el que un *ACK* alcanza el origen con respecto a un esquema tradicional *store-and-forward*. Partiendo de una base similar, Samuel David et al. afirman en [Samu08] que la combinación de los segmentos de datos con sus reconocimientos no mejora sustancialmente el rendimiento de *TCP* en redes *IEEE 802.11*, debido principalmente a la técnica de control de acceso al medio *CSMA/CA* utilizada. Proponen además una

[5] Comparte gran parte de su arquitectura, como la política de no modificar los niveles *TCP* o de enlace. Sin embargo, la mayor diferencia se encuentra en el modo de transmisión; mientras *COPE* se apoya en un modo *pseudo-broadcast* (necesita habilitar el conocido como "modo promiscuo" en los nodos), éste lo hace a través de un modo *pseudo-unicast*, donde la dirección destino de la cabecera *MAC* es la *broadcast*. Como contrapartida, al hacer esto se elimina la posibilidad de utilizar el esquema de retransmisiones de *IEEE 802.11*, por lo que los errores penalizarán en una mayor medida al rendimiento.

modificación distribuida del mecanismo de *back-off* apoyándose en un esquema de *feedback* similar al *Request To Send/Clear To Send (RTS/CTS)*. Los resultados obtenidos en una topología lineal de 10 nodos (*sin errores*) ponen de manifiesto una ganancia superior al 100 %.

También existen propuestas que tratan de reducir el impacto causado por los esquemas de retransmisión extremo a extremo que emplea *TCP*. Algunas proponen un mecanismo *ARQ* paralelo al usado por *TCP* (no interfiere directamente en su comportamiento), siendo el caso de soluciones *NC intra-flujo* [Sund08; Sun09; Soro09], donde el destino genera una confirmación cada vez que consigue recuperar el contenido de algún mensaje original. Otro planteamiento se basa en acercar la información al destino, implementando *buffers* intermedios, en los nodos que forman parte de la red. De esta forma, tras la petición de algún segmento concreto, los dispositivos podrán buscarlo y retransmitirlo, sin que sea necesario de atravesar hacia atrás toda la red. Ejemplos de soluciones que combinan este tipo de técnicas se pueden encontrar en [Li12; Zhen07].

Todas las contribuciones descritas hasta el momento utilizaban escenarios canónicos o topologías con un número limitado de nodos estáticos[6]. Sin embargo, para extender la aplicación de este tipo de técnicas a entornos más realistas deberán incorporarse protocolos que sean capaces de adaptarse a las condiciones cambiantes de una red mallada inalámbrica. Será necesario proporcionar los mecanismos necesarios para seleccionar aquellos nodos que van a ejercer de codificadores, así como las combinaciones que van a efectuar con la información que los atraviesa. Además, esta operación se debería llevar a cabo de una forma óptima y dinámica, minimizando la sobrecarga introducida por la señalización y el porcentaje de aciertos en los procesos de decodificación se acerque al 100 %.

Una técnica habitual es la fusión de los mecanismos de *NC* con soluciones de encaminamiento. Una muestra de ello es *Coding Applied To Wireless On Mobile Ad-hoc Networks (CATWOMAN)* [Hund12], una arquitectura que opera sobre el protocolo *Better Approach To Mobile Adhoc Networking (BATMAN)* [Neum08], diseñado para trabajar sobre redes *ad-hoc* inalámbricas. Los autores implementan un módulo completo que integra una solución *NC* similar a *COPE* [Katt08] con este protocolo de encaminamiento. Aprovechando los mensajes de señalización de este último, se intercambia información entre nodos vecinos para facilitar la identificación de *oportunidades de codificación*. Sin embargo, los autores limitan su análisis a escenarios simples, en los que no se explotan todas las ventajas de esta alternativa. Otra propuesta a destacar es la de Le *et al.* en [Le10], donde presentan el protocolo *Distributed Coding Aware Routing (DCAR)*, para solventar los principales problemas detectados en *COPE*, como su limitación a rutas fijas (no se adapta adecuadamente a escenarios dinámicos) y a que los paquetes se restrinjan a una "región de dos saltos". Gracias a la implementación de un protocolo de encaminamiento reactivo, basado en el estado del enlace, y una nueva métrica, *Coding-aware Routing Metric (CRM)*, se cuentan

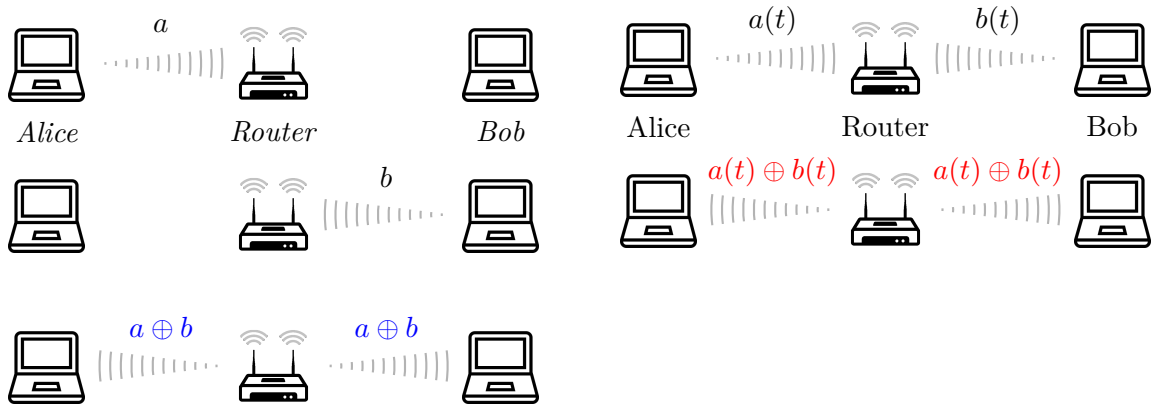
[6]Aunque *COPE* [Katt08] incorpora los procedimientos adecuados para ser empleado sobre cualquier tipo de escenario.

aquellas rutas que proporcionan un *throughput* más elevado. El protocolo detecta la mejor combinación de nodos codificadores, discriminando las situaciones en las que el uso de *NC* es positivo; en caso contrario, establece que la mejor opción es no codificar la información. Posteriormente, Guo *et al.* [Guo11] cuestionan las condiciones propuestas en [Le10] cuando existen varios puntos de intersección (articulación) en las rutas, afirmando que no son suficientes. Como solución, proponen una nueva métrica para seleccionar las rutas, *Free-ride Oriented Routing Metric (FORM)*, capaz de explotar un mayor número de *oportunidades de codificación* en condiciones más generales, sin importar el número de flujos activos.

Uno de los principales objetivos de este bloque de la Tesis es encontrar las condiciones (necesarias y suficientes) para que el uso de *NC* sea beneficioso en un escenario concreto. Existen trabajos similares, como [Tras06], en el que se presentan dos alternativas (subóptimas) para la construcción práctica de códigos (combinaciones en los nodos intermedios) sobre escenarios aleatorios cableados, de manera que el *NC* proporciona un remedio para mitigar la congestión. Se presenta una primera aproximación de carácter lineal, con un menor coste computacional y con un mayor número de variables para incrementar la flexibilidad a la hora de definir la función objetivo. En una versión más compleja, empleando técnicas de programación "entera" (problema *NP* completo), se identifican más restricciones. Con cualquiera de estas técnicas, las mejoras obtenidas no son muy elevadas, adelantando que las ganancias podrían ser superiores sobre redes malladas inalámbricas. En un segundo ejemplo [Para10], ParandehGheibi *et al.* cambian el enfoque, planteando un problema de optimización distribuido, en el que los nodos sacan partido de la información proporcionada por sus vecinos. Con esta formulación, los autores pretenden maximizar el número de oportunidades de codificación para reducir el coste total de la transmisión. Sin embargo, podrían generarse situaciones de congestión en áreas concretas de la red.

Desde una aproximación diferente, existe otra manera de interpretar un esquema de codificación *inter-flujo* en el que la mezcla de información no se lleva a cabo a nivel de símbolo o de *bit*, sino que se realiza a nivel físico (esto es, a través del procesado de la señal que se propaga por el medio radio). En estos casos, conocidos como *Physical layer Network Coding (PNC)* [Zhan06; Naze11] o *NC* analógico [Katt07], la comunicación puede beneficiarse de la interferencia. El ejemplo más básico se describe en la Figura 4.6, que compara el número de transmisiones necesarias en el escenario de *Alice y Bob* en los casos de soluciones *NC inter-flujo* clásicas (Figura 4.6a) o de capa física (Figura 4.6b). Cuando se emplea una solución de capa física, el *router* es capaz de interpretar la superposición de las ondas electromagnéticas correspondientes a las transmisiones correspondientes a *Alice* y a *Bob*. Se buscará por tanto que ambos envíos se realicen simultáneamente. En una segunda transmisión, se reenviará la señal "combinada", que será decodificada en ambos destinos, tras únicamente dos accesos al canal[7].

[7]Las técnicas empleadas en este tipo de *NC* se alejan significativamente de los conceptos tratados en esta Tesis, por lo que no se profundizará en cómo se llevan a cabo estas operaciones.



(a) *NC* clásico (b) *NC* de capa física

Figura 4.6: *NC* clásico *Vs.* *NC* de capa física

Teóricamente, estos mecanismos consiguen duplicar el rendimiento de transmisiones tradicionales, mejorando en un 50 % el proporcionado por las técnicas de *NC* a nivel de paquete. Como aspecto más negativo, no pueden aplicarse sobre ninguna tecnología existente, sino que tendrán que utilizar sus propias soluciones de capa física (y de enlace).

Con el paso del tiempo se ha ido comprobando que la complejidad de la implementación de las técnicas *inter-flujo* no se ve compensada con mejoras de rendimiento acordes, sobre todo cuando la calidad de los enlaces no es ideal. La tendencia que más se observa en la literatura es la de migrar paulatinamente hacia soluciones de codificación *intra-flujo*, que no requiere complicados mecanismos para identificar *oportunidades de codificación*, además de eliminar otros tantos problemas, como la sincronización entre flujos, el alto *overhead* debido a los reportes periódicos, etc. En este caso, el primer paso para la codificación lo dan los propios transmisores, que van generando diferentes combinaciones lineales de los paquetes almacenados[8] de un mismo flujo, enviando mensajes codificados a los niveles inferiores. Aunque existan soluciones que utilizan mecanismos deterministas para generar los coeficientes lineales usados en las operaciones de codificación, la tendencia más habitual es *seleccionarlos aleatoriamente* entre los elementos de un cuerpo finito de la forma $GF(2^q)$. Por su parte, los nodos destino deberán resolver el sistema de ecuaciones construido a partir de los paquetes codificados para recuperar la información original. Además, los nodos intermedios recibirán estos mensajes codificados, pudiendo *recombinar* su contenido, para generar nuevas combinaciones lineales.

[8]Mientras que ciertas soluciones optan por la utilización de bloques (o *generaciones*, como se les suele denominar) de tamaño fijo [Chac07; Kim12a], otras eligen la utilización de ventanas deslizantes [Sund11; Wu15], que reducen el coste computacional de las operaciones de decodificación, así como la sobrecarga que conlleva transportar la información de los coeficientes utilizados en la cabecera. En una tercera vía se posicionan los esquemas que son capaces de decodificar los mensajes de forma "inmediata" [Li11b].

No obstante, esta generación aleatoria de coeficientes puede dar lugar combinaciones lineales nulas. Dicho en otras palabras, coeficientes a través de los cuales un receptor no pueda extraer ninguna información útil. La reacción antes estos casos es unívoca, descartando el mensaje recibido. Existen varios trabajos que evalúan analíticamente el impacto de estas operaciones [Luca09; Trul11], cuantificando las probabilidades de que los valores generados a partir de procesos aleatorios sean linealmente independientes a los recogidos anteriormente, en función del tamaño de los bloques y del orden de los cuerpos finitos utilizados como base para la elección de los coeficientes. Junto a este tipo de análisis, existen otros estudios más avanzados que modelan el rendimiento de este tipo de técnicas (tanto en términos de *throughput* como de retardo) sobre entornos inalámbricos [Swap13; Zhan13].

Al incrementar el tamaño del bloque y el orden del cuerpo de Galois se pueden aumentar las probabilidades de generar independientes pero, al mismo tiempo, crecerá el coste computacional asociado a las operaciones de codificación/decodificación. Existen trabajos que evalúan y comparan el rendimiento para varias combinaciones de estos parámetros en diferentes dispositivos comerciales, como en [Para13], donde los autores comprueban empíricamente las afirmaciones postuladas en [Pede13], que concluye que el cuerpo $GF(2^{32}-5)$ se corresponde con la configuración óptima en términos de complejidad, ya que requiere una búsqueda basada en tablas.

El primer trabajo que hace una referencia práctica a este tipo de esquemas es el protocolo *MAC independent Opportunistic Routing (MORE)* [Chac07]; sus autores implementan una capa adicional (situada entre los niveles de enlace y de red) que combina las funcionalidades de un esquema *RLNC intra-flujo* con un protocolo de encaminamiento oportunista para adaptarse a las particularidades del medio inalámbrico. Los autores se centran en comparar el rendimiento con otras técnicas de encaminamiento oportunista, como *Extremely Opportunistic Routing (ExOR)* [Bisw05], demostrando que la operación conjunta con *NC* mejora notablemente el rendimiento, ya que la generación de combinaciones lineales en los nodos intermedios consigue reducir la redundancia vista cuando reenvían la información que reciben. Sin embargo, no se presta atención a la relación que puede tener *MORE* con los protocolos de nivel de transporte, como *TCP*; así, el retardo asociado a la transmisión de grandes bloques podría penalizar significativamente al *RTT*, perjudicando los mecanismos de control de congestión del propio *TCP*. Otro de los problemas que tendría la operación conjunta de *MORE* con *TCP* sería la utilización de varias rutas (a través de las técnicas de encaminamiento oportunista), lo que provocaría la llegada de paquetes de forma desordenada, afectando al rendimiento de *TCP*, que respondería con la generación de *ACKs* duplicados.

Otra alternativa interesante es el protocolo *TCP/NC*, propuesto por Sundararajan *et al.* [Sund11], que se basa en una capa situada entre los niveles de red y transporte, manteniendo una versión sin modificar de *TCP*, para aprovechar sus mecanismos de control de flujo y congestión. Además, para adaptarse mejor a ellos, *TCP/NC* no considera la codificación de bloques completos. Su principal aportación es la incorporación de un sistema de confirmación propietario de la entidad *NC*, en el que se reconoce la llegada de nuevas

unidades de información (vectores linealmente independientes). Con este esquema de reconocimientos se consigue solventar el problema del encaminamiento oportunista, ya que, ante cualquier combinación lineal "nueva", el receptor generará inmediatamente el *ACK* que confirma al origen la recepción correcta del segmento. En [Sund11] se ofrece una visión más práctica de su comportamiento, demostrando una gran ganancia con respecto al uso aislado de *TCP*, incluso con presencia de errores en el canal (aunque los valores máximos de *FER* no superan el 25 %, cifras relativamente bajas en comparación con las utilizadas en esta Tesis). Existe también un trabajo paralelo [Kim12b], que modela analíticamente el comportamiento del protocolo en función de la tasa de errores, el tiempo de retorno (*RTT*), el tamaño máximo de la ventana de congestión y el tiempo máximo de la conexión. Sin embargo, no se profundiza en la posibilidad de que los nodos intermedios recombinen los paquetes recibidos, asignando el papel de *routers* codificadores a aquellos nodos cuyos enlaces sean de baja calidad.

Sobre las ideas expuestas en [Sund11] aparece más tarde el protocolo *Network Coded TCP (CTCP)* [Kim12a], proponiendo una solución mejorada de *TCP/NC*, más eficiente y robusta. En este caso el módulo *NC* se sitúa en la capa de aplicación, teniendo por debajo a *UDP* (la combinación de éste con un esquema *RLNC* ofrece un servicio confiable, equivalente a *TCP*). Al contrario que en *TCP/NC*, aquí no se emplea ningún mecanismo de ventana deslizante, sino que los datos (ficheros o flujos) se dividen en bloques de tamaño constante (negociado en una etapa previa a la transmisión de datos), con paquetes de longitud fija. Las operaciones de decodificación serán más costosas al aplicarlas sobre un bloque completo, aunque es posible llevar a cabo un control más preciso del retardo. Además, incorpora unos mecanismos de control de flujo basados en la emisión de *tokens* para regular la tasa de transmisión de las fuentes en función de las condiciones dinámicas del canal inalámbrico. Al diseñar el módulo para trabajar en el nivel de aplicación, el protocolo se puede integrar de manera inmediata en cualquier dispositivo. La pruebas realizadas con tráfico real muestran una clara mejoría con respecto al protocolo *TCP*, especialmente cuando las condiciones del canal son hostiles. Aunque hayan extendido el trabajo para permitir multiplexar el tráfico a través de múltiples rutas e interfaces [Kim12c], la solución, al estar situada en el nivel de aplicación, tiene un enfoque extremo a extremo, por lo que no permite la recodificación en nodos intermedios. Así, esta propuesta se acerca más a los esquemas de codificación fuente, por ejemplo, los códigos *Digital Fountain* [Byer98].

Si bien es cierto que la utilización de este tipo de técnicas va ganando popularidad con respecto a la alternativa *inter-flujo*, también presentan una serie de carencias, entre las que destacan: *(1)* la gran carga computacional asociada a las operaciones de codificación/recodificación/decodificación y *(2)* la sobrecarga derivada del transporte de los coeficientes que se han generado para combinar los paquetes *nativos*. Para solucionar el primero de los problemas, se han propuesto soluciones que optan por el uso de ventanas deslizantes o de decodificación "en vuelo", u otras que reducen la densidad de las matrices a través de la división de las operaciones de codificación y decodificación en distintas etapas [Luca14a; Yang14]. Para la segunda limitación, se suele fomentar la la utilización de bloques de menor

tamaño (o dividirlos) y el empleo de ventanas deslizantes más pequeñas; también, como en uno de los protocolos ya presentados [Kim12a], también se ha recomendado la utilización de un generador de números *pseudo-aleatorios*, permitiendo a los receptores generar los mismos coeficientes gracias al uso de la misma semilla.

Una vez descritas las contribuciones más relevantes en cada una de las dos alternativas que se han identificado, merece la pena destacar algunos ejemplos de soluciones duales que combinan ambos esquemas. En primer lugar, Chen et al. presentan en [Chen11] *Combocoding*, que mezcla un esquema de codificación *interflujo* basado en *Piggycode* [Scal07], que combinaba el flujo de datos de *TCP* con sus reconocimientos, con otro *intra-flujo*, implementado por los mismos autores y conocido como *Pipeline Coding* [Chen10]. Esta versión de *RLNC* presenta ciertas particularidades, como la codificación incremental (codifica a medida que el *buffer* del transmisor se va llenando) o la utilización de un factor de redundancia, limitando el número de transmisiones por bloque. De esta forma, la decodificación podrá ser gradual (la matriz de coeficientes será triangular inferior) y su coste se verá notablemente reducido. No obstante, sólo tiene en cuenta los errores en el canal como fuente de pérdidas, despreciando la posibilidad de que las combinaciones recibidas tengan que ser descartadas. En una campaña de simulación sobre escenarios simples, los autores comparan el rendimiento de varias configuraciones de su solución (esquemas por separado y operando en conjunto) con una transmisión clásica *TCP*. Los resultados muestran un mejor comportamiento en condiciones ideales cuando actúa el protocolo *inter-flujo* en solitario (la redundancia causada por el esquema *intra-flujo* se traduce en una contención adicional); sin embargo, se aprecia que, a medida que la tasa de pérdida de paquetes aumenta, el protocolo *Combocoding* ofrece una mayor robustez, apoyado en gran medida en las propiedades de corrección de errores de los mecanismos de codificación aleatoria. Otra aproximación es I^2NC [Sefe11], ya que los autores utilizan una nueva filosofía a la hora de implementar el módulo de *NC*. En este caso, sitúan una primera capa *intra-flujo* entre los niveles de red y de transporte, encargada de enmascarar los errores derivados de la transmisión; la segunda (*inter-flujo*) se localiza entre los niveles de enlace y de red. Al estar esta última basada en *COPE* [Katt08], los nodos intermedios combinarán los paquetes de los diferentes flujos con operaciones *XOR* pero, gracias a la redundancia introducida por la primera etapa, no será necesario intercambiar mensajes de señalización, reduciendo la sobrecarga. A través de una función de utilidad propia, los autores tratan de maximizar el porcentaje de paquetes combinados. Sus resultados (basados en simulaciones sobre topologías canónicas) muestran mejoras significativas en el rendimiento al compararlo con el del propio *COPE* o con esquemas de transmisión *store-and-forward* tradicionales. Finalmente, una contribución más reciente es *CORE*[9] [Krig13], que vuelve a combinar un esquema *RLNC intra-flujo* con una solución *inter-flujo* similar a *COPE*. La principal novedad en este trabajo es la detección de puntos de intersección entre los diferentes flujos para combinar sus datos a través de operaciones *XOR*. En situaciones normales, las rutas entre diferentes conexiones son tratadas de manera independiente, mientras que con *CORE* los nodos intermedios tendrán la

[9]No confundir con *CORE* [Yan10], un protocolo de encaminamiento oportunista que opera con una solución de *NC inter-flujo* para transmisiones sobre redes malladas inalámbricas.

capacidad de descubrir la aparición de oportunidades de codificación. Además de reportar notables mejoras en el rendimiento con respecto a alternativas como *COPE* o soluciones *store-and-forward*, incluso sobre canales ideales, los resultados ponen de manifiesto uno de los mayores problemas de las técnicas *inter-flujo*, ya que su rendimiento se deteriora significativamente cuando los enlaces *promiscuos* son de baja calidad.

Es necesario destacar la existencia de diferentes proyectos de investigación que tienen al *NC* entre sus líneas de actuación más relevantes. Por citar algunos ejemplos, dentro del *Séptimo Programa Marco* de la Comisión Europea, *N-CRAVE*[10], finalizado en diciembre del 2010, es un proyecto completamente dedicado a mejorar la robustez del *NC* en redes inalámbricas. Fruto de este trabajo surgió *NEtwork COding simulator (NECO)* [Ferr09], una herramienta de simulación de alto rendimiento para la evaluación del comportamiento de diferentes protocolos de *NC*. Actualmente (marzo de 2015) se encuentran activos los proyectos *Dense Cooperative Wireless Cloud Network (DIWINE)* [DIWINE] y *COMANDER* [COMANDER], centrados en la utilización de estos mecanismos sobre grandes redes malladas inalámbricas y en la convergencia *Fiber Wireless (FiWi)*. Además, a este interés se han unido grandes organizaciones, como el *Internet Research Task Force (IRTF)*, formando un grupo de trabajo centrado en la utilización de técnicas de *NC* [IRTF].

Una de las discusiones más habituales plantea las dudas acerca de la complejidad asociada a las tareas de codificación y decodificación, cuestionando si sería posible utilizar una solución completa de *NC* sobre dispositivos reales. Pueden encontrarse en la literatura diferentes trabajos que han demostrado que es sí es posible integrarlas en teléfonos móviles, para transmisiones de vídeo *multicast* [Para13; Ving11]. Unos años antes, Microsoft mostró un gran interés por estos mecanismos y mantuvo durante mucho tiempo activo el proyecto *Avalanche* [Yeun07], generando un prototipo para la distribución de ficheros en una red *peer-to-peer* con *NC* como elemento central de la arquitectura.

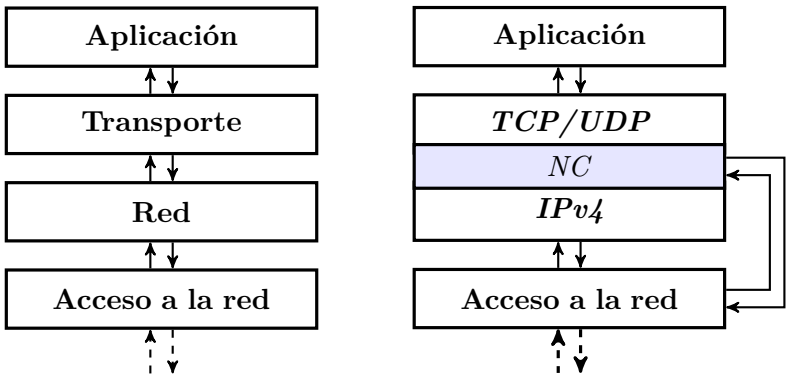
Antes de concluir, resulta interesante incluir una breve visión de todas aquellas implementaciones *NC* que se pueden encontrar en el marco del simulador `ns-3`. Partiendo de la definición del protocolo *COPE*, Yang Chi et al. desarrollaron *Yet Another Network Coding Implementation (YANCI)* [YANCI], una versión simplificada de esta solución de *NC inter-flujo*. No existe sin embargo ninguna publicación o documento que recoja algún resultado de su comportamiento; además, en los ejemplos propuestos no se tiene en cuenta la presencia de errores. Los mismos autores firman *MUlti Radio with network COding (MURCO)* [Chi12; MURCO], una evolución del anterior que permite la transmisión simultánea a través de múltiples interfaces. Los resultados mostrados presentan mejoras significativas en cuanto a *throughput* y retardo extremo a extremo. No obstante, al igual que en el trabajo anterior, no contemplan la posibilidad de que se pierda información durante los intercambios de información. De forma adicional, y aunque no se trata de desarrollos sobre la plataforma del simulador, existen herramientas que permiten enlazar a `ns-3` como una

[10]No se incluye referencia porque la *web* del proyecto ha dejado de estar activa.

librería externa, para emular el comportamiento del *NC* sobre diferentes tipos de redes, como `Kodo` [Kodo] o `NECO` [Ferr09].

4.2 Módulo implementado en la plataforma `ns-3`

En el marco de la presente Tesis se han desarrollado *dos aproximaciones diferentes* con dos objetivos comunes: *(1)* mejorar las pobres prestaciones ofrecidas por el protocolo *TCP* cuando se utiliza sobre canales hostiles; *(2)* ofrecer un servicio fiable, que garantice la entrega ordenada de toda la información generada en las aplicaciones transmisoras. La primera de ellas combinará el tráfico *TCP* con un esquema *NC inter-flujo*, codificando de manera conjunta la información perteneciente a diferentes sesiones de transporte, mientras que en la segunda aproximación utilizará un esquema que compensará la sencillez del protocolo *UDP* con la robustez de un sistema de codificación lineal aleatoria o *Random Linear Coding (RLC)* para proporcionar, en conjunto, un servicio fiable, con un rendimiento superior al ofrecido por la dupla *TCP*/encaminamiento tradicional, especialmente cuando las condiciones de la red son adversas.



(a) Modelo *TCP/IP* estándar (b) Modelo *TCP/IP* modificado

Figura 4.7: Integración del nivel *NC* en el módulo *TCP/IP*

Ante la naturaleza tan dispar que presentan ambas propuestas parece difícil encontrar una solución única; sin embargo, gracias a la filosofía de la programación orientada a objetos del simulador `ns-3`, se ha diseñado una arquitectura conjunta que permite la reutilización de buena parte de la infraestructura en ambos casos. Como primer paso, ha sido necesario decidir el nivel de la pila de protocolos en el que operan las soluciones propuestas. En la literatura no hay unanimidad clara en la ubicación del nivel de *NC* en la pila *Open System Interconnection (OSI)*, habiendo soluciones que lo sitúan en el nivel físico [Naze11], entre las capas de enlace y de red [Katt08], de red y de transporte [Sund11], entre los niveles 2 y 4 [Chac07], o incluso en la capa de aplicación, como es el caso del almacenamiento distribuido [Dima10]. En consecuencia, y considerando que el principal objetivo para el que fueron concebidas ambas propuestas fue ofrecer un servicio fiable sobre enlaces

inalámbricos, mejorando el rendimiento obtenido por *TCP*, en este trabajo se ha optado por ubicar una entidad de *NC* entre los niveles de red y de transporte, como se muestra en la Figura 4.7. Así, se mantiene la información de las cabeceras de los niveles inferiores en claro, permitiendo la operación transparente de los protocolos subyacentes (e.g. encaminamiento, *IP*, etc.). Tras el proceso de codificación, tanto los datos provenientes del nivel de aplicación (*payload*) como las cabeceras *TCP* (o *UDP*) se encontrarán codificados dentro de la unidad de datos de protocolo generada en la capa *NC*.

Sin embargo, tal y como muestra la figura anterior, existen una serie de conexiones *cross-layer* entre la capa de *NC* y los niveles inferiores. Entre ellas destaca la utilización de un *modo promiscuo* en todos los nodos, esencial para poder explotar la naturaleza *broadcast* del medio inalámbrico. En concreto, cualquier nodo podrá ejercer las tareas de codificación/decodificación, por lo que deberá ser capaz de procesar todos los paquetes que reciba, ya sea de manera directa o no (*escucha oportunista* o *promiscua*). Junto con ésta se han ido añadiendo otras funcionalidades *cross-layer* (inyección de paquetes en función de la tasa de salida del *buffer* del nivel *MAC IEEE 802.11*, borrado selectivo de su contenido, etc.) a raíz de las necesidades impuestas por los propios protocolos durante su desarrollo, que serán discutidas más adelante.

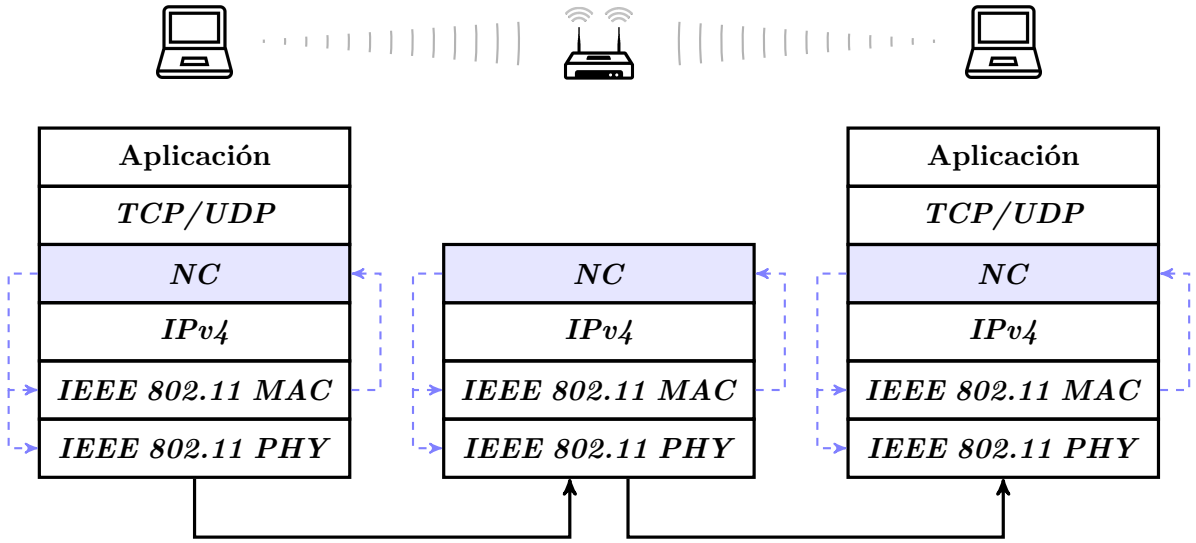


Figura 4.8: Transmisión extremo a extremo con *NC*

Volviendo al ejemplo de "*Alice* y *Bob*", en la Figura 4.8 se representan los niveles (protocolos) por los que fluye la información durante una comunicación entre ambos usuarios, asumiendo que la transmisión se realiza a través de interfaces *IEEE 802.11*. En un esquema estándar (*store-and-forward*), los nodos intermedios sólo procesarían el contenido de los paquetes recibidos hasta el nivel de red (*IP*), cuando la dirección de nivel de enlace coincide con la del terminal en cuestión[11]. En estos casos, el nodo comprobaría en

[11]Recuérdese que en un direccionamiento *unicast*, aquellas tramas cuya dirección *MAC* destino no coincida con la del nodo receptor serán descartadas con carácter inmediato.

sus tablas de rutas la dirección del siguiente salto, reenviando la trama hacia los niveles inferiores. Tal y como se aprecia en la figura, en el esquema *NC*, el nuevo módulo tendrá una especial relevancia en los nodos intermedios. Los paquetes llegan desde dos caminos diferentes: en primer lugar, en el caso de formar parte de la ruta activa entre los nodos origen y destino, el nivel *IP* se encargará de enviarlos hacia la capa *NC* (la dirección *MAC* destino es la del propio nodo); por otra parte, el *modo promiscuo* habilitado en los nodos hará que todos los paquetes escuchados "oportunistamente" sean redirigidos desde el nivel de enlace a la capa *NC*. Posteriormente, el protocolo en cuestión (ya sea *inter* o *intra flujo*) se encargará de llevar a cabo las operaciones pertinentes sobre los paquetes (almacenarlos temporalmente, combinarlos, eliminarlos, etc.), en función de sus criterios de codificación. Por otra parte, es importante destacar que este tipo de técnicas no se aplica exclusivamente en los nodos intermedios de una red, sino que los extremos también realizarán una serie de tareas, como la agregación o extracción de cabeceras, la decodificación para recuperar la información original (en el destino) o la codificación en el nodo fuente (solamente en el protocolo *intra-flujo*), entre otras.

5

Mecanismos de Network Coding Inter-flujo

Para afrontar el pobre comportamiento de *TCP* sobre enlaces inalámbricos, se presenta en este Capítulo un protocolo *NC inter-flujo*, que combina segmentos de diferentes sesiones de nivel de transporte. De este modo, se consigue incrementar la cantidad de información transmitida en cada envío, reduciendo la contención del canal inalámbrico (serán necesarias menos transmisiones), con el consecuente incremento del rendimiento global del sistema. Para ello, se seguirá la estructura descrita seguidamente.

- En primer lugar, se presentará detalladamente la versión inicial del protocolo *inter-flujo* diseñado, cuya operación básica se resume en la codificación de paquetes pertenecientes a diferentes conexiones, sin tener en cuenta ninguna información adicional (e.g. segmentos ya confirmados por el receptor). En su rendimiento cobra una especial relevancia la correcta configuración de los parámetros que definen el comportamiento de los elementos involucrados en el proceso de codificación, como por ejemplo el tamaño de un *buffer* que utilizarán los nodos codificadores para esperar la aparición de una oportunidad de codificación, así como el tiempo máximo que cada segmento puede permanecer en el mismo.

- Posteriormente se añadirá al esquema la capacidad de *encapsular* los reconocimientos *TCP* que viajan por la red en sentido inverso al flujo de datos. La transmisión de un *ACK* de *TCP* genera una sobrecarga apreciable, sobre todo en canales basados en contienda. Para minimizar su impacto, se combinarán tantos reconocimientos como sea posible en un único paquete, manteniendo la información de los *ACKs* encapsulada dentro de la cabecera del protocolo *NC*, para poder ser extraída en los destinos correspondientes.

- También se modificará el protocolo para monitorizar el tráfico que atraviesa los nodos. De esta manera, se podrán establecer mecanismos que permitan un control en tiempo real del estado de la transmisión de cada uno de los flujos. Gracias a esta nueva versión del protocolo, se conseguirá mitigar el impacto de los mecanismos de control de flujo y congestión inherentes al protocolo *TCP*, reduciendo notablemente el número de retransmisiones producidas en el nivel de transporte. Para ello, se incorporará un

nuevo esquema *ARQ* en el nivel de *NC*, basado en mantener temporalmente paquetes (*caching*) en los nodos intermedios, lo que permitirá recuperar la información faltante sin necesidad de solicitar una retransmisión extremo a extremo.

- Otro de los aspectos fundamentales de un esquema de *NC* es la elección de los nodos que codificarán los paquetes, así como la identificación en cada instante de las combinaciones de segmentos *nativos* que van a ser codificados en un único paquete, garantizando su decodificación en el mayor número de nodos posible. En otras palabras, el objetivo básico será el de maximizar el número de paquetes codificados, manteniendo la tasa de éxito de decodificación lo más alta posible. En escenarios sencillos, la elección de los nodos codificadores puede resultar trivial; no obstante, en topologías más complejas, la solución no será tan sencilla. Se presentará un algoritmo adaptado al esquema utilizado que, utilizando la teoría de grafos, permitirá determinar: *(1)* la idoneidad de utilizar un esquema de *NC* en despliegues aleatorios y *(2)* la lista de potenciales *nodos codificadores*, responsables de combinar la información de los diferentes flujos que los atraviesan.

- El comportamiento de los diferentes esquemas se estudiará mediante extensas campañas de simulación sobre `ns-3`, utilizando varias configuraciones para los canales inalámbricos y comparando los resultados con el rendimiento del protocolo *TCP* nativo.

5.1 Descripción del protocolo

Tomando como referencia algunas de las principales características de una de las propuestas *inter-flujo* más populares, *COPE* [Katt08], se ha diseñado un nuevo protocolo que combina la información correspondiente a diferentes sesiones de nivel de transporte (o flujos). Para ello, se ha introducido una nueva capa, situada entre los niveles de red y de transporte. Se ha optado por *no modificar* ninguno de los niveles anexos a la entidad de *NC*, por lo que su comportamiento permanecerá inalterado[1], asegurando de tal manera una transparencia total en la operación del nuevo módulo. En un primer acercamiento, se ha prescindido además de cualquier mensaje de señalización, con lo que la capa de *NC* se limitará exclusivamente a los procesos de codificación en los nodos intermedios y decodificación en los nodos finales[2] (además de la gestión de cabeceras en los nodos origen y destino). Posteriormente, se incluirá un control de flujo más avanzado, que permite descartar segmentos del *buffer* de recepción del nivel de *TCP* del nodo receptor en función

[1] Existen otros trabajos, como [Scal07; Samu08], que realizan modificaciones en el nivel *MAC IEEE 802.11*, alterando los mecanismos de control de acceso al medio definidos en el estándar.

[2] Una restricción que se mantiene a lo largo de todo el Capítulo es que la decodificación será llevada a cabo exclusivamente en los destinos, en los que la dirección *IP* correponde con la de alguno de los segmentos *nativos* que se encuentran dentro de un paquete codificado.

de su contenido, así como la solicitud explícita de retransmisión a través de diferentes mensajes definidos en el protocolo.

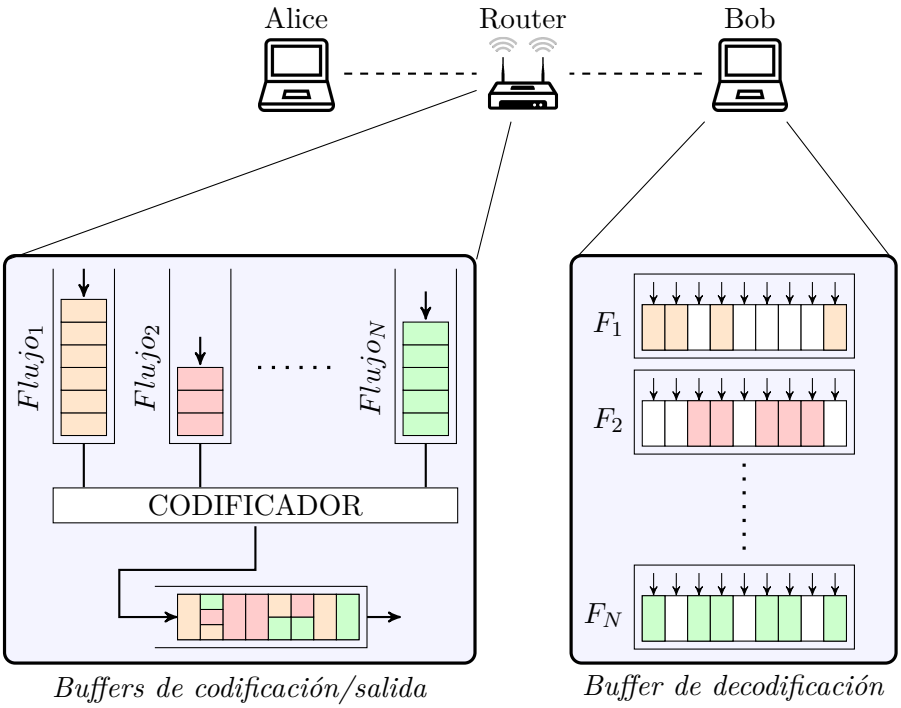


Figura 5.1: Utilización de los *buffers* en el protocolo *NC* inter-flujo

Para poder comprender de una forma más detallada el funcionamiento de este tipo de técnicas, merece la pena detenerse de nuevo en el ejemplo canónico de "*Alice y Bob*", descrito en el Capítulo 4, profundizando en aquellos aspectos necesarios para poder completar con éxito una transmisión que se beneficie de la utilización de las técnicas de *NC*. En este escenario, asumiendo que la capacidad del canal es de un paquete por unidad de tiempo, el primer usuario en obtener acceso al medio será el que transmita la primera trama de información. Como ya se mencionó anteriormente, deberá mantener una copia local del paquete (en la capa de *NC*), ya que podría ser utilizado posteriormente para recuperar la información que se ha recibido codificada. Es por tanto necesaria la presencia de un *buffer de decodificación* (*Decoding Buffer*, *DB*), que almacene los paquetes transmitidos por los propios nodos. Como elemento central del esquema de *NC*, el nodo intermedio deberá procesar la trama recibida, tratando de combinarla con otras que tenga previamente almacenadas, aumentando la eficiencia en el siguiente envío, ya que se incrementará la información transmitida en la misma unidad de tiempo. En este caso, se hace necesario un retardo artificial en los nodos intermedios, que habilite la aparición de *oportunidades de codificación*. Tras la primera recepción, el router tendrá que guardar el paquete en un *buffer de codificación* (*Coding Buffer*, *CB*), bien hasta la llegada de un nuevo segmento *TCP* con el que pueda combinarse o hasta la expiración de un tiempo máximo de permanencia, instante tras el cual se transmite el segmento *nativo*. Por lo tanto, tras la recepción de un

paquete enviado por el otro usuario (i.e. *Bob*), el módulo de codificación del *router* detectará que se ha producido una *oportunidad de codificación* y combinará su contenido con el que tenía almacenado en el *Coding Buffer (CB)*, enviando el paquete generado hacia los niveles inferiores. Gracias a la naturaleza *broadcast* del canal inalámbrico, ambos usuarios serán capaces de recibirlo en la misma unidad de tiempo[3], procediendo a su decodificación (utilizando el paquete almacenado previamente en el *Decoding Buffer (DB)*) y recuperando la información del paquete nativo dirigido a ellos. Por último, el segmento reconstruido será enviado de forma transparente hacia la capa *TCP*, donde se realizarán las tareas oportunas (e.g. generación de un reconocimiento, envío al nivel de aplicación, etc.). En la Figura 5.1 se representan los elementos que deberán incorporarse a los nodos para permitir la correcta operación de un esquema *NC inter-flujo*, entre los que se incluye un *buffer* adicional en los nodos codificadores (i.e. *buffer de salida*), cuya presencia se justificará más adelante.

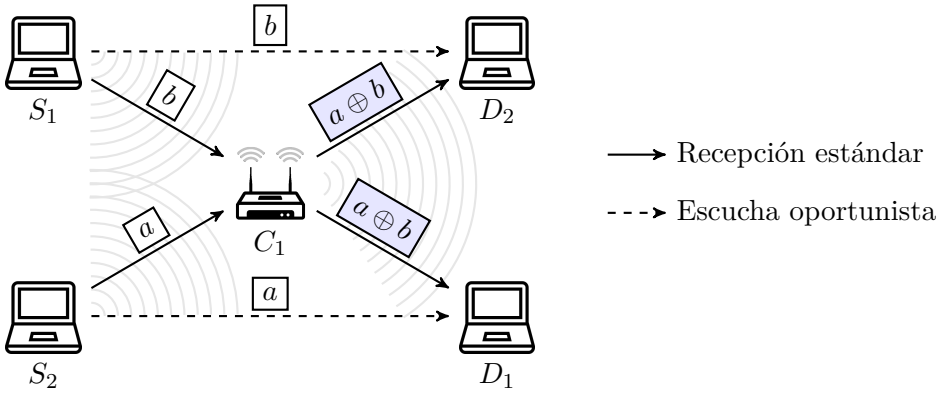


Figura 5.2: Topología en X

El protocolo *NC* puede aprovecharse de la naturaleza *broadcast* del medio inalámbrico, en el que los nodos que se encuentren dentro del área de cobertura del transmisor podrán recibir la trama enviada, aun sin ser los destinatarios originales del mensaje. En el escenario que se representa en la Figura 5.2, "*topología en X*", los nodos S_1 y S_2 establecen una conexión con los destinos cruzados D_1 y D_2, respectivamente, apoyándose en el nodo intermedio C_1, que hará las veces de *router* y *nodo codificador*, combinando (siempre que sea posible) los paquetes de datos pertenecientes a ambos flujos. En esta topología, tanto D_2 como D_1 serán capaces de escuchar las transmisiones realizadas por S_1 y S_2, respectivamente, por lo que dispondrán de la información necesaria para recuperar la información codificada y enviada por C_1. Para que el proceso de decodificación pueda realizarse con éxito, deberán cumplirse las siguientes condiciones: por un lado, será imprescindible que los nodos trabajen en *modo promiscuo*; de lo contrario, descartarán todas las tramas *unicast* cuya dirección *MAC* destino no coincida con la suya (en el caso de la figura, todas las tramas que se reciban por los enlaces $S_1 \dashrightarrow D_2$ y $S_2 \dashrightarrow D_1$). Por otra parte, mientras que en el escenario de "*Alice y Bob*" los nodos finales almacenaban los paquetes que

[3]Aunque se trate de una transmisión originalmente *unicast* (tanto en la cabecera *MAC* como en la *IP* aparece una única dirección destino), el envío será considerado *pseudo-broadcast*.

transmitían en el *buffer de decodificación*, para poder recuperar la información codificada en los paquetes enviados por el nodo intermedio, en un escenario genérico, los nodos que deberían decodificar la información no disponen de antemano de los paquetes *nativos* necesarios para ello, por lo necesitarán haberlos recibido en un instante anterior a la llegada del paquete codificado. En este caso en particular, los nodos D_1 y D_2 deberán almacenar (en el *buffer de decodificación*) todos aquellos paquetes que envíen S_2 y S_1, respectivamente. En conclusión, los nodos guardarán en su *buffer de decodificación* todos los paquetes nativos que procesen, tanto los transmitidos como los recibidos, porque podrían ser necesarios posteriormente para recuperar la información de paquete codificado.

Con los dos ejemplos anteriores se han presentado los elementos que se implementarán en el módulo *NC*. Junto a ellos, se describen a continuación las principales características que definen el comportamiento del protocolo *inter-flujo* desarrollado en el marco de esta Tesis.

- Con el fin de reducir al máximo la complejidad computacional en todo el proceso, las operaciones de codificación/decodificación se realizarán a través de la función *XOR*, equivalente a una suma en el cuerpo de Galois binario $GF(2)$. Los operandos deberán tener la misma longitud (rellenando con ceros en el caso en el que sean diferentes), y el resultado será una combinación de los paquetes de entrada, conservando la longitud del segmento más largo. Mientras que en el proceso de codificación no existe limitación alguna en el número de paquetes a combinar, en el proceso inverso, para recuperar la información correspondiente a un segmento *nativo* que se encuentra combinado en un paquete codificado con otros $N-1$ segmentos, será necesario disponer de todos los demás $(N-1)$ para poder completar la decodificación y recuperar dicho mensaje. Será crucial, por lo tanto, que los nodos codificadores actúen según unos criterios que maximicen el porcentaje de acierto en la decodificación; de lo contrario, se reduciría la ganancia inherente al esquema de *NC*, derivando en la transmisión aislada de los paquetes nativos que faltan para que la decodificación se lleve finalmente a cabo.

- La operación de los *nodos codificadores* se basa en dos aspectos principales:

  1. Solamente los segmentos de datos (i.e. aquellos que transportan información perteneciente a una aplicación) podrán ser sometidos al proceso de codificación. Ya en la propia definición de *COPE* [Katt08], se da prioridad a la codificación de paquetes de tamaños similares, ya que combinar segmentos con longitudes muy diferentes reduciría notablemente el ahorro en ancho de banda. Por lo tanto, en esta primera versión del protocolo *inter-flujo*, sólo se mezclarán segmentos de datos, cuyo tamaño será siempre cercano al *Maximum Segment Size (MSS)* de *TCP*.
  2. Por otra parte, el único requisito para que se genere una *oportunidad de codificación* es que los paquetes pertenezcan a diferentes flujos (i.e. conexiones *TCP*). Para ello, se ha definido un nuevo elemento, el *identificador de flujo*, obtenido a

partir de una función *hash* `MD5` aplicada a la tupla formada por las direcciones *IP* y los puertos (*TCP*), tanto origen como destino, generando una variable única de 2 *bytes* de longitud, que reducirá la sobrecarga subyacente en la cabecera del protocolo.

- Los nodos intermedios, elementos clave de un esquema de *NC inter-flujo*, serán los encargados de combinar el *payload*, a nivel de la capa de *NC*[4], de los paquetes seleccionados para ser codificados. Teniendo en cuenta que el canal inalámbrico basado en el estándar *IEEE 802.11* es un medio compartido, un nodo no podrá recibir dos tramas simultánemente (se produciría una colisión), por lo que será imprescindible "esperar" hasta que la llegada de otro paquete genere una *oportunidad de codificación*. Por lo tanto, deberán almacenar, de manera temporal, los paquetes recibidos, aguardando a que surjan dichas *oportunidades*. Para ello, la entidad de *NC* incorpora un *buffer de codificación*, compuesto por una cola *First In First Out (FIFO)* asociada a cada *identificador de flujo* activo en el escenario[5]. Cada uno de estos *buffer* tendrá una capacidad máxima (en número de paquetes), a través del parámetro CB_S. Además, se habilitará un temporizador, definido por el tiempo CB_{TO}, que marca el tiempo máximo que un segmento nativo puede permanecer en el propio *buffer* (de lo contrario, el proceso de codificación podría introducir una latencia inaceptable para cierto tipo de servicios).

- Junto con el mencionado *buffer de codificación*, estos nodos implementan otro contenedor adicional, llamado *buffer de salida* (*Output Buffer*, *OS*), donde se almacenan los paquetes como último paso previo a su envío a los niveles inferiores. Tras haberse producido una *oportunidad de codificación*, los paquetes pasarán a almacenarse en este contenedor, en el que permanecen a la espera de la llegada de otro segmento de datos que pueda combinarse en aquel. De este modo, una vez se alcance el número máximo de paquetes codificables (variable configurable del sistema[6]) o expire el temporizador del paquete que lleve más tiempo en el *OS*, se realizará una llamada a la entidad *NC* para que genere el nuevo paquete (codificado o no) y proceda a su envío, eliminándolo de esta *cola de salida*. Cabe destacar que en este *buffer* los segmentos se almacenarán de manera nativa (i.e. en forma de lista), siendo la entidad de *NC* la encargada de combinarlos en el instante previo a su transmisión a los niveles inferiores.

- Dado que el esquema de direccionamiento utilizado en este protocolo es *unicast*, una vez que se haya generado un paquete codificado, y antes de ser enviado hacia

[4]Se considerará información (i.e. *payload*) tanto a la cabecera *TCP* como a los datos del nivel de aplicación, por lo que el proceso de codificación incluirá a ambos.

[5]Se asumirá la existencia de un flujo activo mediante la monitorización de los procesos de establecimiento y finalización de las conexiones *TCP*. Los nodos procesarán aquellos segmentos con los *flags* `SYN` o `FIN` habilitados, creando o destruyendo un contenedor interno que almacene los datos pertenecientes a la sesión de nivel de transporte (e.g. direcciones, puertos, etc.) mientras esté activa.

[6]Durante la realización de esta Tesis se ha fijado a dos, debido a que solamente se estudiarán escenarios con dos flujos de datos, y se pretende limitar el impacto de retardos adicionales.

los niveles inferiores, será necesario especificar la dirección destino del mismo. Para ello, se escogerá aleatoriamente la de uno de los paquetes *nativos*. Sin embargo, esta limitación de envío a un único destino puede suponer un problema, ya que podría determinar la ruta a seguir, por lo que si los otros paquete *nativos* con los que comparte la transmisión se encuentran físicamente alejados de ella, puede darse el caso de que no reciban el paquete. De hecho, existen trabajos que limitan el número de saltos de los paquetes codificados, evitando este efecto de "dispersión". Por ejemplo, en [Le10] los autores afirman, utilizando al protocolo *COPE* como ejemplo, que la región de codificación no debe exceder los dos saltos.

- Como ya se ha discutido con anterioridad, uno de los puntos clave que hacen posible el uso de esta solución radica en la naturaleza *broadcast* del medio inalámbrico, permitiendo a los nodos la *escucha oportunista* de los paquetes transmitidos dentro de su área de cobertura. Gracias a esta propiedad, los dispositivos dispondrán de una segunda vía para recibir los paquetes *nativos* que serán utilizados en futuras labores de decodificación (tal y como se explicó en la descripción de la Figura 5.2).

- Otra de las circunstancias a tener en cuenta a la hora de analizar el comportamiento de esta solución guarda relación con la transmisión conjunta de paquetes dirigidos a destinos diferentes en un único datagrama *unicast* (es decir, sólo tiene una dirección *IP* destino). El nivel de enlace *IEEE 802.11* generará una trama *unicast*, con una única dirección *MAC* destino. Esto significa que los mecanismos de retransmisión del estándar *IEEE 802.11* sólo serán efectivos en aquellos enlaces en los que esta dirección *MAC* coincida con la del propio nodo, dejando fuera al resto (el modo *promiscuo* permite el procesado en la capa *NC*, pero no se enviará un *ACK IEEE 802.11* que confirme la correcta recepción de la trama). Por lo tanto, ni los enlaces en los que los nodos realicen una *escucha oportunista*, ni en los que se reciba un paquete codificado y la dirección *IP* no coincida con la del propio nodo serán protegidos por el esquema de recuperación de errores propuesto en el estándar *IEEE 802.11*. Sin duda alguna, este efecto tendrá una gran repercusión sobre el rendimiento final de la red, cuando la presencia de errores de transmisión aleatorios en el canal inalámbrico no sea despreciable.

- Los nodos finales (destino) deberán almacenar todos los paquetes *nativos* que atraviesen el nivel de *NC*, ya sea en sentido descendente (provienen de la aplicación) como ascendente (a través del modo *promiscuo*), en el denominado *buffer de decodificación*, que almacenará por separado (un contenedor por flujo) una lista ordenada en función del número de secuencia del segmento de datos, facilitando su localización cuando sea necesario. Para evitar su crecimiento indefinido se ha limitado, tanto en términos de espacio como de tiempo, el número máximo de paquetes por flujo y el periodo de vida útil de cada segmento *TCP* almacenado. Aquí, al contrario que con el *buffer de codificación*, el único factor limitador es la memoria del dispositivo, ya que mantener los paquetes en este contenedor no tiene un impacto negativo sobre el rendimiento. Por lo tanto, cuanto mayor sea su capacidad, mayor será la posibilidad

de que contenga el paquete que se busca en el proceso de decodificación y, por lo tanto, mayor será el porcentaje de acierto.

- A diferencia de otras soluciones *inter-flujo* como *COPE*, *las tareas de decodificación se han incluido únicamente en los nodos finales*. Se ha querido mantener un esquema de codificación/decodificación sencillo, dando un mayor peso al análisis del comportamiento del protocolo (i.e. impacto de la retención de los paquetes en los nodos intermedios, sincronización entre flujos, etc.).

- Uno de los elementos clave a la hora de diseñar esta primera versión del protocolo se ha basado en asegurar la completa independencia entre los niveles *NC* y de transporte. De este modo, la capa *NC* no realizará ningún tipo de control adicional, reduciendo su complejidad de manera notable. Además, esta solución tampoco utilizará ningún tipo de mensajes propios de señalización, evitando la introducción de sobrecarga adicional al tráfico presente en la red.

- Finalmente, la integración de un nuevo protocolo exige el intercambio de información entre entidades pares. En el caso del esquema *inter-flujo*, se ha definido una cabecera que transporta los elementos necesarios para el proceso de codificación/decodificación, con un formato como el que se ilustra en la Figura 5.3, donde se observa la división de la misma en dos partes, una fija y una variable, cuyos campos se detallan a continuación:

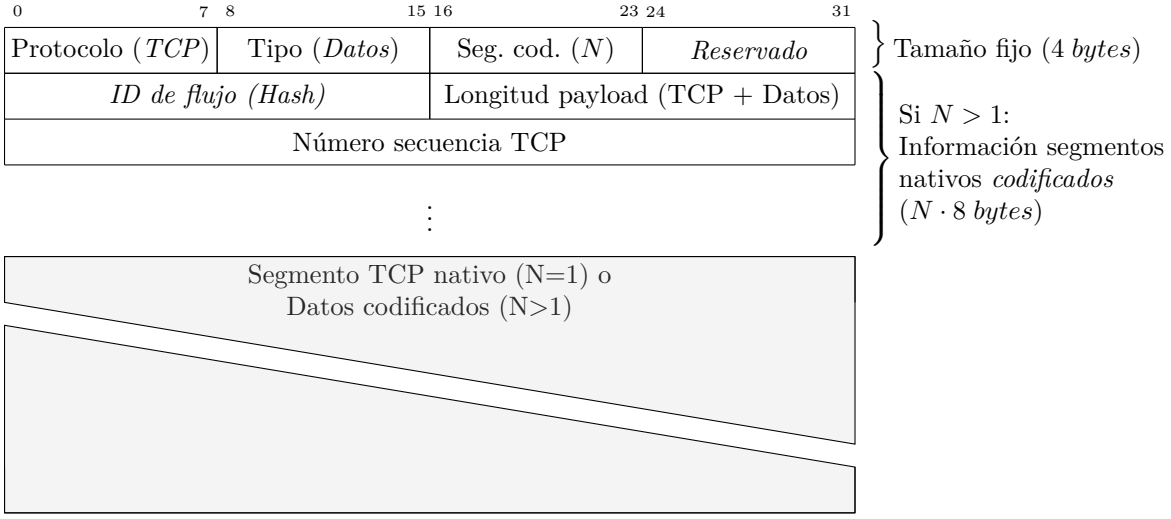


Figura 5.3: Formato de la cabecera utilizada en el protocolo *NC inter-flujo*

1. La parte de longitud fija, con un tamaño total de 4 *bytes*, contendrá cuatro campos, ocupando cada uno de ellos 1 octeto: el código asignado por la *IANA*

para el *protocolo de nivel de transporte* (6 para *TCP* o 17 para *UDP*[7]), necesario a la hora de enviar los paquetes a los niveles superiores, el *tipo de paquete* (en esta versión del protocolo sólo se contemplan paquetes de datos, cuyo código será 0), el *número de paquetes nativos* que transporta y un último campo, que se *reserva* para versiones futuras del protocolo.

2. Por otro lado, en el caso de que se trate de un paquete codificado con varios segmentos *nativos*, será necesario añadir cierta información de los mismos, ya que sus cabeceras *TCP* estarán codificadas (forman parte del *payload* en el nivel de *NC*). En concreto, para cada segmento se añadirá un total de 8 *bytes*, incluyendo el *identificador de flujo*, la longitud de los datos (cabecera *TCP* y datos del nivel de aplicación) y el número de secuencia, utilizado para localizar al segmento en el *buffer de decodificación*. En el caso de transportar segmentos nativos "en claro" no será necesario utilizar este campo, ya que la información podrá leerse directamente de la cabecera *TCP*.

Existen soluciones como [Seng10] donde las cabeceras *TCP* se reducen antes de ser codificadas, aprovechando que un conjunto de sus campos de información van a aparecer en la cabecera del protocolo *NC* (puertos, número de secuencia y *flags*, concretamente), evitando transmitir datos duplicados. Sin embargo, el beneficio es prácticamente despreciable, teniendo en cuenta los tamaños de trama utilizados en el análisis de esta Tesis, por lo que las cabeceras *TCP* permanecerán intactas.

5.1.1. Proceso de codificación

El elemento clave de los mecanismos de codificación se sitúa en una capa de inteligencia adicional que se añade en los *nodos intermedios*, dotándoles de la capacidad de procesar los paquetes que reciben, modificando su contenido antes de enviarlos nuevamente. Dentro del protocolo *inter-flujo* presentado en este capítulo, su funcionalidad se divide en dos categorías: en la primera de ellas se encuentran los nodos *estándar*, que actuarán como meros *routers*, reenviando la información con el menor retardo posible, tras la consulta de su destino (o siguiente salto) en sus tablas de rutas. Por otro lado, los *nodos codificadores* serán los responsables de combinar la información a partir de los criterios de codificación establecidos en el protocolo.

La elección de estos *nodos codificadores* tendrá un impacto fundamental en el posterior rendimiento de la comunicación, ya que su principal objetivo será el de codificar la mayor cantidad de información posible, maximizando la probabilidad de éxito en la decodificación. En la literatura existen soluciones que se apoyan en el intercambio de mensajes

[7] Aunque no se comente explícitamente en esta Tesis, el esquema *inter-flujo* permite la posibilidad de operar con datagramas *UDP*, como se estudió en un Proyecto Fin de Carrera de la Universidad de Cantabria [Pomp13], que propuso un nuevo esquema de secuenciamiento para poder distinguir los datagramas que estaban siendo codificados.

de señalización entre nodos vecinos, que anuncian los segmentos que posee cada uno en un instante dado [Katt08; Hund12]. Esto permitirá a cualquier nodo ejercer el papel de codificador a partir de la información recibida de sus vecinos. Sin embargo, para el protocolo *inter-flujo* definido en el marco de esta Tesis se ha optado por utilizar una versión simplificada de estos trabajos[8], de manera que los nodos definidos como codificadores se mantendrán así durante todo el proceso.

La funcionalidad de codificación queda resumida en la Figura 5.4, que representa el flujo que sigue un paquete desde su llegada a la capa *NC*[9] hasta que la abandona. En primer lugar se deberá comprobar si el paquete recibido se encuentra codificado ya que, por motivos de simplicidad, no se recodificarán paquetes "en tránsito". De ser así, se pasará nuevamente al nivel inferior, que se encargará de reenviarlo al siguiente salto, siguiendo el procedimiento *store-and-forward* estándar.

En la situación opuesta (recepción de un paquete nativo) se tratará de encontrar una *oportunidad de codificación*, empleando dos etapas (debido a la implementación del módulo *NC*). En la primera de ellas, se buscará en el *buffer de salida* la presencia de algún paquete (codificado o no) con el que pueda combinarse (comparando los *identificadores de flujo*), uniéndose al mismo en caso afirmativo. Si no se ha podido emparejar con ningún elemento del *buffer de salida*, el siguiente paso será buscar en el *buffer de codificación*. En los casos en los que se cumpla esta condición, tanto el segmento recibido como el más antiguo del *buffer de codificación* (que será eliminado de la cola) darán lugar a un nuevo paquete codificado en el *buffer de salida*, con la información de todos los segmentos que lo conforman.

Aunque en esta Tesis se limita el número máximo de segmentos que pueden combinarse en un único paquete a 2, el módulo desarrollado mantiene una implementación genérica ($N > 2$) en la que estos mensajes permanecerán en la *cola de salida*, esperando la llegada de nuevos segmentos que puedan combinarse, aumentando la *ganancia de codificación* del sistema. En estos casos, el procesado en la entidad *NC* concluye estableciendo un temporizador con el tiempo restante correspondiente al segmento más antiguo, evitando así retener la información de manera indefinida. Si se ha alcanzado el número máximo de segmentos, el proceso pasará directamente a la fase de generación del paquete de salida.

Hasta este punto se han descrito todos los eventos que conducen a una *oportunidad de codificación*, que dará como resultado el envío de un paquete codificado a los niveles inferiores. A continuación se describen los dos casos que generarían un *error de codificación*, donde la información pasará por la entidad de *NC* de manera transparente (solamente se añadirá la cabecera más básica - 4 *bytes*), atravesando la red sin codificación alguna:

[8]Teniendo en cuenta las condiciones de simulación que se van a fijar, según las que los nodos permanecerán estáticos durante todo el proceso, será posible determinar la posición de los codificadores a la hora de generar el escenario, en una etapa previa a la simulación.

[9]Hay que tener en cuenta la entrada del paquete en el sistema, ya que, por defecto, la entidad *NC* lo recibirá a través de dos procesos diferentes (i.e. recepción promiscua y a través del nivel *IP*). Para evitar un procesado duplicado, se descartarán las recepciones promiscuas de paquetes nativos en los *nodos codificadores.*

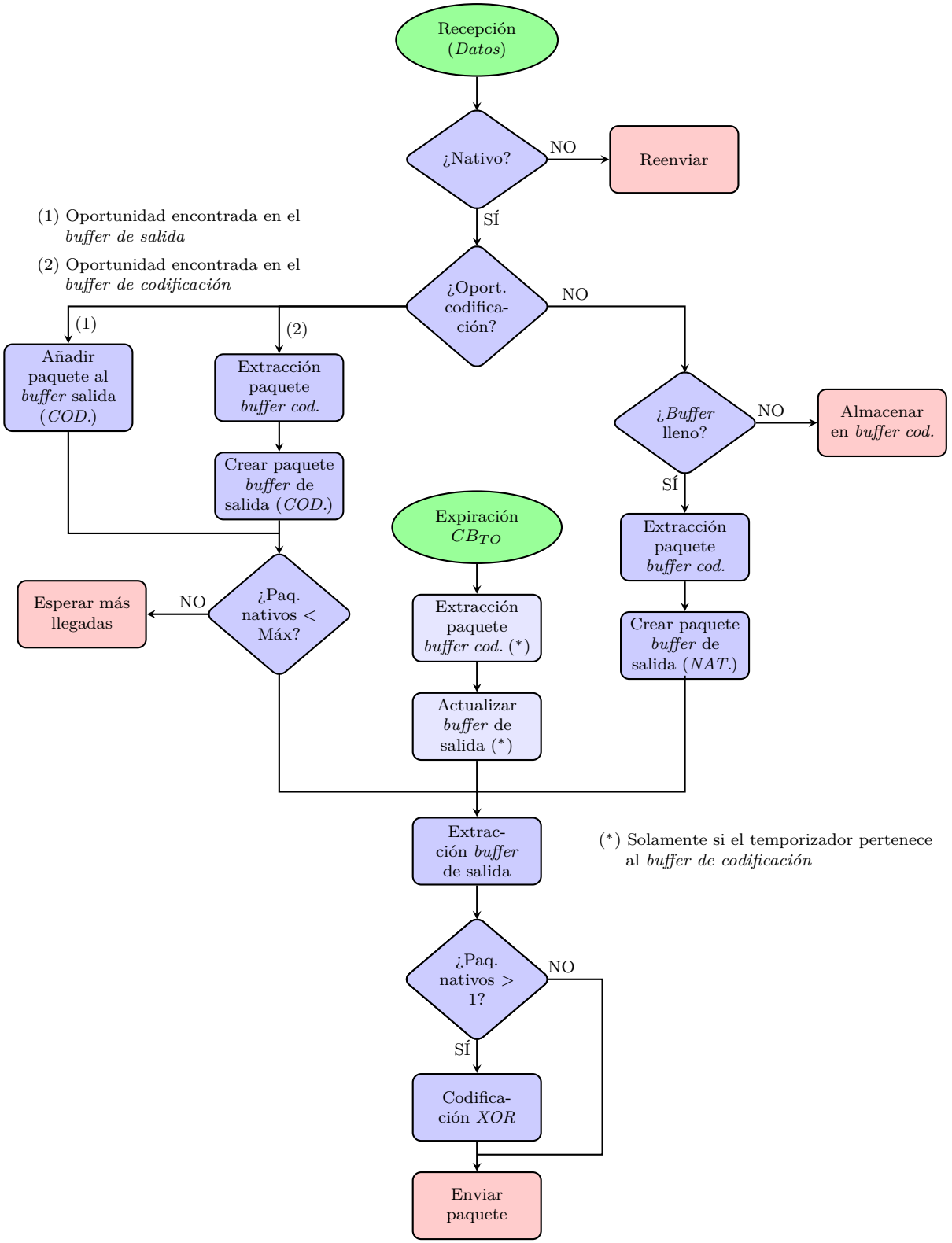


Figura 5.4: Diagrama de flujo de la etapa de codificación de un paquete en el protocolo de *NC inter-flujo*

1. De forma paralela a la recepción de los segmentos nativos, los *nodos codificadores* controlan el tiempo que los paquetes almacenados permanecen en sus diferentes contenedores, limitando su estancia a un valor máximo, definido a través de la variable CB_{TO}. En el caso de que se genere un *evento producido por la expiración de dicho temporizador*, el proceso a seguir variará ligeramente en función del *buffer* al que esté asociado: mientras que en el *buffer de salida* el proceso a seguir será el habitual, descrito posteriormente, en el *buffer de codificación*, deberá extraerse primero el segmento *nativo*, moviendo su contenido al *buffer de salida*. Este último caso se considerará como un *error de codificación*, ya que el paquete saliente transporta un único segmento *TCP*.
2. En la otra parte derecha de la figura se representa el flujo en el caso en el no se haya producido una *oportunidad de codificación*, circunstancia que se da cuando *se recibe un paquete y solamente se encuentran almacenados segmentos con el mismo identificador de flujo*. El siguiente paso será realizar una copia local del mismo en el *buffer de codificación* donde, dada su capacidad limitada, aparecen dos opciones: *(1)* guardar el segmento (si hay espacio), dando por finalizado el proceso en el nivel de *NC*; *(2)* cuando el *buffer* se encuentre completo, se eliminará el paquete más antiguo para poder almacenar al recién recibido. Este último punto es la segunda causa que conduce a un *error de decodificación*, con la consecuente transmisión de un segmento *TCP* nativo.

Como último paso (zona inferior de la figura), se observa que todo el esquema converge en una única fase común, en la que se generará la información que va a ser enviada al nivel de red. El proceso a seguir es el siguiente: a partir de la extracción del elemento más antiguo del *buffer de salida*, se comprueba el número de segmentos *nativos* que contiene. En el caso en el que haya más de uno, se realizará la operación *XOR* entre todos los segmentos de la lista; por el contrario, si sólo hay uno, se pasará directamente a la generación de la cabecera del nivel *NC*. Una vez se haya construido el paquete completamente, se enviará a los niveles inferiores.

5.1.2. Proceso de decodificación

En el esquema *NC inter-flujo*, la decodificación será llevada a cabo *exclusivamente por los nodos finales*, que posteriormente enviarán los segmentos recuperados a los niveles superiores. En la Figura 5.5 se detalla el flujo seguido desde la llegada de un paquete a estos nodos[10] hasta que finaliza su procesado.

En primer lugar, tras la recepción de un paquete en el nivel de *NC* se comprobará si transporta información codificada, a través del campo "*Paquetes nativos*" de la cabecera

[10]De nuevo, para evitar un procesado duplicado, el procedimiento de recepción estándar se encargará del análisis de aquellos paquetes que no contengan información codificada, dejando al esquema promiscuo esta tarea.

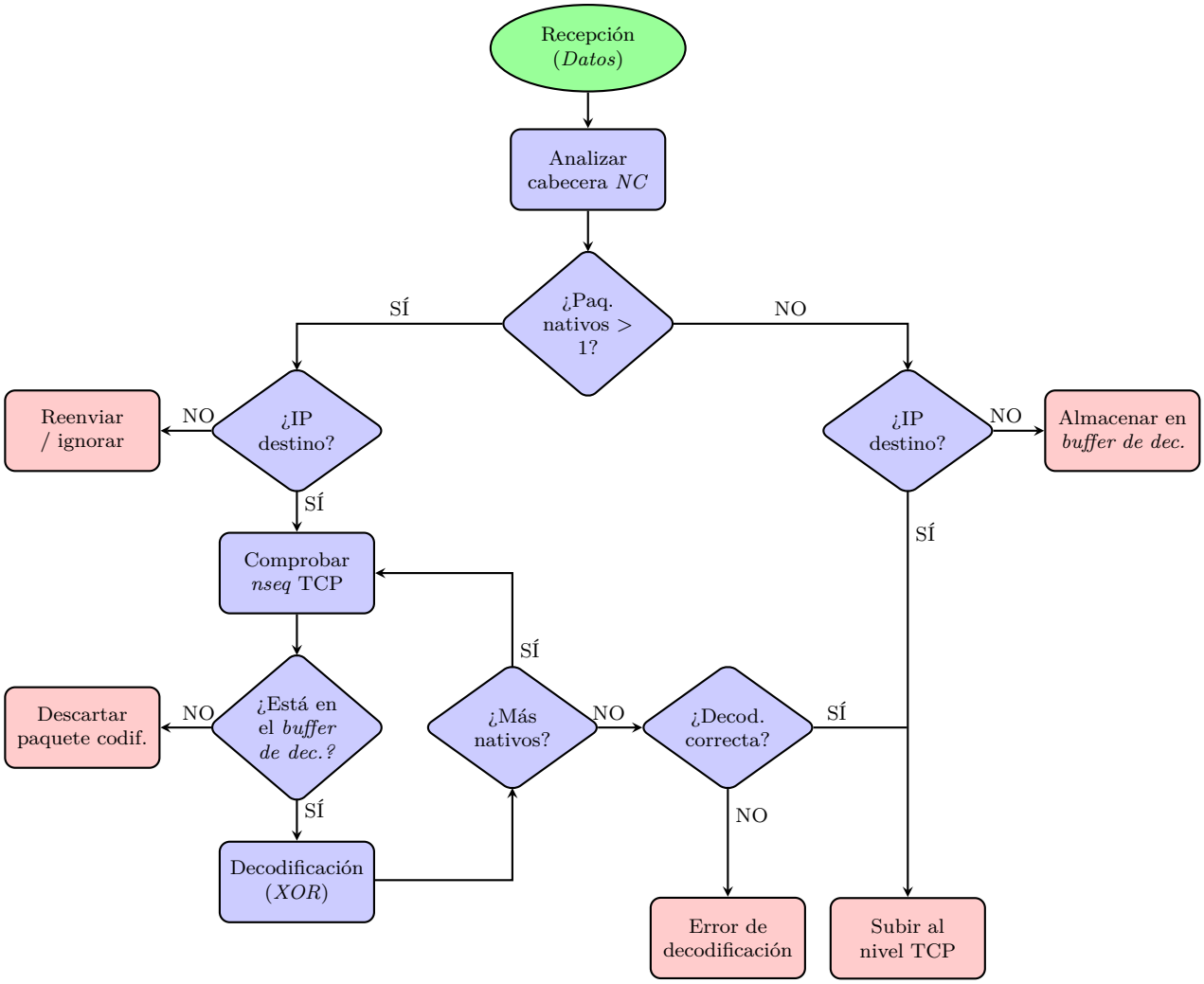


Figura 5.5: Diagrama de flujo de la etapa de decodificación de un paquete en el protocolo de *NC inter-flujo*

NC. En caso de tratarse de un segmento *nativo*, se enviará al nivel *TCP* cuando vaya destinado al nodo; de no ser así, se almacenará en el *buffer de decodificación*, esperando a la posible llegada de paquetes codificados que lo incluyan.

Por otra parte, en las situaciones en las que los paquetes recibidos sean producto de la combinación de varios segmentos *nativos*, deberá analizarse en primer lugar si alguno de ellos va dirigido al nodo en cuestión, descartando el paquete en caso contrario. A partir de este punto, el nodo comenzará con el proceso de decodificación, tratando de recuperar el segmento *nativo* correspondiente[11]. Para ello, se buscarán en el *buffer de decodificación* los

[11]En este proceso solamente se podrá recuperar un único segmento *nativo* por paquete codificado, debido a las propiedades del operador *XOR*.

segmentos que componen el paquete codificado, a partir de su *identificador de flujo* y el número de secuencia, presentes en la parte variable de la cabecera *NC*. Si la búsqueda es exitosa, se realizará la operación *XOR* entre el paquete codificado y el segmento extraído del *buffer de codificación*, reduciendo el número de segmentos codificados en una unidad. El proceso se repetirá con todos los segmentos que integran el paquete, dando por válido el proceso si y sólo si el *buffer* disponía de $N-1$ de los N segmentos que componen el paquete codificado, con lo que se logra recuperar el mensaje destinado al nodo. En caso de no disponer de todos ellos, se considerará un *error de decodificación*, eliminando el paquete codificado[12]. En caso de éxito, el último paso será subir el segmento recuperado al nivel *TCP*, que seguirá con su procesado habitual.

5.1.3. Modelado de las oportunidades de codificación

Una vez descrita la operación seguida para llevar a cabo las tareas de codificación/decodificación, se presenta a continuación un análisis del porcentaje de *oportunidades de codificación* que aparecen en un *nodo codificador*, en función de la sincronización de los flujos y del tamaño del *buffer de codificación*, CB_S.

Como se ha descrito anteriormente, el esquema de codificación propuesto en este protocolo se puede ver como una simplicación de otras soluciones más complejas, basadas en la monitorización de los *buffers* en los nodos vecinos a través del intercambio de mensajes de señalización. En la solución propuesta, los únicos requisitos para que dos (o más) segmentos puedan ser combinados son: primero, que se trate de segmentos de datos (con longitud elevada), evitando de este modo combinar paquetes de tamaños muy dispares; además, es necesario que pertenezcan a diferentes flujos (lo que se distingue gracias al campo *identificador de flujo* transmitido en la cabecera *NC*).

Debido a las condiciones de codificación impuestas en este esquema, en el *buffer de codificación* (CB) sólo se va a guardar una copia local de un segmento cuando el CB está vacío o todos los segmentos almacenados pertenecen al mismo flujo (recuérdese que existe una cola individual por cada flujo activo). De esta manera, el CB solamente podrá contener segmentos pertenecientes al mismo flujo.

En cuanto a los eventos de salida, el esquema presentado previamente establece cuatro vías diferentes a la hora de generar un paquete y enviarlo a los niveles inferiores, donde dos de ellas se corresponden con una *oportunidad de codificación*, mientras que las otras dos tienen como consecuencia a un *error de codificación*, que deriva en la transmisión de un paquete *nativo*:

- ✓ Si el número máximo de segmentos que pueden combinarse en un mismo paquete está limitado a 2, cuando el CB contenga al menos un segmento de datos y se produzca

[12]En una versión más avanzada se implementa una *cache de paquetes codificados*, que los almacenará, esperando una posible recepción posterior que permita su decodificación.

la llegada de otro que pertenezca a un flujo con identificador diferente, se generará un *paquete codificado*.

✓ En una situación más genérica, en la que el límite de segmentos a combinar no se encuentre restringido a 2, se mantendrá una copia de los paquetes que van a ser codificados en un único elemento en el OB, esperando a la llegada de nuevos segmentos de datos que puedan ser añadidos. En todas estas situaciones, el evento que propicia la creación del paquete (*codificado*) es la expiración del temporizador del segmento más antiguo de los que se encuentran en el OB[13].

✗ Si la cola correspondiente a un flujo está completa, y se produce la recepción de otro segmento, se producirá una situación de desbordamiento del CB, por lo que el paquete más antiguo se sustituye por el más reciente, siendo además *enviado de forma nativa* a los niveles inferiores.

✗ La expiración del BC_{TO} en una de las entradas del BC implicará que, durante ese periodo, no se ha producido ninguna *oportunidad de codificación*, por lo que el *segmento nativo* más antiguo pasa directamente a las capas inferiores.

A continuación se presenta un modelo analítico para la probabilidad de que se produzca una *oportunidad de codificación* en función de los parámetros de configuración del CB. Para facilitar el análisis, se han considerado las siguientes limitaciones:

1. En el escenario se contemplan dos conexiones simultáneas, que atravesarán un punto de articulación común en el grafo que define la red. El nodo ubicado en esta posición tendrá el papel de *nodo codificador*. Puede utilizarse como ejemplo la topología en X mostrada en la Figura 5.2.

2. Los *errores de codificación* asociados a la expiración del CB_{TO} no serán tenidos en cuenta en el análisis (se asumirá que no se impone esta limitación temporal). Por lo tanto, solamente influirán aquellos que sean producto de un desbordamiento en el CB.

3. Únicamente los segmentos *TCP* de datos podrán ser codificados; cualquier otro tipo de mensaje (reconocimientos *TCP*, mensajes *ARP*, etc.) atravesarán la entidad de *NC* de manera transparente.

4. El número máximo de segmentos *nativos* que pueden combinarse en un mismo paquete será 2, por lo que, tras la aparición de una *oportunidad de codificación*, se generará un paquete codificado con carácter inmediato.

[13]La implementación que se presenta en el marco de la presente Tesis no contempla los eventos producidos en este punto, ya que el número de segmentos codificables se ha limitado siempre a 2.

5. El análisis se llevará a cabo en condiciones ideales, de manera que la presencia de errores derivados de la propagación por el canal inalámbrico o consecuencia de los mecanismos de control de acceso al medio del estándar *IEEE 802.11* (e.g. colisiones, terminal oculto, etc.) puede considerarse despreciable.

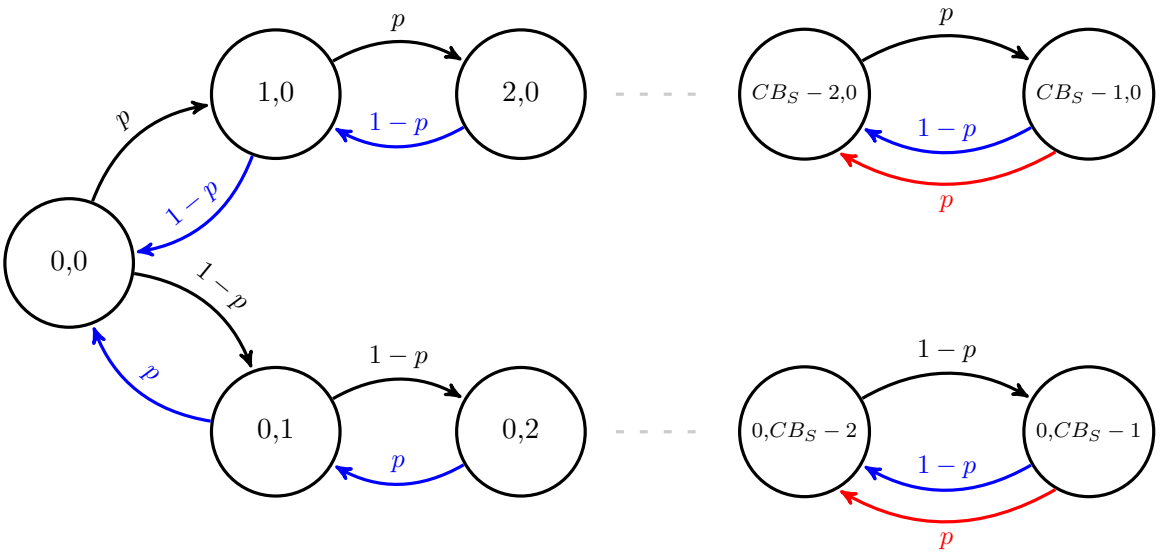


Figura 5.6: Proceso de Markov que representa los eventos discretos que definen la evolución del buffer de codificación en un escenario con *dos flujos*. En azul (→) se muestran aquellos cambios de estado que generan una *oportunidad de codificación*; por el contrario, en rojo (→) aparecen aquéllos que darán lugar a un *error de codificación*, cuya consecuencia será la transmisión de un paquete nativo

Manteniendo estas condiciones, la Figura 5.6 representa el proceso de Markov que define la secuencia de eventos discretos que modela el comportamiento de los *nodos codificadores* ante la llegada de nuevos segmentos de datos. Siguiendo con la descripción del proceso de codificación, cada flujo de datos mantiene una cola independiente, con capacidad para almacenar CB_S segmentos *nativos*. Como ya se ha discutido anteriormente, el protocolo imposibilita mantener segmentos de flujos diferentes en un mismo instante de tiempo, ya que la aparición de una *oportunidad de codificación* deriva en la extracción de un segmento almacenado en la cola activa (sin mantener una copia del recibido). La llegada de segmentos pertenecientes al mismo flujo provocará el aumento paulatino de la cola correspondiente al mismo, mientras que si *existe al menos un segmento nativo almacenado* y se recibe un segmento con un *identificador de flujo* diferente, se producirá una *oportunidad de codificación*. Finalmente, en los casos en los que la *cola* esté llena y el segmento recibido pertenezca al mismo flujo, se producirá un *error de codificación*, y la entidad *NC* enviará el paquete más antiguo a los niveles inferiores.

En base a la descripción previa, se define p como la probabilidad de que un paquete proceda de uno de los nodos transmisores, correspondiéndose el suceso complementario,

$1-p$, con la segunda de las fuentes. Siguiendo con el esquema ilustrado en la figura anterior, la Ecuación (5.1) expresa la probabilidad de que se produzca una oportunidad de codificación.

$$\mathcal{P}\ (\text{Oportunidad codificación}) = (1-p)\cdot\sum_{i=1}^{CB_S} p^i + p\cdot\sum_{i=1}^{CB_S}(1-p)^i \tag{5.1}$$

Por el contrario, para que se produzca un *error de codificación*, el *nodo codificador* deberá recibir una ráfaga de CB_S+1 segmentos consecutivos del mismo flujo. En este caso, la probabilidad correspondiente es la que se muestra en la Ecuación (5.2).

$$\mathcal{P}\ (\text{Salida sin codificar}) = p^{CB_S+1} + (1-p)^{CB_S+1} \tag{5.2}$$

En la Figura 5.7 se representa la probabilidad de encontrar una *oportunidad de codificación* en función del tamaño del *buffer de codificación* (CB_S) y p. Como es lógico, cuanto mayor sea CB, mayor será la tasa de éxito en el proceso de codificación, ya que las ráfagas de segmentos consecutivos del mismo flujo deberán ser más largas. Por otra parte, se puede deducir que el sistema presentará un mejor comportamiento cuando los flujos se encuentren más sincronizados. En el mejor de los casos, con $p = 0.5$, que se corresponde con un acceso alterno al canal entre las fuentes (i.e. *Round-Robin*), el comportamiento del sistema experimentará las siguientes mejoras: *(1)* se maximizan las *oportunidades de codificación*, *(2)* se reduce el retardo artificial introducido por almacenar los paquetes en el CB, *(3)* se incrementa la *ganancia de codificación* (la tasa de paquetes codificados respecto a segmentos *nativos* transmitidos será mayor) y *(4)* el número de transmisiones totales necesarias se verá reducido. Como resultado, el rendimiento global del sistema mejorará, tanto en términos de *throughput*, como de retardo o energía.

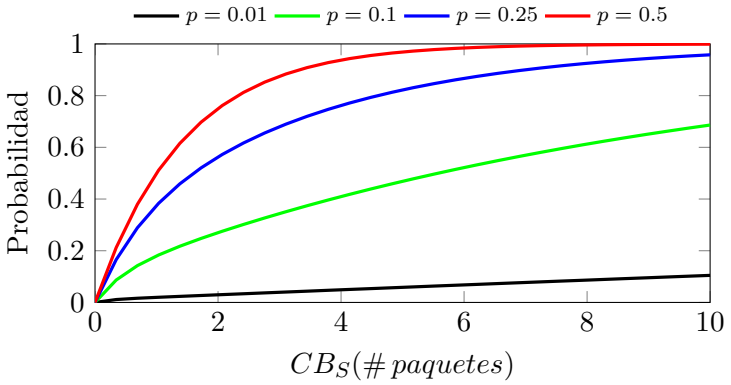


Figura 5.7: Probabilidad de encontrar una oportunidad de codificación

Por el contrario, a medida que la sincronización entre los flujos empeora, disminuye la probabilidad de codificar segmentos en un mismo paquete, por lo que los beneficios mencionados anteriormente serán menos claros, pudiendo llegar al punto de que el rendimiento

sea inferior al que se obtendría utilizando *TCP* bajo las mismas condiciones. Por lo tanto, será esencial encontrar una configuración óptima de los parámetros del *buffer de codificación* que permita mantener los flujos lo más sincronizados posible. Es preciso destacar la importancia de la relación del nivel *NC* con los mecanismos de control de flujo y congestión de *TCP*, ya que, como se verá más adelante, la pérdida de paquetes en la red tendrá diferentes consecuencias, en función de si se produce en enlaces en los que predomine el tráfico de paquetes codificados o si, por el contrario, las pérdidas correspondan exclusivamente a paquetes nativos, afectando por lo tanto a un único flujo.

5.2 Encapsulación de los reconocimientos *TCP*

Antes de describir la propuesta para reducir la sobrecarga producida por la transmisión aislada de los reconocimientos de *TCP*, resulta interesante mencionar el trabajo de *Sinha et al.* [Sinh07], donde los autores elaboran un completo análisis de la distribución de la longitud de los paquetes que circulan por Internet, cubriendo diferentes aspectos, tanto en términos temporales (el tráfico varía notablemente a lo largo del día), como espaciales, repitiendo el experimento en diferentes lugares. Se destaca un cambio que rompe con el tradicional comportamiento *bimodal* que ha caracterizado el tráfico de datos desde sus orígenes, según el que se observaban dos picos, situados en los 40 y 1500 octetos de longitud, que cubrían alrededor un 60 % del total de paquetes analizados[14]. Sin embargo, en los últimos años se ha observado la aparición de un nuevo máximo, centrado en los 1300 *bytes*, consecuencia del uso cada vez más extendido de redes privadas virtuales (cuyo *MTU* se fija a esa cifra [VPNMTU]), así como por las recomendaciones de algunos proveedores de servicio *Digital Suscriber Line (DSL)*. Esto viene a demostrar que una porción no despreciable de los paquetes disponen de un espacio aprovechable (asumiendo un *MTU* de 1500 *bytes* en un enlace *IEEE 802.11*) para incluir información adicional en un mismo paquete, sin que los niveles inferiores tengan que recurrir a la fragmentación en múltiples tramas.

Como ya se puso de manifiesto en el Capítulo 2, el flujo inverso producido por los reconocimientos enviados por la entidad receptora de *TCP* introduce una sobrecarga que debe ser tenida en cuenta, sobre todo en canales *half-duplex* con acceso basado en contienda, como es el caso de los enlaces *IEEE 802.11*. Por lo tanto, la transmisión de un simple mensaje *ACK* comparte el medio con los segmentos de datos, que son los que transportan la información útil (del nivel de aplicación) para el usuario. Para cuantificar el impacto

[14]Esta naturaleza dual se explica a partir de los siguientes factores: por un lado, la tecnología de acceso dominante ha sido durante muchos años *Ethernet*, que define una longitud máxima de trama (típicamente) de 1500 *bytes*. Además, otras tecnologías, como *IEEE 802.11*, se comportan como interfaces *Ethernet* virtuales, manteniendo el mismo valor de *MTU*. Para obtener un mayor rendimiento, la tendencia general será la de transmitir paquetes lo más largos posibles, explicando la presencia del mencionado pico centrado en los 1500 octetos. En el otro extremo se encuentran los mensajes de control y señalización que intercambian los diferentes protocolos que atraviesan la red, caracterizados por estar formados únicamente por las propias cabeceras de los protocolos, dando lugar a paquetes notablemente más cortos.

Tabla 5.1: Parámetros utilizados para calcular el retardo medio en una transmisión *IEEE 802.11*

Parámetro	*Valor*
DIFS	50 μseg
SIFS	10 μseg
$\overline{BO}$ (32 *slots*)	310 μseg
t_{PLCP} (Preámbulo largo)	192 μseg
Cabecera *TCP*	20 *bytes*
Cabecera *IP*	20 *bytes*
Cabecera *LLC*	8 *bytes*
Cabecera *IEEE 802.11*	26 *bytes*
Tasa binaria (datos)	11 *Mbps*
Tasa binaria (control/*broadcast*)	2 *Mbps*

producido por estos reconocimientos, se utilizará un ejemplo básico con dos nodos A y B conectados a través de un enlace *ad hoc IEEE 802.11*. El nodo A genera un tráfico de datos *TCP* (por ejemplo, enviando un fichero utilizando *File Transfer Protocol (FTP)*) hacia B, que únicamente transmitirá los correspondientes *ACKs*. Para obtener el mayor rendimiento posible, los segmentos de datos generados tendrán una longitud de 1460 *bytes*, correspondientes al *MSS* típico de *TCP*. Partiendo de los cálculos del retardo medio de transmisión en un enlace *IEEE 802.11b* a 11 *Mbps* (presentado en la Sección 2.1.1), se añade tiempo desde que un transmisor envía un segmento de datos hasta que recibe su correspondiente *ACK*. Para ello, la Tabla 5.1 recoge los principales parámetros a tener en cuenta. Por motivos de simplicidad, se asumirá que no se producen errores derivados de la propagación por el medio inalámbrico, descartando también las colisiones generadas por las transmisiones simuláneas entre ambos nodos.

Con todo ello, y siguiendo el esquema de tiempos definido en el estándar *IEEE 802.11b*, tal y como se resume en la Ecuación (5.3), la transmisión de un segmento de datos de longitud máxima (y su correspondiente acuse de recibo a nivel *MAC*) captura el canal inalámbrico durante aproximadamente 2 milisegundos.

$$\begin{aligned}\overline{\tau_{datos}}(1460\ B) = DIFS + \overline{BO_0} + t_{PLCP} + \ t_{TX_{Cabeceras}}(26+8+20+20)+ \\ + t_{TX_{datos}}(1460\ B) + SIFS + t_{PLCP} + t_{TX_{ACK}} \simeq 2000\ \mu segs\end{aligned} \tag{5.3}$$

Por otro lado, tras la recepción del segmento de datos, la capa de *TCP* del nodo B procede a la generación de un reconocimiento o, lo que es lo mismo, un segmento *TCP* sin datos (contiene exclusivamente las cabeceras de los protocolos que intervienen en la comunicación). En este caso se necesitarán, a partir de los tiempos definidos en la Ecuación (5.4), aproximadamente 900 microsegundos para completar la transmisión. Esto supondrá una sobrecarga adicional de prácticamente un 45 % cada vez que el nodo destino genera un

reconocimiento. De todo este tiempo, es preciso remarcar que solamente un 6 % de este tiempo es utilizado para transmitir el propio reconocimiento, ya que la duración estricta de la transmisión del *ACK* es de 54 microsegundos (cabeceras *MAC*, *LLC*, *IP* y *TCP* a una tasa binaria de 11 *Mbps*) del total, poniendo de manifiesto la gran sobrecarga generada por la transmisión aislada de un segmento de reconocimiento *TCP*.

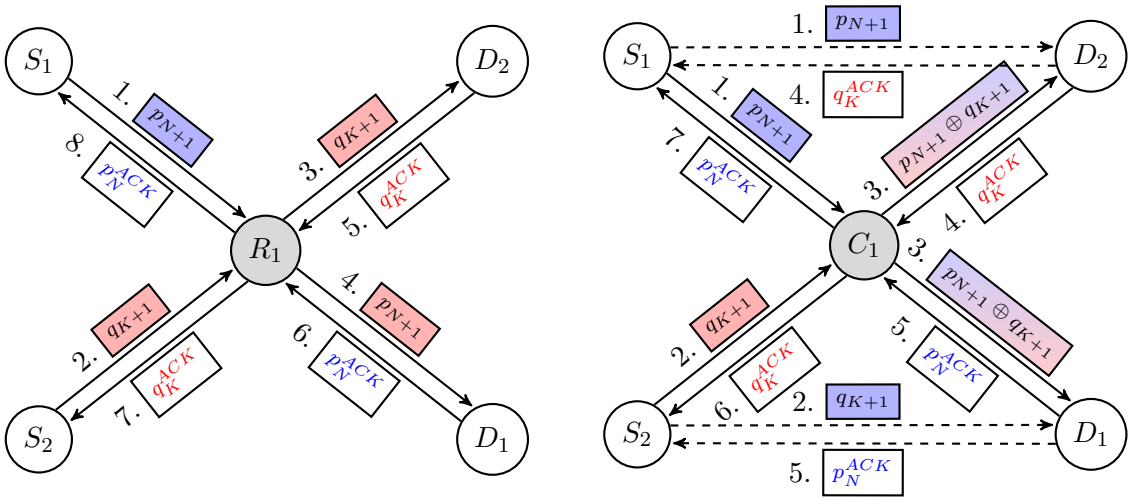
$$\overline{\tau_{ACK_{TCP}}} = DIFS + \overline{BO_0} + t_{PLCP} + t_{TX_{Cabeceras}}(26 + 8 + 20 + 20) + \\ + SIFS + t_{PLCP} + (t_{trans})^{ACK} \simeq 914\ \mu segs \quad (5.4)$$

Combinando ambos retardos, se obtiene una latencia promedio de unos 3 milisegundos desde que el segmento de datos sale del nodo transmisor hasta que se recibe su confirmación (*TCP*). Este tiempo se define con el parámetro *RTT*, que como ya se ha comentado en capítulos anteriores, resulta esencial en los mecanismos de control de congestión definidos en el protocolo *TCP*.

Sin embargo, hay que tener en cuenta que el protocolo *TCP* habitualmente genera (técnica conocida como "*delayed ACKs*") un reconocimiento por cada B segmentos recibidos correctamente, siendo típicamente $B = 2$. Así, el *overhead* promedio producido por los reconocimientos *TCP* será ligeramente inferior, suponiendo un total de $\approx 22\,\%$.

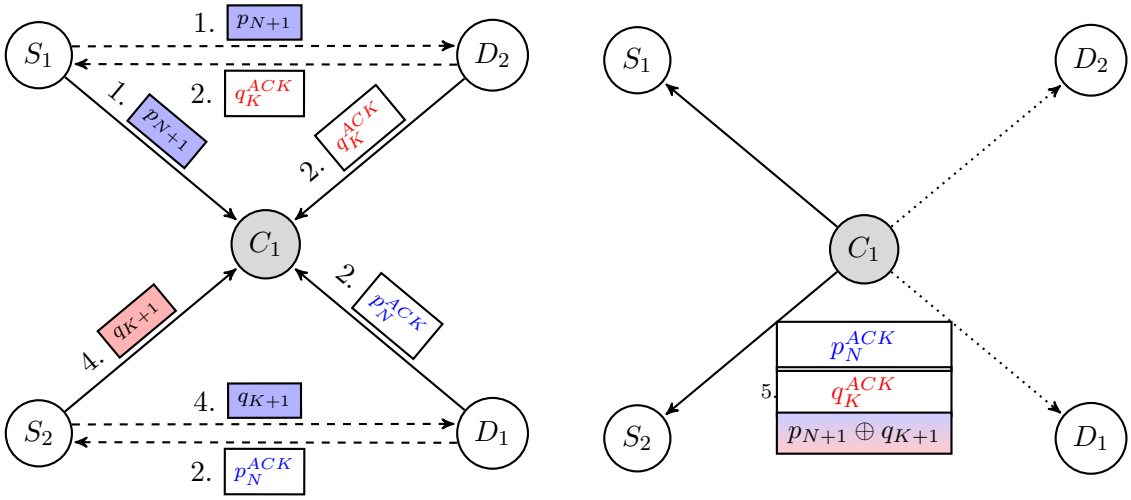
Para limitar el impacto de los *ACKs*, se agregará una nueva funcionalidad a la entidad de *NC* en los nodos intermedios, donde se encapsularán los reconocimientos *TCP* dentro de las cabeceras *NC*, aprovechando el espacio disponible que se observa en un buen porcentaje de los paquetes que circulan por la red [Sinh07]. Gracias a este nuevo enfoque, aquellos *ACKs* que se transporten embebidos dentro de los paquetes de datos conseguirán eliminar toda su sobrecarga correspondiente a los niveles físico y *MAC*.

A modo de ejemplo, se muestra en la Figura 5.8 el comportamiento de las dos alternativas, utilizando para ello la topología en *X*. En este caso, se representa, la transmisión de un segmento de datos y su correspondiente *ACK* para cada una de las dos conexiones activas en el escenario, analizando el número de transmisiones necesarias. Para un esquema tradicional *store-and-forward* (ver Figura 5.8a) se requieren hasta ocho envíos, ya que el nodo central R_1 simplemente reenviará los datos previamente recibidos. En cambio, utilizando el esquema de *NC* descrito en la Sección 5.1, el nodo C_1 podría ahorrar una transmisión, combinando los segmentos de datos. Gracias a la naturaleza *broadcast* del medio inalámbrico, los destinos D_1 y D_2 podrán escuchar los paquetes nativos transmitidos originalmente por S_2 y S_1, respectivamente, permitiendo llevar a cabo el proceso de decodificación. Sin embargo, no se contempla la codificación de paquetes cortos, por lo que los reconocimientos *TCP* atraviesan la red según un esquema tradicional. En total, serán 7 las transmisiones requeridas, suponiendo un ahorro de un 20 % (asumiendo que los nodos receptores envían un *ACK* hacia atrás).



(a) Encaminamiento tradicional (b) *Network Coding*

Figura 5.8: Impacto de los reconocimientos *TCP*



(a) Recepción y almacenamiento (b) Transmisión única

Figura 5.9: Combinación de las técnicas de *NC* con la encapsulación de los reconocimientos *TCP*

Sin embargo, la utilización de las técnicas de *NC* puede verse aún más beneficiada si se combina con el esquema de encapsulación presentado, cuya operación se resume en la Figura 5.9. El nodo codificador almacena (durante un tiempo finito) los reconocimientos recogidos a lo largo de la comunicación; mientras tanto, el tráfico de datos mantendrá su comportamiento habitual. En el caso de la Figura 5.9a, C_1 esperará hasta que se produzca una *oportunidad de codificación*, generada por la llegada del segmento q_{k+1}, ya que el propio nodo había almacenado previamente el segmento p_{N+1}, perteneciente a un flujo diferente. En este momento, y antes de enviar el paquete codificado a las capas inferiores, se comprueba si hay espacio disponible en el paquete. De ser así, se introducen (sin codificar) tantos *ACKs* como sea posible, consiguiendo evitar la sobrecarga que producirían las transmisiones aisladas de los reconocimientos, enviando la información correspondiente a los cuatro segmentos en una única transmisión "física" (ver Figura 5.9b). Cada vez que se consigue completar un proceso combinando la codificación de dos segmentos de datos más la encapsulación de dos *ACKs* (el número no está limitado a dos), el ahorro global se sitúa alrededor del 37.5 %.

Una vez descrito el esquema de encapsulación a través de un ejemplo práctico, se procederá a describir su funcionamiento con un mayor nivel de detalle:

a) Tras la recepción de un segmento, el nodo intermedio, comprobará si se trata de un *ACK puro*, esto es, un segmento que no transporta datos de aplicación.

b) En caso afirmativo, el reconocimiento se almacenará en un nuevo contenedor, denominado *buffer de reconocimientos* (B^{ACK}), donde se mantendrá a la espera de una *oportunidad de encapsulación*. En otro supuesto, el paquete atravesará de manera transparente el esquema de encapsulación de *ACKs* y continuará su flujo habitual. Al igual que el *buffer de codificación*, la operación de este contenedor dependerá de dos parámetros principales: el tamaño del *buffer* (número de reconocimientos total que puede almacenar), B_S^{ACK}, y el tiempo máximo de permanencia, fijado por el parámetro B_{TO}^{ACK}. Sin embargo, la estructura del *buffer* será ligeramente distinta: mientras que para los datos se utilizará una cola para cada flujo activo, en el caso de los reconocimientos, éstos se almanenarán en una única cola *FIFO*, por lo que no se hará distinción a la hora de proceder a su encapsulación.

c) En el momento en el que la entidad *NC* del nodo intermedio vaya a generar un paquete se comprobará, en primer lugar, si dispone del espacio suficiente para incluir reconocimientos *TCP* (incluyendo los 2 *bytes* correpondientes a la *identidad del flujo*) sin sobrepasar el *MTU*. En caso afirmativo, irá incluyendo *ACKs* guardados en el B^{ACK}, bien hasta que se alcanza el tamaño máximo o hasta que se vacía el propio *buffer*.

d) Cuando no se consiga encapsular ningún reconocimiento, ya sea porque el temporizador expira antes de que se haya producido una *oportunidad de encapsulación* o porque se recibe un *ACK* estando el *buffer* lleno, la entidad *NC* se encargará de generar un

nuevo paquete, en el que se incluyen todos los reconocimientos almacenados hasta el momento en el B^{ACK} (siempre y cuando no se sobrepase el MTU).

e) En recepción, el nodo procesará la cabecera *NC*, analizando, a través del identificador del flujo, si alguno de los reconocimientos va dirigido a él; en caso afirmativo, extraerá el (los) *ACK*(s) correspondientes, alojado(s) (sin codificar) en el interior de la cabecera *NC*, entregándoselo(s) a la capa superior, que será la encargada de procesarlo(s).

f) Por último, es preciso remarcar que esta funcionalidad no está directamente conectada con la codificación de los paquetes, por lo que se podrá utilizar en combinación directa con el protocolo *TCP*.

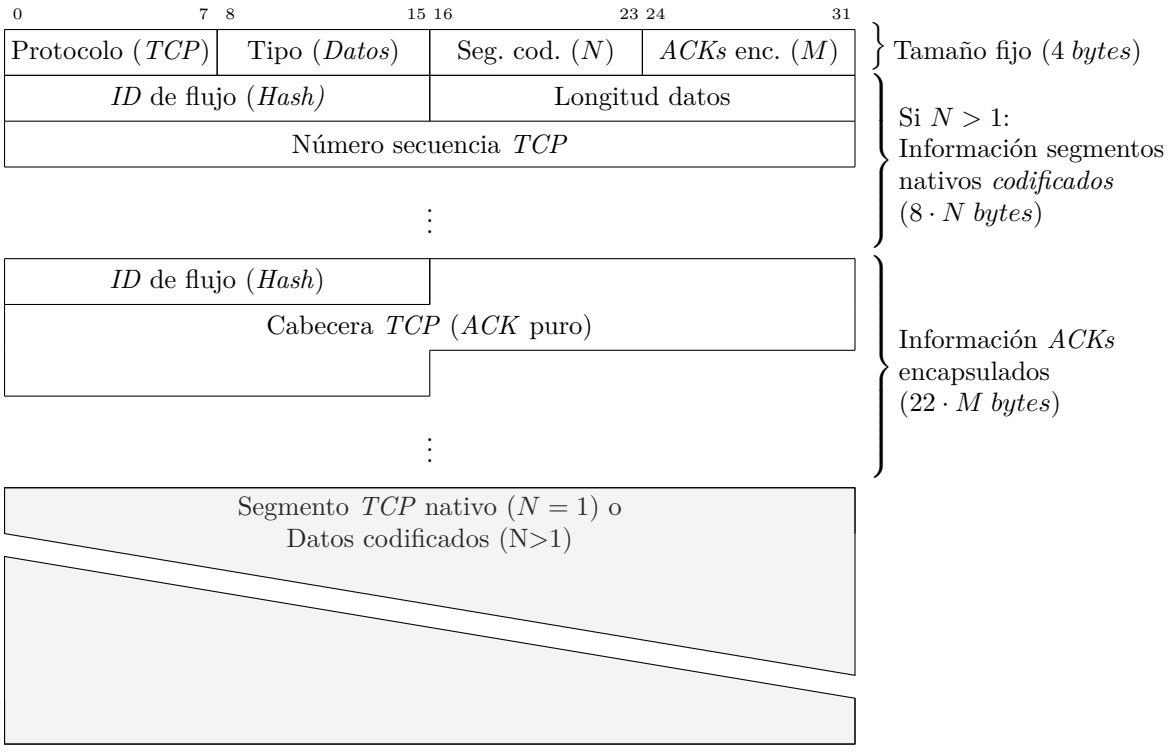


Figura 5.10: Formato de la cabecera *NC* con *ACKs* encapsulados

Para habilitar esta funcionalidad, será necesario añadir al protocolo nuevos elementos dentro la cabecera *NC*, como muestra la Figura 5.10. En primer lugar, se añade un campo para indicar el número de *ACKs* encapsulados en el paquete. Posteriormente, se agregarán los *ACKs* al final de la cabecera *NC*, junto con su *identificador de flujo*, que permitirá su inmediata extracción en el destino correspondiente, ocupando un total de *22 octetos por cada segmento encapsulado*.

De un modo similar a la solución mostrada en [Seng10], podría plantearse en un futuro la reducción de la información incluida en los reconocimientos encapsulados, incluyendo

solamente los campos imprescindibles (i.e. número de secuencia y *ACK*, *flags*, tamaño de la ventana de congestión y *Cyclic Redundancy Check (CRC)*, por ejemplo). Para ello, sería necesario generar una versión "reducida" de la cabecera *TCP* en los nodos intermedios, que se combinaría con la anterior en un proceso de reconstrucción (en el nivel *NC*) previo a su entrega a la capa de transporte.

5.3 Versión avanzada

Tras un profundo análisis del rendimiento mostrado por los mecanismos de *NC inter-flujo* presentados hasta este punto, cuyos principales resultados se resumirán en la Sección 5.5, se puede afirmar que la combinación de las técnicas de *NC* con el protocolo *TCP* no ofrece unas prestaciones adecuadas sobre canales hostiles.

En la versión inicial del protocolo, la capa *NC* se encargaba exclusivamente de las tareas de codificación/decodificación y las correspondientes a la gestión de cabeceras. Como se verá más adelante, es capaz de mejorar significativamente el rendimiento sobre canales ideales, consiguiendo incrementar el *throughput* gracias a la reducción del número de transmisiones.

Sin embargo, en el momento en el que las conexiones empiezan a experimentar errores de transmisión, se comienza a observar un empeoramiento importante en el comportamiento global, debido a la reacción de los mecanismos de control de congestión de *TCP* ante las pérdidas. Además, como ya se ha mencionado con anterioridad, la presencia de paquetes codificados en la red hace que se incremente la probabilidad de que se produzcan procesos de decodificación erróneos. Hay que tener en cuenta que, junto a la recepción correcta en los nodos finales de los paquetes codificados, deberían haberse escuchado con anterioridad al menos uno de los paquetes *nativos* que componen el mensaje, para que el proceso de decodificación sea completado con éxito. Por lo tanto, serán necesarias cuatro recepciones correctas para que dos nodos[15] consigan extraer la información de un paquete codificado. A esto hay que sumarle el problema derivado del direccionamiento *unicast*, que limita el uso de los mecanismos de retransmisión del estándar *IEEE 802.11* en sólo uno de los cuatro eventos de recepción (aquél en el que la dirección *MAC* destino de la trama coincida con la del nodo que la está procesando), correspondiente a uno de los que reciben el paquete codificado. En consecuencia, estas pérdidas, interpretadas por el nivel de transporte como indicadores de congestión en la red, inducirán la retransmisión de los segmentos que la entidad *TCP* asume como perdidos, dando lugar a una penalización en la tasa de envío (i.e. ventana de congestión) de cada una de la fuentes, disminuyendo el *factor de acoplamiento* de las diferentes conexiones. Dicho en otras palabras, cuando los diferentes flujos pierden su sincronía, el número de *oportunidades de codificación* se verá seriamente reducido (o incluso anulado), lo que provocará que los retardos artificiales causados por el almacenamiento

[15] Asumiendo un paquete que contiene dos segmentos *nativos*.

temporal de los segmentos en los *nodos codificadores* incremente el RTT de la conexión o, lo que es lo mismo, que el rendimiento se reduzca significativamente.

Para paliar este deterioro, se presenta en esta sección una nueva versión del protocolo *inter-flujo*. A partir de una serie de modificaciones en su comportamiento, se añaden un conjunto de funcionalidades utilizadas para mitigar el pobre rendimiento observado cuando la entidad de *NC* no es capaz de enmascarar la pérdida de segmentos producida durante el intercambio de información. Es importante destacar que el *gran objetivo* de estos cambios radica en *conseguir reducir al máximo las retransmisiones extremo-a-extremo producidas en el nivel* TCP. Para ello, la contribución más importante será la introducción de un esquema paralelo de retransmisión nativo en el nivel *NC*, donde los nodos intermedios almacenarán los segmentos recogidos de manera que, en caso de que se detecte una situación de error o pérdida, se pueda recuperar la información sin la necesidad de llegar hasta el extremo de la conexión (el nodo fuente).

A continuación se presentarán los principales elementos que han sido incorporados en esta versión del protocolo, así como una breve descripción de su influencia en el comportamiento del mismo:

- El primer y gran cambio en el diseño del protocolo viene dado por el hecho de que la entidad *NC* ya no ignorará los flujos *TCP* que la atraviesan, dejándolos pasar de manera transparente, sino que deberá monitorizar el estado de las conexiones, reaccionando ante posibles situaciones adversas. Para ello, se analizará el contenido de toda cabecera *TCP* recibida en el nivel *NC*, actualizando la información correspondiente en el contenedor que almacena la información asociada al *identificador de flujo* del segmento procesado. En este caso en concreto, se añadirán, como campos de información, tanto el número de secuencia como el de reconocimiento. Además, cobrará una especial relevancia la presencia de los *flujos inversos*, donde el tráfico ascendente de los datos de aplicación lleva intrínsecamente asociado otro de reconocimientos, que recorre la red en sentido inverso. De esta manera, la recepción de un *ACK* en un nodo permitirá eliminar todos los segmentos almacenados en los diferentes *buffers* de la entidad *NC* que ya hayan sido confirmados (i.e. $Num_{seq} < Num_{ACK}$), evitando así ocupar el canal con transmisiones innecesarias. Es preciso destacar también que los nodos extremos podrán acceder directamente al nivel *TCP*, para obtener el número de *ACK* correspondiente, sin tener que esperar a que el reconocimiento atraviese la capa *NC* (recuérdese que el protocolo *TCP* [Post81] normalmente no confirma cada segmento recibido individualmente).

  El coste asociado a este procedimiento es el incremento de la sobrecarga necesaria para transportar el número de reconocimiento en la parte correspondiente a la información de los segmentos nativos en la cabecera *NC* (ver Figura 5.3). En total, se necesitarán 12 octetos adicionales por segmento.

- En la implementación inicial del protocolo *NC* se incluía un *buffer de decodificación* en los nodos finales de la red, cuya labor era la de almacenar los segmentos *nativos*

recibidos, bien desde los niveles inferiores (en el caso de las estaciones receptoras) o bien desde la capa de transporte (en las transmisoras). Esta información era utilizada posteriormente para llevar a cabo la decodificación de un paquete codificado.

Sin embargo, en la versión mejorada, todo nodo que reciba un segmento *TCP* guardará una copia local del mismo, manteniendo los parámetros del *buffer* (i.e. número de paquetes que puede almacenar por flujo, tiempo máximo de permanencia), sin añadir campos adicionales. El nombre de este contenedor pasará a ser *cache de paquetes nativos*, ya que su funcionalidad no estará exclusivamente ligada a las tareas de decodificación.

Como ya se ha adelantado, el elemento clave en esta actualización del protocolo consiste en la utilización de un esquema paralelo de retransmisiones a nivel *NC*, cuyo principal objetivo será evitar, en la medida de lo posible, la generación de retransmisiones en las entidades *TCP* transmisoras. Para ello, se definirán nuevos mensajes de *solicitud y respuesta de retransmisión* en el protocolo. En el caso de recibir una *solicitud de retransmisión*, la entidad *NC* buscará si la *cache de segmentos nativos* guarda una copia, generando, en caso afirmativo, un mensaje de *respuesta* con el segmento en cuestión. Más adelante se detallará su operación con una mayor profundidad.

- Junto con la introducción del citado almacenaje de los segmentos *nativos* presentada en el punto anterior, los nodos receptores mantendrán una segunda *cache*, donde se guardará una copia de aquellos paquetes codificados en los que no haya sido posible recuperar el segmento nativo correspondiente. Así, en el momento de producirse un *error de decodificación*, la estación receptora generará una *solicitud de retransmisión* del segmento (o segmentos) restante(s).

- La introducción de las nuevas funcionalidades requiere la modificación de algunos de los mecanismos presentados en la Sección 5.1. Mientras que el esquema de codificación (Figura 5.4) permanece, en esencia, sin sufrir grandes alteraciones[16], en la parte que corresponde a la decodificación (cuya operación inicial se presenta en la Figura 5.5) se introducen dos cambios fundamentales, ilustrados en la Figura 5.11:

  1. En primer lugar, la reacción de la versión inicial ante un *error de decodificación* era el descarte directo del paquete codificado. Sin embargo, la información recibida podría haber sido utilizada en el futuro para recuperar algún segmento nativo, por lo que se guardarán copias de todos aquellos paquetes de los que no se haya podido extraer toda la información. Posteriormente, se generará un mensaje de *solicitud de retransmisión*, solicitando el reenvío de alguno de los segmentos nativos que componen el mensaje. Es importante destacar en este punto que la reacción de los nodos intermedios tras la recepción de este tipo

[16]Los cambios más significativos guardan relación con la eliminación de los segmentos ya confirmados por el número de ACK recibido del flujo inverso antes del envío de un paquete a los niveles inferiores.

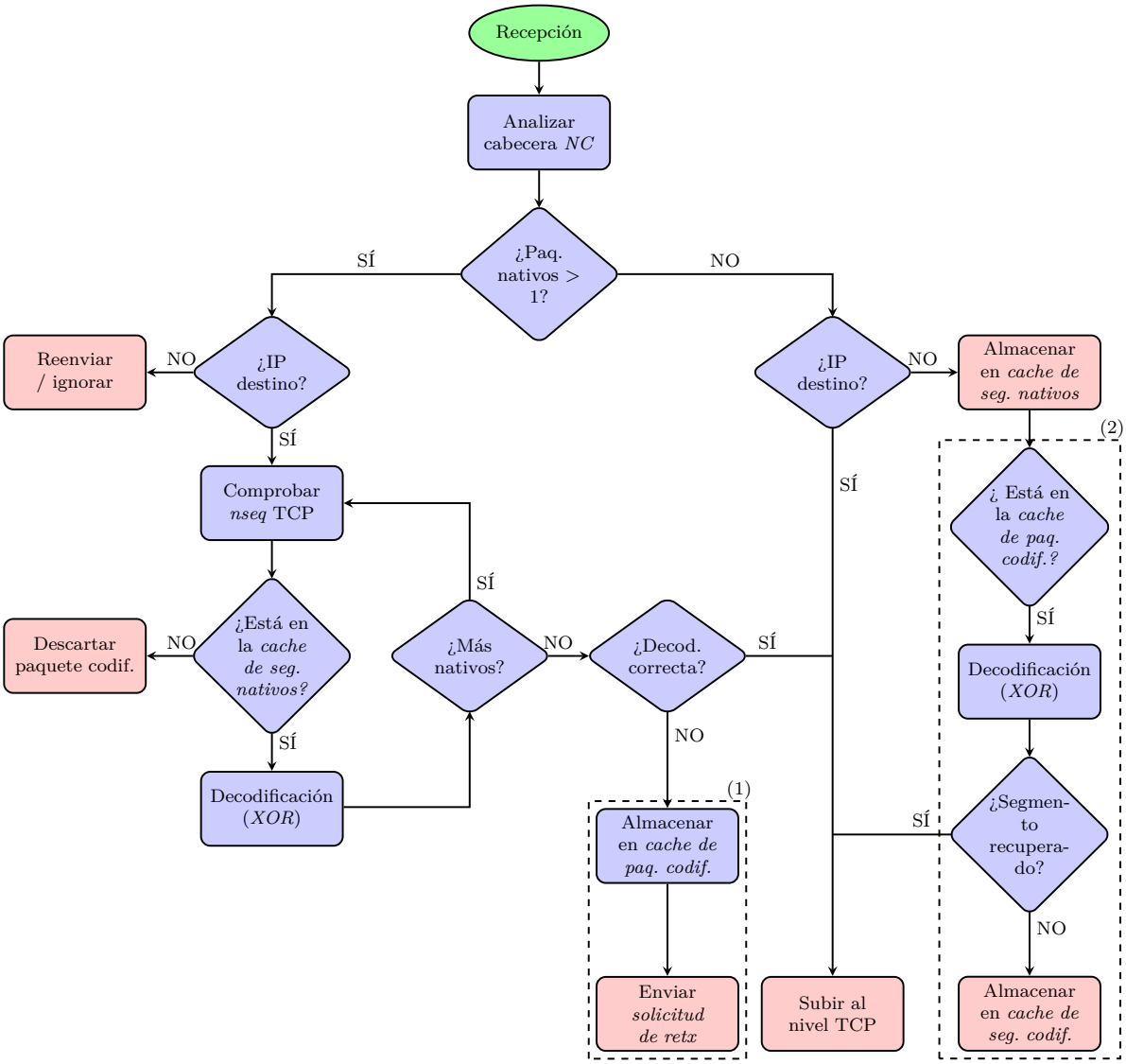


Figura 5.11: Diagrama de flujo de la etapa de decodificación de un paquete en el protocolo de *NC inter-flujo* (Versión avanzada)

de mensajes *no supone el envío inmediato* del segmento (o segmentos) solicitado(s), activando los mecanismos de gestión de retransmisiones del nivel *NC* presentados más adelante.

2. El segundo de los cambios está ligado a la respuesta del nodo receptor ante la llegada de un segmento *nativo*, donde en el diseño original solamente se guardaba una copia en el *buffer de decodificación* (ahora llamado *cache de segmentos nativos*). En esta nueva versión se buscará en la *cache de segmentos codificados* la presencia de alguna combinación que contenga al segmento *nativo* recibido. Si se encontrase, se efectuaría una operación *XOR* entre ambos, comprobando posteriormente si la información resultante se corresponde con un segmento *nativo* destinado al nodo, enviándolo inmediatamente al nivel superior. Por el contrario, si el número de segmentos codificados en el paquete siguiese siendo mayor que uno, se guardaría de nuevo en la *cache de paquetes codificados*, a la espera de una nueva *oportunidad de decodificación*. Merece la pena recordar que la aplicación del operador *OR exclusivo* entre un paquete codificado (con N segmentos) y un segmento *nativo*, equivale a la eliminación de éste del primero, reduciendo en una unidad el número de segmentos "mezclados".

- Como se ha incidido repetidamente en los puntos anteriores, la clave de esta nueva versión del protocolo radica en la utilización de un esquema de retransmisiones paralelo al propio de *TCP*. Utilizado una idea similar a la presentada por Amir et al. en el protocolo *Snoop* [Amir95], se ha implementado un sistema basado en *caches* en los nodos intermedios, que permitirán una reacción más rápida ante la pérdida de segmentos, ya que las retransmisiones no tendrán que recorrer toda la red para alcanzar al destino. Gracias a ellas, cualquier nodo que reciba una *solicitud de retransmisión* podrá enviar el segmento en cuestión, reduciendo en gran medida la sobrecarga asociada a la transmisión extremo a extremo, sobre todo en canales basados en contienda. Además de la respuesta a *solicitudes de retransmisión* explícitas, la entidad *NC* procesará los reconocimientos recogidos en el nodo, detectando la presencia de *ACKs* duplicados. Manteniendo una solución similar a la clásica de *TCP*, tras la recepción de *tres ACKs duplicados* consecutivos, el nodo interpretará que se ha producido una pérdida, y procederá al envío del segmento en cuestión (si se encuentra en la *cache de segmentos nativos*) o a la propagación de una *solicitud de retransmisión* en el caso de no tener una copia del mismo. Con el fin de reducir el número de retransmisiones tras *triple ACK duplicado* en la entidad *TCP* transmisora, se evitará en la medida de lo posible la propagación de los mismos hacia el origen. Sin embargo, esto podrá tener un efecto adverso, ya que este bloqueo puede derivar en un aumento del *RTT* e incluso en la aparición de retransmisiones por *RTO*, por lo que será esencial llevar a cabo una caracterización del comportamiento de esta solución, estableciendo unos criterios que optimicen la operación de este mecanismo.

  En la Figura 5.12 se representa el diagrama de flujo seguido en la entidad de *NC* tras la aparición de un evento de retransmisión. Como ya se ha mencionado anteriormente,

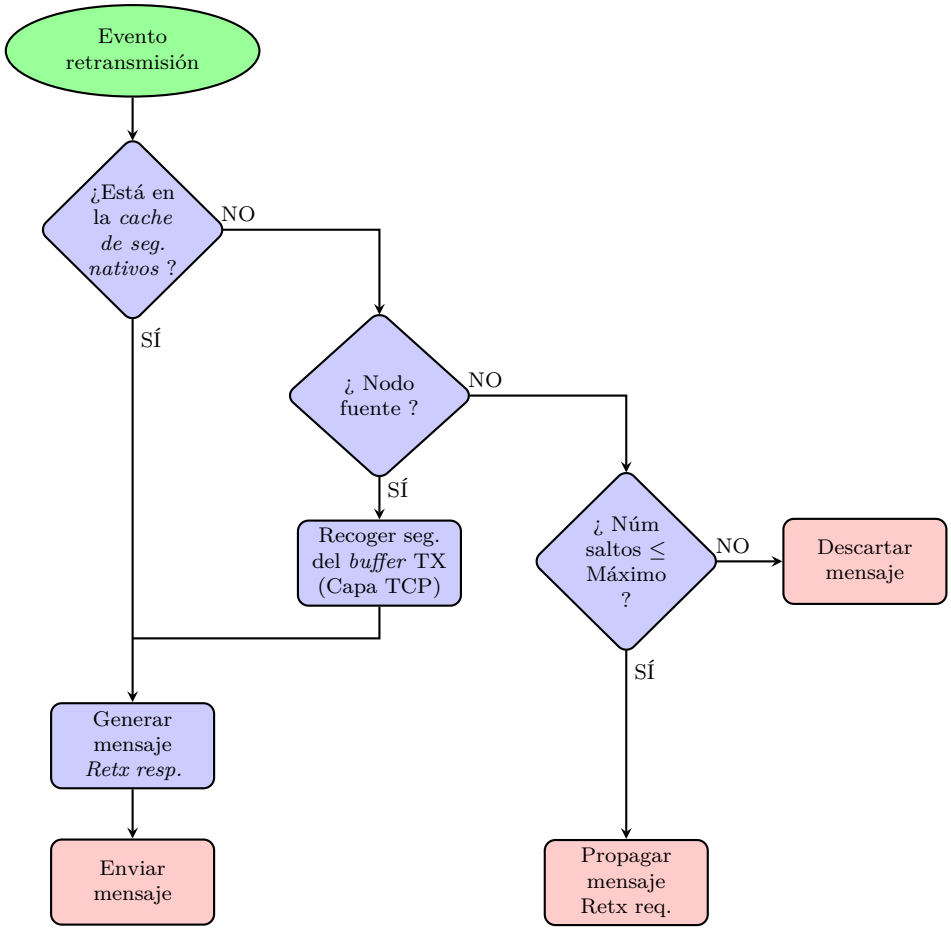


Figura 5.12: Diagrama de flujo de la etapa de decodificación de un paquete en el protocolo de *NC inter-flujo* (Versión avanzada)

el primer paso a seguir será la búsqueda del segmento *nativo* en el interior de la *cache de segmentos nativos*. Si se encontrase allí, bastaría con copiar el segmento (no se eliminará hasta que se reciba el *ACK* que confirme su correcta recepción) y generar un mensaje de respuesta, enviándolo a los niveles inferiores. Por el contrario, si no se hubiera copia del mismo, surgirían dos posibilidades diferentes: en la primera de ellas, cuando se trata de un nodo fuente, se extraerá la información directamente del *buffer* de transmisión de la capa *TCP*, pasando directamente a generar un mensaje de *respuesta de retransmisión*. En el resto de los casos, si no se dispone del segmento requerido, se propagará el mensaje de *solicitud de retransmisión* hacia atrás. Ahora bien, para evitar inundar la red con estos mensajes, se limita su reenvío a un número máximo de saltos, descartando cualquier retransmisión tras superarlo.

Además, se descartará el envío de solicitudes o respuestas de retransmisión si se ha detectado previamente la transmisión de la misma por parte de algún nodo vecino, evitando así tráfico redundante en la red.

- Para poder intercambiar los mensajes correspondientes a este esquema de retransmisión propio del nivel *NC*, se añadirán al protocolo las nuevas cabeceras necesarias para cada una de las primitivas. En el caso de una *solicitud de retransmisión*, como muestra la Figura 5.13, se distinguen dos partes, al igual que para los mensajes de datos: una de longitud fija (4 *bytes*) y otra variable, que incluye la información correspondiente a los segmentos solicitados. En el momento de generar estos mensajes, se podrán solicitar tantos segmentos como sea necesario, siempre y cuando vayan dirigidos a la misma dirección *IP* destino. La parte fija incluye el protocolo de nivel superior, el tipo de mensaje (*solicitud de retransmisión*), el número de saltos[17] y el número de segmentos incluidos en el mensaje. La parte variable, al igual que en los mensajes de datos, transportará el *identificador de flujo*, la longitud de los datos del nivel de aplicación y el número de secuencia de los mismos en la sesión de transporte correspondiente. En total, serán 8 *bytes* por segmento.

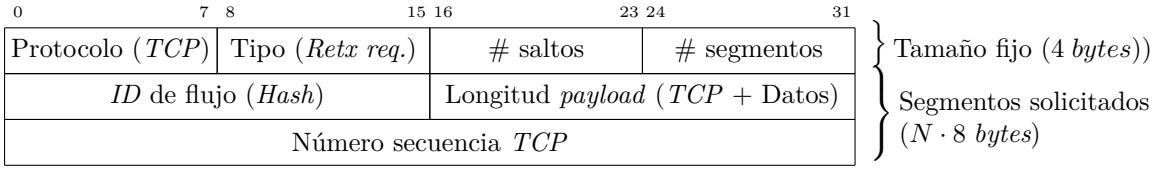


Figura 5.13: Formato de la cabecera para la solicitud de retransmisión del protocolo *inter-flujo* (*Retx request*)

- Estrechamente ligada a una solicitud se encuentra la *respuesta de retransmisión*, cuyo formato se resume en la Figura 5.14. En este caso se incluye, en la cabecera *NC*, la información correspondiente al protocolo de nivel superior, el tipo de mensaje (*respuesta de retransmisión*) y el *identificador de flujo*, añadiendo posteriormente el segmento *nativo* en cuestión. En total, la cabecera *NC* tiene una longitud en total de 4 octetos.

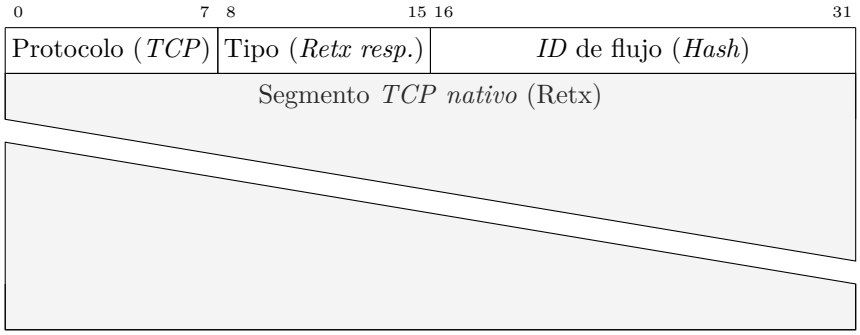


Figura 5.14: Formato de la cabecera para la respuesta a la solicitud de retransmisión del protocolo *inter-flujo* (*Retx response*)

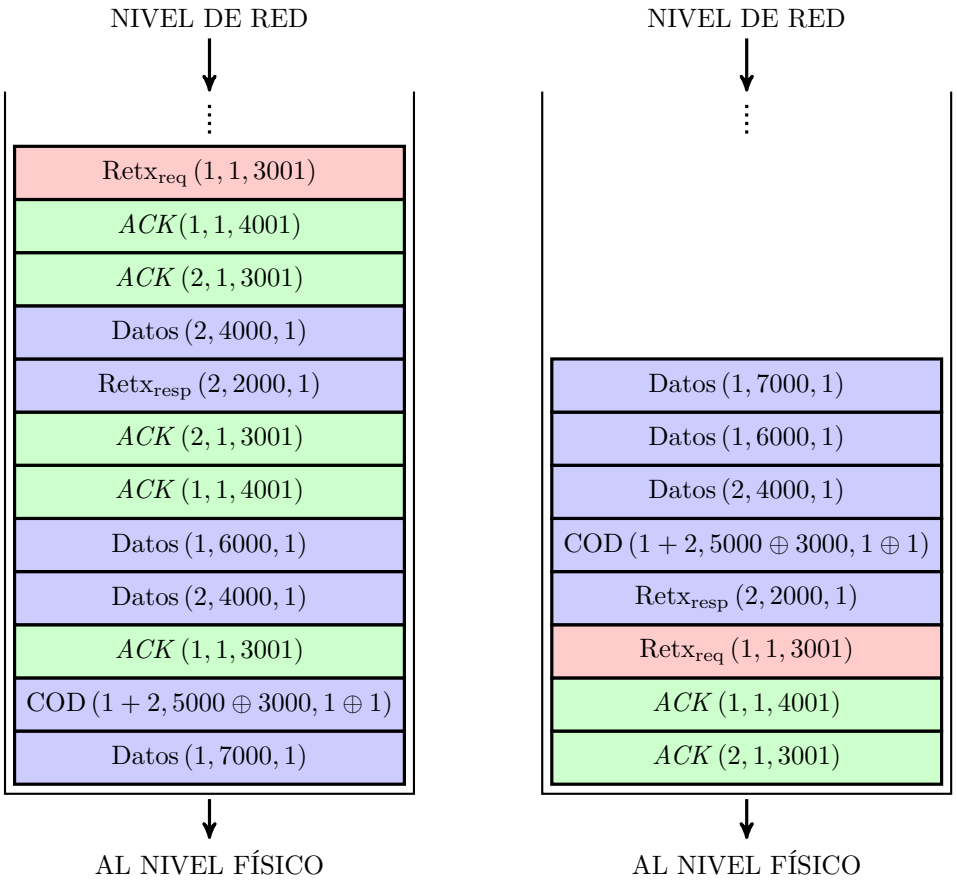
[17]Como se ha dicho anteriormente, se limitará el número de saltos que pueden dar estas solicitudes, evitando la inundación de la red.

- Por último, se ha definido una solución *cross-layer*, mediante la cual se tratará de maximizar la utilización del canal, evitando la transmisión de información innecesaria (o duplicada) y ordenando los datos antes de su transmisión, lo que reducirá la generación de *ACKs* duplicados por parte de los nodos finales. Para ello, se ha establecido una comunicación (señal) entre la entidad *NC* y el nivel de enlace *IEEE 802.11* (que es donde se gestiona la cola de transmisión), para que se ordene y "limpie" el *buffer* de nivel *MAC*, como paso previo a la transmisión de una trama.

  La Figura 5.15 muestra una captura con el contenido del *buffer* de transmisión en un instante cualquiera ante una situación de saturación (Figura 5.15a), pudiendo observar la diversidad de tramas almacenadas. Al tratarse de una cola *FIFO*, el nivel de enlace irá dando salida a los paquetes respetando el orden de llegada, lo que hará que la información pueda estar almacenada en desorden o duplicada, factores que conducirán a un deterioro en el rendimiento, sobre todo en *TCP*, donde la llegada al destino de segmentos "fuera de orden" conducirán al envío inmediato de un *ACK* duplicado. Además, se incrementará el *RTT*, aumentando las posibilidades de que expire el temporizador *RTO* en alguno de los nodos transmisores, dando lugar a nuevas retransmisiones y a la disminución de la tasa de envío, así como el tamaño de la ventana de congestión del nivel *TCP*.

  Para reducir el impacto de las situaciones anteriores, cuando se trate de un nodo fuente, se añadirán al esquema los mecanismos necesarios para manipular la información almacenada en la *cola de transmisión* del nivel de enlace *IEEE 802.11*, en tiempo real. Para ello, tal y como se representa en la Figura 5.15b, cada vez que el nivel de enlace reciba una llamada de la entidad *NC*, se procederá a la ordenación de dicho *buffer* (por ejemplo, tras el procesado de un *ACK* de *TCP* en el nivel *NC*) incluyendo, como parámetros de entrada, un *identificador de flujo* (*hash*) y un número de reconocimiento asociado a dicha conexión, eliminando todos aquellos segmentos de datos con un número de secuencia menor. Junto con este proceso de limpieza de información ya confirmada, se ha establecido un sistema de prioridades mediante el cual los reconocimientos *TCP* pasarán a ocupar la cabeza de la cola, seguidos por las solicitudes de retransmisión del nivel *NC*; finalmente, los mensajes de datos, que quedarán relegados a la parte final del *buffer*. A pesar de que pueda parecer que con esta solución se penaliza al tráfico de datos, la frecuencia del resto de paquetes es muy inferior, por lo que el impacto se limita notablemente. Además, al utilizar el esquema de encapsulación de reconocimientos *TCP* combinado con las técnicas de *NC*, se evitaría el problema que pueda generar la "priorización" del tráfico de *ACKs*.

  Junto con la asignación de los diferentes niveles de prioridad, surge la cuestión de cómo ordenar los mensajes del mismo tipo cuando los datos pertenecen a diferentes flujos. En este supuesto, se seleccionará aleatoriamente, en cada llamada, el orden a seguir en el proceso, siguiendo un reparto equitativo ($\sim$ *Round Robin*) mientras haya paquetes. No obstante, existen matices que deben gestionarse en cada uno de los tres niveles de prioridad: en primer lugar, en lo que respecta a los reconocimientos *TCP*, se transmitirán exclusivamente aquéllos con el número de reconocimiento más alto,



(a) Estado de la cola en un instante t (b) Estado de la cola tras la reordenación

Figura 5.15: Esquema de reordenación por prioridades de la cola de transmisión en el nivel de enlace *IEEE 802.11* en un nodo intermedio de la red. La representación de los datos se corresponde con la notación "Tipo (ID_{flujo}, Seq_{num}, ACK_{num})" en un instante arbitrario de una transmisión

evitando la transmisión por duplicado de los mismos. Es importante destacar que la entidad *NC* monitoriza el estado de las conexiones, con lo que puede detectar la presencia de *ACKs* duplicados, bloqueando además el reenvío de los reconocimientos hacia atrás, con lo que se evita la posible aparición de *triples ACKs duplicados* en los nodos fuentes. Para las *solicitudes de retransmisión*, el único criterio será la eliminación de mensajes duplicados, ordenándolos según el número de secuencia de los segmentos solicitados. Por último, el tráfico de datos incluye tanto segmentos *nativos* como paquetes codificados o *respuestas de retransmisión*. Al igual que para el caso anterior, se reordenarán los elementos en función de su número de secuencia, con la salvedad de los paquetes codificados (contienen varios), donde se tomará como referencia el número de secuencia de uno de ellos, escogido aleatoriamente.

5.4 Algoritmo de selección de nodos codificadores

En los casos presentados hasta este momento no se ha considerado el proceso de asignación de los nodos que van a llevar a cabo las tareas de codificar los paquetes. En un escenario canónico, donde los nodos se encuentran colocados de manera sintética, para mostrar los beneficios del *NC*, es inmediato identificar cuál de todos es el más idóneo para convertirse en el *elemento de codificación* de la red. A simple vista, aquellos nodos que se encuentren en un punto de articulación del grafo que modela la red serán los potenciales candidatos, ya que lo más probable es que el tráfico de los diferentes flujos los atraviese. Recuérdense las condiciones expuestas en la definición del protocolo, donde se exige que los nodos decodificadores deban ser capaces de escuchar los segmentos *TCP nativos* de otros flujos. De no cumplirse, no podrán recuperar la información contenida en los paquetes codificados, y los beneficios asociados a las técnicas de *NC* quedarán anulados, ofreciendo un rendimiento inferior al de un esquema *TCP* estándar.

Sin embargo, en un caso genérico, con topologías más complejas, la elección de los *nodos codificadores* aparece como un problema más complicado. De hecho, en un gran porcentaje de escenarios, la utilización de los mecanismos de *NC* no conllevará ningún tipo de beneficio, bien porque las rutas seguidas por los flujos de datos no favorecen su combinación (no se encuentra un *nodo codificador apropiado*) o bien porque el rendimiento observado resulta inferior al obtenido con esquemas de comunicación clásicos.

En esta sección se presenta un nuevo algoritmo que, basándose en [Le10; Guo11], permite identificar la presencia de *oportunidades de codificación* en cualquier red mallada inalámbrica. Para ello, y a partir del grafo $\mathcal{G}(V, E)$ que subyace de un escenario y los nodos entre los que se intercambiará la información (las parejas origen-destino), el algoritmo resolverá las siguientes cuestiones:

1. Definir si el escenario reúne las condiciones necesarias para que la utilización de los mecanismos de *NC* resulte beneficiosa.
2. Si se cumple el punto anterior, deberá establecer, de entre todos los nodos intermedios, cuál (o cuáles) debería(n) codificar los paquetes pertenecientes a flujos diferentes.

A modo de ejemplo, la Figura 5.16 muestra un escenario ilustrativo en el que se puede deducir cuál de los nodos es el más apropiado para "mezclar"el contenido de las dos conexiones. Se observa la presencia de dos flujos (uno en azul y otro en rojo), siendo los nodos 1 y 6 los transmisores, mientras que 7 y 3 se corresponderán con los destinos. Puede apreciarse claramente como el nodo 5 aparece como el único punto de articulación entre ambas rutas, por lo que los segmentos generados por ambas fuentes pasan a través de él, haciendo que sea la elección obvia para codificar la información.

Por otro lado, gracias a las propiedades *broadcast* que ofrece el medio inalámbrico, ambos nodos 3 y 7 serán capaces de escuchar los datos transmitidos por las fuentes de los

otros flujos, 1 y 6, respectivamente. Por lo tanto, puede afirmarse que la utilización de las técnicas de *NC* en este escenario podrán generar un beneficio con respecto a un uso tradicional del protocolo *TCP* (suponiendo un ahorro bruto de un 16.6 % de las transmisiones necesarias).

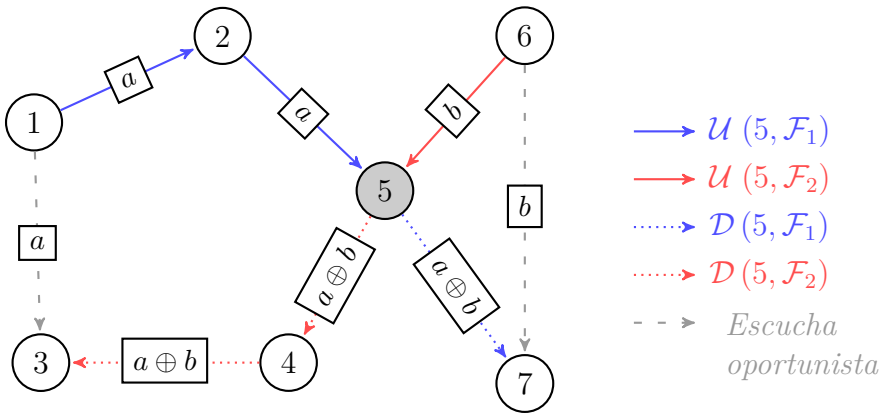


Figura 5.16: Topología sencilla para demostrar los criterios que deben seguirse para la elección de los *nodos codificadores*

Con el fin de mantener una notación apropiada, se resumen a continuación los principales parámetros y condiciones utilizados durante la definición de los algoritmos:

- En primer lugar, se utiliza la variable $\mathcal{N}(i)$ para definir el conjunto de nodos adyacentes al nodo i-ésimo, que tendrán una especial relevancia para el aprovechamiento de la *escucha oportunista*.
- $\mathcal{F}_i$ representa al flujo i-ésimo en el escenario. Además, $j \in \mathcal{F}_i$ identifica a cada uno de los nodos atravesados por dicho flujo.
- Manteniendo la misma tónica de los análisis anteriores sobre escenarios sintéticos, se limitará el tráfico a dos flujos activos simultáneos ($S_1 \rightarrow D_1$ y $S_2 \rightarrow D_2$).
- Como se verá más adelante, la técnica utilizada por el algoritmo se basa en un esquema "*dentro-fuera*". En concreto, se calculará el camino (o caminos) más cortos desde cualquier nodo hasta los cuatro nodos origen-destino. Para ello, será necesario distinguir entre dos tipos de flujos:

  1. Los flujos de subida (*upstream*) van desde un nodo j hasta los transmisores. El conjunto $\mathcal{U}(j, \mathcal{F}_i)$ contiene a todos los nodos que son atravesados por el flujo i-ésimo; la información viajará en forma *nativa*, ya que aún no se habrá alcanzado al nodo codificador.
  2. Los flujos de bajada (*downstream*) entre un nodo j y los receptores. En este caso, el conjunto de nodos $\mathcal{D}(j, \mathcal{F}_i)$ podrá transportar la información en forma de *paquetes codificados*.

- Finalmente, aquellos elementos que cumplan las condiciones impuestas por los algoritmos serán incluidos en la lista de potenciales *nodos codificadores de la red*: $\mathcal{C} = \{c_0, c_1, ..., c_{N-1}\}$.
- La posición de los nodos permanecerá constante durante todo el proceso de análisis.

En relación a las condiciones que deben darse en un red mallada inalámbrica para que los mecanismos de *NC inter-flujo* generen beneficios, Le et al. plantean en [Le10] el conjunto de *condiciones necesarias* para que exista la posibilidad de encontrar, al menos, un nodo codificador:

DEFINICIÓN **5.1.** *Condiciones de codificación en un escenario con dos flujos $\mathcal{F}_1$ y $\mathcal{F}_2$. Deberán cumplirse la primera y una de dos siguientes*

1. *Existe un nodo* $c \mid c \in \mathcal{F}_1 \cap \mathcal{F}_2$.
2. *Existe un nodo* $x \in \mathcal{D}(c, \mathcal{F}_j) \mid x \in \mathcal{N}(y)$, *con* $y \in \mathcal{U}(c, \mathcal{F}_k)$, $j, k = [1, 2]$ *y* $j \neq k$.
3. *Existe un nodo* $x \mid x \in \mathcal{D}(c, \mathcal{F}_j) \cap \mathcal{U}(c_i, \mathcal{F}_k)$, *donde* $j, k = [1, 2]$ *y* $j \neq k$.

En otras palabras, los flujos que atraviesan un escenario cualquiera podrán ser codificados entre sí *si y sólo si* existe un elemento situado en un *punto de intersección* de ambos caminos (punto de articulación). Además, alguno de los nodos perteneciente a los flujos de bajada, $\mathcal{D}(c, \mathcal{F}_i)$, con $i \in [1, 2]$, debería poder "escuchar" los paquetes *nativos* correspondientes al flujo cruzado, bien a través de la diversidad espacial de las transmisiones sobre un canal *IEEE 802.11* (*condición 2*), o bien porque pertenece al mismo tiempo al conjunto de nodos del flujo de subida de la otra conexión (*condición 3*). Asumiendo un canal sin errores, se puede afirmar que estas condiciones son *necesarias y suficientes* para que las *oportunidades de codificación* generadas en el nodo c sean recuperadas satisfactoriamente en los destinos.

Sin embargo, la implementación del protocolo *NC* desarrollado establece una restricción adicional que no está reflejada en este conjunto de condiciones: *las tareas de decodificación serán ejecutadas única y exclusivamente en los nodos finales*. Por lo tanto, los criterios de búsqueda de escenarios "codificables" expuestos en la Definición 5.1 deberán verse modificados para adaptarse a la solución empleada, dando pie a la siguiente definición de restricciones:

DEFINICIÓN **5.2.** *Condiciones de codificación en un escenario con dos flujos $\mathcal{F}_1$ y $\mathcal{F}_2$ y decodificación exclusiva en los nodos destino D_1 y D_2. Al igual que el caso anterior, deberán cumplirse la primera y una de las dos siguientes*

1. *Existe un nodo* $c \mid c \in \mathcal{F}_1 \cap \mathcal{F}_2$.

2. *Existe un nodo* $x \in \mathcal{U}(c, \mathcal{F}_j) \mid x \in \mathcal{N}(D_k)$, *con* $j, k = [1, 2]$ *y* $j \neq k$.

3. *Existe un nodo* $y \in \mathcal{U}(c, \mathcal{F}_j) \mid y = D_k$, *con* $j, k = [1, 2]$ *y* $j \neq k$.

Estas restricciones vienen a expresar lo mismo que en la Definición 5.1, con la salvedad de que los nodos del flujo de bajada deberán corresponderse estrictamente con los destinos de los mismos, adaptándose a las condiciones de decodificación impuestas por el propio protocolo. En concreto, los nodos finales deberán ser capaces de escuchar directamente el flujo *nativo* cruzado de manera promiscua (*condición 2*), o pertenecer directamente al conjunto de nodos del flujo de subida inverso (*condición 3*). Con todo ello, se puede deducir fácilmente que, bajo *condiciones de canal ideales*, las restricciones impuestas en esta Definición 5.2 serán *necesarias y suficientes* para encontrar escenarios "codificables".

Tras la presentación de las condiciones impuestas para la búsqueda de un *codificador* y volviendo al ejemplo de la Figura 5.16, los cuatro flujos asociados al posible codificador (5) serán los siguientes: $\mathcal{U}(5, \mathcal{F}_1) = \{1,2\}$, $\mathcal{D}(5, \mathcal{F}_1) = \{7\}$, $\mathcal{U}(5, \mathcal{F}_2) = \{6\}$ y $\mathcal{D}(5, \mathcal{F}_1) = \{4,3\}$. Además, se aprecia claramente como los nodos 3 y 7 pertenecen al conjunto de vecinos $\mathcal{N}(1)$ y $\mathcal{N}(6)$, respectivamente, por lo que los nodos finales serán capaces de escuchar los paquetes nativos (en este caso, directamente de sus fuentes originales) de los flujos cruzados.

Una vez se ha establecido el conjunto de condiciones para que el protocolo pueda ser aplicado en cualquier topología de red, el Algoritmo 5.1 describe el proceso de selección del nodo codificador, determinando si el escenario analizado presenta las características necesarias para que la utilización de las técnicas de *NC inter-flujo* sean viables.

Como parámetros de entrada, el algoritmo recibe el grafo $\mathcal{G}(V, E)$ que modela la topología de red, estableciendo los enlaces a partir de un área de cobertura predefinida[18]. Se necesita añadir también los identificadores de los nodos que van a intercambiar la información (S_1, S_2, D_1 y D_2) a través de los flujos $\mathcal{F}_1$ y $\mathcal{F}_2$, así como el número de caminos que desean analizarse (parámetro K). A modo de resumen, el algoritmo recorrerá toda la lista de nodos (a excepción de los finales) encontrando, a través del *Algoritmo de Yen* [Yen72], los K caminos más cortos entre el nodo analizado y cada uno de los extremos S_1, S_2, D_1 y D_2. Una vez obtenidas las rutas, se aplicarán las condiciones expuestas en la Definición 5.2, marcando, en caso afirmativo, al nodo analizado como un potencial elemento codificador. Tras la finalización de todo el proceso, el algoritmo devolverá una lista que contiene todos los nodos que cumplen con las restricciones de codificación ya discutidas, seleccionando aquél que implique un número total de saltos menor.

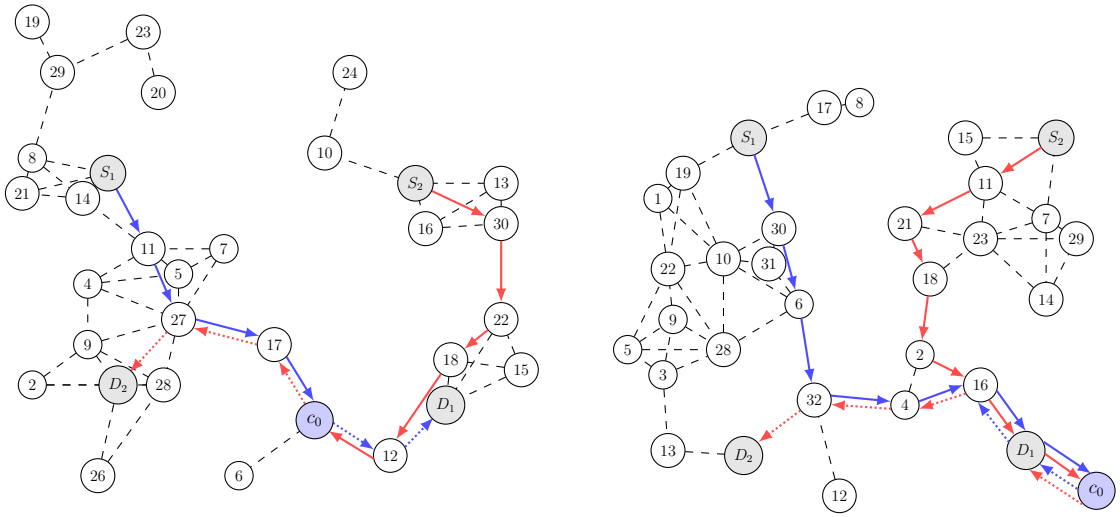
[18] Al igual que en el Capítulo 3, la capacidad de un nodo en un despliegue aleatorio estará limitada a una región circular definida por su radio de cobertura.

Algoritmo 5.1: Búsqueda de los nodos codificadores en una red mallada (versión inicial)

Entrada: $\mathcal{G}(V,E), S_1, S_2, D_1, D_2, K$
Variables auxiliares: $\mathcal{G}'(V,E')$, $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{C}$
Inicialización: $\mathcal{G}'(V,E') \leftarrow \mathcal{G}(V,E)$, $\mathcal{G}_1 \leftarrow \emptyset$, $\mathcal{G}_2 \leftarrow \emptyset$, $\mathcal{C} \leftarrow \emptyset$
for *cada nodo* $i \in \mathcal{G}(V,E) \mid i \neq S_1, S_2, D_1$ *y* D_2 **do**
  for $j = 1 : K$ **do**
    A. Dijkstra: $\mathcal{U}_j(i,F_1)$, $\mathcal{D}_j(i,F_1)$, $\mathcal{U}_j(i,F_2)$, $\mathcal{D}_j(i,F_2)$;
    for *cada nodo* $n \in \mathcal{U}_j(i,F_1)$ **do**
      $\mathcal{G}_1 \leftarrow \mathcal{G}_1 \bigcup n \bigcup \mathcal{N}(n)$;
    for *cada node,* $m \in \mathcal{U}_j(i,F_2)$ **do**
      $\mathcal{G}_2 \leftarrow \mathcal{G}_2 \bigcup m \bigcup \mathcal{N}(m)$;
    if $(D_1 \in \mathcal{G}_2)$ & $(D_2 \in \mathcal{G}_1)$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \bigcup i$;
    A. Yen: $\mathcal{G}'(V,E') \leftarrow \mathcal{G}'(V,E' - \{e\})$;
  $\mathcal{G}_1 \leftarrow \emptyset, \mathcal{G}_2 \leftarrow \emptyset$;
Salida: Lista de posibles nodos codificadores ($\mathcal{C}$)

Sin embargo, los resultados ofrecidos por esta versión del algoritmo no pueden ser considerados como definitivos, ya que se pueden producir "falsos positivos", en los que, aunque el algoritmo haya determinado la existencia de al menos un nodo codificador en el escenario, las rutas generadas para adaptarse a un esquema de codificación *inter-flujo* pueden implicar un gran número de saltos adicionales. Para ilustrar este problema, la Figura 5.17 muestra dos claros ejemplos que reflejan las limitaciones mostradas por el Algoritmo 5.1. En primer lugar, la Figura 5.17a representa un escenario en el que la elección del *nodo codificador* parece la adecuada, viéndose como el nodo c_0 se encuentra en un punto de convergencia entre los dos flujos (*Condición 1* en la Definición 5.2) y los nodos finales serán capaces de escuchar las transmisiones pertenecientes a alguno de los nodos que conforman el flujo de subida *nativo* inverso, $\mathcal{U}(c_0, \mathcal{F}_j)$, con $i,k = [1,2]$ y $i \neq j$, cumpliendo con la *Condición 2*. En este caso, es fácil deducir que las rutas utilizadas por los mecanismos *NC* serán las mismas que las que usaría una transmisión *store-and-forward* clásica.

Por otro lado, puede darse el caso de que la elección de un *nodo codificador* perjudique al rendimiento global del sistema, como ilustra la Figura 5.17b. En este escenario en concreto, el Algoritmo 5.1 identifica al nodo c_0 como elemento codificador en la red, aunque se necesitarían rutas sensiblemente más largas. Mientras que con una transmisión estándar *store-and-forward* el nodo S_1 necesitaría 6 saltos para alcanzar a D_1 y S_2 de otros 7 para poder comunicarse con D_2, la utilización de las rutas que ofrece el algoritmo implicaría un total de $8 + 12$ saltos, esto es, 7 saltos adicionales en comparación con una transmisión estándar. Además, se puede apreciar que el algoritmo no es capaz de distinguir y eliminar



(a) Codificación viable

(b) *Falso positivo*

Figura 5.17: Ejemplo de *falso positivo* en el Algoritmo 5.1

la aparición de *ciclos*. Como consecuencia, el esquema *NC* no será capaz de compensar en absoluto la pérdida de rendimiento producida por el incremento de longitud de las rutas.

A raíz de las limitaciones mostradas por esta primera solución, se proponen una serie de modificaciones, recogidas en el Algoritmo 5.2. En concreto, se introducen dos nuevos elementos para determinar la viabilidad del *NC* en un escenario dado.

1. En primer lugar se obtendrán los caminos directos más cortos que conectan los nodos fuente con los finales ($S_1 \rightarrow D_1$ y $S_2 \rightarrow dD2$) a través del *Algoritmo de Dijkstra*, definiendo los parámetros $\mathcal{F}_i^{S\&F}$, con $i = [1, 2]$. Estas rutas serán utilizadas como referencia en el siguiente parámetro.

2. Con el fin de limitar la obtención de soluciones no coherentes, se presenta un nuevo parámetro, δ, cuyo papel será el de evitar la aparición de escenarios en los que la presencia de *nodos codificadores* implique la utilización de rutas más largas.

Algoritmo 5.2: Búsqueda de los nodos codificadores en una red mallada (versión modificada)

Entrada: $\mathcal{G}(V,E), S_1, S_2, D_1, D_2, K, \delta$
Variables auxiliares: $\mathcal{G}'(V,E')$, $\mathcal{F}_1^{S\&F}$, $\mathcal{F}_2^{S\&F}$, $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{C}$
Inic.: $\mathcal{G}(V,E') \leftarrow \mathcal{G}(V,E)$, $\mathcal{F}_1^{S\&F} = \mathcal{F}_2^{S\&F} = \mathcal{G}_1 = \mathcal{G}_2 = \mathcal{C} \leftarrow \emptyset$
A.Dijkstra: $\mathcal{F}_1^{S\&F}, \mathcal{F}_2^{S\&F}$;
for *cada nodo* $i \in \mathcal{G}(V,E) \mid i \neq S_1,\ S_2,\ D_1\ y\ D_2$ **do**
  for $j = 1 : K$ **do**
    A. Dijkstra: $\mathcal{U}_j(i,F_1)$, $\mathcal{D}_j(i,F_1)$, $\mathcal{U}_j(i,F_2)$, $\mathcal{D}_j(i,F_2)$;
    if
    $\left(N_{saltos}\left(\mathcal{F}_1\right) \leq N_{saltos}\left(\mathcal{F}_1^{S\&F}\right) + \delta\right)$ & $\left(N_{saltos}\left(\mathcal{F}_2\right) \leq N_{saltos}\left(\mathcal{F}_2^{S\&F}\right) + \delta\right)$
    then
      for *cada nodo* $n \in \mathcal{U}_j(i,F_1)$ **do**
        $\mathcal{G}_1 \leftarrow \mathcal{G}_1 \bigcup n \bigcup \mathcal{N}(n)$;
      for *cada node,* $m \in \mathcal{U}_j(i,F_2)$ **do**
        $\mathcal{G}_2 \leftarrow \mathcal{G}_2 \bigcup m \bigcup \mathcal{N}(m)$;
      if $(D_1 \in \mathcal{G}_2)$ & $(D_2 \in \mathcal{G}_1)$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \bigcup i$;
      A. Yen: $\mathcal{G}'(V,E') \leftarrow \mathcal{G}'(V,E' - \{e\})$;
  $\mathcal{G}_1 = \mathcal{G}_2 \leftarrow \emptyset$;
Salida: Lista de posibles nodos codificadores ($\mathcal{C}$)

En esencia, el comportamiento del algoritmo será básicamente el mismo que en la primera versión, con la excepción de que en este caso se descartarán todos los *nodos codificadores* que impliquen incrementar la longitud de las rutas en un número mayor de δ saltos. Se filtrarán así aquellos escenarios en los que, a pesar de cumplir con las Condiciones expuestas en la Definición 5.2, la utilización de los mecanismos de *NC* no serán capaces de compensar las pérdidas producidas por los saltos adicionales. La consecuencia más directa será la disminución significativa del porcentaje de escenarios en los que la codificación *inter-flujo* sea una solución viable.

Para estimar la complejidad inherente a la ejecución del Algoritmo 5.2, el peso principal de la operación recaerá en el *Algoritmo de Dijktra*, cuya operación conllevará la mayor parte del tiempo (hasta el punto de que el impacto del resto de componentes se puede considerar despreciable). Teniendo en cuenta que se ha utilizado una versión optimizada del mismo, su coste, sobre un grafo $G(V,E)$, será $\mathcal{O}_{Dijkstra} = \mathcal{O}(E + V \log V)$. Además, se puede afirmar [Hers07] que, en el peor de los casos, el *Algoritmo de Yen* realiza un total de $K \cdot N$ llamadas a *Dijkstra* para el cómputo de todas las rutas (donde el factor N se puede aproximar, en redes densas, por $N \approx \log V$). Finalmente, el proceso se repite 4 veces por

nodo diferente a S_1, S_2, D_1 y D_2 (cada nodo intermedio buscará las K rutas más cortas hasta los cuatro "extremos" de los flujos). Con todo, la complejidad del Algoritmo 5.2, en el caso más pesimista, se resume en la Ecuación (5.5).

$$\mathcal{O}_{Algoritmo} = 4 \cdot K \cdot N \cdot (N-4) \cdot \mathcal{O}_{Dijkstra} \tag{5.5}$$

5.5 Simulación y resultados

Una vez descrito el protocolo *NC inter-flujo* desarrollado en el marco de esta Tesis, en esta sección se presentan los principales resultados obtenidos a través de extensas campañas de simulación, comparándolos con el rendimiento que se obtendría con la combinación clásica del protocolo *TCP* con unos mecanismos de encaminamiento tradicionales. El análisis se dividirá a su vez en cuatro etapas diferentes:

1. La primera de ellas se centra en el estudio (a través de `ns-3`) del comportamiento del protocolo *NC* sobre escenarios sintéticos con condiciones ideales, donde la presencia de errores en el canal se puede considerar despreciable. En primer lugar, se cuantificará la ganancia del rendimiento sobre una transmisión *TCP* estándar. Asimismo, se evaluará la influencia de la configuración de los parámetros de operación en los *nodos codificadores*.

2. Tras el estudio de las técnicas de *NC* sobre enlaces sin errores, el siguiente paso será el de introducir pérdidas en el sistema, comprobando la interacción entre los mecanismos de control de flujo y congestión de *TCP* con el esquema de codificación definido en el protocolo *inter-flujo*.

3. Mientras que en escenarios canónicos la posición de un *nodo codificador* es fácilmente localizable, en despliegues aleatorios ya no resultará tan evidente conocer qué nodo(s) deberá(n) llevar a cabo las tareas de codificación, en el caso de que sea factible. En esta etapa se caracterizará el algoritmo presentado en la Sección 5.4, analizando la viabilidad del uso de técnicas de *NC* sobre diferentes escenarios generados aleatoriamente. Merece la pena destacar que este punto se llevará a cabo en una campaña independiente al simulador.

4. En una última etapa se analizará el rendimiento del esquema de *NC inter-flujo* en aquellas topologías en las que su uso sea viable, utilizando de nuevo la plataforma `ns-3`.

A lo largo de todo el proceso de medidas se utilizará el mismo grupo de estadísticos para caracterizar el comportamiento de las diferentes técnicas. A continuación se definen aquéllos que no han sido introducidos hasta este punto:

- **Tasa de codificación**. Esta primera métrica se define como el cociente entre los paquetes codificados y el número total de mensajes *de datos* transmitidos por los nodos codificadores (los mensajes que no se consideran en el proceso de codificación, como *ACKs*, mensajes cortos, etc., no contabilizarán para esta estadística). Además, este parámetro permite analizar el *nivel de sincronización* entre los flujos (y, por tanto, las ventanas de congestión en la capa de transporte de los nodos transmisores)
- **Tasa de decodificación**. Tan importante como generar paquetes codificados es tener la capacidad para decodificarlos. Este estadístico se limitará a los *nodos decodificadores* (esto es, los destinos), y se define como el cociente entre las decodificaciones correctas y el número total de paquetes codificados recibidos.
- **Número de transmisiones**. La ganancia derivada de este esquema de codificación se basa en aumentar la utilización del canal, transportando una mayor cantidad de información en el mismo espacio que en una alternativa más clásica. Este hecho, unido a las propiedades físicas del medio radio, hará que, en conjunto, el número de transmisiones necesarias sea menor.

Juntos a las métricas presentadas, se recurrirá a otras ya mencionadas en capítulos anteriores, como el *throughput* medido en el nivel de aplicación, la tasa de error de trama (*FER*), el número de retransmisiones *TCP* o diversos parámetros utilizados en el control de flujo de *TCP*, como el *RTT*, el *RTO* o el tamaño de la ventana de congestión.

5.5.1. Escenarios canónicos en condiciones ideales

Antes de proceder a la descripción de la plataforma de medidas seguida en esta primera etapa, se describirán los escenarios sintéticos correspondientes a las dos primeras etapas del análisis, en los que se despliega un reducido número de nodos, que conforman unas topologías regulares y definidas, donde la elección de los *nodos codificadores* resulta trivial. En estos escenarios canónicos se cuantificarán los beneficios asociados al empleo del *NC*.

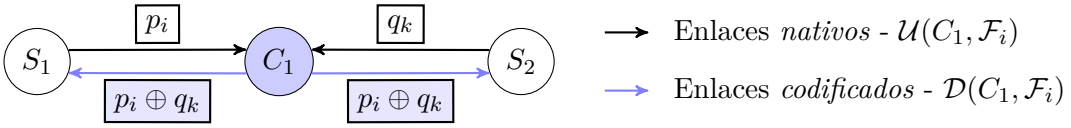


Figura 5.18: Topología en *línea* empleada para estudiar la sincronización de los flujos *TCP* a partir de la distribución de los errores aleatorios

Al primero de los escenarios, presentado en la Figura 5.18, se le denominará *topología en línea* o *topología de 3 nodos* (visto previamente en el Capítulo 4), y cuenta con dos flujos en sentido contrario, actuando los nodos S_1 y S_2 tanto de transmisores como de receptores. Como es lógico, el nodo intermedio, C_1, ejercerá la tarea de combinar los mensajes (de

datos) de ambas conexiones. Como ya se ha discutido anteriormente, esta topología no permite explotar la diversidad espacial del medio inalámbrico, por lo que los nodos deberán utilizar su propia información para recuperar el contenido de los paquetes codificados. En consecuencia, se puede asegurar que *la tasa de decodificación será siempre igual a 1*, ya que los segmentos *nativos* se encontrarán con total certeza en el momento de la decodificación (se almacenan antes de transmitirse). A partir de la definición de *ganancia de codificación* [Katt08], donde se define como el cociente entre el número de transmisiones necesarias en un esquema *store-and-forward* tradicional y uno *NC*, con esta topología se tendría un factor $\mathcal{G} = \frac{4}{3} = 1.333$, aunque el valor que se obtendrá finalmente será inferior, ya que el tráfico de reconocimientos *TCP* no se incluye en el esquema de codificación escogido.

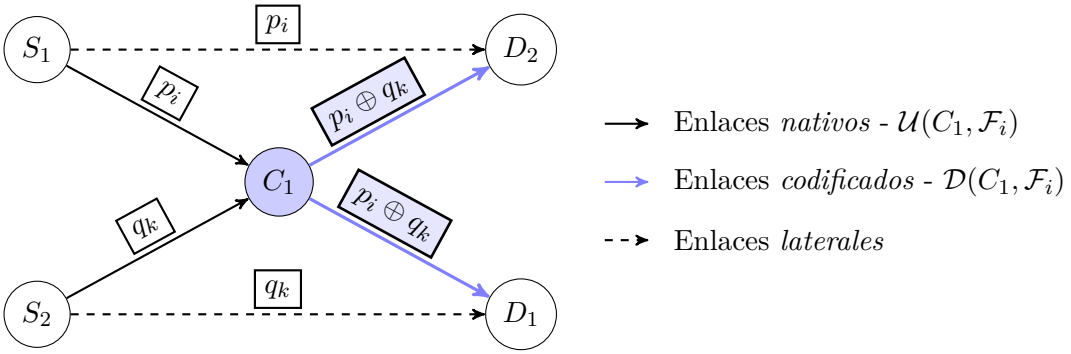


Figura 5.19: Topología en *X* empleada para estudiar la sincronización de los flujos *TCP* a partir de la distribución de los errores aleatorios

En la Figura 5.19 se presenta la *topología en X*, descrita también con anterioridad (Figura 5.2). En este caso, los nodos S_1 y S_2 establecen sendas conexiones *TCP* con D_1 y D_2, respectivamente, teniendo que apoyarse en un nodo intermedio para alcanzar a sus destinos. Como puede verse en la figura, ambos flujos se cruzan en C_1, que hará las veces de *nodo codificador*. El principal matiz que difiere a este escenario con el visto previamente se encuentra en la capacidad de los nodos D_2 y D_1 para escuchar los segmentos *nativos* de los transmisores cruzados, S_1 y S_2, respectivamente. Estos enlaces, que en un esquema clásico `store-and-forward` carecerían de sentido, tendrán un papel esencial en el rendimiento del sistema, ya que cualquier pérdida producida en ellos podría derivar en un *error de decodificación* posterior, impidiendo la recuperación de un segmento *nativo*. Además, deberá tenerse en cuenta que la recepción de segmentos a través de estos enlaces no podrá beneficiarse de los mecanismos de retransmisión del estándar *IEEE 802.11*, ya que las direcciones físicas de la trama y del dispositivo no coinciden. Por otra parte, se aprecia como el flujo descendente de paquetes codificados, $\mathcal{D}(C_1, \mathcal{F}_i)$, se divide en dos, a través de los enlaces $C_1 \rightarrow D_2$ y $C_1 \rightarrow D_1$, y sólo uno de ellos podrá utilizar el *feedback* del protocolo *MAC* como elemento de reacción ante posibles pérdidas eventuales. Finalmente, al igual que para la *topología de 3 nodos*, la *ganancia de codificación* será $\mathcal{G} = 1.333$.

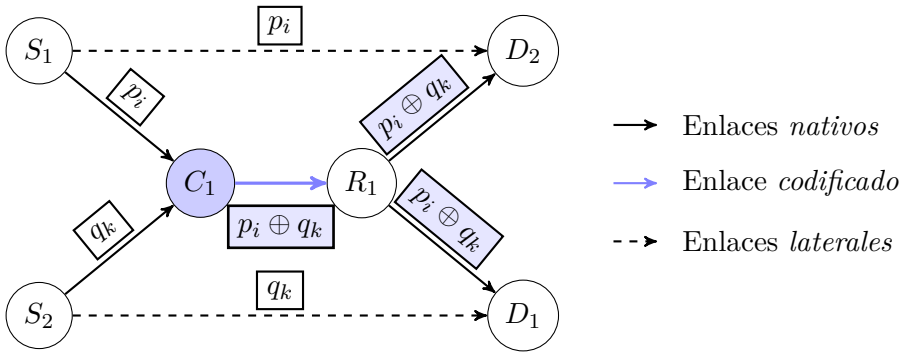


Figura 5.20: Topología *mariposa* empleada para estudiar la sincronización de los flujos *TCP* a partir de la distribución de los errores aleatorios.

Por último, se ha incluido una tercera opción, conocida como *mariposa o butterfly*, cuya topología se ilustra en la Figura 5.20. Para este último escenario, basado en el anterior, se añade un nodo intermedio adicional, R_1, por lo que los flujos cruzados necesitarán 3 saltos para alcanzar a su destino. Al igual que para la X, S_1 y S_2 iniciarán una conexión *TCP* cruzada con sus entidades pares, D_1 y D_2, apoyándose en los nodos intermedios C_1 y R_1. Teniendo en cuenta la posición de los nodos, puede observarse que existen dos posibles *nodos codificadores*; considerando que la *la ganancia será mayor cuanto mayor sea la región recorrida por paquetes codificados*, C_1 será el elegido por encontrarse más cercano a los nodos transmisores[19]. Como elemento distintivo, este escenario se caracteriza por la presencia de un enlace central entre C_1 y R_1, protegido por el esquema de retransmisión *IEEE 802.11*, y por el que pasará todo el tráfico codificado. Esto tendrá un especial interés, puesto que los errores aleatorios en este enlace en concreto provocarán la pérdida de información simultánea en los dos flujos, lo que dará lugar, como se verá más adelante, a un efecto de "sincronización" entre los mecanismos de control de congestión de ambas conexiones *TCP*. Como puede deducirse fácilmente, un esquema tradicional (TCP + encaminamiento *store-and-forward*) necesitaría de 6 transmisiones para que ambos flujos hagan llegar un segmento de datos a cada uno de sus destinos; sin embargo, la utilización del protocolo *inter-flujo* necesitaría un total de 4 envíos (los segmentos viajarían codificados a través del enlace central $C_1 \rightarrow R_1$ y en un único envío simultáneo por $C_1 \rightarrow D_1$ y $C_1 \rightarrow D_2$), ahorrando hasta un tercio de las transmisiones necesarias. En concreto, la ganancia de codificación resultante sería $\mathcal{G} = 1.5$.

Para llevar a cabo la caracterización del rendimiento de ambas soluciones, se procederá al envío de 10000 segmentos *TCP* de longitud máxima en cada una de las conexiones; es decir, para la solución basada en *TCP* (*NewReno* [Hend12]), la aplicación enviará a los niveles inferiores 1460 *bytes* por paquete, mientras que para la combinación de *TCP* con *NC* se reducirá el *payload*, para poder introducir la cabecera de este último sin incurrir en la fragmentación en los niveles inferiores, resultando en un total de 1440 octetos (siendo 20

[19]Se presentarán los resultados correspondientes al supuesto caso de que R_1 fuese el *nodo codificador*, demostrando la importancia de realizar una elección óptima.

bytes el tamaño de la cabecera para transportar dos segmentos codificados). Para asegurar que el canal se encuentre saturado, la tasa de envío desde el nivel de aplicación en los nodos será siempre superior a la capacidad del nivel físico. Al igual que en todos los procesos de simulación seguidos hasta este punto, se ha seguido un método *Monte Carlo*, con un total de 50 simulaciones por cada uno de los puntos de configuración, representando siempre que sea posible el intervalo de confianza del 95 %. Por último, se ha fijado un tiempo máximo (de simulación) de 1000 segundos.

En relación a los parámetros que van a ser modificados a lo largo de la campaña, se encuentran principalmente agrupados en tres categorías:

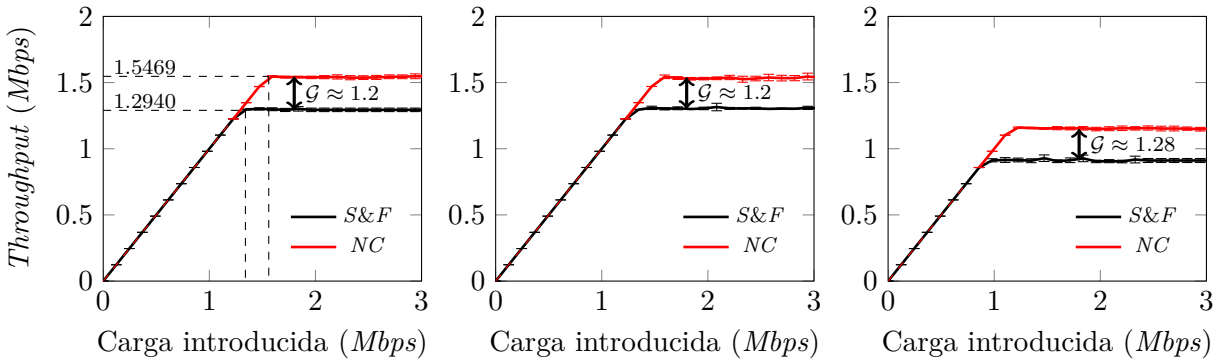
1. **Configuración del *buffer de codificación***. Se irá variando el tamaño(CB_S) y el tiempo de permanencia de los segmentos *nativos* (CB_{TO}), para averiguar empíricamente la configuración óptima en función de las condiciones del tráfico.
2. **Configuración del *buffer de reconocimientos TCP***. También será importante elegir los valores adecuados del tamaño (B_S^{ACK}) y tiempo de permanencia (B_{TO}^{ACK}), puesto que una retención excesiva de los *ACKs* de *TCP* puede dañar notablemente el rendimiento final del sistema.
3. **Tasa de error de trama (*FER*)**. Se alterará la calidad de la señal en cada uno de los enlaces que componen la red de manera independiente. Como se detallará en profundidad en la Sección 5.5.2, se han utilizado diferentes patrones de errores para analizar el efecto de la pérdida de información en diferentes partes de la conexión.

Finalmente, se comparará el comportamiento de las siguientes soluciones:

- **Transmisión *TCP* estándar**. Se tomará como referencia un esquema tradicional.
- **Esquema *NC inter-flujo***. La primera de las soluciones a estudiar se basará en la utilización del protocolo *inter-flujo* en su versión más básica.
- **Encapsulación de reconocimientos *TCP* (aislados)**. Antes de evaluar el comportamiento de esta técnica junto con los mecanismos de *NC inter-flujo*, se caracterizará el rendimiento aislado de la combinación (sin codificar) de los flujos de datos y los *ACKs* de *TCP*.
- **Esquema *NC inter-flujo* + Encapsulación de reconocimientos *TCP***. Finalmente, se procederá al análisis de la operación conjunta de ambas aproximaciones.

La Figura 5.21 muestra el primero de los resultados, en concreto, la ganancia obtenida en el proceso de codificación (versión inicial del protocolo), representando el *throughput* promedio de los dos flujos de datos en función de la tasa de inyección de las aplicaciones a

los niveles inferiores. El procedimiento a seguir ha consistido en ir incrementando el valor de la tasa binaria en las aplicaciones en los nodos fuente, monitorizando el caudal en el otro extremo de la red. Se muestra también el intervalo de confianza del 95 % en cada punto de análisis, exhibiendo un bajo nivel de variabilidad. Además, este estudio permite obtener de manera empírica el punto de saturación de la red, en el que el valor del *throughput* alcanza su máximo. Efectuando una comparación entre el comportamiento de *TCP* con el de *NC*, se aprecia claramente como la segunda solución soporta una carga de tráfico superior, gracias a la disminución de la contención en los nodos intermedios, asociada a la combinación de los segmentos de datos pertenecientes a los diferentes flujos. Además, la *tasa de codificación* para aquellas tasas que no saturan la red será siempre igual a 1.



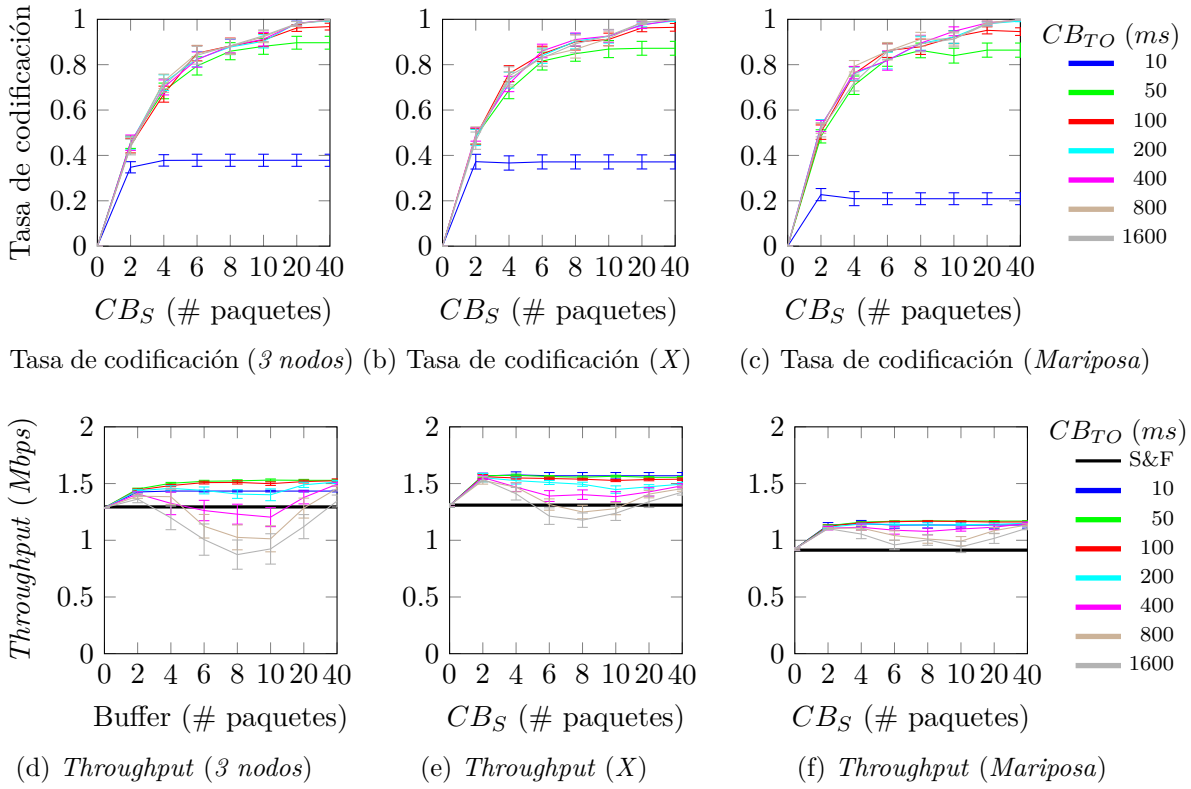
(a) Topología en línea (b) Topología *X* (c) Topología *mariposa*

Figura 5.21: Comparativa de la capacidad de la red en los diferentes escenarios sintéticos

Como puede desprenderse de las curvas mostradas, el comportamiento (en condiciones ideales) sobre las *topologías de tres nodos* y *X* son prácticamente idénticos, aunque hay un matiz que merece la pena comentar: mientras que la *tasa de decodificación* en la *línea de tres nodos* será siempre perfecta, existe la posibilidad de que se produzcan errores en la *X*, asociados a las colisiones en los enlaces laterales, que impedirán la recepción de los segmentos *nativos*. Finalmente, el rendimiento observado en la *mariposa* parte de unos valores significativamente inferiores, explicados por el salto adicional necesario para alcanzar a los destinos.

En relación a las discrepancias observadas entre los valores de ganancia de codificación teórica (i.e. =1.3333, 1.3333 y 1.5 para las *topologías en línea*, *X* y *mariposa*, respectivamente), y las obtenidas a partir de los resultados experimentales, éstas se explican por el hecho de que el tráfico de reconocimientos que recorre los flujos en sentido inverso *no forma parte del esquema de codificación*, por lo que contiende por el acceso al canal, atravesando la red de manera *nativa*. Deberá tenerse en cuenta de nuevo que no todos los segmentos recibidos correctamente en el nodo destino son confirmados de inmediato, sino que se genera un reconocimiento *TCP* tras la recepción correcta de un número de segmentos (normalmente 2). Por lo tanto, los valores que se obtienen pueden considerarse adecuados. Para

estas simulaciones se ha utilizado la configuración del *buffer de codificación* (cuyos valores se detallarán más adelante).



(a) Tasa de codificación (*3 nodos*) (b) Tasa de codificación (X) (c) Tasa de codificación (*Mariposa*)

(d) *Throughput* (*3 nodos*) (e) *Throughput* (X) (f) *Throughput* (*Mariposa*)

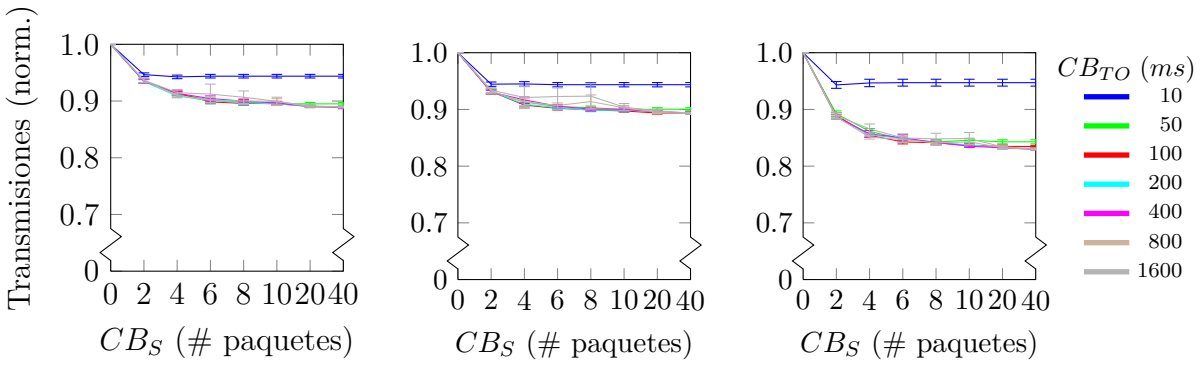
Figura 5.22: Caracterización del protocolo *NC inter-flujo* en condiciones ideales, variando la configuración de los atributos del *buffer de codificación*

La Figura 5.22 muestra los valores de la *tasa de codificación* y *throughput* para cada una de las topologías presentadas, en función de los atributos del *buffer de codificación*, CB_S y CB_{TO}, asumiendo condiciones de saturación. Como podía esperarse a raíz de los cálculos realizados en la Sección 5.1.3, a medida que aumenta el número de paquetes en el CB, se incrementa el porcentaje de *oportunidades de codificación*. Además, un mayor tiempo de permanencia en el nodo afectará de manera positiva, comprobando como se acerca a 1 cuanto mayor es el valor del parámetro CB_{TO}. Sin embargo, puede verse una misma tendencia en todos los casos, donde a partir de un punto se llega a una situación en la que el crecimiento de esta tasa deja de ser apreciable, alcanzando un valor de rendimiento máximo. Puede deducirse que, cuanto más elevada es la *tasa de codificación* en el *nodo codificador*, mayor será el *factor de sincronización* entre los flujos de datos que lo atraviesan, beneficiando la aparición de oportunidades de codificación y, por lo tanto, maximizando la utilización del canal y reduciendo al mismo tiempo el retardo artificial introducido en la capa *NC* de los *nodos codificadores*. Merece la pena destacar el valor obtenido para un $CB_{TO} = 10$ *msegs* en la *topología mariposa* (Figura 5.22c), el cual es aproximadamente *la mitad* que en los otros escenarios, debido al retardo causado por el salto adicional en la

red, lo que reduce significativamente la aparición de *oportunidades de codificación* en un tiempo menor al de este temporizador.

A pesar de que la *tasa de sincronización* presente una tendencia creciente con los parámetros del CB, el rendimiento final no va a experimentar la misma respuesta, tal y como se aprecia en las Figuras 5.22d, 5.22e y 5.22f. Se observa como la utilización de altos tiempos de permanencia puede afectar al rendimiento del sistema, hasta tal punto de resultar inferior incluso al ofrecido por un esquema tradicional *TCP* ($CB_{TO} \geq 400\ msegs$). En el otro extremo, se observa que los valores máximos de *throughput* se obtienen con tiempos reducidos (i.e. $CB_{TO} \leq 100\ msegs$). En valores relativos, la mejora introducida tanto para la *topología de tres nodos* como para la X es de aproximadamente un 20 %, con unos valores de $CB_S \geq 8$ paquetes en ambos casos y un tiempo $CB_{TO} = 50\ msegs$ (*línea*) o 100 $msegs$ (X). En cuanto a la mariposa, el porcentaje es aún mayor, cercano al 28 %, con un tamaño de *buffer* superior a 8 paquetes y un tiempo de entre 50 y 100 $mseg$. En condiciones ideales, las ventanas de congestión de la entidad *TCP* en los nodos transmisores presentarán un crecimiento uniforme y sincronizado, ya que el mecanismo de acceso al medio *DCF* que define el estándar *IEEE 802.11* tiende a repartir equitativamente el ancho de banda entre todos los dispositivos que contienden por el canal. Así, es improbable la aparición de grandes ráfagas procedentes del mismo flujo, que deriven en la transmisión forzada de segmentos nativos (recuérdese que un *error de codificación* es consecuencia, bien de la expiración del CB_{TO} o de la recepción de una ráfaga de $CB_S + 1$ segmentos pertenecientes al mismo flujo).

Si bien el tamaño del *buffer de codificación* no reporta grandes diferencias a partir de un valor superior a 10 paquetes, retener los segmentos más de lo debido conlleva, incluso en condiciones ideales, a una severa degradación del rendimiento.

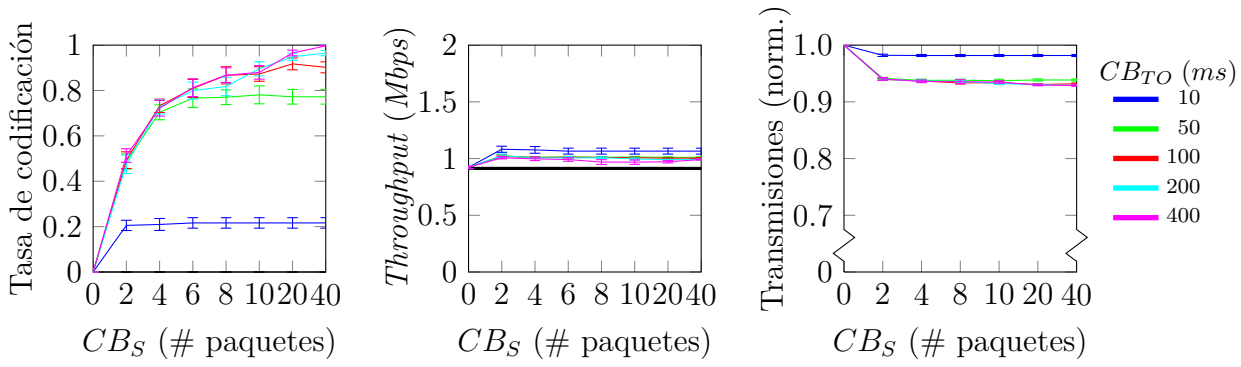


(a) *Línea* (b) X (c) *Mariposa*

Figura 5.23: Número de transmisiones normalizado para cada uno de los escenarios analizados (condiciones ideales)

Por otra parte, la Figura 5.23 representa el número de transmisiones necesarias para completar la transferencia de los 10000 paquetes de cada una de las aplicaciones, normalizando el resultado con respecto al valor más elevado (correspondiente a la combinación de

TCP con un encaminamiento clásico). Nuevamente, los valores obtenidos para el escenario *en línea* y *X* muestran resultados parejos, con un ahorro cercano al 12 % en el mejor de los casos; mientras tanto, en el caso de la *mariposa*, con una mayor *ganancia de codificación*, la reducción de transmisiones alcanza prácticamente un 18 %. Se constata con esto que el incremento del temporizador en el *CB* no resulta perjudicial, poniendo de manifiesto la relación con la tasa de codificación.



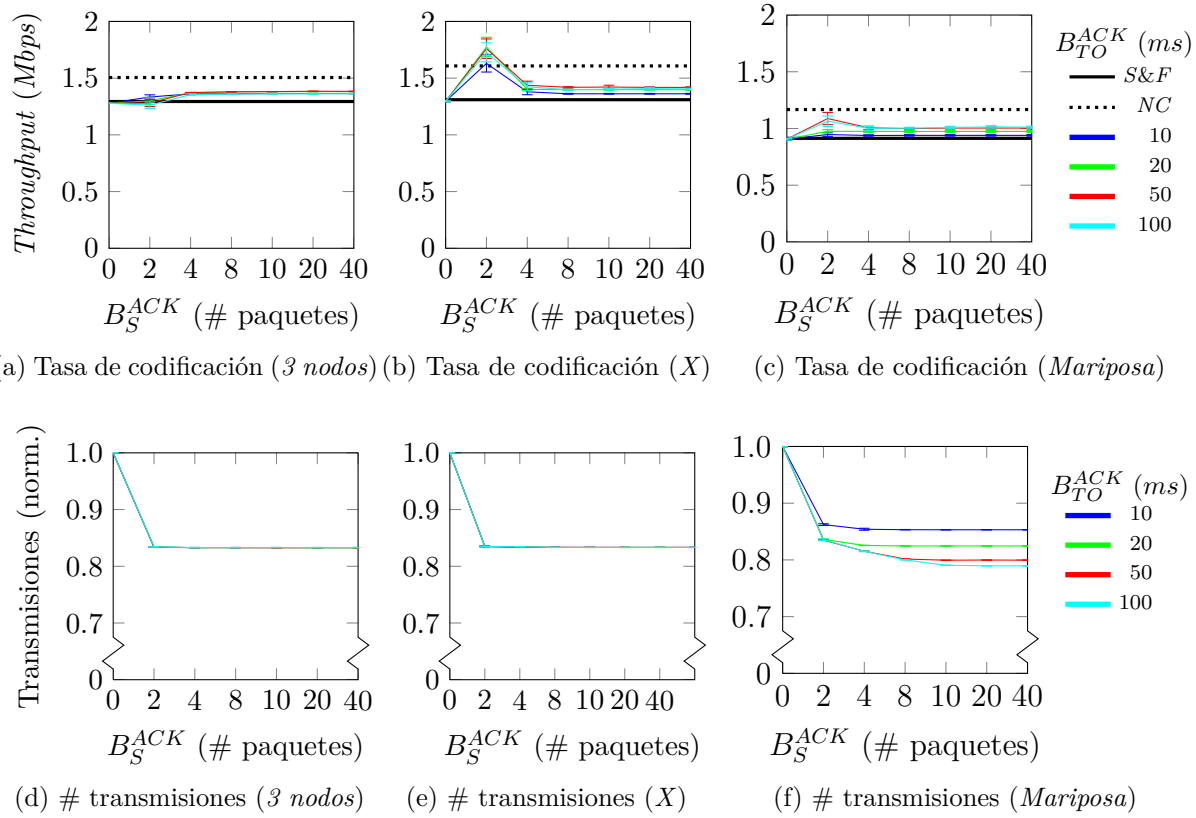
(a) Tasa de codificación (b) *Throughput* (c) Número de transmisiones

Figura 5.24: Principales métricas de rendimiento del protocolo *inter-flujo* con la topología *mariposa* en condiciones ideales (configuración subóptima)

Se analiza a continuación el impacto de una elección subóptima del *nodo codificador*, tomando como referencia la topología de *mariposa* (Figura 5.20). Si en lugar del nodo C_1, fuese R_1 el encargado de realizar las tareas de codificación, la ganancia del proceso se vería reducida a $\mathcal{G} = 1.2$ (en lugar de 1.5), ya que los segmentos *nativos* deberían atravesar individualmente el enlace central $C_1 \rightarrow R_1$ antes de alcanzar al *nodo codificador*. En la Figura 5.24 se muestran los resultados de las simulaciones realizadas y, aunque la *tasa de codificación* (Figura 5.24a) mantenga un comportamiento similar, tanto el *throughput* (Figura 5.24b) como el número de transmisiones necesarias (Figura 5.24c) son notablemente peores, quedándose en unos lejanos 17 y 8 %, respectivamente. Como se estudiará más adelante, la elección de los nodos codificadores puede verse como un problema de optimización en redes malladas inalámbricas, en las que la utilización de este tipo de técnicas pudiera no ser viable bajo determinadas condiciones.

Tras la caracterización individual de las mejoras introducidas por las técnicas de *NC inter-flujo*, la Figura 5.25 ilustra los resultados obtenidos en el caso de que solamente se encuentre activo el esquema de encapsulación de los reconocimientos *TCP*. Como dato a tener en cuenta, se ha tenido que reducir la longitud del *payload* generado en el nivel de aplicación para poder dar cabida a los reconocimientos. En concreto, se utilizarán paquetes de 1390 *bytes*, permitiendo el encapsulado de hasta 3 ACKs en un segmento de datos.

En primer lugar, el fenómeno más llamativo guarda relación a los picos observados en el *throughput* en los escenarios de la *X* y la *mariposa* (Figuras 5.25b y 5.25c) con tamaños



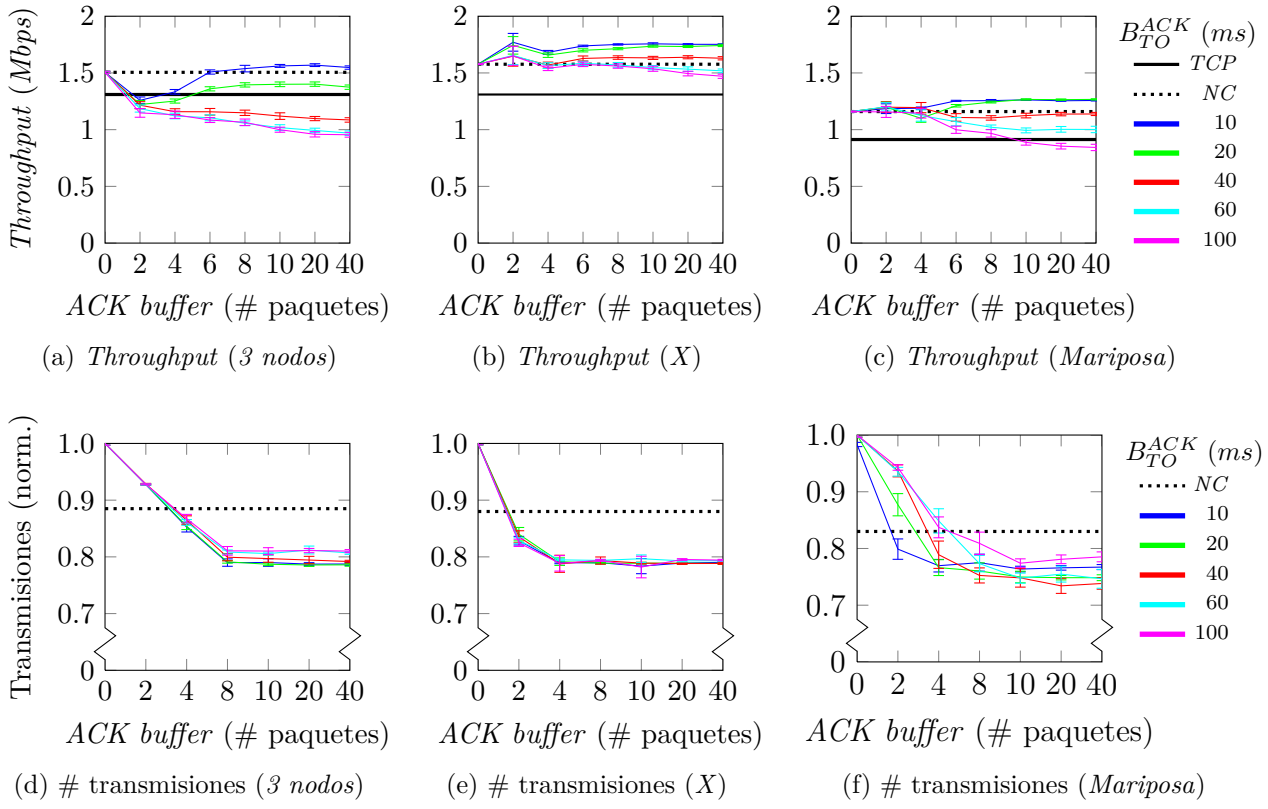
(a) Tasa de codificación (*3 nodos*) (b) Tasa de codificación (X) (c) Tasa de codificación (*Mariposa*)

(d) # transmisiones (*3 nodos*) (e) # transmisiones (X) (f) # transmisiones (*Mariposa*)

Figura 5.25: Caracterización de la solución basada en la encapsulación de los reconocimientos *TCP* en condiciones ideales, variando la configuración de los atributos del *buffer de reconocimientos* B^{ACK}

de *buffer* bajos (i.e. $B_S^{ACK} = 2$ paquetes); estos resultados no pueden ser considerados como válidos, ya que se produce un efecto de captura de uno de los dos flujos, que monopoliza el canal hasta su finalización, momento en el que relanza la segunda de las conexiones; además, puede verse como el intervalo de confianza es mucho mayor en estas zonas, dando muestras de gran variabilidad en los resultados. Tras esta primera fase de inestabilidad, se aprecia claramente como los resultados convergen hacia unos valores fijos, mejorando el rendimiento mostrado por *TCP* en aproximadamente un 7, un 9.5 y un 12 % para cada uno de los escenarios analizados (*línea*, X y *mariposa*, respectivamente), comprobándose que prácticamente la totalidad de los *ACKs* de *TCP* viajan embebidos en los diferentes segmentos de datos que atraviesan la red en sentido contrario. Como puede apreciarse a raíz de los resultados mostrados, la ganancia de esta técnica aislada es inferior a la observada por los mecanismos de *NC inter-flujo*. Lo que sí es capaz de mejorar es el número de transmisiones necesarias, aprovechando la transmisión de los segmentos de datos para incorporar la información correspondiente a los reconocimientos de la capa *TCP*, llegando a superar el 20 % de ahorro en la *topología de mariposa*, cuando el parámetro B_{TO}^{ACK} es elevado, como se observa en la Figura 5.25f.

Finalmente, se analiza el comportamiento al combinar los dos esquemas, tratando de aprovechar los beneficios de ambas aproximaciones. Actuando como procesos paralelos, no se produce interferencia alguna entre ellos: mientras que los mecanismos de *NC* tratan de encontrar, de manera independiente, sus propias *oportunidades de codificación*, el motor de encapsulación de reconocimientos utilizará el espacio disponible en los paquetes salientes para introducir tantos *ACKs* como sea posible. Para poder incorporar, al menos, dos reconocimientos a un paquete codificado con dos segmentos *nativos* (recuérdese que la longitud de la cabecera *NC* correspondiente sería de 20 octetos), se reducirá el tamaño del *payload* del nivel de aplicación a 1392 *bytes* por paquete, para evitar así la fragmentación en las capas inferiores.



(a) *Throughput* (*3 nodos*) (b) *Throughput* (*X*) (c) *Throughput* (*Mariposa*)

(d) # transmisiones (*3 nodos*) (e) # transmisiones (*X*) (f) # transmisiones (*Mariposa*)

Figura 5.26: Caracterización de la operación conjunta del protocolo *NC inter-flujo* y la encapsulación de reconocimientos *TCP* en condiciones ideales, variando la configuración de los atributos del *buffer de reconocimientos* B^{ACK}, fijando los atributos del *buffer de codificación a valores de* $CB_S = 10$ *paquetes y* $CB_{TO} = 50$ *mseg*

La Figura 5.26 muestra los principales resultados obtenidos. El primer aspecto a destacar se encuentra en el *throughput* obtenido en el escenario *en línea* (Figura 5.26a), donde se aprecia que el comportamiento resultante es, exceptuando el caso en el que $B_S^{ACK} \geq 6$ *segmentos* y $B_{TO}^{ACK} = 10$ *msegs*, inferior al uso aislado del esquema de codificación *inter-flujo* (e incluso al propio *TCP* cuando $B_{TO}^{ACK} > 40$ *msegs*). Por el con-

trario, y a pesar de guardar un gran parecido con éste, la *topología en X* muestra un comportamiento completamente diferente (Figura 5.26b), mejorando significativamente el rendimiento observado al utilizar únicamente protocolo *inter-flujo*. Los resultados más interesantes se obtienen cuando el tiempo de permanencia de los reconocimientos es bajo ($B_{TO}^{ACK} \leq 40\ msegs$), llegando a incrementarse el *throughput* hasta un 35 % cuando $B_{TO}^{ACK} = 10\ msegs$, lo que supone un 12 % adicional con respecto a la solución *NC* aislada. Hay que tener en cuenta que, mientras en la *X* cada *cola de transmisión* (del nivel de enlace) en los nodos finales se utiliza para tráfico del mismo tipo (datos o *ACKs*), en la *topología en línea* ambos comparten un único *buffer*, lo que penalizará el envío de reconocimientos. Al no producirse errores, la ventana de contención de la capa de transporte irá enviando cada vez más carga a los niveles inferiores, reduciéndose las oportunidades de transmisión de los *ACKs*. Ésta ha sido la principal razón para implementar, en la *versión avanzada* del protocolo, un nuevo esquema que priorice el envío de reconocimientos sobre el tráfico de datos (Figura 5.15). Así, se reducen los tiempos de espera adicionales introducidos por los elementos intermedios, ajenos al control de flujo gestionado por la capa de transporte.

El comportamiento sobre la *mariposa* (Figura 5.26c) se sitúa en una situación intermedia, ya que solamente los tiempos más bajos resultan beneficiosos ($B_{TO}^{ACK} \leq 20\ msegs$), induciendo una mejora cercana al 39 % con respecto a *TCP*. Por último, a la vista del número de paquetes transmitidos (Figuras 5.26d, 5.26e y 5.26f), se aprecia claramente el ahorro adicional que aporta la combinación de ambas técnicas.

De todos los resultados vistos hasta el momento, puede concluirse que aumentar los tiempos de permanencia en los diferentes *buffers* (tanto de codificación como de reconocimientos) no conlleva necesariamente una mejora en el comportamiento de una transmisión, sino que la excesiva retención de los paquetes en los nodos intermedios no hará sino perjudicar al sistema, penalizando el tiempo de retorno de los diferentes flujos, reduciendo el rendimiento global del mismo. En lo que respecta a su tamaño, éste no tendrá un impacto tan relevante, ya que a partir de ciertos valores la mejora en el rendimiento se estabiliza. Será esencial, por lo tanto, encontrar un equilibrio óptimo entre el retardo artificial introducido en los nodos intermedios, asegurando que la aparición de *oportunidades de codificación* y *encapsulación de reconocimientos* de *TCP* aporte beneficios.

5.5.2. Escenarios canónicos con errores

En el estudio anterior se han comprobado los beneficios asociados a los mecanismos de codificación *inter-flujo* cuando las condiciones del canal de comunicación son óptimas. Sin embargo, en un escenario real, el medio inalámbrico podría presentar unas propiedades más hostiles, siendo lo más habitual la aparición de errores aleatorios durante una transmisión. En esta segunda fase de simulaciones se van a modificar las condiciones de los escenarios, de tal manera que los enlaces serán ahora propensos a la pérdida de tramas. Se utilizarán diferentes patrones de errores, comprobando la reacción que provocan en los mecanismos de control de congestión de *TCP*.

Es necesario, por tanto, introducir nuevos estadísticos, que complementan a los presentados al comienzo de esta sección (i.e. *throughput*, tasas de codificación y decodificación, número de transmisiones, etc.), definidos a continuación:

- **Número de retransmisiones**. La pérdida de segmentos podría reducir de manera notable el rendimiento final de una sesión *TCP*, que reducirá su ventana de congestión y, por tanto, la tasa media de envío.
- **Causa de las retransmisiones**. Teniendo en cuenta que la versión de *TCP* utilizada ha sido *NewReno*, pueden darse *tres* situaciones que lleven a la retransmisión de un segmento: recepción de un *triple ACK duplicado*, de un *ACK parcial*, usado para mitigar el efecto de ráfagas de errores, y expiración del *RTO*, que es la que mayor impacto puede tener en el rendimiento.
- **Evolución de las ventanas de congestión**. Se utilizará para estudiar la sincronización entre flujos, permitiendo interpretar con mayor detalle el efecto de las pérdidas sobre las conexiones presentes en la red.

Finalmente, en ciertos puntos del análisis se incluirán los resultados correspondientes a la utilización de los modelos de canal con memoria presentados en el Capítulo 2 (concretamente, el modelo *HMP*), para analizar el impacto de los errores a ráfagas, ya que, como ya se vio, perjudicará en mayor medida el rendimiento del protocolo *TCP*.

Errores en todos los enlaces

En un primer acercamiento se estudiará el caso más realista, donde los errores de propagación pueden aparecer en cualquiera de los enlaces. Con el fin de mantener un criterio uniforme que simplifique la representación de los resultados obtenidos, se utilizará un valor único de *FER*. El impacto de las pérdidas será diferente según se produzcan en:

1. Flujos de subida, $\mathcal{U}(C_1, \mathcal{F}_i)$. Transportan segmentos nativos (salvo en el *escenario de 3 nodos*), por lo que podrán apoyarse en los mecanismos de retransmisión del estándar *IEEE 802.11* para combatir la aparición puntual de errores de trama[20]. Como puede intuirse, la pérdida de un segmento en estos enlaces afectará exclusivamente a un único flujo.

[20]Recuérdese que, al igual que en capítulos anteriores, el número máximo de transmisiones por trama se ha fijado a 4 intentos, adaptándose a los valores utilizados por las tarjetas de red inalámbricas empleadas durante la etapa de caracterización experimental del Capítulo 2. Además, se configura el simulador para que no se produzcan errores en tramas cortas (inferiores a 250 octetos), por lo que tanto los reconocimientos del protocolo *IEEE 802.11* como de *TCP* se recibirán correctamente.

2. Flujos de bajada, $\mathcal{D}(C_1, \mathcal{F}_i)$. En este caso, la utilización del esquema de retransmisiones *IEEE 802.11* depende del tipo de trama: en el caso de corresponder con un segmento *nativo*, el proceso de confirmación será el habitual; por el contrario, en las situaciones en las que contengan varios segmentos codificados, solamente uno de los receptores enviará el correspondiente *ACK* de nivel *MAC*. A diferencia de en los flujos de subida, puede darse el caso de que una pérdida puntual afecte simultáneamente a los dos flujos activos en la red, derivando en una mayor sincronización entre ambos.

3. Los enlaces "promiscuos", producidos en los *enlaces laterales* $S_1 \rightarrow D_2$ y $S_2 \rightarrow D_1$, a través de los cuales los destinos escuchan la información *nativa* procedente directamente de las fuentes. En este caso, toda recepción errónea conlleva a la pérdida definitiva del segmento en cuestión, lo que, en el protocolo *NC*, podrá tener graves consecuencias, ya que impedirá la correcta decodificación de los paquetes codificados. En cualquier caso, los errores sobre estos enlaces afectarán a un único flujo.

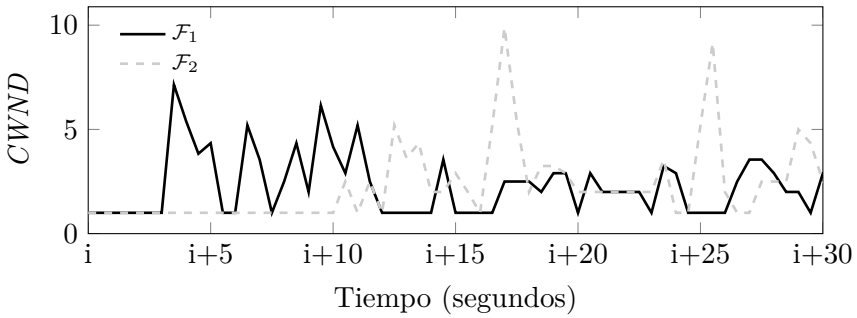
A modo de ejemplo, se muestran en la Figura 5.27 una serie de capturas correspondientes a la evolución temporal[21] del tamaño de las ventanas de congestión de las conexiones *TCP* activas, para ilustrar los efectos producidos por las pérdidas en los diferentes enlaces de la red (utilizando como base la topología de *mariposa*). En el primer caso, la Figura 5.27a muestra el caso más genérico, en el que los errores pueden aparecer en cualquier parte, lo que causa que la evolución del ancho de banda sea aleatoria (existe una dependencia clara entre las ventanas de congestión y el ancho de banda, ya que cuanto mayor sea la ventana de un flujo, mayor será su ocupación del medio inalámbrico); se ve que cualquiera de los flujos puede capturar el canal, dominando sobre el otro. Además, pueden apreciarse errores que afectan simultáneamente a ambos flujos (entre los segundos $i + 20$ e $i + 25$), debidos seguramente a un error en un paquete codificado, lo que causa que la ventana de ambos flujos se reduzca simultáneamente.

En un segundo ejemplo (Figura 5.27b), la presencia de errores se limita al *enlace central* presente entre los nodos C_1 y R_1, por lo que todas las pérdidas de paquetes codificados afectan a ambos destinos, ya que los dos flujos de bajada atraviesan este enlace, que actúa como cuello de botella. Se puede ver como las ventanas de congestión se encuentran prácticamente *sincronizadas*[22] a lo largo de todo el intervalo, lo que permitirá que el ancho de banda se reparta equitativamente. En este caso concreto, las técnicas de *NC* podrán verse beneficiadas de los mecanismos de control de congestión de *TCP*, ya que cuanto más equitativo sea el reparto de recursos, mayor será la probabilidad de encontrar *oportunidades de codificación*.

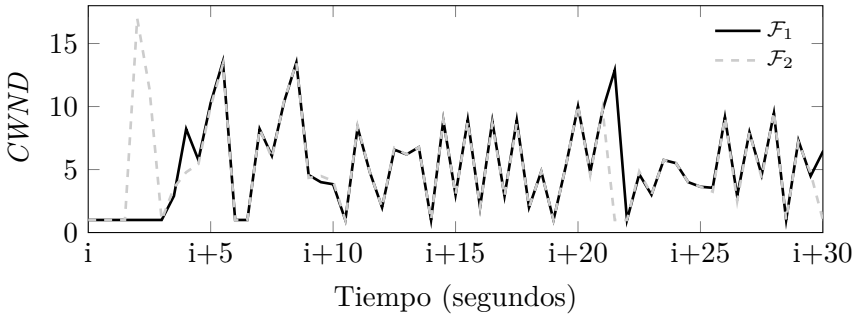
Finalmente, la Figura 5.27c muestra un comportamiento opuesto, observado cuando las pérdidas tienen lugar en los *enlaces laterales* ($S_1 \rightarrow D_2$ y $S_2 \rightarrow D_1$), situación en la

[21] Para una mejor visualización, se representa solamente un intervalo de 30 segundos.

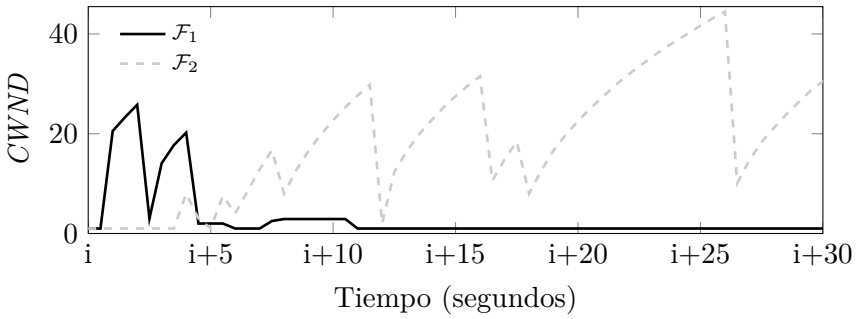
[22] Los errores que dan lugar al desajuste entre las dos ventanas se corresponden con colisiones producidas en los *enlaces laterales*, que darán lugar a un posterior *error de decodificación*.



(a) Errores en todos los enlaces



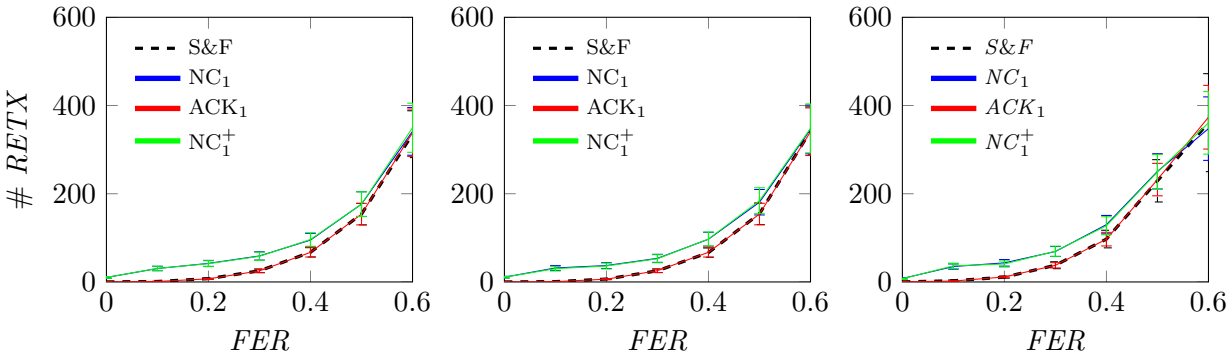
(b) Errores en los flujos de bajada $\mathcal{D}(C_1, \mathcal{F}_i)$



(c) Errores en los enlaces laterales

Figura 5.27: Tamaño de las ventanas de congestión de los flujos *TCP* en función del tiempo de conexión, mostrando los diferentes efectos según dónde se produzcan las pérdidas de información ($FER = 0.3$).

que la correlación de la evolución de las ventanas es nula, demostrando que los eventos de error entre los flujos son completamente independientes entre sí. El efecto más inmediato será la falta de sincronización entre los mismos, pudiendo llegar al extremo en el que uno de los flujos adquiere el canal por completo, anulando al otro, bien hasta que se produzca un número lo suficientemente elevado de errores como para poder recuperarse, o hasta que su transmisión finalice (similar al efecto "*flappiness*" observado en algunos algoritmos de control de congestión del protocolo *MPTCP* en el Capítulo 3).

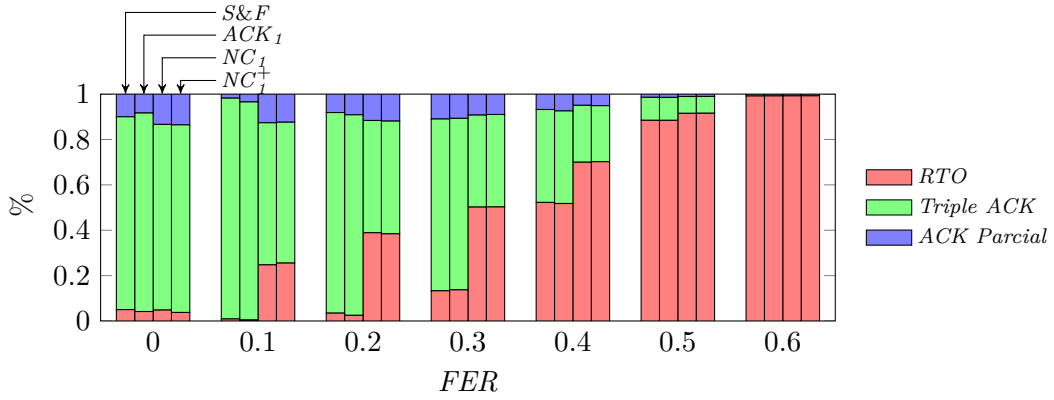


(a) Topología *en línea* (b) Topología *X* (c) Topología *mariposa*

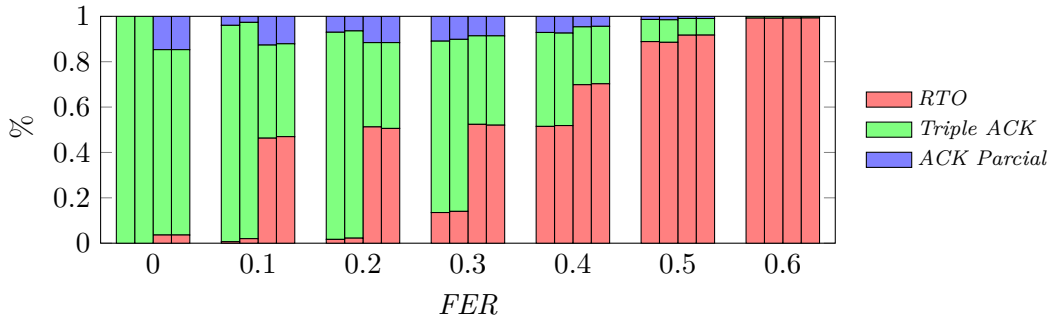
Figura 5.28: Número medio de retransmisiones en los escenarios canónicos con errores en todos los enlaces

Antes de comenzar con el estudio del rendimiento global y de las métricas propias del nivel *NC*, resulta interesante profundizar en la respuesta que el protocolo *TCP* proporciona ante los diferentes patrones de errores aleatorios, pues facilitará el análisis de los resultados posteriores. En primer lugar, la Figura 5.28 compara el número medio de retransmisiones (así como el intervalo de confianza del 95 %) para cada uno de los escenarios y configuraciones estudiados, utilizando los parámetros que muestren un mejor comportamiento. A la vista de los resultados, parece clara la diferencia entre la utilización tradicional de *TCP* o combinándolo con el protocolo *NC inter-flujo*, ya que este último causa la generación de un mayor número de retransmisiones, incluso en condiciones ideales. Los motivos son claros: la correcta decodificación de un paquete codificado en todos los destinos requerirá que, además de la llegada de éstos a los receptores, se haya producido, con carácter previo, la recepción de los segmentos nativos a través de los enlaces *promiscuos*. Por lo tanto, la probabilidad de que se produzca un error será significativamente mayor. Además, la introducción de un retardo artificial en los nodos intermedios, provocará la aparición de retransmisiones derivadas de la expiración del temporizador *RTO*.

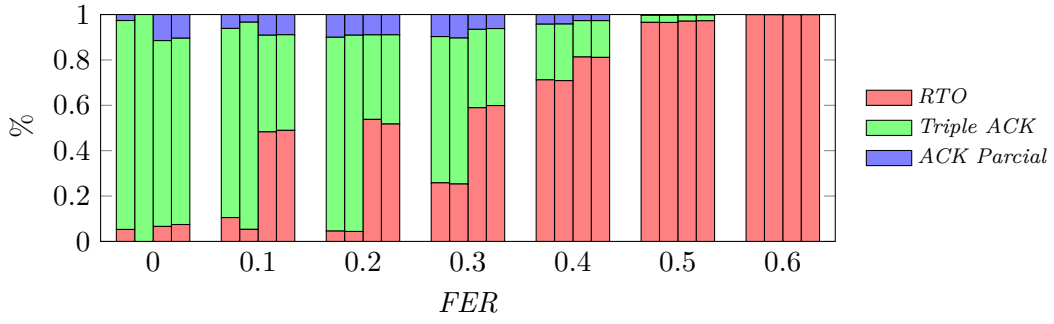
Entrando ahora en un mayor nivel de profundidad, la Figura 5.29 desglosa los eventos que generaron las citadas retransmisiones, representando el porcentaje de las mismas con respecto al número total de reenvíos de un mismo segmento. En lo que respecta al comportamiento aislado del protocolo *TCP* (columna de la izquierda), puede verse como las retransmisiones se ven claramente dominadas por la llegada de *triples ACKs duplicados*



(a) Topología *en línea*



(b) Topología *X*



(c) Topología *mariposa*

Figura 5.29: Causa de las retransmisiones *TCP* para los escenarios canónicos con errores en todos los enlaces

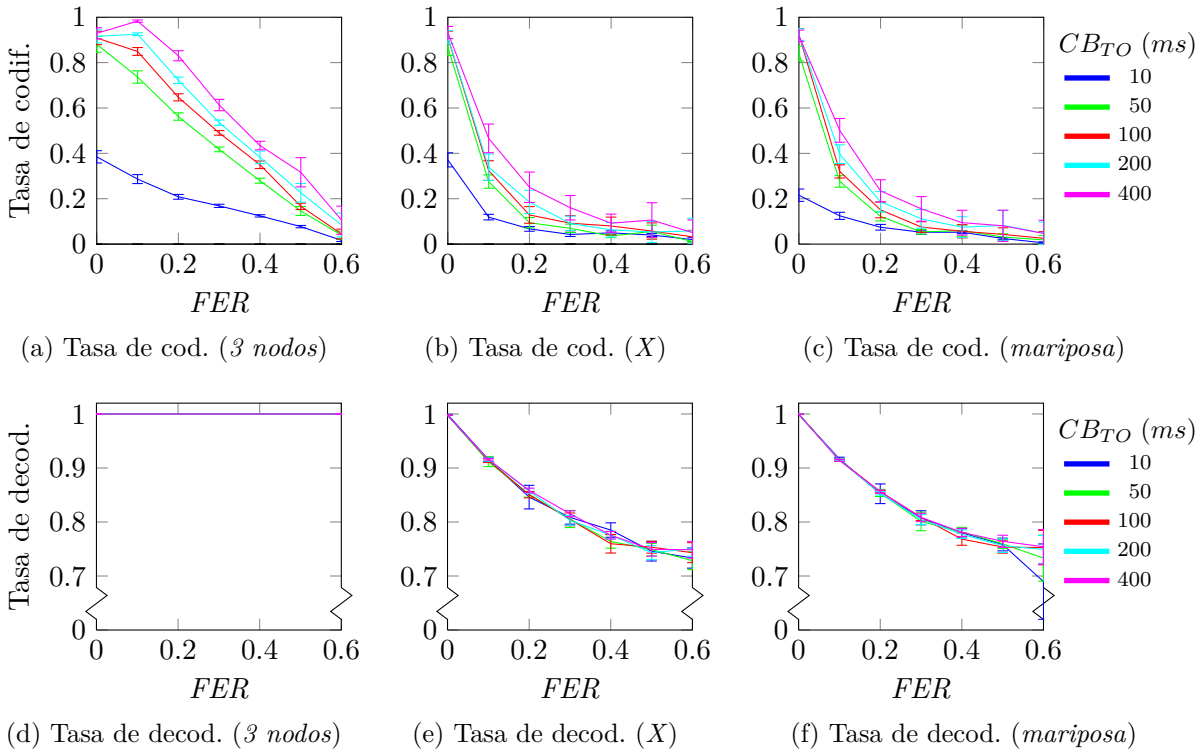
o, en su defecto, de *ACKs parciales*, para valores de *FER* inferiores a 0.4. Con estas condiciones, tanto los mecanismos de retransmisión del estándar *IEEE 802.11* como el algoritmo *Fast Retransmit* de la versión *New Reno* de *TCP* [Hend12] son capaces de sobreponerse a las pérdidas, generalmente aisladas[23], evitando así la aparición de retransmisiones por *RTO*. Sin embargo, para condiciones más hostiles ($FER \geq 0.4$), el porcentaje de retransmisiones por expiración de *RTO* irá aumentando, hasta convertirse en mayoritarias cuando el canal presenta las peores condiciones ($FER = 0.6$). En la siguiente columna se observa el comportamiento mostrado por la solución que únicamente encapsula reconocimientos *TCP*, que guarda un gran parecido con el uso estándar de *TCP*, demostrando que el valor del temporizador en el *buffer de reconocimientos* (10 milisegundos) no es lo suficientemente elevado como para generar retransmisiones por expiración del *RTO*.

En lo que se refiere a la respuesta de los mecanismos de control de congestión de *TCP* cuando interactúan con el protocolo *NC inter-flujo* (tercera y cuarta columnas), además de comprobar que el número medio de retransmisiones es significativamente más elevado puede apreciarse que, en canales con errores, el porcentaje de retransmisiones disparadas por *RTO* es sensiblemente mayor para valores de *FER* entre 0.1 y 0.4. Presumiblemente, la consecuencia más directa del comportamiento de estos mecanismos de control de congestión sea una disminución del rendimiento global de la comunicación.

Una vez vista la respuesta de la capa *TCP* ante la presencia de errores de propagación aleatorios, se presentan a continuación los resultados correspondientes a las métricas de rendimiento empleadas para comparar el comportamiento de las diferentes soluciones. La Figura 5.30 muestra las tasas de codificación y decodificación medidas en el nivel *NC* de los nodos codificadores y decodificadores, respectivamente. A medida que el ratio de errores empieza a crecer, se aprecia un rápido descenso en la sincronización de los flujos, interpretada a través de la primera de las métricas. Al perderse paquetes en cualquiera de los enlaces de la red, serán mayoritarios aquéllos en los que afecta a un único flujo (como en los enlaces laterales). En estos casos, la reacción inmediata de los mecanismos de control de congestión será reducir la tasa de envío (ventana de congestión) de la conexión correspondiente; sería posible, incluso, cesar el envío durante largos intervalos de tiempo (asociado a múltiples expiraciones del temporizador *RTO* consecutivas). Esto hará que el flujo que presente unas mejores condiciones acapare los recursos, produciéndose un "desbalanceo" entre ambas conexiones, con lo que los eventos de llegada a los *nodos codificadores* tenderán a pertenecer a un único flujo, lo que dificultará la aparición de *oportunidades de codificación*. Además, el tiempo medio de espera en estos nodos crecerá ostensiblemente, aumentando el *RTT* y, en consecuencia, reduciendo el rendimiento global.

En este caso, las dos tasas presentan comportamientos similares en las topologías en *X* y en *mariposa*, en las que se aprecia un rápido descenso a medida que el canal deja

[23]Como se estudió en el Capítulo 2, en un modelo de canal sin memoria, la probabilidad de que se produzca una ráfaga de K tramas erróneas será FER^K, siendo K el número máximo de transmisiones por trama del nivel *IEEE 802.11*. Además, se comprobó que para valores de *FER* inferiores a 0.4, la pérdida de datagramas *IP* era prácticamente nula.



(a) Tasa de cod. (*3 nodos*) (b) Tasa de cod. (X) (c) Tasa de cod. (*mariposa*)

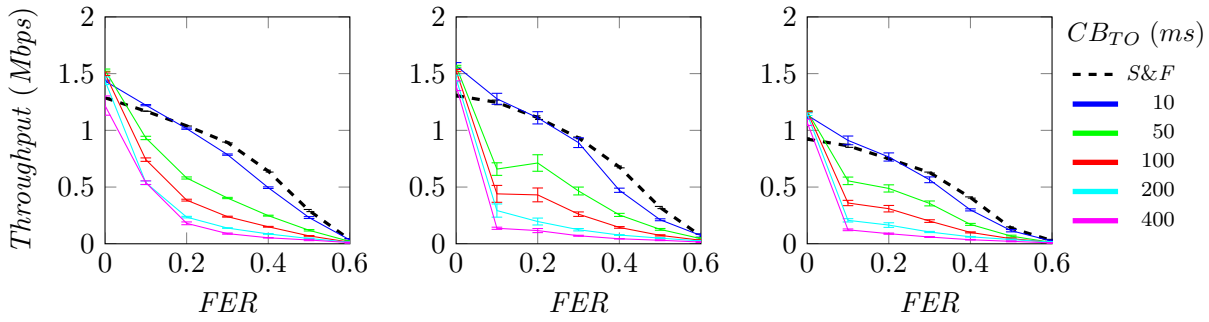
(d) Tasa de decod. (*3 nodos*) (e) Tasa de decod. (X) (f) Tasa de decod. (*mariposa*)

Figura 5.30: Tasas de codificación/decodificación obtenidos en los escenarios canónicos con errores en todos los enlaces (*NC puro*)

de presentar condiciones ideales. Por su parte, en el caso de la topología de *tres nodos*, la pendiente en función de la calidad del canal es bastante más reducida (Figura 5.30a), ya que, en este escenario en concreto, los nodos finales poseen la información nativa ya almacenada de antemano, al ser al mismo tiempo los transmisores del flujo inverso. De este modo, todo paquete codificado será recuperado correctamente, como refleja la Figura 5.30d.

En lo que respecta al rendimiento neto del sistema, la Figura 5.31 muestra que el *throughput* no es proporcional al comportamiento del nivel *NC*, como podría haberse intuido; de hecho, salvo con $CB_{TO} = 10$ *msegs*, la codificación de paquetes en la red perjudicará muy significativamente a la comunicación. Además, puede comprobarse que el hecho de aumentar el tiempo de permanencia en los *nodos codificadores* para incrementar la probabilidad de encontrar *oportunidades de codificación* no hará sino reducir más si cabe el *throughput*. La desincronización de los flujos, unida al incremento del retardo en los nodos intermedios (y, por tanto, el valor del *RTT*), harán que la entidad *TCP* interprete que la red se encuentra más saturada, por lo que generará un mayor número de retransmisiones, manteniendo el valor de la ventana de congestión más bajo. Así, la tasa de envío en las fuentes será más reducida. Como ya se ha mencionado, la única excepción se da para $CB_{TO} = 10$ *msegs*, configuración que no maximiza los beneficios del *NC* en condiciones ideales (ofrece unos márgenes más bien bajos en cuanto a *tasa de codificación* se refiere),

pero que mantiene el rendimiento por encima del observado con el encaminamiento tradicional, hasta valores de tasa de error de trama cercanos al 30 %, donde empieza a decaer abruptamente, debido a que más del 50 % de las retransmisiones son causadas por la expiración del *RTO*. Por último, con una $FER = 0.6$, la cantidad de paquetes perdidos es tan elevada que en un gran porcentaje de los experimentos, las simulaciones (limitadas a 1000 segundos) terminan antes que las transmisiones, por lo que el valor final del *throughput* puede considerarse despreciable.



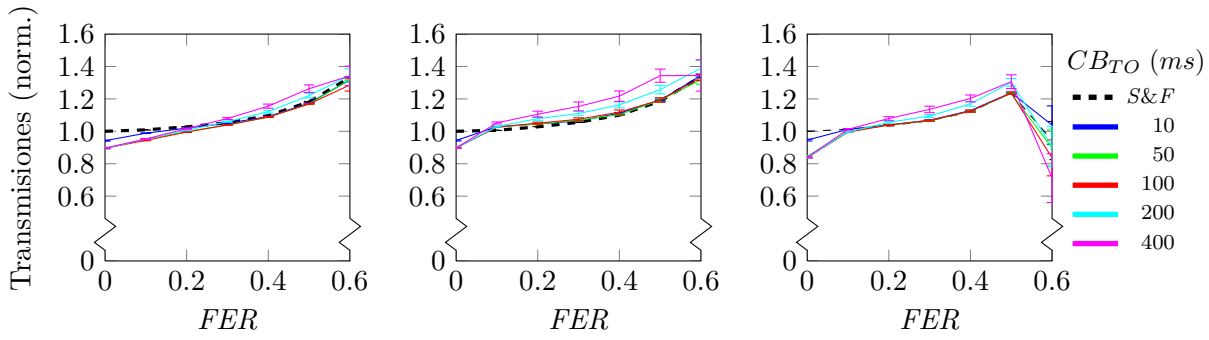
(a) Topología *en línea* (b) Topología *X* (c) Topología *mariposa*

Figura 5.31: *Throughput* obtenido en los escenarios canónicos con errores en todos los enlaces (*NC* puro)

Para terminar de caracterizar el comportamiento del esquema *NC*, la Figura 5.32 representa los paquetes (de nivel de red) transmitidos por todos los nodos, normalizando los resultados con respecto al valor medio al utilizar *TCP* y un encaminamiento *store-and-forward* en condiciones ideales. Como ya se vio en la Sección 5.5.1, en un canal sin errores, cuanto mayor sea la *tasa de codificación*, mayor será la ganancia y menor el número de transmisiones necesarias. Sin embargo, con la aparición de las pérdidas por propagación, este beneficio desaparece y la solución *NC* requiere más transmisiones que las alternativas tradicionales. Se demuestra también que el hecho de alargar la permanencia de los datos en los *nodos codificadores* causa un aumento en el número total de transmisiones. Merece la pena destacar la disminución observada en el escenario de la *mariposa* para valores de $FER = 0.6$ (Figura 5.32c), que es consecuencia de que las transmisiones no finalizan correctamente en el tiempo de simulación, con lo que la interpretación en este punto no es posible.

Se analiza ahora el comportamiento del esquema de integración de los reconocimientos *TCP*, sin que actúe el esquema *NC* en paralelo. Para ello, se reducirá el *payload* generado por paquete en el nivel de aplicación, permitiendo el encapsulado de hasta 3 *ACKs* por segmento de datos.

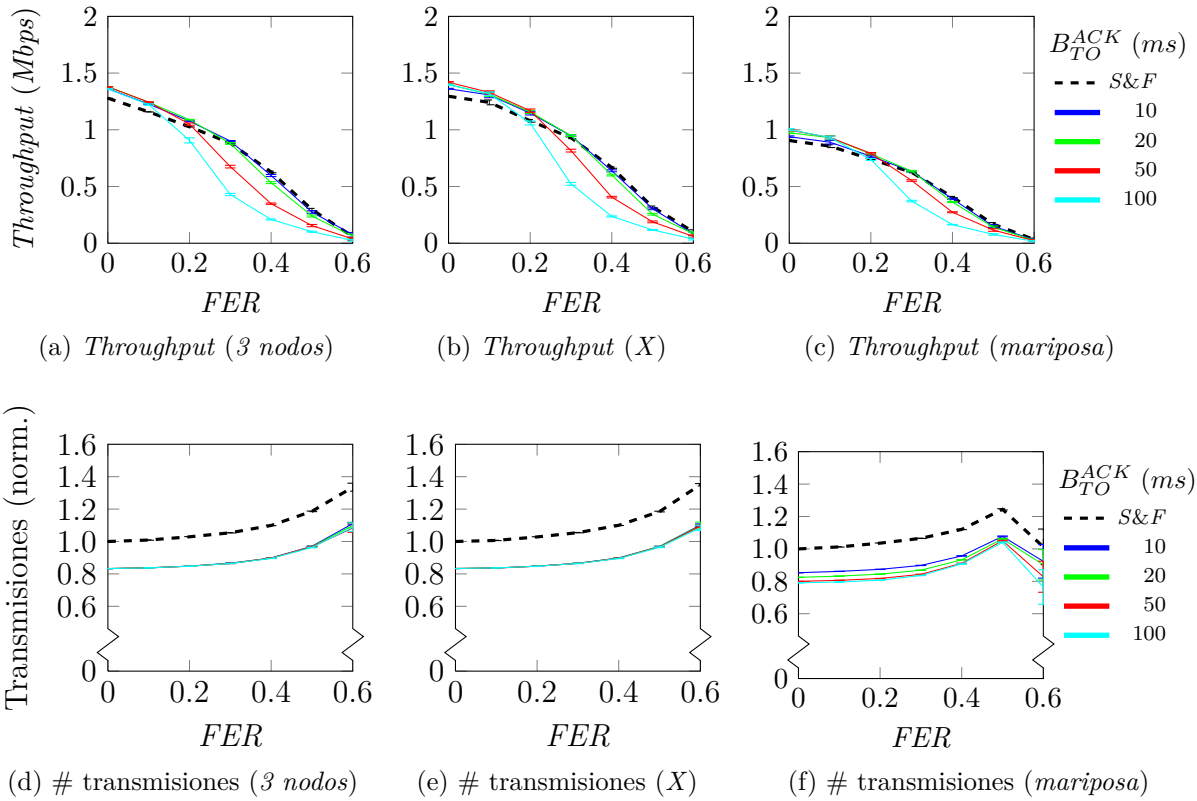
A la vista de los resultados mostrados en la Figura 5.33, donde se representan los valores de *throughput* y el número de transmisiones normalizado, puede afirmarse que el deterioro que se produce, a medida que el porcentaje de errores de propagación crece,



(a) Topología de 3 nodos (b) Topología *X* (c) Topología *mariposa*

Figura 5.32: Número de transmisiones normalizado, obtenido en los escenarios canónicos con errores en todos los enlaces (*NC* puro)

es menos apreciable que el observado para la solución *NC* "pura". Al igual que en el caso anterior, una excesiva retención de los datos en los nodos intermedios traerá unas connotaciones negativas en el rendimiento, que ve como a medida que el valor del parámetro B_{TO}^{ACK} sube, mayor es la disminución del *throughput* (Figuras 5.33a, 5.33b y 5.33c).



(a) *Throughput* (*3 nodos*) (b) *Throughput* (*X*) (c) *Throughput* (*mariposa*)

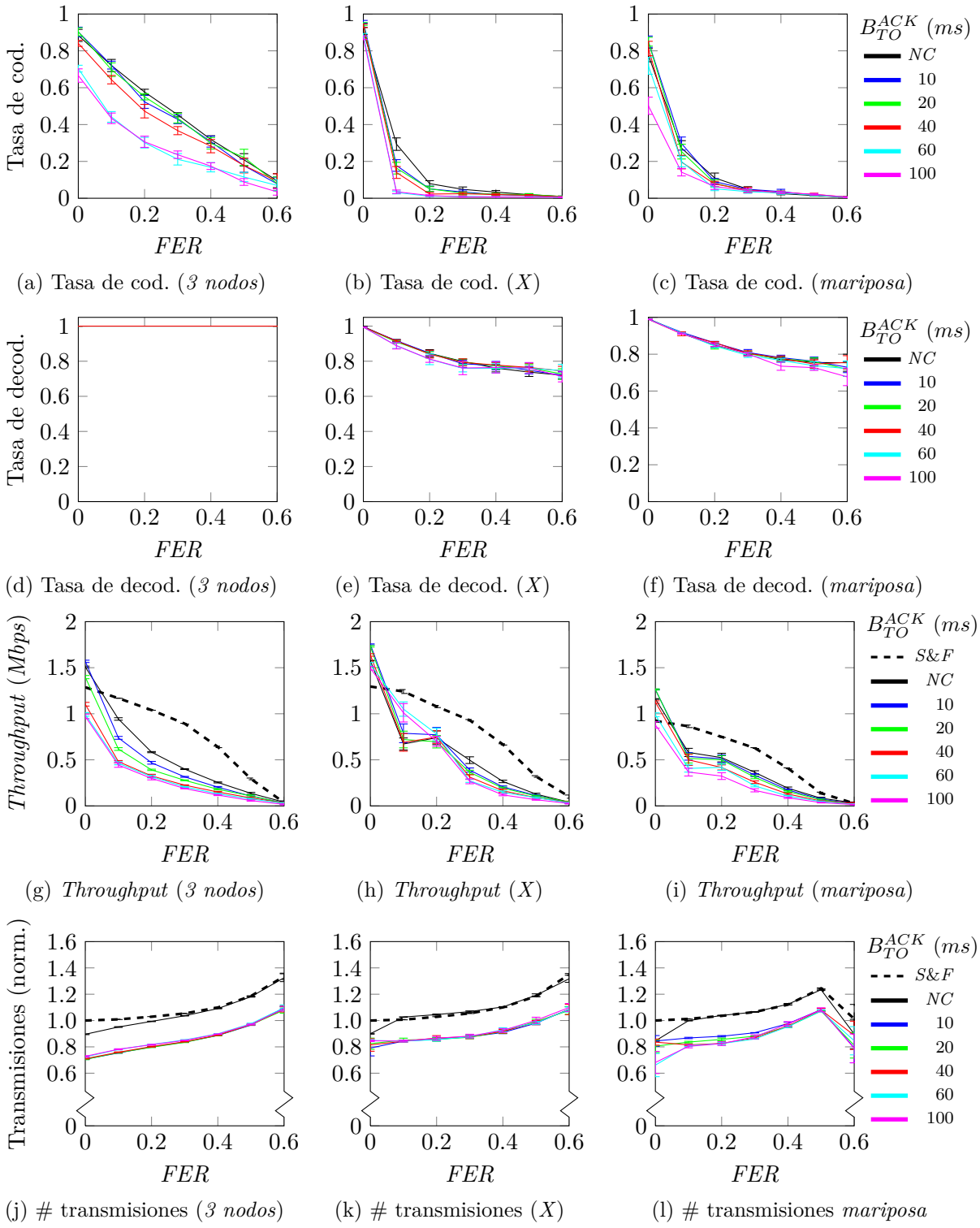
(d) # transmisiones (*3 nodos*) (e) # transmisiones (*X*) (f) # transmisiones (*mariposa*)

Figura 5.33: Principales métricas de rendimiento obtenidas en los escenarios canónicos con errores en todos los enlaces (encapsulación de ACKs de *TCP*)

No obstante, se observa una mejora notable en cuanto al número de transmisiones necesarias, ya que al combinar los reconocimientos sin codificar en los segmentos de datos, se produce una reducción (en torno a un 20 %) frente al protocolo *TCP* sobre un esquema de encaminamiento tradicional, tal y como se recoge en las Figuras 5.33d, 5.33e y 5.33f. Se comprueba además que la influencia del parámetro B_{TO}^{ACK} es prácticamente nula, y, salvo en el escenario de la *mariposa*, las curvas convergen en torno a un mismo valor, cercano a un 17 % de ahorro.

En un último grupo, las Figuras 5.34 y 5.35 resumen los resultados correspondientes a la combinación de las dos primeras soluciones (codificación de datos y encapsulación de reconocimientos *TCP*) en escenarios en los que los errores de propagación pueden aparecer en cualquier enlace de la red. Primero (Figura 5.34) se representa el comportamiento asociado a los parámetros $CB_S = 10$ paquetes y $CB_{TO} = 10$ $msegs$, correspondientes a la configuración con mayor rendimiento en *condiciones ideales*. En lo que respecta a la *tasa de codificación* (Figuras 5.34a, 5.34b y 5.34c), se ve que la cota superior está marcada por el esquema de *NC sin* encapsulación de reconocimientos, viendo como se reducen las *oportunidades de codificación* a medida que el B_{TO}^{ACK} crece. Por otro lado, las *tasas de decodificación* (Figuras 5.34d, 5.34e y 5.34f) presentan un comportamiento parejo, poniendo de manifiesto que la decodificación no depende de la configuración de los *buffers* en los *routers*. El *throughput* (Figuras 5.34g, 5.34h y 5.34i), muestra un comportamiento similar, ya que el aumento del tiempo de permanencia en el *buffer de reconocimientos* causa un incremento en el *RTT* por lo que, además de significar un aumento en la latencia de la conexión, repercute negativamente en la estimación del parámetro *RTO*. En consecuencia, cuando se producen errores, el rendimiento será más bajo cuanto más tiempo estén retenidos los reconocimientos en los nodos intermedios. Finalmente, la principal ventaja que aporta esta solución conjunta frente a la utilización exclusiva de la codificación es que consigue reducir drásticamente el número de transmisiones utilizadas, tal y como se muestra en las Figuras 5.34j, 5.34k y 5.34l.

En la segunda de las configuraciones, cuyos resultados se recogen en la Figura 5.35, se reduce el tiempo de permanencia en el *buffer de codificación* a 10 $msegs$, suavizando la reacción ante la pérdida de información en el medio inalámbrico. Con estos parámetros, los cambios más relevantes se producen en la *tasa de codificación* y el *throughput*. En el primero caso (ver Figuras5.35a, 5.35b y 5.35c), la reducción del parámetro CB_{TO} implica sacrificar en gran medida la aparición de *oportunidades de codificación*. Al intentar mitigar la latencia artificial introducida en los *nodos codificadores* en caso de error, el precio a pagar es que se tendrá una menor ganancia sobre escenarios ideales. El *throughput* (Figuras 5.35g, 5.35h y 5.35i), es sensiblemente mayor al visto con $CB_{TO} = 50$ milisegundos (Figuras 5.34g, 5.34h y 5.34i), aunque sigue siendo inferior al ofrecido por los mecanismos de encaminamiento tradicionales, sobre todo si el temporizador B_{TO}^{ACK} es alto.



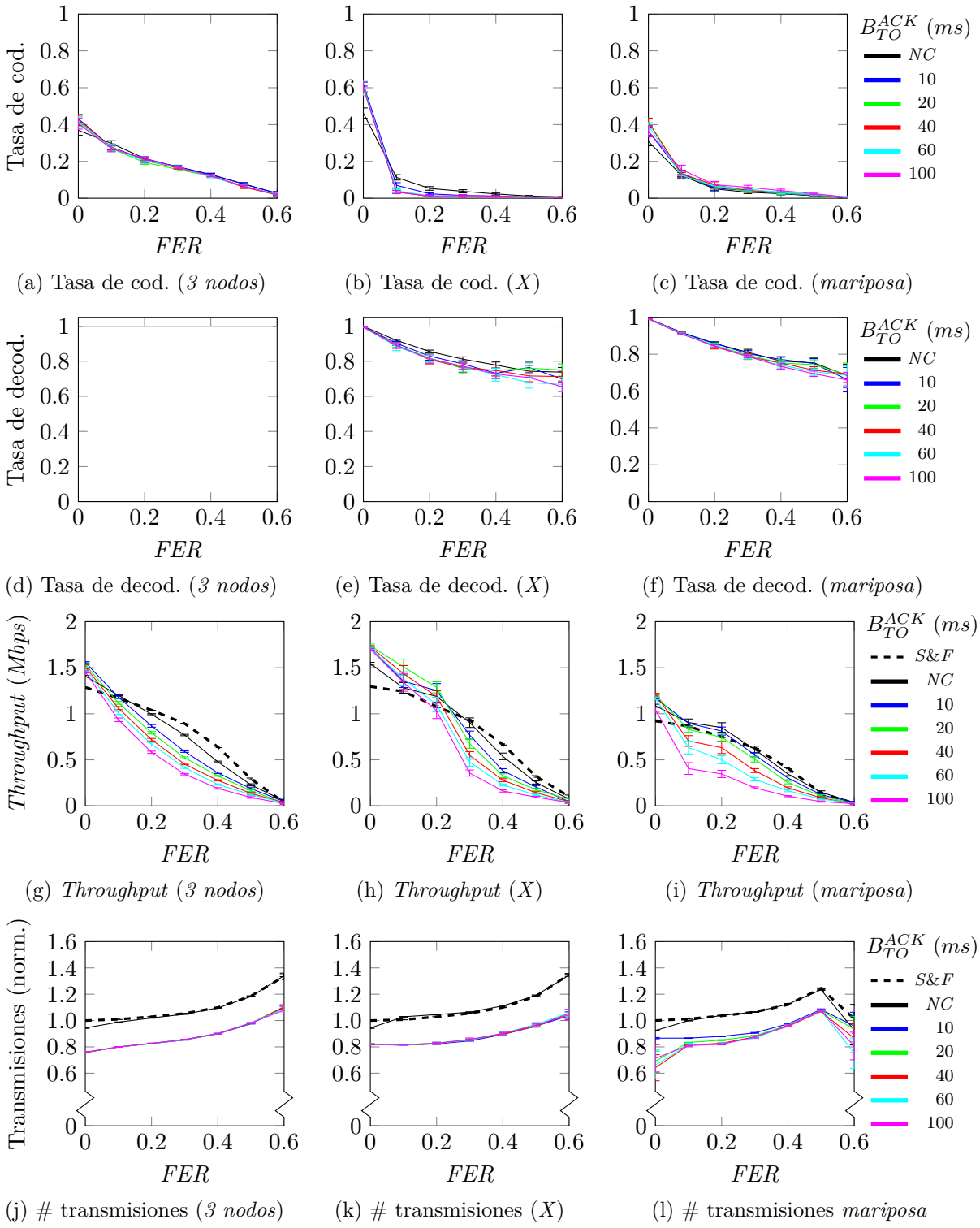
(a) Tasa de cod. (*3 nodos*) (b) Tasa de cod. (X) (c) Tasa de cod. (*mariposa*)

(d) Tasa de decod. (*3 nodos*) (e) Tasa de decod. (X) (f) Tasa de decod. (*mariposa*)

(g) *Throughput* (*3 nodos*) (h) *Throughput* (X) (i) *Throughput* (*mariposa*)

(j) # transmisiones (*3 nodos*) (k) # transmisiones (X) (l) # transmisiones *mariposa*

Figura 5.34: Principales métricas de rendimiento obtenidas en los escenarios canónicos con errores en todos los enlaces para la combinación de los esquemas NC y de encapsulación de reconocimientos TCP ($CB_S = 10$ paquetes y $CB_{TO} = 50$ milisegundos)



(a) Tasa de cod. (*3 nodos*) (b) Tasa de cod. (X) (c) Tasa de cod. (*mariposa*)

(d) Tasa de decod. (*3 nodos*) (e) Tasa de decod. (X) (f) Tasa de decod. (*mariposa*)

(g) *Throughput* (*3 nodos*) (h) *Throughput* (X) (i) *Throughput* (*mariposa*)

(j) # transmisiones (*3 nodos*) (k) # transmisiones (X) (l) # transmisiones *mariposa*

Figura 5.35: Principales métricas de rendimiento obtenidas en los escenarios canónicos con errores en todos los enlaces para la combinación de los esquemas NC y de encapsulación de reconocimientos TCP ($CB_S = 10$ paquetes y $CB_{TO} = 10$ milisegundos)

Comparativa

Por último, se resume el comportamiento de todas las soluciones en un mismo lugar, para comparar las prestaciones de cada una de ellas. Para ello, se representarán aquellas configuraciones de los diferentes *buffers* (i.e. CB y B^{ACK}) que hayan ofrecido unas mejores prestaciones, recogidas en la Tabla 5.2.

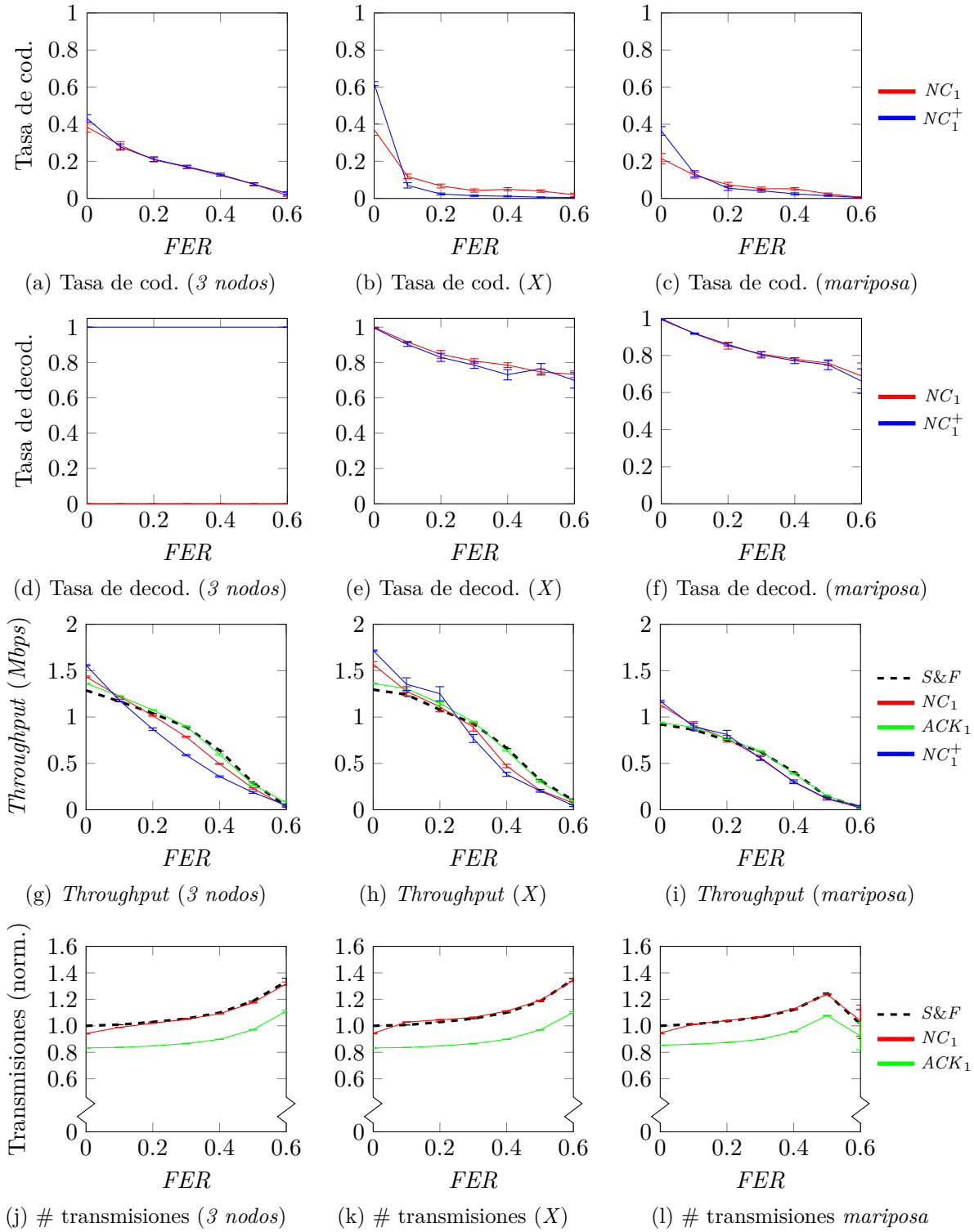
Tabla 5.2: Configuración de los diferentes atributos del nivel *NC*

	$\boldsymbol{CB_S}$ [paq.]	$\boldsymbol{CB_{TO}}$ [*msegs*]	$\boldsymbol{B_S^{ACK}}$ [paq.]	$\boldsymbol{B_{TO}^{ACK}}$ [*msegs*]
$\boldsymbol{NC_1}$	10	10	0	0
$\boldsymbol{NC_2}$	10	50	0	0
$\boldsymbol{ACK_1}$	0	0	10	10
$\boldsymbol{ACK_2}$	0	0	10	50
$\boldsymbol{NC_1^+}$	10	10	10	10
$\boldsymbol{NC_2^+}$	10	50	10	10

La Figura 5.36 muestra los resultados correspondientes a las configuraciones NC_1, ACK_1 y NC_1^+, ya que las otras ofrecen unas prestaciones claramente inferiores. Las métricas específicas a la capa *NC* presentan una gran similitud, con una mayor sincronización de los flujos en el caso de mezclar los *ACKs* con los segmentos de datos en situaciones con baja *FER*, como se observa en las *tasas de codificación* (Figuras 5.36a, 5.36b y 5.36c). Por otra parte, a la vista del *throughput* parece claro que, en presencia de errores, codificar la información perteneciente a diferentes flujos no sólo no aporta ningún beneficio sobre un esquema de transmisión tradicional, sino que además puede conducir incluso el rendimiento a cotas inferiores, como se observa en las Figuras 5.36g, 5.36h y 5.36i. La *desincronización* de los flujos producida por la reacción de los mecanismos de control de flujo resulta determinante, ya que el desequilibrio producido en las tasas de envío provoca la llegada continua de segmentos del mismo flujo, reduciendo significativamente la aparición de *oportunidades de codificación*. Además, la ventaja que aportaba la combinación de ambas técnicas con una $FER = 0$ desaparece a medida que la tasa de errores crece, sobre todo en el escenario de *3 nodos*, siendo inferior que al usar únicamente la codificación del tráfico de datos. Por último, merece la pena incidir de nuevo en que el supuesto ahorro asociado a la combinación de información en la red deja de ser apreciable cuando las conexiones pierden su sincronismo, y el número de transmisiones se acerca al observado al emplear un método de encaminamiento tradicional. Solamente cuando la encapsulación de reconocimientos se encuentra activa se reduce el total de envíos (Figuras 5.36j, 5.36k y 5.36l).

Comportamiento sobre un canal a ráfagas

Antes de pasar al siguiente escenario, se presentarán los resultados correspondientes a la utilización de un modelo de canal con memoria (*HMP*), asumiendo que los errores pueden aparecer en cualquier enlace.



(a) Tasa de cod. (*3 nodos*) (b) Tasa de cod. (X) (c) Tasa de cod. (*mariposa*)

(d) Tasa de decod. (*3 nodos*) (e) Tasa de decod. (X) (f) Tasa de decod. (*mariposa*)

(g) *Throughput* (*3 nodos*) (h) *Throughput* (X) (i) *Throughput* (*mariposa*)

(j) # transmisiones (*3 nodos*) (k) # transmisiones (X) (l) # transmisiones *mariposa*

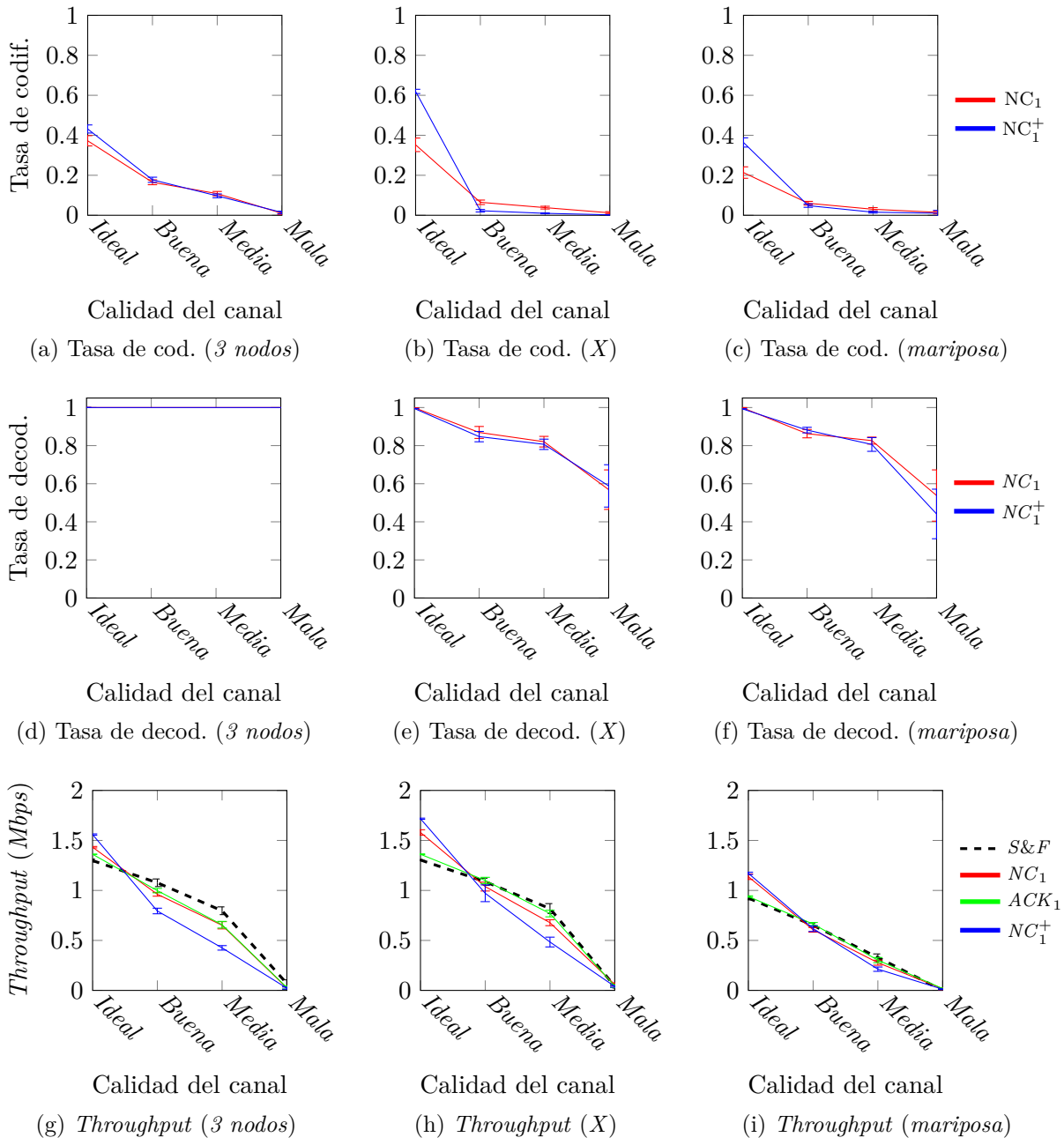
Figura 5.36: Comportamiento de los esquemas analizados sobre los escenarios canónicos

La Figura 5.37 recoge las principales métricas de rendimiento obtenidas al sustituir el modelo de canal *Nativo* del simulador por el *HMP*. Éste se configura atendiendo a tres posibilidades: un canal *bueno*, con una tasa de error de trama cercana al 16 %, otro *medio*, cuya *FER* ronda el 30 % y, por último, uno *malo*, en el que aproximadamente la mitad de las recepciones de datos serán erróneas. Resumiendo las principales diferencias observadas, la pérdida de paquetes penaliza más rápidamente a la sincronización entre los flujos, sobre todo en los casos de la *X* y la *mariposa* (Figuras 5.37b y 5.37c), e incluso en el canal *bueno* las *tasas de codificación* se ven prácticamente anuladas, tanto para la solución NC_1 como para la NC_1^+. Con respecto a las *tasas de decodificación*, se observa una caída significativa entre los canales *medio* y *malo*, alcanzando un valor cercano al 50 % en este último. La explicación se encuentra en que la recepción de paquetes codificados es tan reducida (la *tasa de codificación* es prácticamente nula) que la tasa de *errores de decodificación* se ajusta a la *FER* del canal.

Finalmente, el *throughput* presenta un descenso mucho más acusado, debido a la aparición de largas ráfagas de errores, que van a producir la pérdida de múltiples segmentos en intervalos de tiempo cortos, circunstancia que los mecanismos de control de congestión de *TCP* no consiguen combatir adecuadamente. En comparación con un modelo de canal sin memoria (Figura 5.36), el rendimiento del protocolo *inter-flujo* se ve todavía más perjudicado, siendo más apreciable aún la pérdida frente a un encaminamiento tradicional.

Resulta asimismo interesante estudiar la reacción de *TCP* ante estos errores a ráfagas. En primer lugar, en la Figura 5.38 se representa el número total de retransmisiones generadas en los dos flujos en función de la calidad del canal y de la solución adoptada; en comparación con los resultados ofrecidos sobre un canal sin memoria (Figura 5.28), se observa que, para una tasa de error de trama similar (15 % para el canal *bueno*, 30 % para el *medio* y 50 % para el *malo*) el número de retransmisiones es mayor. La diferencia más apreciable se encuentra en el tercero de los escenarios (*mariposa*).

La Figura 5.39 permite analizar las causas de las retransmisiones, representando el porcentaje de su origen (*triple ACK duplicado*, *ACK parcial* o *RTO*), normalizado con respecto al total. Al igual que en el canal sin memoria, aparecen dos grupos claramente diferenciados por el esquema de encaminamiento utilizado, donde el tradicional es capaz de mantener el porcentaje de retransmisiones basadas en el *RTO* más bajo que la solución basada en *NC* (salvo en el canal *malo*, donde la pérdida de segmentos es tan elevada que la capa *TCP* no puede recuperar la información a tiempo). La mayor diferencia entre el canal *Nativo* (Figura 5.29) y uno con memoria es que, para una misma *FER*, las ráfagas de tramas corruptas provocan una mayor pérdida de información en el lado del receptor en intervalos cortos de tiempo, lo que genera un mayor número de retransmisiones por *RTO*, perjudicando de manera clara el rendimiento, de manera que no es posible finalizar una conexión en canales *malos*, con tasas de error de trama cercanos al 50 % (mientras que en el modelo `Nativo`, esto no se producía hasta valores de *FER* cercanos a 0.6).

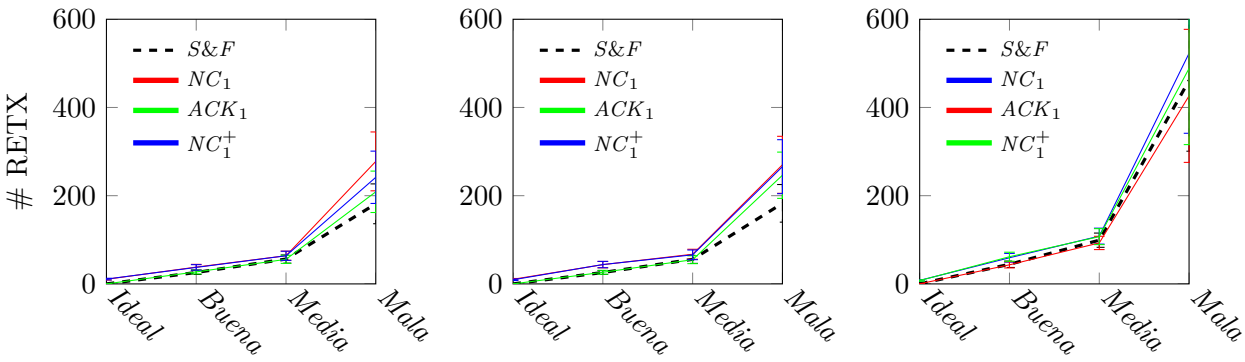


(a) Tasa de cod. (*3 nodos*) (b) Tasa de cod. (X) (c) Tasa de cod. (*mariposa*)

(d) Tasa de decod. (*3 nodos*) (e) Tasa de decod. (X) (f) Tasa de decod. (*mariposa*)

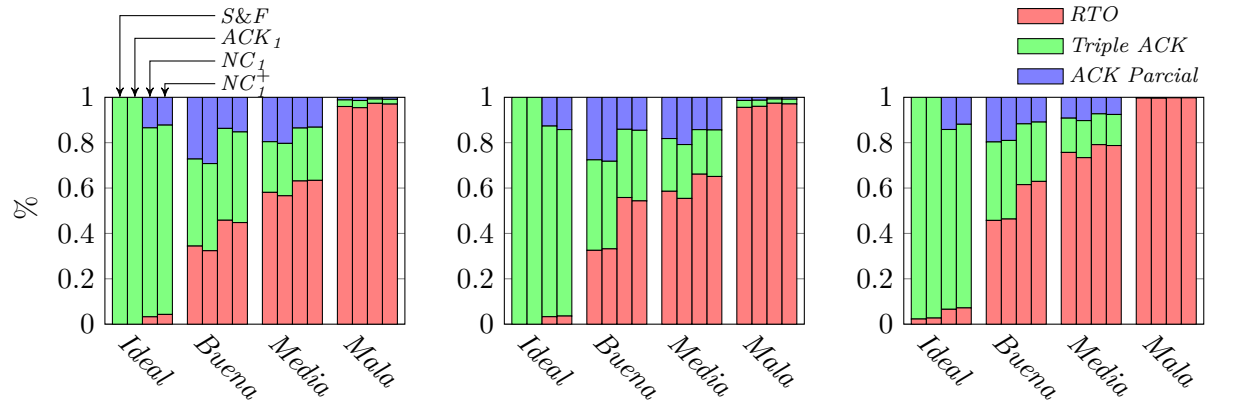
(g) *Throughput* (*3 nodos*) (h) *Throughput* (X) (i) *Throughput* (*mariposa*)

Figura 5.37: Comportamiento de los esquemas analizados sobre los escenarios canónicos con errores en todos los enlaces. Para comprobar el impacto de un canal a ráfagas más realista, se sustituirá el modelo tradicional por el *HMP* presentado en el Capítulo 2



(a) Topología *en línea* (b) Topología *X* (c) Topología *mariposa*

Figura 5.38: Número medio de retransmisiones en los escenarios canónicos con errores en todos los enlaces (modelo de canal a ráfagas)



(a) Topología *en línea* (b) Topología *X* (c) Topología *mariposa*

Figura 5.39: Causa de las retransmisiones *TCP* para los escenarios para los escenarios canónicos con errores en todos los enlaces (modelo de canal a ráfagas)

Errores en los enlaces laterales (recepciones promiscuas)

En esta configuración se limita la presencia de errores a los enlaces laterales, que permite que la información *nativa* llegue a los nodos receptores, para ser utilizada posteriormente en la fase de decodificación. Como su proceso se limita a este proceso de recuperación de los datos originales, no se analizará la topología de *3 nodos*, ya que el *buffer de decodificación* siempre tiene los segmentos *nativos* necesarios para llevar a cabo correctamente la decodificación. Para las otras dos (*X* y *mariposa*), la pérdida de segmentos en los enlaces $S_1 \rightarrow D_2$ y $S_2 \rightarrow D_1$ en los nodos finales provoca la aparición de *errores de*

decodificación, lo que conlleva el descarte del paquete codificado[24]. Deberá tenerse en cuenta además que, al tratarse de recepciones promiscuas, no están protegidas por el esquema de retransmisiones del nivel de enlace.

La Figura 5.40 recoge las principales métricas de rendimiento; en primer lugar, destaca la rápida caída de la *tasa de codificación* para los dos escenarios (Figuras 5.40a y 5.40d), lo que pone de manifiesto que las pérdidas provocadas por los *errores de decodificación* afectan, principalmente, a una única conexión. Como ya se ha mencionado, la reacción de los mecanismos de control de congestión de *TCP* en estos casos rompe el equilibrio entre los dos flujos, reduciendo la tasa de envío del que experimenta más errores, habilitando una cantidad mayor de recursos a la conexión con mejores condiciones. Así, se produce un efecto de transferencia secuencial de las dos sesiones. En lo que se refiere a las *tasas de decodificación* (Figuras 5.40b y 5.40b), se puede comprobar como la pendiente con la que decrece se corresponde con la probabilidad de perder una trama en estos enlaces. Por su parte, el *throughput* (Figuras 5.40c y 5.40f), muestra bajadas puntuales en los esquemas que utilizan *NC* para valores de *FER* entre 0.1 y 0.2, justificados por la presencia de paquetes codificados en la red, que derivan en *errores de decodificación* y en la consiguiente reducción de las prestaciones. Como era de esperar, en los esquemas que no utilizan codificación (*S&F*, ACK_1 y ACK_2), el valor del *throughput* es completamente independiente de las pérdidas en los enlaces laterales.

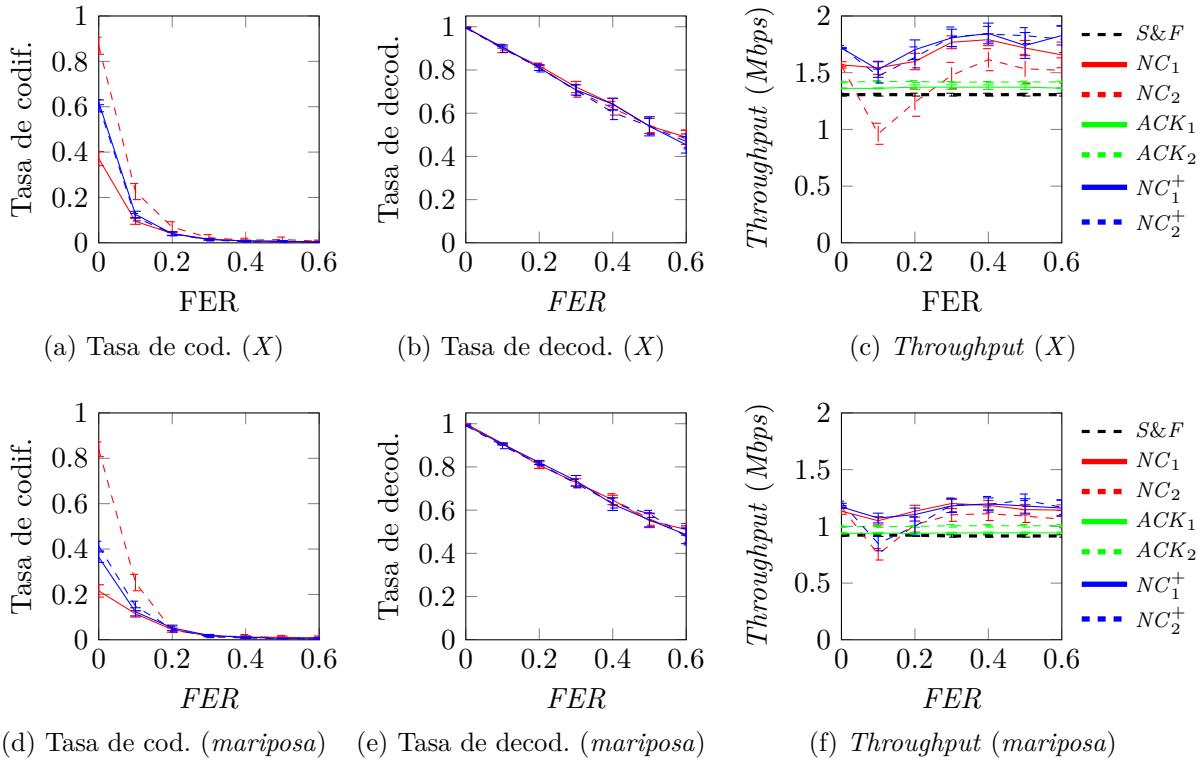
Puede afirmarse por tanto que, en estas condiciones, las técnicas de combinación de paquetes de diferentes flujos no aportan beneficio alguno.

Errores en los enlaces centrales

En este caso, los enlaces propensos a causar errores se corresponden con los flujos de bajada, cuyo contenido puede transportar segmentos codificados. En concreto, para el escenario de *3 nodos*, serán los enlaces $C_1 \rightarrow S_1$ y $C_1 \rightarrow S_2$; para el escenario en *X*, $C_1 \rightarrow D_1$ y $C_1 \rightarrow D_2$; por último, en la *mariposa* únicamente va a existir un enlace corrupto ($C_1 \rightarrow R_1$).

Para este último grupo de resultados, la Figura 5.41 muestra las principales métricas de rendimiento. El hecho de que los errores estén en los caminos de bajada $\mathcal{D}(C_1, \mathcal{F}_i)$, con $i = [1, 2]$ asegura que prácticamente todos los mensajes codificados sean recuperados con éxito (*tasa de decodificación* $\approx$ *1*). Se ve que en los escenarios de *tres nodos* y *X* destaca la caída en las *tasas de codificación* (Figuras 5.41a y 5.41b), más suave que en las configuraciones anteriores, manteniendo unos valores elevados incluso cuando la calidad del medio inalámbrico es baja (por ejemplo, se tiene una tasa cercana a un 20 % para una $FER = 0.4$). Hay que tener en cuenta que existe un porcentaje más elevado de pérdidas que afectan de manera simultánea a los dos flujos. Además, su valor será limitado, ya

[24]La versión avanzada del protocolo *inter-flujo*, descrita en la Sección 5.3, busca mitigar el impacto de estos errores de decodificación, mediante la introducción de un nuevo *buffer* que mantenga estos paquetes.



(a) Tasa de cod. (X) (b) Tasa de decod. (X) (c) *Throughput* (X)

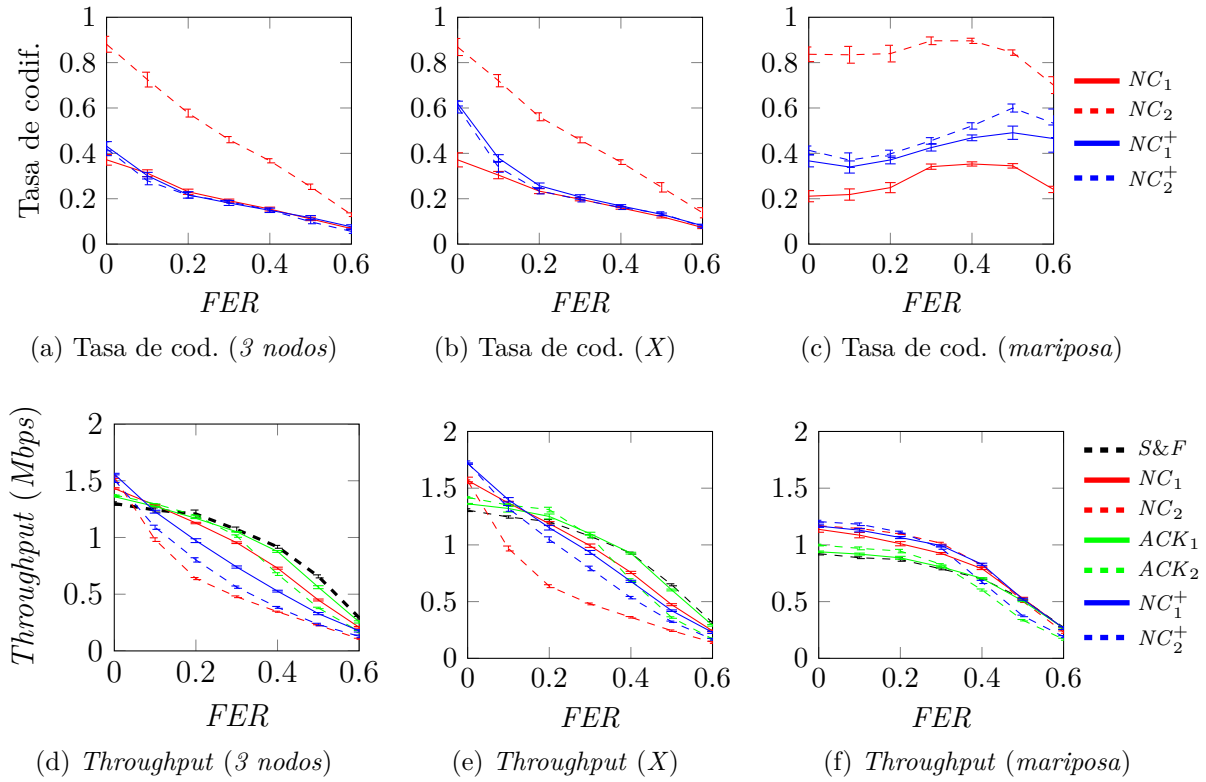
(d) Tasa de cod. (*mariposa*) (e) Tasa de decod. (*mariposa*) (f) *Throughput* (*mariposa*)

Figura 5.40: Comportamiento de los esquemas analizados sobre los escenarios canónicos con errores en los enlaces laterales

que los enlaces están (parcialmente) protegidos por los mecanismos de retransmisión del estándar *IEEE 802.11*.

Los valores observados de *throughput* (Figuras 5.41d y 5.41d) vuelven a evidenciar un efecto similar a lo visto a lo largo de toda esta campaña de simulaciones, donde toda la *ganancia de codificación* desaparece en el momento en el que el porcentaje de errores empieza a ser apreciable. En estos casos en particular, para valores de error superiores al 20 %, el esquema tradicional *store-and-forward* proporciona unas prestaciones superiores. Nuevamente, la encapsulación de los reconocimientos de *TCP* en segmentos de datos aparece como una solución atractiva, ya que, además de tener un *throughput* similar, logra reducir en gran medida el número de transmisiones necesarias.

Los resultados más interesantes se dan en el escenario de la *mariposa*, en el que la pérdida de un paquete codificado afecta a ambos flujos de la misma manera, lo que causará retransmisiones por parte de las entidades *TCP* correspondientes. A medida que aparecen más errores, las ventanas de congestión de las fuentes se irán sincronizando en mayor medida, hasta que llegan a estar prácticamente acompasadas. En los momentos de máxima sincronía, el ancho de banda se repartirá equitativamente, favoreciendo la aparición



(a) Tasa de cod. (*3 nodos*) (b) Tasa de cod. (*X*) (c) Tasa de cod. (*mariposa*)

(d) *Throughput* (*3 nodos*) (e) *Throughput* (*X*) (f) *Throughput* (*mariposa*)

Figura 5.41: Comportamiento de los esquemas analizados sobre los escenarios canónicos con errores en los enlaces centrales

de numerosas *oportunidades de codificación*, que permiten transportar más información en menos envíos.

Así, las Figuras 5.41c y 5.41f muestran como, a medida que el número de errores va en aumento, la *tasa de codificación* no sólo no disminuye, sino que aumenta. De este "equilibrio" se va a beneficiar el *throughput*, siendo la caída en los esquemas de codificación (NC y NC$^+$) notablemente menos acusada que en los casos anteriores.

5.5.3. Caracterización del algoritmo de búsqueda de nodos codificadores

En esta sección se extenderá el estudio a escenarios más realistas, donde la posición de los nodos no facilita la aparición de *oportunidades de codificación*. Para ello, se dividirá el análisis en dos fases: en la primera de ellas se evaluará la viabilidad del uso de *NC* sobre despliegues aleatorios, analizando si la topología correspondiente (y los flujos presentes) puede beneficiarse del uso del protocolo *NC*. Posteriormente, y basándose en estos resultados previos, se analizarán aquellos escenarios en los que la utilización del pro-

tocolo *inter-flujo* pueda resultar positiva. En este caso, se utilizará el simulador `ns-3` para cuantificar la mejora introducida por la combinación de paquetes, en comparación con un esquema tradicional de *almacenamiento y reenvío*.

En una primera fase, al igual que para los algoritmos de encaminamiento multi-camino *MPTCP* descritos en el Capítulo 3, se ha llevado a cabo un proceso independiente, cuya operación consta de los siguientes pasos: *(1)* desplegar N nodos en un escenario cuadrado de área $M \times M$ aleatoriamente[25]; *(2)* ejecutar el algoritmo presentado en la Sección 5.4; y *(3)*, en caso de encontrar un nodo codificador, generar los ficheros necesarios para poder estudiar las prestaciones sobre el simulador `ns-3`, para emular la presencia de tráfico sobre la red desplegada.

Con el fin de favorecer la aparición de nodos codificadores en los diferentes escenarios, se ha establecido la siguiente lista de condiciones en el proceso de búsqueda de nodos codificadores:

1. Se desplegarán un total de 32 nodos sobre un área cuadrada de $100 \times 100\, m^2$.
2. Como filtro previo, se descartarán todos aquellos despliegues cuyos grafos subyacentes no sean conexos.
3. La posición de los nodos permanecerá estática durante todo el proceso.
4. El área de cobertura de los nodos se corresponde con una región circular de 20 metros de radio.
5. Para reducir el número de escenarios en los que no se puede encontrar, al menos, un *nodo codificador*, se favorecerá que ambos flujos converjan en algún nodo articulación de la red. Para ello, los nodos extremos (S_1, S_2, D_1 y D_2) serán los que se encuentren más cercanos a las coordenadas $(80, 20)$, $(80, 80)$, $(20, 20)$ y $(80, 20)$, respectivamente.
6. El número de flujos activos durante las simulaciones se limitará a 2, habilitando asimismo un único *nodo codificador*. Además, por razones de simplicidad, se asumirá que ambos flujos, $\mathcal{F}_1$ y $\mathcal{F}_2$, están activos durante toda la simulación.

Como ejemplo ilustrativo, la Figura 5.42 representa un despliegue arbitrario en el que se resumen los principales condicionantes a la hora de desplegar los nodos en el escenario. En este caso en particular, los nodos 20 y 3 ejercerán el papel de transmisores S_1 y S_2, al ser los más cercanos a las coordenadas $(20, 80)$ y $(80, 80)$, mientras que los terminales 25 y 31, como elementos más próximos a los puntos $(80, 20)$ y $(20, 20)$, harán las veces de estaciones receptoras (D_1 y D_2, respectivamente). El *Algoritmo de Dijkstra* establecerá los caminos más cortos para los flujos $\mathcal{F}_1$ y $\mathcal{F}_2$ con las rutas $32 \rightarrow 11 \rightarrow 27 \rightarrow 17 \rightarrow 1 \rightarrow 12 \rightarrow 25$

[25]La posición en cada uno de los ejes del plano XY se realizará a través de una variable aleatoria uniforme en el intervalo $[0, M]$.

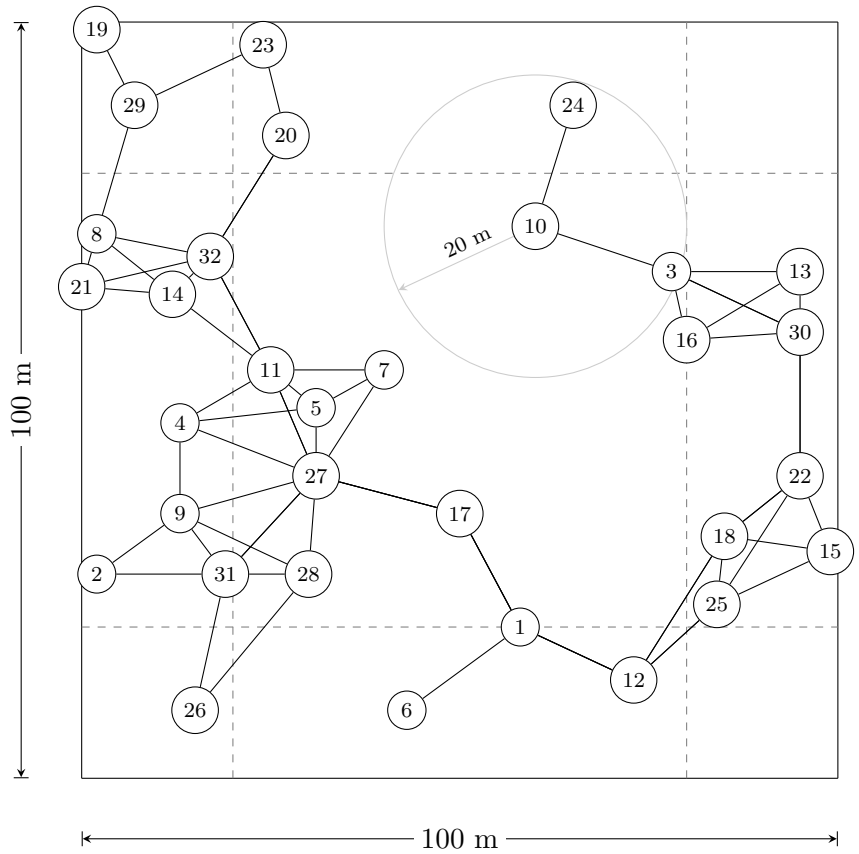
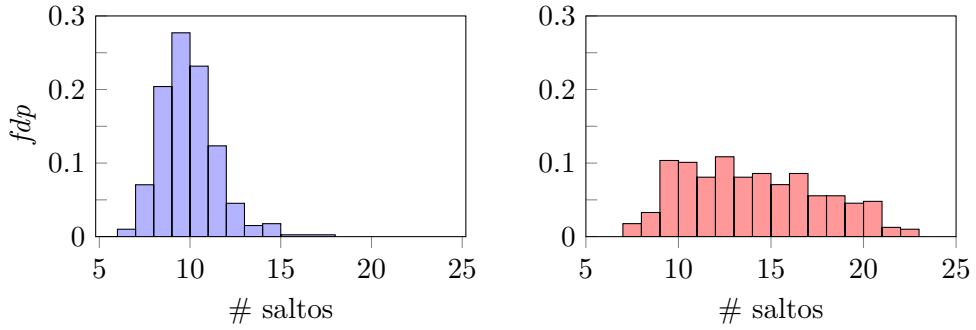


Figura 5.42: Topología aleatoria para identificar *oportunidades de codificación*

y $3 \rightarrow 30 \rightarrow 22 \rightarrow 18 \rightarrow 12 \rightarrow 1 \rightarrow 17 \rightarrow 27 \rightarrow 31$, respectivamente (aunque existen otras alternativas con el mismo número de saltos). Como puede observarse, existen varios nodos por los que pasa el tráfico perteneciente a ambos flujos (en este ejemplo, 27, 17, 1 y 12), cumpliendo con la primera de las condiciones impuestas en la Definición 5.2 de la Sección 5.4. Además, los destinos D_1 y D_2 están dentro del área de cobertura de alguno de los nodos intermedios que retransmiten los flujos de subida $\mathcal{U}(c_i, \mathcal{F}_j)$, con $j \in [1, 2]$, siendo c_i cualquiera de los potenciales codificadores, por lo que podrán adquirir la información necesaria para recuperar la información contenida en los paquetes codificados. Esto hará que el escenario reúna las condiciones necesarias para que la utilización de las técnicas de *NC* resulten viables.

Para dotar a esta etapa de análisis de cierta validez estadística, se utilizará el método de *Monte Carlo* para generar el número de experimentos necesarios para poder garantizar un total de 1000 escenarios viables diferentes, según las condiciones descritas anteriormente. Con los datos obtenidos en este primer paso (grafo y rutas asociadas a los dos flujos), se procederá a la ejecución del algoritmo mejorado presentado en la Sección 5.4, añadiendo los parámetros δ y K (número de rutas calculadas por el *Algoritmo de Yen*) como argumentos de entrada. Finalmente, el algoritmo devolverá como resultado una lista con los

potenciales *nodos codificadores*, en el caso de que se cumplan con los requisitos impuestos en la Definición 5.2.



(a) Esquema *store-and-forward* (b) Esquema *NC* (K=1)

Figura 5.43: Función densidad de probabilidad del número de saltos

La Figura 5.43 muestra la función densidad de probabilidad del número de saltos presentes en ambos flujos[26] para los 1000 escenarios generados, para los casos de un encaminamiento tradicional *store-and-forward* (i.e. el resultado tras ejecutar el *Algoritmo de Dijkstra*) y del esquema *inter-flujo* propuesto en este capítulo, limitando el número de rutas a 1 ($K = 1$) como parámetro de entrada en el *Algoritmo de Yen*. En la Figura 5.43a se observa que la solución tradicional es capaz, en más de un 90 % de los casos, de encontrar rutas con menos de 15 saltos, siendo más probables valores en el rango de $9 - 11$ saltos; por el contrario, la búsqueda de un nodo codificador en la alternativa *NC* (Figura 5.43b) incrementa la longitud de los caminos, presentando una distribución mucho más uniforme, llegando a valores de hasta 23 saltos para conectar un nodo con su destino. De esto se deduce que, en un porcentaje significativo de los experimentos, la utilización de las técnicas de *NC* implicará la introducción de saltos adicionales en las rutas que unen los terminales, factor que puede ser determinante en el rendimiento de las comunicaciones.

La Figura 5.44 representa la función de distribución del número de saltos a utilizar en cada una de las rutas, pudiendo comprobarse que valores de K más elevados no afectan de manera significativa (siendo la longitud en cualquier caso claramente superior a la de un encaminamiento clásico). La principal diferencia en el descubrimiento de múltiples caminos está en el porcentaje de escenarios en los que será viable la utilización del protocolo *inter-flujo*, ya que a medida que K crezca, aparecerán más alternativas sobre las que encontrar un posible *nodo codificador*.

La Figura 5.45 presenta las longitudes de las rutas obtenidas para ambas alternativas en cada uno de los despliegues. Se ve, inicialmente, que el número de saltos mínimo está marcado por el *Algoritmo de Dijkstra*. Sin embargo, el elemento que más llama la atención es la tendencia a utilizar rutas más largas con la solución *NC* en la mayoría de los casos

[26]Por comodidad, a la hora de representar los resultados, se tendrá en cuenta únicamente el resultado correspondiente a la ruta más larga.

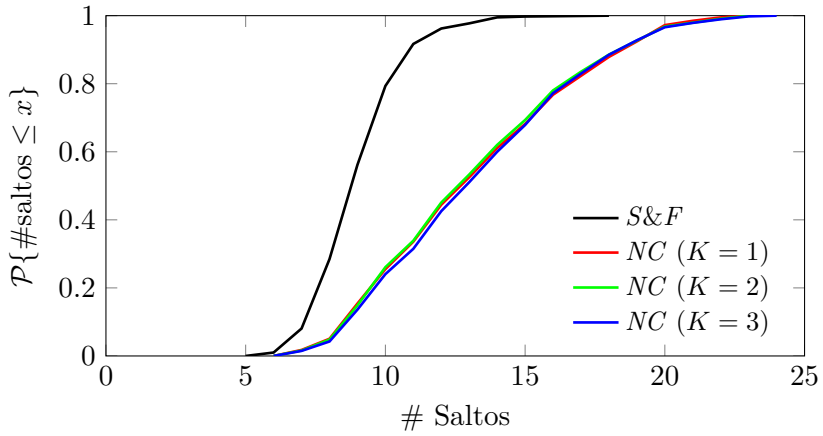


Figura 5.44: Función de distribución del número de saltos necesarios (*Algoritmo de Yen*)

(se han generado escenarios con hasta 12 saltos adicionales con respecto al esquema de encaminamiento tradicional). Esto, en canales basados en contienda, reducirá significativamente el rendimiento final. Será necesario cuantificar hasta qué punto la utilización de los mecanismos de *NC* van a ser capaces de compensar este efecto, provocado para forzar la presencia de algún *nodo codificador* en el escenario.

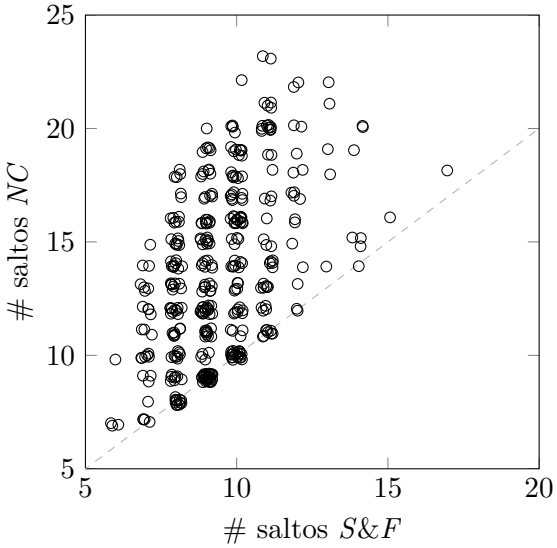


Figura 5.45: Rutas de mayor longitud (esquema tradicional *Vs. NC*)

Finalmente, se analiza la probabilidad de que exista, al menos, un *nodo codificador*, como se ve en la Figura 5.46. El eje X representa la diferencia de longitud entre las rutas de las dos soluciones analizadas, viéndose, en primer lugar, que el incremento de K sí que aporta un cierto beneficio, pues aumenta las posibilidades de encontrar escenarios viables para el uso de *NC*. En términos absolutos, si no se tiene en cuenta la longitud de las rutas generadas, el Algoritmo propuesto encuentra *nodos codificadores* en aproximadamente la

mitad de los escenarios (con un valor de $K = 3$). Sin embargo, no en todos ellos existirá una mejora en el rendimiento, que podría ser incluso inferior al de un esquema *store-and-forward*. Si se limitase la diferencia a 2 saltos, eligiendo este valor de acuerdo al límite de la región de codificación definido en [Katt08], el porcentaje de escenarios en los que el uso de *NC* es factible sería inferior al 10 %.

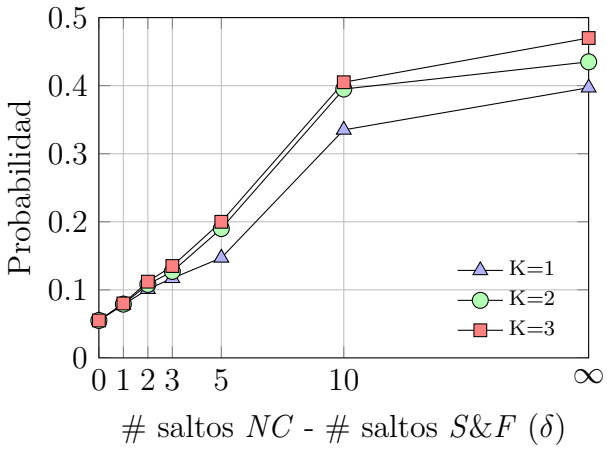


Figura 5.46: Probabilidad de encontrar al menos un *nodo codificador* en los escenarios generados aleatoriamente en función de la restricción impuesta por el parámetro δ del Algoritmo 5.2

Además, junto con estas cifras, no hay que olvidar las restricciones que se impusieron inicialmente, a la hora de generar los escenarios, para favorecer la aparición de elementos codificadores en los despliegues. Con todo, se puede decir que, en condiciones completamente aleatorias, la probabilidad de que sea viable el uso del protocolo *NC* será aún más baja, poniendo en duda su verdadera utilidad sobre topologías *no sintéticas*. A todo esto hay que sumar los problemas derivados de los mecanismos de control de congestión de *TCP* cuando la presencia de errores deja de ser despreciable.

5.5.4. Medidas de rendimiento en escenarios aleatorios

Tras el análisis realizado sobre la viabilidad del uso de *NC* sobre redes malladas inalámbricas, se cuantificarán ahora las mejoras derivadas de la codificación en los elementos intermedios. Para ello, como ya se ha adelantado, se llevarán a la plataforma `ns-3` aquellos escenarios en los que el Algoritmo haya asignado al menos un *nodo codificador*, para analizar el rendimiento a través de la generación de tráfico *TCP*.

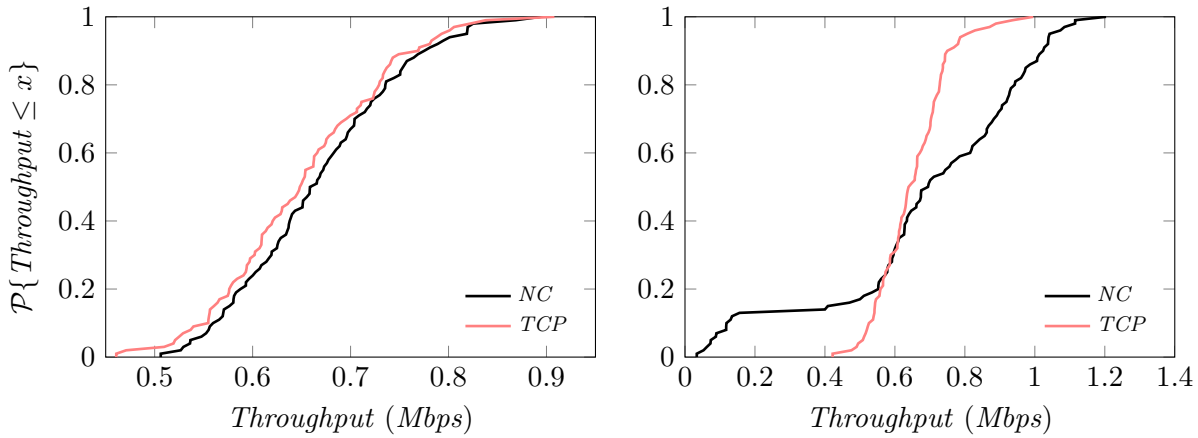
A continuación se resumen los principales parámetros de la configuración usados durante la campaña de simulaciones que se ha llevado a cabo:

- Con los datos obtenidos tras la ejecución del Algoritmo propuesto (posición de los 32 nodos, identidad de los dispositivos S_1, S_2, D_1 y D_2 y tablas de rutas estáticas de

los flujos $\mathcal{F}_1$ y $\mathcal{F}_2$), se simularán 1000 escenarios en los que el esquema *NC inter-flujo* sea viable. El experimento se repetirá para diferentes valores de δ, para comprobar el impacto de emplear rutas más largas para habilitar el uso de las técnicas de *NC*.

- Cada uno de los nodos fuente transmitirá un total de 5000 paquetes de información, con un tamaño por unidad de 1432 octetos.
- En lo que respecta a la configuración del nivel *NC*, se ajustarán los atributos del *buffer de codificación* con unos valores de $CB_S = 10$ paquetes y $CB_{TO} = 100$ *msegs*. Solamente se estudiará el comportamiento del protocolo básico, manteniendo el esquema de encapsulación de reconocimientos *TCP* deshabilitado.
- Los enlaces inalámbricos se configurarán a través del estándar *IEEE 802.11b*, trabajando con su tasa binaria máxima (11 *Mbps*) para los datos y 2 *Mbps* para los mensajes de control y/o *broadcast*. Además, los mecanismos *RTS/CTS* estarán desactivados.
- En este caso no se considerarán los errores de transmisión, ya que el análisis se centra en la aleatoriedad del despliegue.

El primero de los estadísticos a analizar, presentado en la Figura 5.47, se corresponde con la representación de la función de distribución del *throughput* para diferentes configuraciones del parámetro δ utilizado en el Algoritmo 5.2. Como se pudo ver en la Sección 5.2, este valor establece la diferencia entre las longitudes de las rutas obtenidas a través del algoritmo.



(a) $\delta = 0$ (b) $\delta = 1$

Figura 5.47: Función de distribución del *throughput* para diferentes valores de δ

Destaca la gran variabilidad en los resultados mostrados, consecuencia de la heterogeneidad de los escenarios generados aleatoriamente. Como puede intuirse, el rendimiento

muestra una tendencia inversamente proporcional a la longitud de las rutas. En la parte izquierda (Figura 5.47a) se muestra la función de distribución del rendimiento en escenarios con $\delta = 0$, observando una mejora constante, cercana a un 2 % con respecto a una transmisión estándar, cifra muy alejada de los valores que se obtuvieron sobre escenarios sintéticos en condiciones ideales (Sección 5.5.1).

Sin embargo, en el momento en el que se relaja la restricción y se permite la utilización de rutas más largas, los resultados comienzan a deteriorarse. Por ejemplo, con $\delta = 1$, tal y como muestra la Figura 5.47b, se aprecia una primera zona (cercana al 30 % de los casos, con un rendimiento inferior a los 0.6 *Mbps*) en la que *TCP* proporciona unos valores superiores a la solución basada en *NC*; sin embargo, cuando el *throughput* sea superior a 0.6 *Mbps*, se invertirá la tendencia y la codificación pasará a ofrecer unas mejores prestaciones. En estos casos se puede decir que el propio esquema *inter-flujo* compensa la contención adicional introducida en el sistema por el salto adicional. Además, el comportamiento estudiado presentará una variabilidad mucho mayor a la observada con $\delta = 0$.

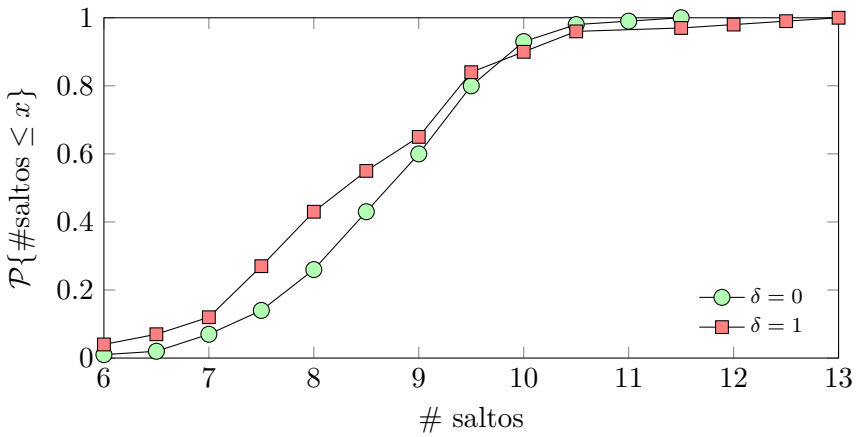


Figura 5.48: Función de distribución del número medio de saltos

Para encontrar una explicación a esta diferencia, la Figura 5.48, representa la función de distribución acumulada del número de saltos que obtiene el algoritmo propuesto para los diferentes valores de δ. Puede verse que la relajación de la restricción ($\delta = 1$) permite, en un gran porcentaje de los escenarios analizados ($\approx 90\,\%$), rutas más cortas que cuando se fuerza al algoritmo a encontrar *nodos codificadores* en caminos de igual longitud que las obtenidas con *Dijkstra* ($\delta = 0$). Al necesitar un menor número de reenvíos en los nodos intermedios, la *ganancia de codificación* presentará unos valores más elevados, incrementando por lo tanto el *throughput* en mayor medida. Por contra, existen los escenarios en los que la elección de rutas más largas (un salto adicional) implica un incremento en el retardo que no se compensa con la transmisión conjunta de paquetes combinados, penalizando notablemente el rendimiento.

En el momento en el que el valor de δ es superior a 1, la combinación de paquetes no consigue mejorar, en ningún caso, las prestaciones de la solución tradicional, siendo más perjudicial a medida que el valor de la restricción δ crece.

5.6 Conclusiones y líneas futuras

A pesar de que existe un gran número de trabajos que caracterizan la interacción entre el protocolo *TCP* y diferentes soluciones de *NC inter-flujo* sobre redes malladas inalámbricas, no existe a día de hoy una conclusión clara acerca de sus beneficios. La tendencia en los últimos tiempos se inclina hacia la utilización conjunta de mecanismos *intra-flujo RLNC* con técnicas de *encaminamiento oportunista*.

En este capítulo se han presentado las características de un protocolo *NC inter-flujo* novedoso, diseñado a partir de los conceptos fundamentales de este tipo de soluciones, cuyo ejemplo más destacable es el protocolo *COPE* [Katt08]. A partir de un módulo común desarrollado en la plataforma `ns-3`, se añade una nueva capa entre los niveles de red y de transporte[27] que incorpora los mecanismos propios del protocolo. En un primer acercamiento, se propone una versión básica de la solución, en la que la presencia de *nodos codificadores* en una red mallada inalámbrica permite combinar el tráfico perteneciente a diferentes flujos de datos. Para poder establecer las llamadas *oportunidades de codificación*, estos elementos deberán almacenar temporalmente los segmentos *nativos* recibidos en un contenedor llamado *buffer de codificación*, cuyos atributos (tamaño y tiempo máximo de permanencia) tienen un gran impacto en el rendimiento global del sistema. Cuando haya al menos dos segmentos de diferentes flujos, bastará con realizar una simple operación *XOR* para generar el mensaje cifrado. Desde el punto de vista de un receptor, será necesario mantener todos los segmentos *nativos* recibidos (o transmitidos), ya que podrían servir para recuperar la información contenida en el interior de un paquete codificado, extraída mediante una nueva operación *XOR*.

Posteriormente, gracias a la capa intermedia implementada, se añade una funcionalidad adicional, para que los nodos intermedios puedan retener los ACKs (de *TCP*) el tiempo suficiente, hasta encontrar algún paquete de datos con el espacio necesario para encapsularlo (dentro de la cabecera de nivel *NC*), disminuyendo de este modo la contención que produciría el envío individual de cada mensaje por separado.

Tras un exhaustivo análisis del comportamiento del módulo *inter-flujo*, se ha comprobado que su rendimiento presenta una degradación notable cuando se pierde información por las condiciones hostiles del medio inalámbrico. Con el fin de mejorarlo, se añade al protocolo una funcionalidad adicional, que permite a la entidad *NC* mantener información actualizada del estado de las conexiones que la atraviesan. Esta monitorización permite una gestión más optimizada de la memoria utilizada, además de un sistema de retransmisiones

[27] Sin modificar el comportamiento tradicional de ninguno de los niveles.

propio de la capa *NC*, cuya finalidad será la de conseguir reducir al máximo el número de retransmisiones *TCP* extremo a extremo.

Uno de los problemas fundamentales de un esquema de codificación *inter-flujo* será la selección óptima de los nodos que van a ejercer las tareas de codificación. En escenarios sencillos y canónicos la elección es inmediata, mientras que un despliegue real la asignación oportuna de los nodos y los patrones de codificación se vuelve muy compleja. De hecho, en un gran porcentaje de los casos la topología subyacente no cumplirá con las condiciones necesarias para que la utilización del protocolo *NC* aporte algún tipo de beneficio. Para acometer este análisis se ha desarrollado un algoritmo que, a partir del grafo que define el escenario, devuelve una lista con los potenciales *nodos codificadores*, en el caso de que los hubiese.

Para evaluar las prestaciones ofrecidas por el protocolo *inter-flujo* se ha llevado a cabo un extenso análisis sistemático que compara su rendimiento con el ofrecido por un esquema de transmisión clásico basado en un encaminamiento *store-and-forward*. Para ello, se ha dividido el proceso de medida en cuatro etapas claramente diferenciadas:

1. Inicialmente, se ha caracterizado el rendimiento de las soluciones sobre varios escenarios canónicos ideales. Se ha comprobado que la utilización de *buffers* propios del nivel *NC* (de codificación y de reconocimientos) más grandes no redunda necesariamente en una mejora del rendimiento final. En concreto, retener los paquetes en exceso tiene un doble efecto, ya que por un lado se aumenta el número de *oportunidades de codificación*, pero por otro sin embargo se incrementa el retardo global, penalizando al *throughput*. En el supuesto de emplear una configuración adecuada, se pueden conseguir mejoras significativas, tanto en el rendimiento (hasta un 39 % cuando se combinan las dos técnicas implementadas) como en la reducción del número de transmisiones necesarias (con ganancias cercanas al 30 %).

2. Posteriormente, utilizando los mismos escenarios, se ha analizado el impacto de los errores de transmisión. Las conclusiones en este punto no son tan positivas, ya que el rendimiento observado decae a una gran velocidad en el momento en el que la presencia de errores empieza a ser apreciable ($FER \geq 0.1$), obteniendo *throughputs* claramente inferiores a los ofrecidos por soluciones basadas en un encaminamiento tradicional. Se ha estudiado también el efecto de restringir las pérdidas a enlaces concretos, especialmente en la sincronización de los flujos mediante los mecanismos de control de congestión de *TCP*.

3. Se ha estudiado la viabilidad de las técnicas de *NC* en despliegues aleatorios a través de un algoritmo que determina la presencia de nodos codificadores. Se ha comprobado que el porcentaje de escenarios que presentan las condiciones necesarias para la codificación es bajo (menor del 50 %) y que, además, en la mayor parte de los casos, las rutas necesarias para poder combinar segmentos implican un gran número de saltos adicionales con respecto a los caminos más cortos.

4. Finalmente, se han analizado con el simulador aquellos escenarios en los que el algoritmo ha encontrado al menos un nodo codificador, comparando el *throughput* de los esquemas de *NC* y *store-and-forward*. Se ha comprobado que la ganancia de la primera alternativa es muy baja (en torno a un 2 %), y únicamente en un porcentaje muy reducido de los escenarios, ya que solamente se obtienen beneficios cuando las longitudes de las rutas son iguales (lo que sucede en menos de un 10 % de los escenarios).

Se ha puesto de manifiesto que los mecanismos de codificación *inter-flujo* no son la solución más apropiada para ser trabajar sobre redes reales. Sin embargo, todavía quedan abiertas numerosas preguntas que deberían ser tratadas en un futuro:

- Se deberían analizar las prestaciones de la versión avanzada del protocolo, poniendo especial atención en el comportamiento de la capa *TCP*. Al introducir un nuevo sistema de retransmisiones independiente, se reducirían las retransmisiones extremo a extremo propias de *TCP*.
- Se ha visto que la configuración óptima de los parámetros de los diferentes *buffers* varía según la calidad del enlace. Sería interesante incorporar un sistema que permita modificar *dinámicamente* estos atributos en función de las condiciones del tráfico o del medio inalámbrico.
- Está previsto también extender el estudio a un mayor número de flujos, lo que aumentaría la *ganancia de codificación* en función del número de segmentos que van a combinarse en los nodos intermedios.
- Se tendría que ampliar el alcance del algoritmo desarrollado para identificar nodos codificadores, para que tenga en cuenta otros aspectos: número de flujos, tiempo de vida, características del tráfico, movilidad de los nodos, etc. Además se debería hacer frente a la limitación del direccionamiento *unicast* para segmentos dirigidos a varios destinos, lo que reduce la validez de los paquetes codificados.

Conviene hacer hincapié en que todo el código que se ha desarrollado del protocolo *inter-flujo* está disponible en el siguiente repositorio público [Góme].

6

Mecanismos de Network Coding Intra-flujo

En este capítulo se analiza un esquema de codificación completamente diferente al presentado en el Capítulo anterior, basado en combinar la información de mensajes que pertenecen a un mismo flujo (disciplina conocida como *intra-flujo* o *intra-sesión*). Además, se buscará asimismo asegurar la sencillez de las operaciones de codificación/decodificación, mediante la utilización de un sistema *RLC*, donde la aleatoriedad que rige el proceso permitirá disminuir drásticamente cualquier tipo de control o gestión durante el intercambio de información. En pocas palabras, un nodo fuente generará de manera indefinida combinaciones lineales aleatorias de bloques de paquetes nativos hasta que el destino haya sido capaz de decodificarlos y recuperar la información original, cuando enviaría un mensaje de confirmación, notificando a la fuente la correcta recepción. Tras esto, se dará la transmisión del bloque por concluida, pasando al siguiente.

Por otro lado, en los últimos años se ha observado el interés de la comunidad científica por buscar una solución alternativa a *TCP* como protocolo de transporte fiable *de facto*, encontrando un buen número de soluciones que se apoyan en versiones modificadas de *UDP*. Por ejemplo, el protocolo *Reliable User Datagram Protocol (RUDP)* [Part90] incorpora algunas de las funcionalidades propias de *TCP* a *UDP*, como el reconocimiento de la información recibida, el control de flujo a través de una ventana deslizante o la retransmisión de mensajes perdidos, entre otras opciones. También es interesante destacar la aproximación de *Google* y su protocolo experimental *QUIC* [QUIC], que busca optimizar el rendimiento de *TCP* para aplicaciones *web* orientadas a la conexión, ofreciendo al mismo tiempo una seguridad comparable a la que se obtendría con el uso del protocolo *Transport Layer Security (TLS)* [Dier08]. En la propuesta de esta Tesis se combina el esquema *RLC* con el protocolo *UDP*, ya que gracias a las propiedades de corrección de errores proporcionadas por la codificación, se garantiza que toda la información original transmitida por la fuente será recuperada de manera ordenada en el destino.

En concreto, a lo largo de este capítulo se cubrirán los siguientes aspectos:

- Como punto de partida, se describirán las principales diferencias entre las técnicas de *NC* con dos de los mecanismos de codificación más conocidos: los *códigos bloque* y los

Digital Fountain, identificando sus principales ventajas, especialmente las que aporta la capa de inteligencia adicional, que permite utilizar los elementos intermedios con tareas más complejas que el mero reenvío de la información.

- Posteriormente se llevará a cabo una exhaustiva descripción del protocolo *NC intra-flujo* desarrollado en el marco del simulador `ns-3`, detallando sus elementos clave, como son los mecanismos de codificación, decodificación y recombinación de la información en los nodos intermedios. Además, se presentarán una serie de técnicas *cross-layer* que permitirán interactuar con diferentes entidades situadas en niveles inferiores, obteniendo un mayor rendimiento.

- Teniendo en cuenta que el análisis se lleva a cabo sobre enlaces inalámbricos, es interesante aprovechar la recepción simultánea de los mensajes en todos los nodos que se encuentran dentro de la región de cobertura del transmisor, de manera que la información podrá llegar al destino por diferentes vías. Además, esta característica *broadcast* podrá servir de base para el uso de *esquemas de encaminamiento oportunista* más avanzados, que permitan obtener una ganancia adicional sobre los mecanismos clásicos *store-and-forward*.

- Tras describir la operación del protocolo, se realizará una caracterización analítica del mismo, en función de diferentes parámetros, como el tamaño del bloque utilizado K, el orden q en un cuerpo de Galois extendido sobre una base binaria $GF(2^q)$ o el número de retransmisiones a nivel *IEEE 802.11* empleadas. Se llevará a cabo un estudio del rendimiento obtenido por diferentes librerías de código libre en `C++` que incluyen las operaciones necesarias para completar los procesos de codificación y decodificación.

- Finalmente, para validar el protocolo desarrollado, se ha llevado a cabo un profundo análisis sistemático sobre el simulador `ns-3`, caracterizando el rendimiento de la solución propuesta, comparándola con el obtenido con *TCP*.

6.1 *Network Coding Vs.* Codificación fuente

Es importante, antes de pasar a su descripción, aclarar las principales diferencias entre el esquema de codificación propuesto en esta Tesis con otros mecanismos correctores de errores. Desde un punto de vista cronológico, los *códigos bloque* (lineales) [Pete72], propuestos originalmente durante la primera mitad del siglo XX, supusieron un importante avance en el campo de la recuperación frente a los errores de transmisión en canales con errores, donde soluciones como los códigos *Hamming* [Hamm50], *Low Density Parity Check (LDPC)* [Gall63] o *Reed-Solomon* [Reed60], entre otros, permiten localizar y recuperar un número finito de símbolos erróneos en un bloque. Apoyándose en el uso del álgebra sobre cuerpos finitos $GF(2^q)$, su comportamiento se puede definir a través de las matrices generadoras y de chequeo de paridad, de tal manera que la entrada de un bloque de k *bits* de

información produce un mensaje de salida de n *bits*, siendo $n-k$ la redundancia introducida por el esquema de codificación. La principal limitación de este tipo de códigos (junto con su capacidad correctora) es la complejidad inherente al proceso decodificador, ya que la multiplicación de matrices requerida es un proceso computacionalmente costoso; se pueden destacar propuestas que reducen dicho coste a través de diferentes técnicas de resolución de sistemas de ecuaciones, equivalentes a operaciones matriciales, como el algoritmo de Berlekamp-Massey [Mass69], que se basa en un módulo de registros de desplazamiento con realimentación lineal. Como ejemplos prácticos, los códigos bloque lineales son parte central de un numeroso conjunto de arquitecturas, como la memoria *Error-Correcting Code (ECC)* presente en los chips *Dynamic Random Access Memory (DRAM)*, la codificación de vídeo *Moving Picture Experts Group 2 (MPEG-2)* o las comunicaciones con las sondas espaciales *Voyager II* o *Galileo*.

Por otro lado, la principal característica de los códigos conocidos como *Digital Fountain* [Byer98] (también llamados códigos *rateless*) es que son capaces de generar una secuencia ilimitada de mensajes codificados a partir de un conjunto de K símbolos originales almacenados en el nodo fuente. Una diferencia importante con los códigos bloque es que cualquier subconjunto de símbolos codificados mantiene la misma longitud que la de los símbolos originales. Con este esquema de codificación, un receptor deberá recibir correctamente K' símbolos codificados para poder recuperar la información original, considerándose que el código es óptimo cuando $K' = K$. Dentro de este grupo se pueden encontrar los códigos *Luby Transform (LT)*, propuestos por Michael Luby en 1998 [Luby02].Fueron los primeros en su categoría, presentando una solución cercana a la óptima, basada en la utilización de grafos bipartitos y en la operación *XOR* para minimizar el coste computacional en las operaciones de codificación/decodificación. La siguiente generación de este segundo grupo fue desarrollada por el matemático iraní Amin Shokrollahi en [Shok06], donde introduce los códigos *Rapid Tornado (RAPTOR)*, evolución de los códigos *Tornado* (también creados por Luby en [Luby97]), suponiendo el primer esquema de codificación con complejidad lineal $\mathcal{O}(K)$, tanto en codificación como en decodificación. Además, logra reducir la redundancia hasta obtener valores cercanos a las K recepciones *correctas* para poder llevar a cabo el proceso de decodificación. Para ello, se divide el esquema de codificación en dos etapas: en primer lugar se lleva a cabo una *pre-codificación*, basada en uno o vario módulos de códigos correctores de "tasa fija", seguida de una codificación *LT*. Como muestra de su relevancia en el mundo de las comunicaciones se puede decir que la empresa formada por Michael Luby, *Digital Fountain*, fue adquirida en el año 2009 [QualDF] por *Qualcomm*. Entre sus aplicaciones más destacadas se encuentran la difusión de vídeo en dispositivos móviles, *Digital Video Broadcasting - Handheld (DVB-H)*, o en servicios comerciales de televisión sobre una red IP, *Digital Video Broadcasting - Internet Protocol TV (DVB-IPTV)*, formando parte ya de numerosos estándares del *3rd Generation Partnership Project (3GPP)*.

A pesar de sus semejanzas, siendo la más importante que todos ellos son esquemas concebidos para enmascarar la aparición de errores puntuales en el intercambio de informa-

Tabla 6.1: Diferencias entre *Network Coding* y otras técnicas de codificación

Funcionalidad	***Códigos bloque***	**Digital Fountain**	**Network Coding**
Correción de errores	✓	✓	✓
Lleva el código en cada paquete	✗	✗	✓
Operación distribuida	✗	✗	✓
Decodificación sin utilizar paquetes codificados	✗	✗	✓
Capacidad para generar códigos a partir de paquetes nativos o codificados (recombinación)	✗	✗	✓
Recodificación sin necesidad de decodificar (añade redundancia incremental)	✗	✗	✓
Codificar datos en una ventana deslizante	✗	✗	✓

ción, las técnicas de *NC* se desmarcan de las anteriores en un número elevado de aspectos. En primer lugar, el objetivo fundamental del *NC* es el de maximizar la utilización de recursos durante una transmisión, especialmente en lo que se refiere a la capacidad (ancho de banda), y no en incrementar la fiabilidad de las comunicaciones. Además, su uso se extiende a nuevos tipos de redes con requerimientos muy diversos, como redes *Delay Tolerant Networks (DTN)* o topologías *ad hoc* (e.g. sensores, vehiculares, submarinas, etc). En cualquier caso, la diferencia fundamental radica en que no se trata de un esquema extremo a extremo, de manera que los nodos intermedios tienen la capacidad de modificar la información de los mensajes al mismo tiempo que los reenvían. La Tabla 6.1 recoge las principales diferencias o mejoras que los mecanismos de *NC* aportan sobre los *códigos bloque* o *Digital Fountain*. La integración de los códigos generadores del mensaje cifrado en las cabeceras propias del protocolo[1], evita llevar a cabo ningún tipo de monitorización, ya que cada mensaje es auto-contenido. Además, la capacidad de los *routers* para cambiar el contenido de los mensajes antes de reenviarlos da lugar a una *operación distribuida*.

Las propiedades del álgebra empleado en las funciones de codificación/decodificación permitirán: *(1)* la capacidad de decodificar combinando indistintamente paquetes *nativos* y codificados, reduciendo el coste computacional; *(2)* gracias a las propiedades de linealidad de los cuerpos finitos de Galois, las recombinaciones (a partir de paquetes *nativos* o codificados) realizadas en los nodos intermedios serán, a su vez, combinación lineal de los paquetes originales, por lo que no será necesaria la decodificación en los nodos intermedios. *(3)* finalmente, aunque la versión presentada en esta Tesis utiliza bloques fijos de K paquetes, existen alternativas que buscan disminuir la complejidad de la operación de decodificación, bien a través de una decodificación dinámica, en la que se van extrayendo los paquetes nativos a medida que se construye dicha matriz, utilizando técnicas de eliminación Gaussiana para mantenerla en forma escalonada [Sund11], o mediante la utilización

[1] Aunque existen soluciones que buscan eliminar esta sobrecarga a través de la utilización de generadores de secuencias pseudo-aleatorias [Thom12; Alwi13] o la transmisión de *hashes* en lugar de los vectores de codificación [Adel09].

de *ventanas deslizantes* en lugar de bloques fijos, lo que permite simplificar las operaciones a realizar por el destino y reducir la sobrecarga correspondiente [Wu12a].

Antes de dar por finalizada esta comparativa, merece la pena destacar el intento por parte de ciertos grupos de investigación de combinar la operación de los diferentes esquemas, argumentando que el coste computacional de los esquemas *RLNC* presentan valores muy superiores a los que se consiguen a través de las técnicas *Digital Fountain*. A modo de ejemplo, en [Cham10] los autores utilizan códigos *LT* para codificar la información, habilitando una técnica adicional que permite "refinar" los mensajes a medida que atraviesan la red, reduciendo la complejidad de decodificación un 99 %, aunque a coste de incrementar la sobrecarga un 20 % y el tiempo de convergencia un 30 %.

6.2 Descripción del protocolo

En esta sección se describirá el protocolo *NC intra-flujo* diseñado e implementado para ofrecer un servicio fiable a través de la combinación de un esquema de codificación *RLC* y un protocolo de transporte no orientado a la conexión como *UDP*. Tomando como base trabajos como [Chac07] o [Frag06], donde se definen los pasos a seguir a la hora de codificar y decodificar los paquetes, se reutilizará la clase raíz presentada en la Sección 4.2 para adaptar un módulo correspondiente al protocolo *intra-flujo*. Recuérdese que se introducía una capa intermedia entre los niveles de red y de transporte, alterando el flujo por defecto de los paquetes de datos.

En concreto, la descripción se estructura en cuatro partes, presentando los procesos de codificación, decodificación y recodificación, para finalizar con el formato de la cabecera utilizada para compartir la información con entidades *NC* pares.

6.2.1. Proceso de codificación

Uno de los elementos clave de cualquier esquema de codificación *intra-flujo* (o codificación en fuente) es el criterio a la hora de codificar la información que un nodo tiene almacenada. El proceso que se sigue, únicamente para los paquetes de datos, aparece descrito a continuación.

- Como ya se ha mencionado anteriormente, el transmisor trabajará con bloques de tamaño fijo (así, el parámetro K define el número de paquetes que forman los bloques) como unidad mínima de codificación. Para ello, se implementará un *buffer de codifica-*

ción (en forma de cola *FIFO*), de capacidad ilimitada[2], que almacene la información de forma indefinida hasta que ésta sea confirmada por el nodo destino.

- En el instante en el que llegue el paquete K-ésimo a la capa *NC*, dará comienzo la fase de codificación. Como puede verse en la Figura 6.1, los paquetes p'_i serán el resultado de combinar linealmente los K paquetes que forman el bloque utilizando los coeficientes c_i, siguiendo la expresión $p'_i = \sum_{i=0}^{K-1} c_i \cdot p_i$.

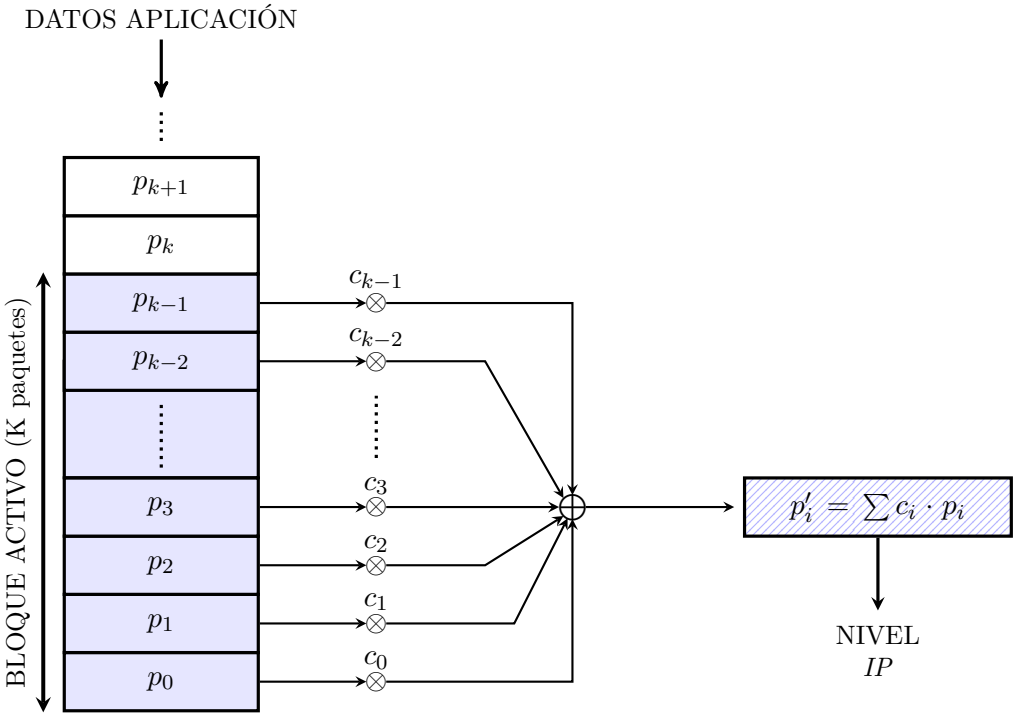


Figura 6.1: Esquema de codificación utilizado por el nodo transmisor en el protocolo *NC intra-flujo*

Como se ha discutido en la Sección 4.1, aunque en los primeros trabajos se utilizaba algún tipo de criterio a la hora de seleccionar estos coeficientes [Li03], se comprobó posteriormente [Ho03b] que su generación aleatoria puede ofrecer un rendimiento óptimo. Con respecto a los coeficientes c_i, serán elegidos uniforme e independientemente a través de una variable aleatoria perteneciente a un cuerpo de Galois de orden $GF(2^q)$. Gracias a sus propiedades lineales, cualquier operación realizada sobre uno de estos cuerpos tendrá como resultado un valor que se encuentra dentro del mismo, por lo que la longitud del del paquete codificado p_i coincidirá con la mayor de los K

[2]En dispositivos reales deberá asignarse un valor máximo, sobre todo en aquellos elementos con escasa capacidad (i.e. sensores). Otra opción que se ha barajado es la de acceder a la zona de memoria correspondiente a los datos de aplicación y gestionar la codificación directamente desde allí, evitando duplicar la información en la capa *NC*.

paquetes *nativos*[3]. En concreto, la utilización de un cuerpo binario base $GF(2^q)$, con $q = 1$ se traduce en las operaciones a nivel de *bit* *XOR* para la suma y *AND* para las multiplicaciones, por lo que el coste de la generación de los paquetes codificados será mínimo. Por otro lado, en el momento en el que se empleen cuerpos de mayor orden, $GF(2^q)$, con $q > 1$, los datos serán procesados formando símbolos de q *bits*, multiplicándose uno a uno con el coeficiente c_i elegido para el paquete en cuestión. En este caso, la suma se sigue correspondiendo con la operación *XOR*, pero la multiplicación depende de la implementación utilizada: en algunos casos, se resolverá a través de productos y divisiones modulares [FFLAS], en otros, con registros de desplazamiento y operadores *XOR*, de manera más habitual, empleando operaciones polinómicas [MatGF].

Expresado en forma matricial, el proceso de codificación quedaría definido como se muestra en la Ecuación (6.1), donde un paquete codificado p'_i es el resultado de la combinación lineal $p'_i = c_{i,0} \cdot p_0 + c_{i,1} \cdot p_1 + \ldots + c_{i,K-1} \cdot p_{K-1}$. Desde un punto de vista de la operación del protocolo, se generan los K coeficientes necesarios para construir el paquete codificado p'_i codificando el bloque activo. Así, se agrupan en un vector fila de K elementos, $\overrightarrow{c_i} = \{c_{i,0}, c_{i,1} \ldots c_{i,K-1}\}$, en la matriz de coeficientes (definida como $\mathcal{C}$ a partir de este momento).

$$\begin{bmatrix} p'_0 \\ p'_1 \\ p'_2 \\ \vdots \\ p'_{K-1} \end{bmatrix} = \begin{bmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,K-1} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,K-1} \\ c_{2,0} & c_{2,1} & \cdots & c_{2,K-1} \\ \vdots & \vdots & \ddots & \vdots \\ c_{K-1,0} & c_{K-1,1} & \cdots & c_{K-1,K-1} \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{K-1} \end{bmatrix} \tag{6.1}$$

- Otro punto a destacar es que cada uno de los flujos presentes en el sistema serán tratados de manera independiente. Para ello, se volverá a hacer uso del *hash* (a partir de las direcciones *IP* y de los puertos *UDP*), gestionando dinámicamente un *buffer de codificación* por cada flujo activo en la red. Por otro lado, al finalizar el intercambio de datos, esto es, cuando dicha cola esté vacía y no se reciban más datos de los niveles superiores, se procederá a eliminar el contenedor.

- Tras generar el paquete codificado y añadir la cabecera del protocolo *NC* (que será explicada más adelante), se procederá al envío del mensaje codificado a los niveles inferiores. Para garantizar que la tasa de transmisión sea la apropiada, se han barajado diferentes opciones, habiendo optado finalmente por una operación *cross-layer*: la entidad *NC* estará conectada con el nivel físico *IEEE 802.11*, que le avisará cada vez que transmita un paquete de un flujo concreto. De este modo se asegura una tasa, garantizando la presencia de, al menos, un paquete codificado esperando a ser transmitido y, además, se consigue evitar la saturación de este contenedor.

[3]En el caso de que presenten diferentes longitudes, los paquetes más cortos se rellenarán con ceros hasta igualar el tamaño del más largo.

- Cuando el receptor sea capaz de decodificar toda la información del bloque, enviará, de manera inmediata, un paquete que informa a la fuente (*ACK*) que no necesita recibir más paquetes codificados. Al recibirlo, el contenido del bloque se dará por transmitido y podrá ser eliminado directamente del *buffer de codificación*; de este modo, si hay almacenados otros K paquetes, se procederá a repetir el proceso con el siguiente bloque. Además, en el caso de que haya varios flujos activos en la red, podrá darse el caso de que el *buffer* del nivel de enlace contenga paquetes codificados pertenecientes al bloque ya confirmado, esperando su turno para ser transmitidos. Para evitar su envío, se ha implementado una segunda solución *cross-layer*, que elimina de manera selectiva todos aquellos paquetes de datos codificados del flujo correspondiente al *ACK* recibido.

- Finalmente, habrá situaciones en las que el *buffer de codificación* no reciba los paquetes necesarios para completar un bloque; en este caso, si no se hiciera nada, la información quedaría retenida indefinidamente, bloqueando la salida de datos. Como solución, se ha incorporado un temporizador, CB_{TO}, que se reinicia cada vez que concluye la transmisión de un bloque (al comienzo, se activará con la llegada del primer paquete procedente de la capa de aplicación). Al expirar, se asumirá que no se va a poder llenar un bloque completo, iniciando el proceso de codificación con los paquetes que se encuentren en ese momento en el *buffer de codificación*[4].

6.2.2. Proceso de decodificación

En el otro extremo, los destinos reciben la información codificada y tratan de descifrar el contenido original del mensaje. En este caso, el proceso a realizar contempla las fases descritas a continuación:

- En primer lugar, el nodo almacenará temporalmente la información hasta que pueda decodificarla y enviarla a la capa de aplicación. Para ello, asumiendo que cada bloque tiene un tamaño igual a K, la entidad *NC* implementa un *buffer de decodificación* (nuevamente en forma de cola *FIFO*) capaz de albergar hasta K paquetes.

- El elemento clave para la decodificación es la matriz de coeficientes $\mathcal{C}$ (ver Ecuación 6.1), cuya dimensión será siempre $K \times K$. En ella se irán almacenando (fila a fila) todos los vectores de coeficientes $\overrightarrow{c_i}$ que proporcionen información "novedosa" (es decir, que sean linealmente independientes a los ya almacenados). El rango de $\mathcal{C}$ tendrá una gran relevancia, y se almacenará en una variable interna del sistema.

- La información en los nodos receptores será la matriz de coeficientes $\mathcal{C}$ y los paquetes codificados p'_i, con $i \in [0, K-1]$. Por tanto, a partir de la Ecuación 6.1 se pueden

[4]En el supuesto de que hubiese un único paquete, se transmitiría directamente sin codificar.

calcular los paquetes originales, a partir de los paquetes codificados y de la *inversa* de $\mathcal{C}$, tal y como se muestra en la Ecuación 6.2.

$$\begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{K-1} \end{bmatrix} = \begin{bmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,K-1} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,K-1} \\ c_{2,0} & c_{2,1} & \cdots & c_{2,K-1} \\ \vdots & \vdots & \ddots & \vdots \\ c_{K-1,0} & c_{K-1,1} & \cdots & c_{K-1,K-1} \end{bmatrix}^{-1} \cdot \begin{bmatrix} p'_0 \\ p'_1 \\ p'_2 \\ \vdots \\ p'_{K-1} \end{bmatrix} \tag{6.2}$$

Como se verá más adelante, el tiempo necesario para calcular el rango y la inversa de estas matrices puede ser, de hecho, el cuello de botella del sistema, con el impacto correspondiente en el rendimiento.

- Para gestionar varios flujos activos (e independientes), se mantendrá una dupla *buffer de decodificación-matriz de coeficientes* para cada conexión.

- Tras la recepción de un paquete codificado p'_i, la entidad *NC* chequea si su contenido es útil para la decodificación del bloque. Para ello, comprueba si su vector asociado $\overrightarrow{c_i}$ es linealmente dependiente de los que ya estaban en $\mathcal{C}$, calculando su rango (a través del *método de eliminación de Gauss-Jordan*). Si al insertar $\overrightarrow{c_i}$ en la fila *r-ésima* de $\mathcal{C}$ (siendo r el rango actual de ésta) se consigue que su rango aumente en una unidad, significará que la nueva fila es linealmente independiente y que el paquete codificado aporta información "nueva" al decodificador. Finalmente, se guardará el paquete codificado en el *buffer de decodificación*. En caso contrario (el vector $\overrightarrow{c_i}$ es combinación lineal del resto de filas), se considerará que la información no es útil, eliminando la fila de la matriz y descartando el paquete.

- Este proceso se repetirá hasta que la matriz sea invertible, esto es, su rango sea igual a K. Si la codificación fuese perfecta, el número de recepciones necesarias serían K paquetes, donde todos ellos son linealmente independientes de los anteriores. No obstante, en una situación real, es probable que se alguno de los vectores recibidos sea combinación lineal del resto, requiriendo un total de $K + \delta$ recepciones, siendo δ el número de paquetes en exceso enviados por la fuente. En la Sección 6.4 se calculará analíticamente su valor medio, en función del tamaño del bloque K y del orden del cuerpo $GF(2^q)$.

- Al multiplicar la fila *i-ésima* de la matriz inversa $\mathcal{C}^{-1}$ con los K paquetes del *buffer de decodificación*, $p_i = \sum_{i=0}^{K-1} c_i^{-1} \cdot p'_i$, se obtendrá el paquete p_i que envió originalmente el nodo fuente. Así, a través del producto de $\mathcal{C}$ con el *buffer*, se recuperarán los K mensajes *nativos*, que serán enviados, de manera ordenada, a los niveles superiores.

- Como paso final, se enviará a la fuente un mensaje de reconocimiento, informando que se ha podido recuperar el bloque con éxito. En paralelo, se reinicia la matriz de coeficientes y se borra el contenido del *buffer de decodificación*, manteniéndose a la

espera de paquetes codificados correspondientes al siguiente bloque. En el supuesto de que este reconocimiento se pierda antes de alcanzar al transmisor, éste continuaría enviando paquetes codificados de un bloque ya recuperado. Como respuesta a este problema, cuando cualquier nodo reciba un segmento "obsoleto", generará de manera inmediata un mensaje *ACK*.

El esquema *RLNC* cumple una propiedad básica de los códigos *rateless* o *Digital Fountain*, donde *cada paquete codificado transmitido pierde su significado individual*, pasando a transportar una fracción equivalente a $1/K$ de la información del bloque completo. Esto es, no importa el orden en la recepción de los mensajes, ni los que se pierdan por el camino, sino que será suficiente con recibir K paquetes cuyos coeficientes sean combinaciones linealmente independientes para recuperar el contenido original del bloque.

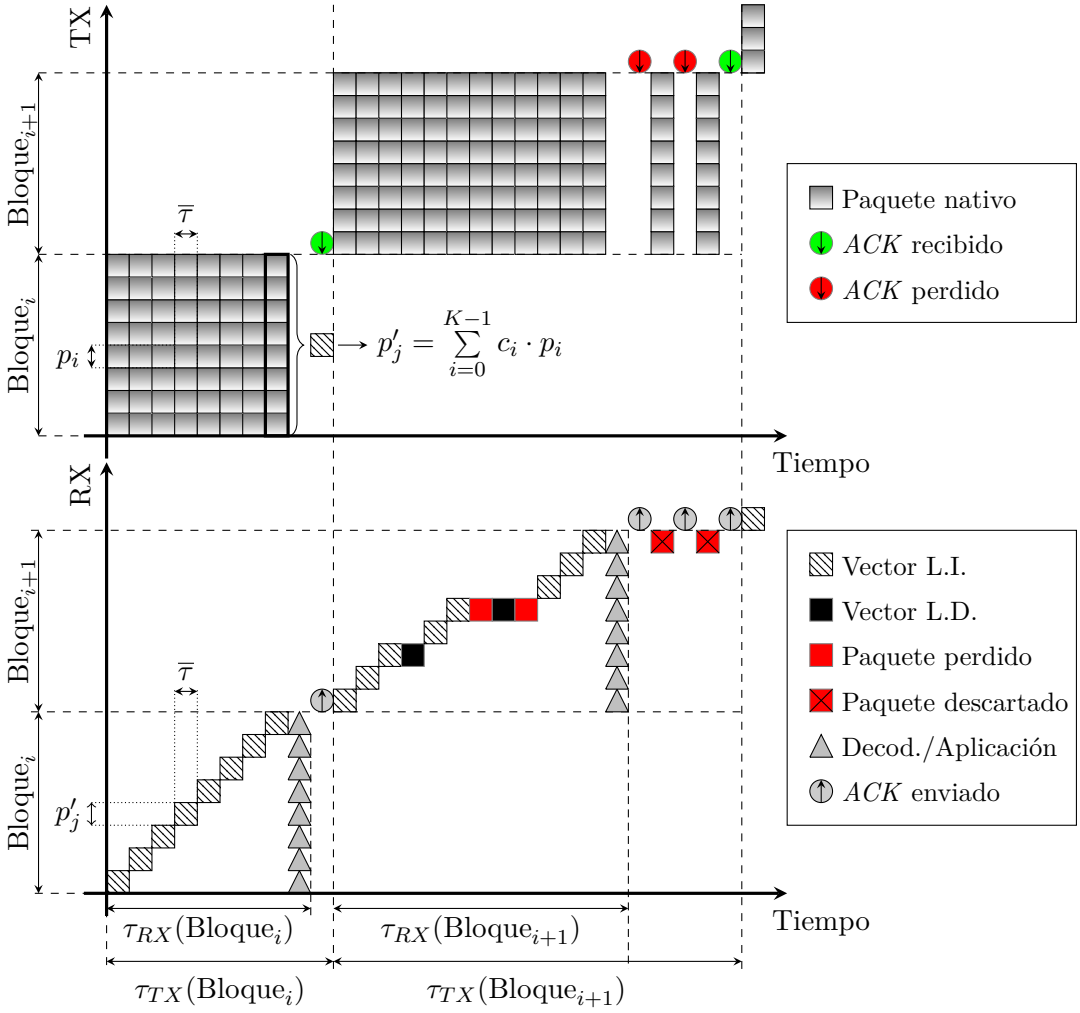


Figura 6.2: Secuencia de eventos discretos de una transmisión arbitraria *RLNC* desde el punto de vista del transmisor (arriba) y del receptor (abajo)

La Figura 6.2 muestra la secuencia de eventos que se producen durante el intercambio arbitrario de dos bloques consecutivos (con $K = 8$), utilizando como referencia temporal los eventos de transmisión y recepción. En el nodo transmisor (parte superior de la figura), se puede observar la combinación de los K paquetes correspondientes a un bloque, produciendo un mensaje codificado p'_j por evento, hasta que se recibe un *ACK*. Al mismo tiempo, el receptor (parte inferior) va recibiendo los paquetes y procesando sus respectivos coeficientes de codificación, incrementando el rango de la matriz a medida que se reciben vectores linealmente independientes. Una vez la matriz sea invertible, el nodo destino decodificará la información, enviando los K paquetes nativos a la aplicación; posteriormente, borrará el contenido del *buffer de decodificación*, reiniciará la matriz $\mathcal{C}$ y generará un mensaje de confirmación, pasando a esperar al siguiente bloque. En el primero de los bloques, todos los vectores recibidos incrementan el rango de la matriz y, por lo tanto, K (8) recepciones han sido suficientes para recuperar su contenido (caso ideal); sin embargo, el segundo bloque pone de manifiesto situaciones en las que se reciben vectores linealmente dependientes o los mensajes (tanto de datos como *ACKs*) no alcanzan correctamente al destino. En estos casos, el número de transmisiones deberá ser superior ($K + \delta$), con el consiguiente incremento en el tiempo requerido para decodificar el bloque. Resulta interesante destacar el comportamiento del receptor al detectar que la fuente esta "desactualizada", descartando inmediatamente cualquier paquete (sin procesarlo) y enviando un *ACK*, para indicar a la fuente que puede comenzar con la transmisión de un nuevo bloque.

6.2.3. Recodificación en nodos intermedios

Posiblemente, el factor más determinante de las soluciones *NC* es que los nodos intermedios tengan un papel más activo a la hora de reenviar la información, pasando de un paradigma clásico *store-and-forward* a uno nuevo, conocido como *compute-and-forward*.

Esta característica es la que lo diferencia de la codificación *Digital Fountain*, cuyo enfoque es fundamentalmente extremo a extremo, y los códigos generados por la fuente no pueden ser alterados por los nodos intermedios. Por el contrario, los esquemas *RLNC* pueden modificar el código indefinidamente en cualquier punto de la red, lo que podría tener un gran impacto en el rendimiento. Considérese, por ejemplo, una red formada por la concatenación de K nodos, como se ve en la Figura 6.3.

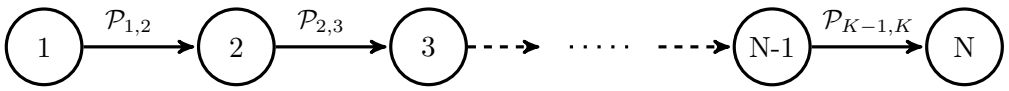


Figura 6.3: Topología *en línea* que ilustra las ventajas de recodificar en los nodos intermedios

Para simplificar, se asume que cada nodo transmite un paquete por unidad de tiempo, y que en cada enlace se pierde un paquete con una probabilidad $\mathcal{P}_{i,i+1} = \mathcal{P}_0$, con i=[1,..., K-1]. Con estas condiciones, y en un esquema tradicional, el *throughput* percibido en el

receptor sería $(1-\mathcal{P}_0)^K$ paquetes por *slot* de tiempo, ya que para que un paquete alcance el destino, deberá atravesar correctamente todos los enlaces. Sin embargo, si los *routers* tuviesen la capacidad de modificar las combinaciones lineales de los paquetes, y asumiendo que los nodos cuentan con la información suficiente como para poder generar sus propios mensajes codificados, el rendimiento observado en recepción sería de $(1-\mathcal{P}_0)$.

Sin embargo, el razonamiento anterior no tiene en cuenta la probabilidad de que los vectores de coeficientes de los paquetes recodificados sean combinación lineal de los anteriores y deban ser por tanto descartados.

Para cuantificar las diferencias en el rendimiento de ambas alternativas, el módulo implementado da la posibilidad de elegir la funcionalidad de los nodos intermedios:

1. Simplemente reenviarán la información que reciban, bien directamente o a través de una escucha promiscua. Se utilizará la notación *Random Linear Source Coding (RLSC)* para referirse a este esquema.
2. Siguiendo la filosofía de *NC*, los nodos intermedios recombinarán libremente los paquetes que los atraviesen. A esta alternativa se la denominará *RLNC*.

A continuación se describen los procesos seguidos por los nodos intermedios.

- En primer lugar, un nodo intermedio almacenará los paquetes recibidos de manera temporal, manteniendo un *buffer de recodificación* y una *matriz de coeficientes* para cada flujo activo que lo atraviese (solamente en un esquema *RLNC*).
- Tras la recepción de un paquete codificado p'_j, la entidad *NC* sigue el mismo procesado que un nodo receptor: calcula el rango de $\mathcal{C}$ para determinar si el paquete transporta información "nueva", guardándolo en el *buffer* y actualizando la matriz si así fuera o descartándolo en caso contrario. En el momento en el que la matriz sea de rango completo, se dejarán de procesar los mensajes recibidos, descartándolos directamente. Cuando detecte el paso de un *ACK*, limpiará sus contenedores (matriz y *buffer*[5]), pasando al siguiente bloque.
- A la hora de generar el paquete de salida, el nodo intermedio puede actuar de dos maneras distintas, en función del número de elementos que tenga almacenados: en la primera recepción no nula del bloque, cuando la matriz se encuentre aún vacía, el nodo no dispone de la información necesaria para generar nuevas combinaciones lineales, por lo que reenviará directamente su contenido de vuelta a la capa *IP*; en un caso más general (con más de un paquete en el *buffer*), la entidad *NC* generará un

[5]Se conservará la misma solución *cross-layer* que se describió en el proceso de codificación, donde, además de borrar el contenedor de la capa *NC*, se enviará una señal al nivel de enlace para que elimine selectivamente los paquetes correspondiente al flujo que acaba de ser confirmado.

nuevo mensaje a partir de la recodificación de los r elementos que se han guardado hasta el momento, siguiendo la expresión $p''_k = \sum_{j=0}^{r-1} d_j \cdot p'_j$. En el momento en el que se llene la matriz, se continuarán generando paquetes recodificados a partir de los K originales hasta que se reciba un *ACK*, de manera que el nodo puede generar su propia información sin necesidad de recibirla con anterioridad.

- Merece la pena aclarar que *las combinaciones lineales de paquetes codificados son combinaciones lineales de los paquetes nativos*. Como se ha visto en el punto anterior, un paquete codificado p''_k se puede expresar como $p''_k = \sum_{j=0}^{r-1} d_j \cdot p'_j$, con $r \leq K$, siendo d_j los coeficientes del cuerpo finito $GF(2^q)$ escogidos aleatoriamente en los nodos intermedios. Dentro de esta expresión, p'_j se puede representar como $p'_j = \sum_{i=0}^{K-1} c_i \cdot p_i$, siendo p_i los K paquetes originales que componen el bloque. Como resultado, descomponiendo p'_j en la primera expresión se obtiene la relación entre paquete recodificado y paquete nativo, tal y como muestra la Ecuación 6.3.

$$p''_k = \sum_{j=0}^{r-1} \left(d_j \cdot \sum_{i=0}^{K-1} c_i \cdot p_i \right) = \sum_{i=0}^{K-1} \left(\sum_{j=0}^{r-1} d_j \cdot c_i \cdot p_i \right) \tag{6.3}$$

  Expresado en formato matricial, el vector de coeficientes resultante, $\overrightarrow{c'_k} = \{c'_{k,0}, c'_{k,1}, ..., c'_{k,K-1}\}$, se obtiene a partir de la multiplicación que aparece en la Ecuación 6.4. En realidad, la operación de multiplicación se aplica al vector generado en la capa *NC*, d_j, y a los paquetes que se encuentren dentro del *buffer de recodificación*.

$$\overrightarrow{c'_k} = \begin{bmatrix} c'_{k,0} & \cdots & c'_{k,K-1} \end{bmatrix} = \begin{bmatrix} d_0 & \cdots & d_{r-1} \end{bmatrix} \cdot \begin{bmatrix} c_{0,0} & \cdots & c_{0,K-1} \\ c_{1,0} & \cdots & c_{1,K-1} \\ c_{2,0} & \cdots & c_{2,K-1} \\ \vdots & \ddots & \vdots \\ c_{r-1,0} & \cdots & c_{r-1,K-1} \end{bmatrix} \tag{6.4}$$

- A la hora de enviar las nuevas combinaciones generadas por el esquema *RLNC*, se volverá a hacer uso de los mecanismos *cross-layer* descritos en el esquema de transmisión, para adoptar la tasa de envío de paquetes de manera dinámica.

- En el supuesto caso en que los nodos intermedios hayan recibido el correspondiente *ACK*, pero los transmisores mantengan sus transmisiones en un bloque anterior, los *routers* podrán generar ellos mismos nuevos mensajes de *ACK*, informando a la fuente que se encuentra transmitiendo un bloque ya recibido.

- Este protocolo se aprovecha del *modo promiscuo* para procesar todo el tráfico de datos que perciben los nodos. Habrá que tener especial cuidado a la hora de reenviar los mensajes, evitando inundar la red. Para los flujos de datos se optará por una solución

probabilística, mientras que para los *ACKs* se seguirá un patrón más clásico, en el que sólo los nodos que pertenezcan a la ruta más corta podrán retransmitirlos.

Siguiendo el esquema utilizado para ilustrar los procesos de codificación/decodificación, la Figura 6.4 representa la secuencia de estados para la transmisión de dos bloques cualquiera por parte de un nodo intermedio. Se distinguen dos supuestos, según se emplee la alternativa *RLSC* (figura central) o *RLNC* (figura inferior); por su parte, la parte superior representa los eventos producidos en el momento de recibir los paquetes codificados, donde se determinará si son linealmente dependientes de los ya guardados. Hay que destacar que en un escenario real, en el que la calidad de los enlaces es variable e impredecible, con errores aleatorios durante las transmisiones, la probabilidad de que las *matrices de coeficientes* de los nodos se encuentren "desfasadas" es elevada.

Con respecto a la solución *RLSC*, es interesante observar lo que sucede cuando se reciben paquetes con vectores linealmente dependientes, que no aportan información nueva (cuadrado negro de la figura superior). Como no hay inteligencia adicional en el nivel *NC*, este mismo vector será reenviado, por lo que será ciertamente factible que tampoco sea útil en el destino. En estos casos, el canal se habrá ocupado con transmisiones innecesarias. Además, en aquellas situaciones en las que se pierda un paquete antes de alcanzar a un nodo intermedio, éste no tendrá información que enviar y se mantendrá inactivo, desaprovechando el tiempo hasta la llegada del siguiente paquete codificado. Por el contrario, con la alternativa *RLNC*, a partir de que el *buffer de recodificación* tenga un mínimo de dos paquetes, escogerá nuevos coeficientes y generará un nuevo mensaje p''_k. Así, tendrá un comportamiento no correlado a la tasa de llegada de la fuente, creando nuevas combinaciones lineales al ritmo determinado por el canal inalámbrico. Un factor a tener en cuenta es que la cantidad de información de la que se dispone en los nodos intermedios será muy inferior a la de la fuente[6], ya que *al no decodificarse los paquetes en los nodos intermedios*, solamente podrá utilizar un coeficiente nuevo por paquete guardado en el *buffer*.

Antes de dar por concluida la descripción de todos los procesos que intervienen en el codificación/recodificación/decodificación de la información, conviene analizar el espacio que se necesita para poder llevar a cabo todas estas operaciones. La Figura 6.5 muestra los principales contenedores que se tienen que implementar en los distintos nodos: el transmisor, que recibe la información de las capas superiores, tiene que poder almacenar (sin restricción) toda la información hasta que el destino confirme su correcta decodificación, bloque a bloque, por lo que idealmente tendría una capacidad infinita, aunque realmente dependerá del tipo de servicio (e.g. tasa de envío, densidad de los datos generados, etc.). Para evitar problemas, en el de esta Tesis se asume que estos *buffers de codificación* tienen capacidad ilimitada, donde además se gestiona uno por cada flujo activo. Por su parte, los nodos intermedios recodificadores y los destinos utilizan *buffers* de tamaño fijo, capaces

[6]En concreto, cualquier recombinación de r paquetes codificados (con $r < K$) sólo podrá generar un total de r vectores linealmente independientes, por lo que en el caso de cortarse el flujo de datos de la fuente, *nunca se podrá completar el bloque*.

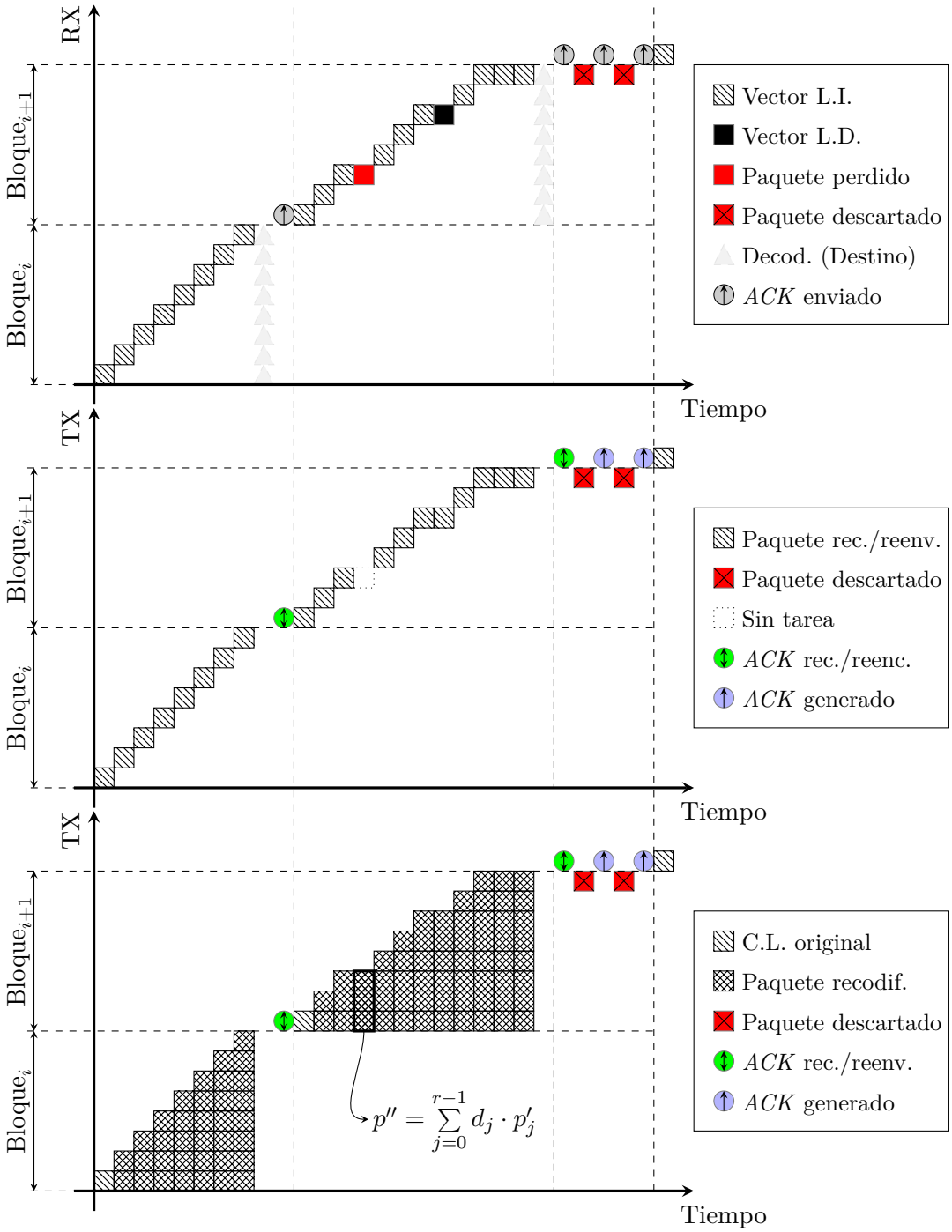


Figura 6.4: Secuencia de eventos discretos de una transmisión *RLC* desde el punto de vista de un nodo intermedio. En concreto, se representan los eventos de recepción (arriba), transmisión *RLSC* (medio) y *RLNC* (abajo)

de albergar K paquetes por flujo, con lo que se necesitaría memoria para almacenar un total de $N \times K$ paquetes, siendo N el total de conexiones en las que interviene el nodo en cuestión. Solamente será necesario guardar los mensajes hasta que se consigan llenar las respectivas matrices. En lo que respecta a estas *matrices de codificación/recodificación*, serán de dimensión fija $K \times K$. Como se reserva el espacio en función del número de flujos, el tamaño correspondiente a estas matrices será de $N \times K \times K$ octetos; los elementos de las mismas serán los coeficientes pertenecientes a los cuerpos finitos $GF(2^q)$, cuyo valor máximo se ha limitado a $GF(256)$, por lo que será suficiente con un *byte* para almacenar cada uno de ellos.

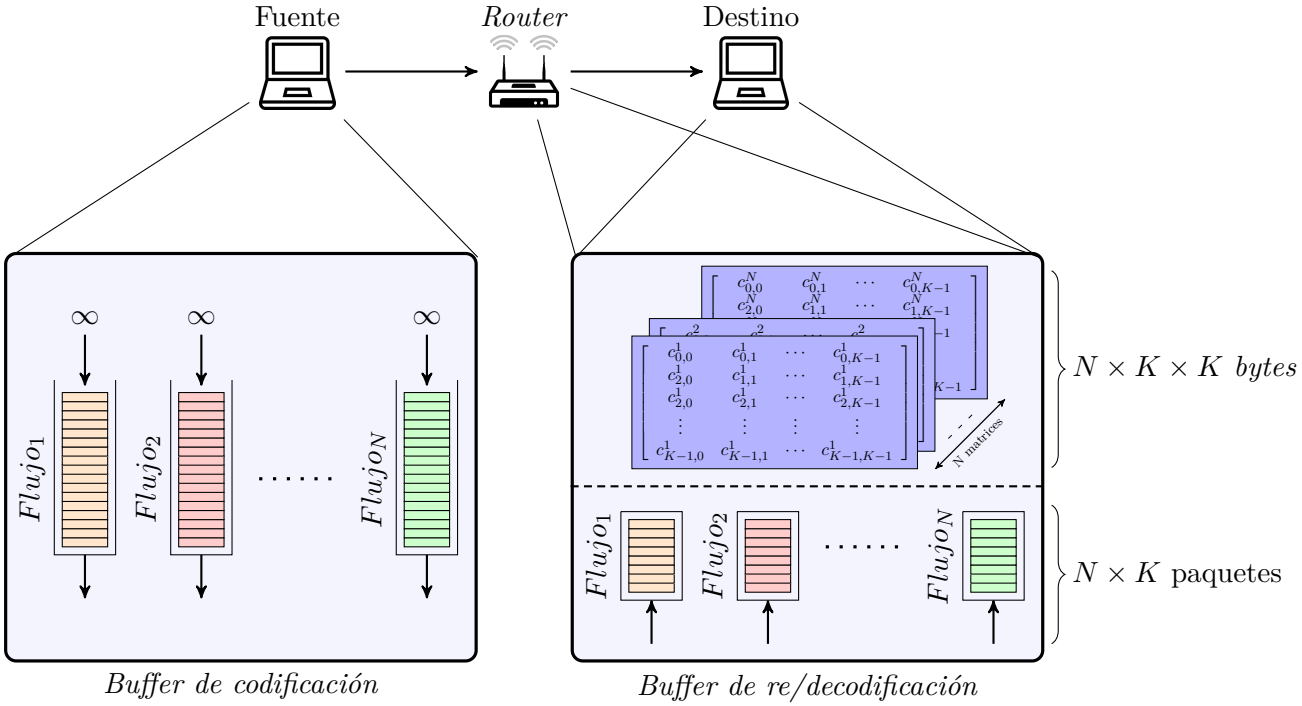


Figura 6.5: Uso de los *buffers* implementados en la entidad *NC* en función de la funcionalidad de los nodos

6.2.4. Formato de la cabecera

La Figura 6.6 muestra el formato de la cabecera que transporta la información correspondiente al protocolo *intra-flujo*. Se pueden apreciar dos partes diferenciadas: la primera, con una longitud fija de 9 *bytes*, contiene los siguientes campos: el tipo de mensaje, el tamaño del bloque utilizado por la fuente, el orden del cuerpo finito extendido, el contador del bloque activo (*approx* número de secuencia) y los puertos *UDP* origen y destino, ya que la cabecera *UDP* forma parte del *payload* en el nivel *NC* y viajará por tanto codificada.

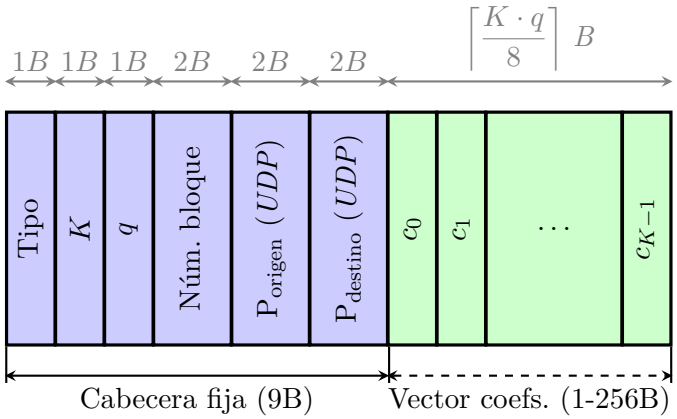


Figura 6.6: Formato de cabecera utilizado en el protocolo *NC intra-flujo*

En una segunda parte, de *longitud variable*, se incluye el vector de coeficientes ($\overrightarrow{c}$) con el que se han generado los paquetes codificados. Su tamaño será proporcional a los valores escogidos para el tamaño del bloque (K) y el orden del cuerpo de Galois $GF(2^q)$, oscilando entre 1 octeto, cuando $K = 2$ y $q = 1$, y 256 *bytes* ($K = 256$ y $q = 8$)[7]. Esta gran variabilidad de la longitud, representada en la Figura 6.7, tendrá un impacto notable en el rendimiento del sistema, ya que la sobrecarga introducida puede limitar el porcentaje de información útil por paquete transmistido.

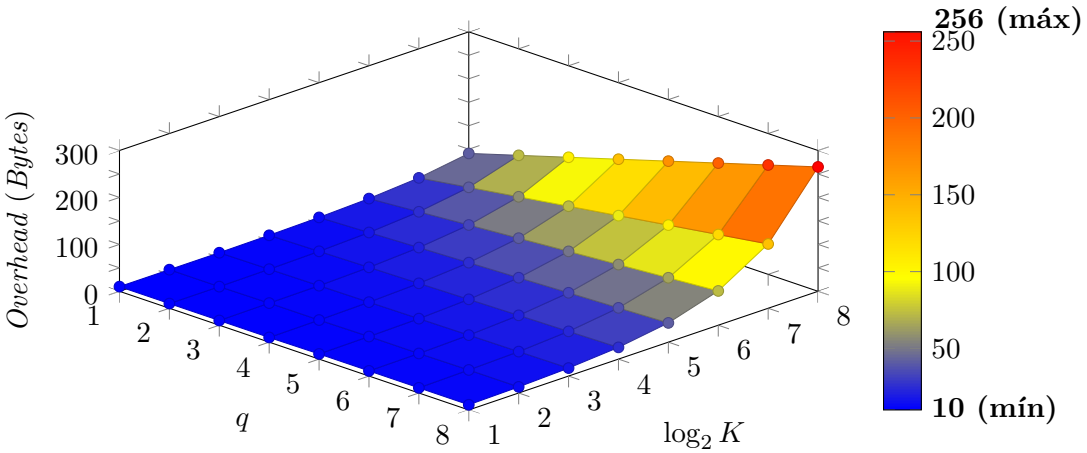


Figura 6.7: Sobrecarga introducida por la cabecera del protocolo en función de los valores del tamaño del bloque (K) y del orden del cuerpo extendido $GF(2^q)$

A continuación se detallarán los campos que componen la cabecera de nivel *NC*:

[7]El tamaño del bloque se limita para que su valor quepa dentro de un único octeto y no incremente en exceso el tamaño de la cabecera. Además, bloques mayores implican matrices más grandes, lo que da lugar a unos tiempos de procesado para calcular rangos e inversas no asumibles.

- El campo **Tipo** identifica la función del paquete, asignando un valor 0 para un mensaje de datos, 1 para un reconocimiento normal extremo a extremo ó 2 para un *ACK* explícito, generado tras la detección de un paquete fuera de bloque. Como se ha reservado un octeto entero, aún queda espacio para incorporar de nuevos tipos, como se discutirá posteriormente.
- El parámetro ***K*** indica el tamaño del bloque o, lo que es lo mismo, el número de paquetes nativos que lo componen. Será esencial encontrar un equilibrio a la hora de escoger este valor, ya que el manejo de bloques grandes implica una gran sobrecarga y un mayor tiempo de procesado (matrices de mayor dimensión), pero, al mismo tiempo, una mayor probabilidad de que los paquetes codificados sean linealmente independientes a los anteriores; además, cuanto mayor sea el bloque, menor tiempo se invertirá en transmitir reconocimientos.
- La variable ***q*** determina la cantidad de *bits* con los que se codifica cada coeficiente, y tiene una relación directa con el orden del cuerpo $GF(2^q)$ utilizado para generar dichos coeficientes aleatorios, ya que podrán elegirse dentro del rango $[0, 2^q - 1]$. El valor de q influirá significativamente en las prestaciones finales, en un modo similar a K: cuando es pequeño ($q = 1$) los posibles valores para cada coeficiente son limitados, en este caso concreto serían "0" y "1", por lo que habrá menos libertad a la hora de generar combinaciones lineales, siendo más difícil obtener vectores linealmente independientes. Sin embargo, para valores más altos (con mejores prestaciones en términos de entropía), se incrementa el tiempo de procesado, por lo que el rendimiento podría verse perjudicado (la caracterización del comportamiento de diferentes librerías se estudiará con un mayor nivel de detalle en la Sección 6.4.3).
- El **número de bloque**, indica el bloque al que pertenece el paquete, permitiendo así a los nodos procesarlo o descartarlo en el caso de que llegue fuera de secuencia. Además, servirá para que un destino confirme la correcta decodificación del mismo.
- Al final de la parte fija de la cabecera se transmiten los puertos origen y destino (extraídos de la cabecera *UDP*). La entidad *NC* los utiliza para generar los *hashes* que permitirán procesar con mayor agilidad la identidad de los diferentes flujos que circulan por la red.
- Por último, se incluye el vector aleatorio de coeficientes $\overrightarrow{c_i}$. Este campo, de longitud variable, se extrae de la cabecera y se introduce en las diferentes matrices $\mathcal{C}$, para comprobar si se trata de una combinación lineal de los paquetes previos o si, por el contrario, contiene información útil.

Cabe destacar que todos los campos descritos hasta este momento son necesarios en los paquetes de datos. Por el contrario, los reconocimientos sólo utilizarán el campo *Tipo* y el *número de bloque* aunque, para mantener un formato único, rellenarán con ceros el resto de variables de la parte fija de la cabecera, ocupando un total de 9 octetos (no incluyen por tanto el vector de coeficientes).

6.3 Funcionalidades a partir de las características del medio inalámbrico

Uno de las ámbitos de mayor relevancia dentro del estudio de las técnicas de *NC* se centra en su aplicación sobre redes inalámbricas, ya que sus beneficios pueden verse potenciados por las características tan particulares de estos medios de transmisión. En el caso concreto del protocolo *intra-flujo*, serán dos los factores los que permitirán mejorar su comportamiento. El primero de ellos se aprovecha de la naturaleza *broadcast* del canal inalámbrico, que hace posible que incluso las transmisiones *unicast* sean recibidas por todos los nodos que se encuentren dentro del área de cobertura del transmisor, lo que podría utilizarse para mejorar el comportamiento de las comunicaciones. Gracias a este primer punto, nace la posibilidad de cambiar de un concepto de encaminamiento *unipath* en redes inalámbricas multi-salto a uno multipath, en el que el tráfico puede alcanzar al destino por múltiples caminos diferentes.

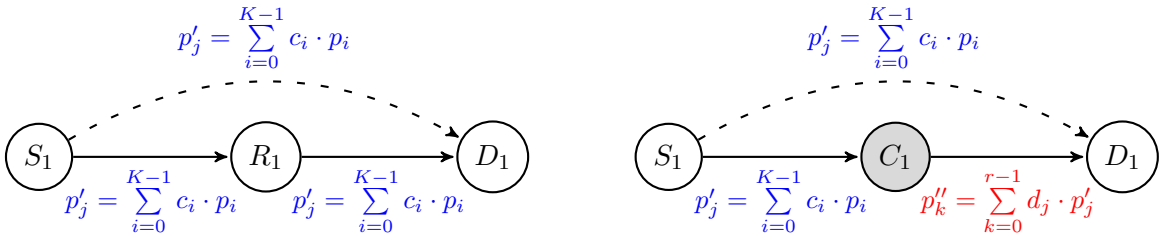
6.3.1. Escucha oportunista

De la misma forma que ya se hizo en el protocolo *inter-flujo*, se mantendrá el *modo promiscuo* de los nodos, evitando descartar aquellas tramas cuya dirección destino no coincida con la del nodo que las está procesando. En este caso, la importancia de la recepción a través de estos "enlaces indirectos" no será tan crítica como en la alternativa *inter-flujo*, pero permitirá acortar las rutas establecidas por defecto entre el origen y el destino[8]. Además, cuando se utiliza adecuadamente, permite evitar la recepción de paquetes duplicados, aumentando la eficiencia y mejorando el *throughput* final.

Para ilustrar sus ventajas, se muestra en la Figura 6.8 un escenario sencillo de tres nodos, en el que el nodo S_1 establece una conexión con D_1, utilizando a R_1 como nodo intermedio. Sin embargo, si S_1 y D_1 se encuentran lo suficientemente cerca, los paquetes generados en el primero podrán ser escuchados directamente en el segundo, con una probabilidad de error que podría ser alta, ya que la distancia entre los extremos será considerable. En un esquema tradicional (sin recepciones promiscuas), todo paquete deberá realizar dos saltos hasta alcanzar correctamente el destino.

Si R_1 simplemente reenviase los paquetes recibidos de S_1 (Figura 6.8), no habría beneficio alguno, ya que todos los paquetes que lleguen a través del enlace directo ($R_1 \rightarrow D_1$) habrían sido recibidos anteriormente a través de la ruta de 2 saltos. Sin embargo, si se permite la recombinación de los paquetes en R_1, como se observa en la Figura 6.8b, las combinaciones lineales recibidas por el enlace directo podrían ser diferentes a aquellas que han llegado directamente desde la fuente. Además, el procesado adicional que se lleva a cabo en R_1, detectando y descartando los vectores que no aportan información, permitirá

[8] Asumiendo que se utiliza un esquema de rutas estáticas *unicast*.



(a) *RLSC* (b) *RLNC*

Figura 6.8: Beneficio de la recodificación en los nodos intermedios

reducir el número de transmisiones no aprovechables. No obstante, merece la pena destacar que si el enlace $S_1 \rightarrow R_1$ no presenta errores, todos los paquetes p''_k generados en R_1 serán combinaciones lineales de los vectores originales recibidos en D_1, por lo que serán igualmente ignorados.

6.3.2. Encaminamiento oportunista

Gracias a la diversidad espacial característica de los medios inalámbricos, la información podrá alcanzar a su destino a través de diferentes vías, permitiendo buscar alternativas a los protocolos de encaminamiento más habituales, que aún a día de hoy no han sabido adaptarse a estas condiciones y mantienen la tendencia de emplear técnicas similares a a aquellas que han sido diseñadas para las redes cableadas. Existen multitud de trabajos que han tratado de buscar la mejor combinación de reenvíos y rutas en los nodos intermedios en una red mallada, desde un punto de vista de teoría de grafos (utilizando una lógica combinatoria) o bien a través de la implementación de un protocolo completo de *encaminamiento oportunista*. En las soluciones pertenecientes al segundo grupo, existen tres aspectos principales: *(1)* la métrica a utilizar para calcular las rutas, *(2)* el algoritmo de selección de candidatos y *(3)* la posterior coordinación entre los nodos. Finalmente, a partir de la señalización recibida en cada nodo, se determina en qué momento se deberá retransmitir un paquete o descartarlo. Éste será el elemento clave del proceso, ya que si la coordinación entre los candidatos no es óptima se producirán transmisiones duplicadas o, en el extremo opuesto, situaciones en lss que ninguna estación envíe nada.

Una de las soluciones más populares se encuentra en el protocolo *ExOR* [Bisw05]; en ella, para llevar a cabo la coordinación entre los vecinos, los autores modificaron el nivel *MAC IEEE 802.11*, reservando un *slot* de tiempo para que cada nodo envíe un *ACK* en el que, además de confirmar la correcta recepción del paquete, los candidatos comparten una lista con la identidad de todos aquellos nodos de mayor prioridad que hayan recibido el mensaje. Existen otros trabajos sobre *NC* que incorporan diferentes mecanismos de encaminamiento oportunista, como *COPE* [Katt08], donde se aprovecha la naturaleza *broadcast* del medio inalámbrico para hacer llegar tanto paquetes *nativos* como codificados

(combinando mensajes de diferentes flujos). Para que los nodos dispongan de la información necesaria para establecer qué combinación de paquetes llevar a cabo en cada ocasión, los vecinos intercambiarán mensajes de señalización, indicando cuáles son los paquetes que tienen almacenados en cada instante. El objetivo es *maximizar la ganancia de codificación* (esto es, en mezclar el mayor número de paquetes posibles), asegurando al mismo tiempo que la información pueda ser posteriormente decodificada. Dentro de los esquemas *intra-flujo*, Chachulski et al. [Chac07] combinan los mecanismos de codificación en red con un protocolo de encaminamiento oportunista basado en la métrica *Expected Transmission Count (ETX)* [Cout03], situando una capa intermedia entre los niveles de enlace y de red. Partiendo de los puntos débiles de *ExOR*, como la falta de compatibilidad con elementos de red tradicionales o la infrautilización del medio inalámbrico, propone una solución alternativa, manteniendo intacto el nivel de enlace. Logra alcanzar prestaciones muy superiores, ya que se aprovecha de todas las ventajas inherentes al uso de *NC*. No obstante, no presenta un sistema de coordinación global, por lo que no sería capaz de adaptarse a situaciones con varios flujos activos en la red; además, tampoco implementa ningún esquema de control de flujo.

Un caso práctico de escenario en el que se pueda obtener beneficio con el uso de técnicas de encaminamiento "múltiple" es el que se representa en la Figura 6.9. En un flujo de S_1 a D_1, se observa como la probabilidad de error $\mathcal{P}$ es diferente para cada uno de los enlaces, siendo más alta cuanto más separados se encuentren los nodos. Si se estableciera la ruta a través de un protocolo tradicional, el camino óptimo entre S_1 y D_1 sería aquel que emplea los nodos R_1 y R_2, necesitando un total de tres saltos para que un paquete para alcance el destino. En este caso, cualquier transmisión que no siga esta ruta (por ejemplo, habrá un 30 % de las veces en las que R_2 reciba correctamente las tramas por S_1), deberá ser descartada. Por el contrario, el encaminamiento oportunista se aprovecha de estas situaciones; en lugar de escoger un único destino para el siguiente salto, se configura un conjunto de potenciales nodos que reenvían la información, teniendo cada uno de ellos una prioridad diferente, escogida en función de un coste, que puede tener en cuenta: número de saltos, consumo energético, calidad del enlace, número de transmisiones esperadas, etc. Así, en consecuencia, el uso de estas técnicas reducirá el número de transmisiones en la red, con lo que su rendimiento se verá beneficiado.

Otra utilidad del encaminamiento oportunista se refleja en la Figura 6.10, donde la transmisión a través de los diferentes nodos intermedios enmascara la calidad deficiente del canal inalámbrico. Cuando se utiliza un protocolo de encaminamiento clásico (Figura 6.10a), sólo un nodo de entre los cuatro será seleccionado como siguiente salto, encargándose de retransmitir los paquetes. En estas condiciones, sólo uno de cada cuatro paquetes llegará correctamente al nodo intermedio, por lo que para recibir un paquete en D_1 se necesitarán 5 transmisiones. En una alternativa oportunista, se podrán seleccionar los 4 nodos como elementos intermedios potenciales. El enlace *virtual*[9] combinado compuesto de

[9]En [Hsu11] se usa esta nomenclatura para equiparar el efecto de enviar el mensaje a través de múltiples nodos intermedios como una única transmisión sobre un enlace de mejor calidad.

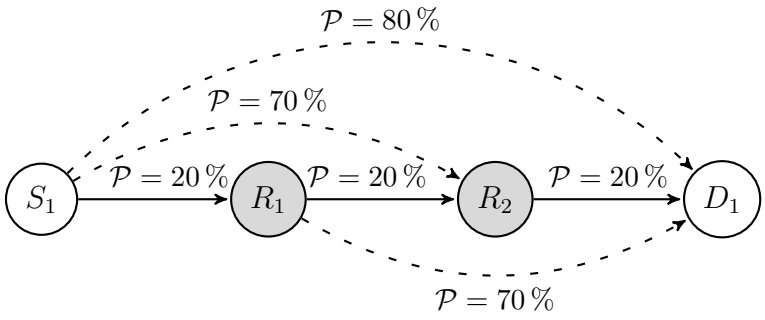
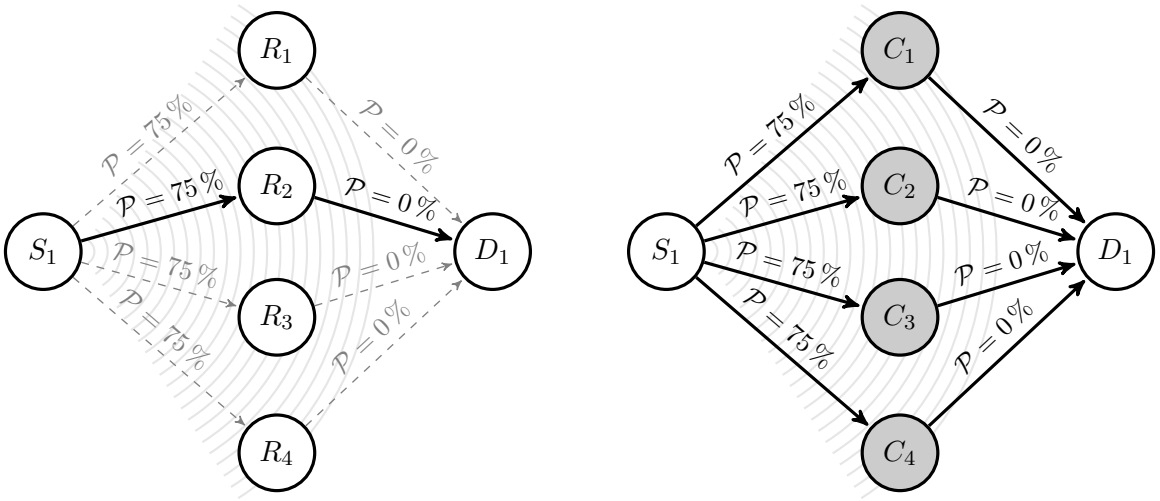


Figura 6.9: Mejoras derivadas del encaminamiento oportunista en redes malladas inalámbricas (*I*)

los cuatro nodos intermedios hará que la probabilidad de que llegue, al menos, un paquete al destino será $(1 - (1 - 0.25)^4) \approx 70\,\%$. Por tanto, el número medio de transmisiones necesarias[10], será $1/0.7 + 1 \approx 1.42$ paquetes, en lugar de los 5 que se necesitaban en un esquema de encaminamiento tradicional.



(a) Esquema tradicional

(b) Con recodificación

Figura 6.10: Mejoras derivadas del encaminamiento oportunista en redes malladas inalámbricas (*II*)

Si a todo esto se le suma la posibilidad de que cada uno de los nodos procese la información y recombine los paquetes recibidos de S_1, se incrementará la probabilidad de que los nuevos mensajes recodificados alcancen al receptor, ya que la correlación entre la información de la que disponen los nodos intermedios (que se encuentra en las *matrices de recodificación*) será baja, debido a la alta probabilidad de pérdida, que hará que éstas se vayan llenando de manera desincronizada. Como consecuencia, el número de vectores

[10]Cuyo valor se calcula a través de la métrica *ETX*, que se corresponde con la suma de los inversos de la probabilidad de recibir correctamente un mensaje en cada uno los enlaces que forman la ruta completa.

linealmente independientes recibidos por D_1 será mayor que en el caso en el que los nodos C_i simplemente reenvían los mensajes.

Sin embargo, si las métricas de selección de nodos retransmisores no son las adecuadas, podrán producirse situaciones con un número excesivo de transmisiones, lo que incrementará la contención en la red y disminuirá el rendimiento final. Por ejemplo, si las condiciones del canal fuesen buenas y se mantuvieran los cuatro nodos reenviando paquetes, el número de recepciones redundantes será mayor y el ancho de banda se vería repartido entre más estaciones, limitando la tasa de la fuente, con la consecuente reducción del *throughput*.

Extendiendo este problema a escenarios reales, con topologías más complejas, la dificultad para seleccionar los mejores conjuntos de nodos que reenvían la información crece ostensiblemente, derivando habitualmente en problemas de optimización no lineal. Existe un gran número de trabajos (resumidos en [Bouk14]) que tratan de encontrar, modificando las propuestas para los aspectos citados anteriormente (i.e. métrica, selección de candidatos y coordinación entre los nodos), la solución óptima, dando lugar a una línea de investigación independiente que, además, se puede complementar con otro tipo de técnicas como *NC* [Katt08; Chac07; Yan10].

6.4 Modelo analítico del rendimiento sobre un canal *IEEE 802.11*

Como paso previo a la caracterización empírica del comportamiento del protocolo *NC intra-flujo*, será interesante llevar a cabo un estudio analítico previo, que permita conocer con mayor exactitud qué es lo que realmente sucede en una transmisión codificada a través de un sistema *RLC*. Por razones de simplicidad, en este análisis solamente se evaluará el impacto de los paquetes codificados directamente en la fuente, asumiendo que la capa *NC* posee toda la información del bloque, esto es, los K paquetes nativos, que combina sin restricciones. Dividido en tres etapas, se comenzará calculando el impacto de los factores asociados a la codificación que penalizan el rendimiento: la generación de combinaciones lineales (que darán lugar a transmisiones espurias) y los reconocimientos que confirman la correcta decodificación de un bloque por parte del destino. Posteriormente, se particularizarán estos resultados sobre un enlace *IEEE 802.11b*, tanto en condiciones ideales como con errores aleatorios. Por último, dada su importancia, se comparará el tiempo de respuesta de diferentes librerías especializadas en el álgebra de cuerpos finitos.

6.4.1. Factores de penalización del esquema de codificación *RLC*

Recuérdese que un nodo fuente transmite combinaciones lineales de un bloque de K paquetes, de manera constante, hasta que el receptor consigue decodificarlo; posteriormen-

te, generará un mensaje de reconocimiento, informando al origen que la transmisión del bloque ha concluido y que puede pasar al siguiente. El nodo destino, por su parte, deberá recibir un total de K paquetes codificados con vectores de coeficientes linealmente independientes, lo que se traduce en un mínimo de K transmisiones (situación ideal). Teniendo en cuenta que el esquema de codificación aleatorio no es perfecto, en cada bloque se generarán δ transmisiones "en exceso", dando lugar a K' transmisiones, siendo $K' = K+\delta$, con $\delta \geq 0$. La idea es por tanto cuantificar, analíticamente, la penalización o sobrecarga asociada a estos δ mensajes, en función del tamaño del bloque K (que establece la dimensión de la matriz $\mathcal{C}$) y del orden del cuerpo de Galois extendido $GF(Q)$. En este apartado se utilizará la notación Q para representar el valor de los cuerpos extendidos, con lo que $Q = 2^q$.

Una de las caracterizaciones más acertadas a la hora de describir la recodificación/decodificación de un bloque se encuentra en [Luca09], donde los autores modelan el proceso como una cadena de Markov, tal y como se muestra en la Figura 6.11. El valor de cada estado lo marca el rango de la matriz de coeficientes $\mathcal{C}$, produciéndose una transición cada vez que se reciba un nuevo paquete codificado[11]. Al cambiar de estado, $i \rightarrow i+1$ $(i < K)$, el vector de coeficientes $\overrightarrow{c_j}$ (el subíndice representa la transmisión j-ésima) pasará a ser la fila i-ésima de $\mathcal{C}$. A medida que la matriz se va completando, la probabilidad de generar vectores linealmente independientes se reduce, ya que el número de combinaciones lineales entre las filas de $\mathcal{C}$ crece exponencialmente, haciendo más difícil generar información nueva. Una vez se haya completado la decodificación, la matriz se vaciará y la cadena volverá al estado inicial.

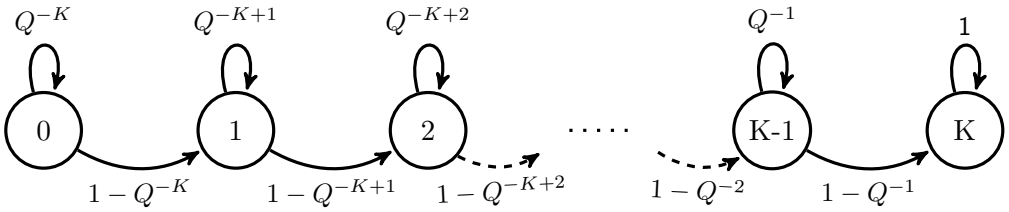


Figura 6.11: Cadena de Markov que representa las probabilidades de recibir vectores de codificación linealmente independientes en función del orden del cuerpo finito y el tamaño del bloque

La Ecuación (6.5) recoge la matriz de transición, de dimensión $K \times K$, entre los diferentes estados del proceso. Es fácil deducir que se necesitarán un mínimo de K transmisiones para alcanzar el estado final, $S_{K-1,K-1}$, marcando el comportamiento óptimo del esquema. Otro aspecto a tener en cuenta es que, de acuerdo con [Lint92], existen un total de $\prod_{j=0}^{K-1}(Q^K - Q^j)$ matrices diferentes con rango K utilizando coeficientes del cuerpo, demos-

[11]Para esta parte del análisis se considera un canal ideal, por lo que todo paquete transmitido será correctamente procesado por el nivel *NC* del receptor.

trando que cuanto mayor sea el cuerpo de Galois, mayor será el número de combinaciones diferentes para generar información linealmente independiente.

$$\mathbb{T} = \begin{bmatrix} Q^{-K} & 1-Q^{-K} & 0 & \dots & 0 & 0 \\ 0 & Q^{-K+1} & 1-Q^{-K+1} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & Q^{-1} & 1-Q^{-1} \\ 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix} \tag{6.5}$$

Asimismo, Lucani *et al.*[Luca09] establecen el número medio de transmisiones necesarias para que la matriz $\mathcal{C}$ tenga rango completo y sea invertible, cuyo margen superior se muestra en la Ecuación (6.6):

$$K' = \text{mín}\left\{K \cdot \frac{Q}{Q-1},\ (K+1) \cdot \frac{1-Q^{-K+1}}{Q-1}\right\} \tag{6.6}$$

No obstante, Trullols-Cruces *et al.* [Trul11] aseguran que la ecuación no captura la dependencia con el tamaño del bloque y proponen un cálculo alternativo, que será el que se emplea como referencia para el resto del análisis presentado aquí[12]. Además, adaptan el modelo a un esquema en el que se descartan de antemano los vectores nulos (en un generador aleatorio, la posibilidad de crear un vector nulo es de $1/Q^K$), evitando así la transmisión de paquetes vacíos.

En primer lugar, se calcula la probabilidad de llevar a cabo la transmisión ideal de un bloque, esto es, de que los K vectores generados sean linealmente independientes entre sí, tal y como se refleja en la Ecuación 6.7:

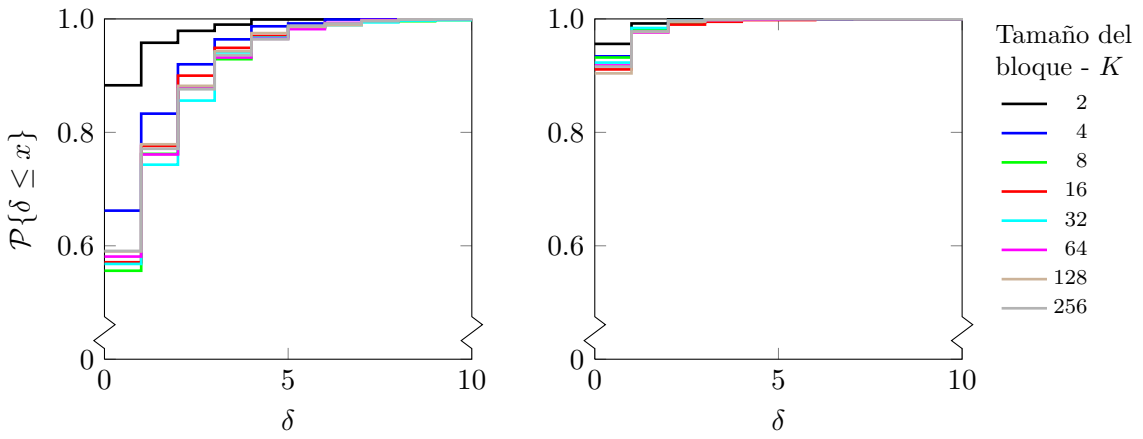
$$\mathcal{P}(K,Q) = \frac{Q^{K^2}}{\left(Q^K-1\right)^K} \cdot \prod_{j=1}^{K}\left(1-\frac{1}{Q^j}\right) \tag{6.7}$$

Será habitual sin embargo que alguno de los vectores generados dependa linealmente de los anteriores, dando lugar a recepciones nulas o *en exceso*. La Ecuación (6.7) permite establecer la probabilidad de que se necesiten $N > K$ recepciones correctas para poder invertir la matriz $\mathcal{C}$, siendo $\delta = N - K$.

$$\mathcal{P}(N,Q)|_{N>K} = \mathcal{P}(K,Q) \cdot \left(\begin{bmatrix} N \\ N-K \end{bmatrix}_Q + \sum_{t=1}^{N-K}(-1)^t \binom{N}{t} \begin{bmatrix} N-t \\ N-K-t \end{bmatrix}_Q\right) \tag{6.8}$$

[12]Existe asimismo un trabajo [Zhao12] en el que el autor simplifica las ecuaciones mostradas en [Trul11], evitando la complejidad de cálculo de algunos coeficientes (sobre todo con valores de Q elevados).

Es interesante destacar que esta expresión se corresponde con la función de distribución acumulativa que representa el número de paquetes (o vectores) espurios que se reciben, en término medio, en función de los tamaños de las matrices de coeficientes y del orden del cuerpo de Galois. La Figura 6.12 ilustra los resultados correspondientes al cuerpo binario (Figura 6.12a) y a su extensión más inmediata, $GF(2^2)$ (Figura 6.12b). El factor común en ambos casos es que, a medida que se incrementa el tamaño de la matriz, mayor es el porcentaje de bloques ideales ($\delta = 0$); no obstante, al normalizar la operación con respecto al tamaño del bloque, se verá que el impacto real sobre la transmisión será el opuesto[13]. Otro aspecto a destacar es el efecto de incrementar Q, pasando de tener alrededor de un 60 % de transmisiones ideales con $K = 256$ y $Q = 2$, a una cifra cercana al 92 % para el mismo tamaño del bloque y un cuerpo $GF(2^2)$. Para cuerpos mayores, se puede asumir que la probabilidad de que todos los vectores sean linealmente independientes es el 100 %, demostrando que aumentar el valor máximo de los coeficientes mitigará notablemente la penalización del rendimiento asociada a las transmisiones de vectores "inútiles". No obstante, el aumento del orden del cuerpo finito tiene un coste asociado en forma de sobrecarga en la cabecera (recuérdese que, con los límites impuestos en esta Tesis[14], se pueden tener vectores de codificación entre 1 octeto hasta 256 *bytes*, cuando $K = 256$ y $q = 8$).



(a) $GF(2)$ (b) $GF(2^2)$

Figura 6.12: Función de distribución del número de paquetes en exceso en función del orden del cuerpo y del tamaño del bloque

[13]El efecto de 1 paquete en exceso con un bloque de tamaño $K = 2$ será mucho mayor que sobre $K = 256$.

[14]En la literatura hay trabajos que utilizan cuerpos mayores, hasta el punto de afirmar en [Pede13] que el tamaño ideal (en términos de eficiencia computacional) es de $GF(2^{32} - 5)$.

Una vez conocida la influencia de los parámetros K y Q, es sencillo obtener el número medio de transmisiones totales N, siendo $N = K+\delta$, a partir de la función de distribución, como expresa la Ecuación (6.9).

$$E[\mathcal{P}(N,Q)|_{N\geq K}] = \sum_{i=N-K}^{\infty} i \cdot \mathcal{P}(i,Q)|_{N\geq Q} \tag{6.9}$$

Con este valor se puede normalizar la expresión en función del tamaño del bloque, cuantificando el impacto (ancho de banda desperdiciado) para cada una de las combinaciones de K y Q. Para mantener una notación común, este factor se denominará *penalización por codificación* (ϵ_{cod}), y su valor se calculará a partir de la Ecuación (6.10), donde $0 \leq \epsilon_{cod} \leq 1$.

$$\epsilon_{cod}(K,Q) = \frac{E[P_{K,q}(n)]}{K + E[P_{K,q}(n)]} \tag{6.10}$$

El otro elemento que hay que tener en cuenta a la hora de calcular la penalización introducida por el protocolo *intra-flujo* es la contención adicional generada con el envío de los mensajes de reconocimiento, tras la decodificación exitosa de cada bloque. En este caso se necesita conocer el retardo medio de un reconocimiento, desde que se transmite en el nodo destino hasta que se recibe por completo en el transmisor. Si se asume que $\overline{\tau_{datos}}$ es el retardo medio de un mensaje de datos y $\overline{\tau_{ACK}}$ el correspondiente a un *ACK*, es posible obtener el parámetro ϵ_{ACK}, que representa el factor de penalización asociado al envío de reconocimientos, tal y como se ve en la Ecuación 6.11. Es importante destacar que el impacto de este parámetro será mayor en medios basados en contienda, como el *IEEE 802.11*.

$$\epsilon_{ACK} = \frac{\overline{\tau_{ACK}}}{(K+\epsilon)\cdot\overline{\tau_{datos}} + \overline{\tau_{ACK}}} \tag{6.11}$$

Finalmente, el rendimiento medio del esquema *RLC*, medido en la capa de aplicación de un nodo receptor, se podrá estimar como refleja la Ecuación 6.12, donde $\overline{Thput_{máx}}$ representa el valor máximo del *throughput* para la tecnología subyacente (asumiendo que el protocolo de transporte no introduce ningún tipo de control de flujo, como es el caso de *UDP*).

$$\overline{Thput_{RLC}}(K,Q) = \overline{Thput_{máx}}(K,Q)\cdot(1-\epsilon_{cod}(K,Q))\cdot(1-\epsilon_{ACK}(K,Q)) \tag{6.12}$$

6.4.2. Caracterización sobre un enlace *IEEE 802.11b*

En el análisis previo se ha modelado el rendimiento del esquema de codificación *intra-flujo* sobre escenarios genéricos, sin tener en cuenta las particularidades de ninguna topología compleja. A continuación, se asumirá el uso del estándar *IEEE 802.11b*. Así, la Ecuación (6.13) permite calcular el retardo medio (ver Sección 2.1.1 para un análisis más detallado) de una transmisión sobre un enlace *IEEE 802.11b* con errores, en función de la longitud de la trama (L) y del número máximo de retransmisiones[15].

$$\overline{\Delta}(k, L, R) = (k+1) \cdot \tau_{det}(L) + \left[\left(16 \cdot \sum_{j=0}^{k} 2^j \mod (R+1) \right) - \frac{k+1}{2} \right] \cdot \sigma \tag{6.13}$$

En la expresión, k representa el número de retransmisiones realizado entre dos recepciones consecutivas (con $k = 0$ se tendría una recepción correcta en el primer intento), τ_{det} es el tiempo total de transmisión que no está sujeto a ninguna componente aleatoria y es, por lo tanto, común para todas las tramas, el envío de los datos correspondientes a las cabeceras y los datos, etc.), R, como ya se ha mencionado, es el número máximo de reintentos y, finalmente, σ se corresponde con la duración del *slot* de la ventana de contención del mecanismo *DCF* del estándar *IEEE 802.11* ($20\,\mu segs$ para *IEEE 802.11b*).

A diferencia de una transmisión clásica (sin codificar) sobre *UDP*, la pérdida de un datagrama en concreto, en un esquema *RLC*, no supone una penalización crucial, ya que cada paquete codificado transporta exactamente el mismo porcentaje del mensaje completo, y no tiene valor alguno si se tratase como una unidad de información aislada. Así, perder una trama particular no es relevante, ya que el objetivo es recibir los K' paquetes codificados para poder recuperar el contenido original del bloque.

Como se verá más adelante, el esquema de retransmisiones del estándar *IEEE 802.11* no aporta beneficio alguno en este tipo de transmisiones, ya que si no es crítico recibir correctamente el contenido individual de un paquete en concreto, no es necesario tratar de recuperarlo al no recibirlo correctamente tras el primer intento. Como se verá posteriormente, aumentar el tiempo de espera tras una transmisión fallida incrementará el retardo medio innecesariamente, lo que redundaría en una disminución del rendimiento.

Si se asume que el canal inalámbrico carece de memoria, la probabilidad de error de trama se puede modelar a través de la variable $\mathcal{P}_{trama}$, a partir de la cual se puede

[15] Recuérdese que en este modelo no se considera el valor máximo del tamaño de la ventana de contención, que siempre se duplica tras cada retransmisión.

aproximar el retardo medio entre dos recepciones consecutivas, tal y como se ve en la Ecuación (6.14).

$$E[\overline{\tau}(k, L, R, \mathcal{P}_{error}(trama))] = \sum_{i=0}^{\infty} [1 - \mathcal{P}_{error}(\text{trama})] \cdot \mathcal{P}_{error}(\text{trama})^i \cdot \overline{\tau}(k, L, R) \quad (6.14)$$

Dado que el retardo medio muestra una dependencia inversamente proporcional con el *throughput* ($Thput = \frac{L}{\tau}$), la Ecuación 6.15 puede utilizarse para calcular el valor del rendimiento al combinar un esquema de codificación *intra-flujo* con el protocolo *UDP* como solución a nivel de transporte.

$$\overline{Thput_{RLC_{802.11b}}} = \frac{L \cdot 8}{E\left[\overline{\tau}(K, L, R, \mathcal{P}_{error}(\text{trama}))\right]} \cdot (1 - \epsilon_{cod}(K, Q)) \cdot (1 - \epsilon_{ACK}(K, Q)) \quad (6.15)$$

6.4.3. Librerías de cuerpos finitos

Uno de los aspectos más críticos de este tipo de técnicas, junto con la sobrecarga asociada a la transmisión explícita de los coeficientes aleatorios utilizados para la codificación, es el coste computacional que subyace en las operaciones algebraicas necesarias: generación de vectores, suma, multiplicación, inserción de filas dentro de una matriz y el cálculo del rango e inversa de matrices. El coste asociado a las cuatro primeras se puede considerar despreciable, por lo que hay que prestar atención al tiempo requerido para las dos últimas (sobre todo al cálculo de la inversa), especialmente cuando la dimensión de las matrices es elevada.

Teniendo en cuenta que el simulador `ns-3` está desarrollado sobre una base de `C++`, será necesaria la utilización de librerías externas que implementen las operaciones sobre cuerpos de Galois mencionadas anteriormente. Además, resultaría esencial que facilitasen una conversión directa a otro tipo de datos más genérico (como una variable entera de 8 *bits*), facilitando la integración con el propio protocolo a la hora de generar/procesar las cabeceras.

Sin entrar en los detalles de implementación de cada una de las librerías, se comparará el rendimiento de cuatro alternativas, todas ellas, salvo `Matlab`, con interfaces compatibles con el simulador, habiéndose incorporado a los motores de codificación/decodificación del protocolo.

1. `IT++` [IT++]. Desarrollada enteramente en `C++`, es una librería genérica utilizada en álgebra lineal, optimización numérica, procesado de señal, sistemas de comunicaciones

y cálculos estadísticos en general. Con respecto a las operaciones sobre cuerpos de Galois, cuenta con la gran limitación de ser compatible exclusivamente con el cuerpo binario base $GF(2)$. Utiliza la eliminación de Gauss para factorizar las matrices y calcular los rangos y las inversas, con un coste total, para esta última operación, de $\mathcal{O}(K^3)$.

2. `FFLAS/FFPACK` [FFLAS] es la combinación de dos librerías diseñadas de manera explícita para realizar operaciones de álgebra lineal sobre cuerpos finitos. La primera se centra en adaptar las técnicas conocidas como *Basic Linear Algebra Subprograms (BLAS)* a este tipo de elementos, mientras que la segunda añade una capa adicional de operaciones, basadas en la triangularización de matrices (eliminación Gaussiana, factorización LU, etc.). Los cálculos correspondientes a su coste computacional se presentan en [Duma], que resulta ser (para la inversa) de orden $\mathcal{O}(w \cdot K^3)$, donde w es un factor de corrección surgido de la variante de Winograd para las multiplicaciones rápidas de Strassen [Pern01].

3. `M4ri` y `M4rie` [M4RI] son dos librerías centradas exclusivamente en el manejo de operaciones aritméticas sobre matrices en cuerpos finitos. La primera de ellas fue desarrollada para trabajar con cuerpos binarios, mientras que la segunda permite operaciones sobre versiones extendidas, hasta $GF(2^8)$. Su característica más significativa es que sus algoritmos se basan en el "*Método de los cuatro rusos*" [Aho74]. Con respecto a su eficiencia, su principal desarrollador afirma en [Albr11] que el coste del cálculo de la inversa es $\mathcal{O}\left(K^3/\log K\right)$. Al igual que en el caso de `FFLAS/FFPACK`, se pueden encontrar dentro de la herramienta `SAGE` [SAGE].

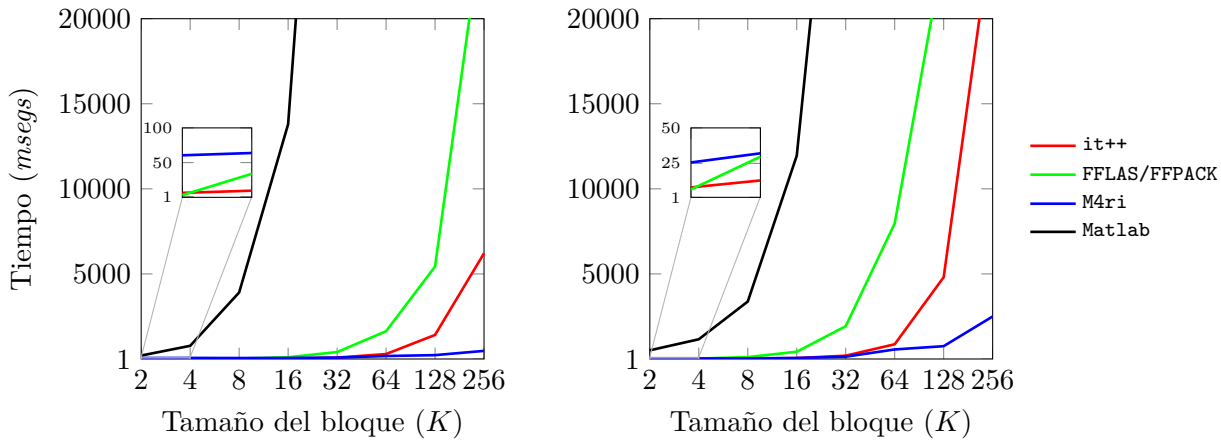
4. `Matlab`. Esta herramienta de propósito general introduce la posibilidad de operar con cuerpos finitos de Galois [MatGF]. Su uso no es recomendable, ya que la integración con `ns-3` sería más costosa y, además, muestra una eficiencia muy inferior a las implementaciones puras sobre `C/C++`.

Para comparar el rendimiento de las cuatro alternativas[16], se ha llevado a cabo un análisis sistemático, a partir de un método *Monte Carlo*, repitiendo el proceso de codificación/decodificación de un bloque hasta 10000 realizaciones para cada una de las combinaciones de K y q. En un primer grupo, se presentarán los resultados normalizados con respecto al menor de los tiempos, obtenido con la librería `M4rie` para $K = 2$ y $q = 6$. Se podrá ver así qué alternativa resulta más eficiente, sin importar los valores de tiempo absolutos (que dependen estrechamente del equipo utilizado). En un segundo análisis se caracterizarán los tiempos de computación absolutos, utilizando tres dispositivos con grandes diferencias en sus prestaciones.

Como primera métrica, se ilustra en la Figura 6.13 el tiempo medio normalizado para el cálculo del rango y la inversa en un cuerpo binario $GF(2)$. Queda patente la

[16]En el caso de `Matlab`, se ha portado el código equivalente al manejo de vectores y las operaciones con la matriz de coeficientes, obviando el resto de elementos que rodean a la simulación.

dependencia con el tamaño de la matriz, viéndose claramente la diferencia de rendimiento entre las cuatro librerías, donde la clasificación sería: `M4ri`/`M4rie` > `It++` > `FFLAS` >> `Matlab`. Además, puede comprobarse que el cálculo de la inversa requiere un mayor tiempo que el del rango (alrededor de cuatro veces mayor). Por último, resulta curioso el efecto observado para valores de K bajos, ya que los resultados no siguen con la tendencia general y la operación a través de `M4ri` resulta más pesada que la de *IT++* o `FFLAS/FFPACK`, mostrándose estas últimas más eficientes para bloques pequeños.



(a) Rango (b) Inversa

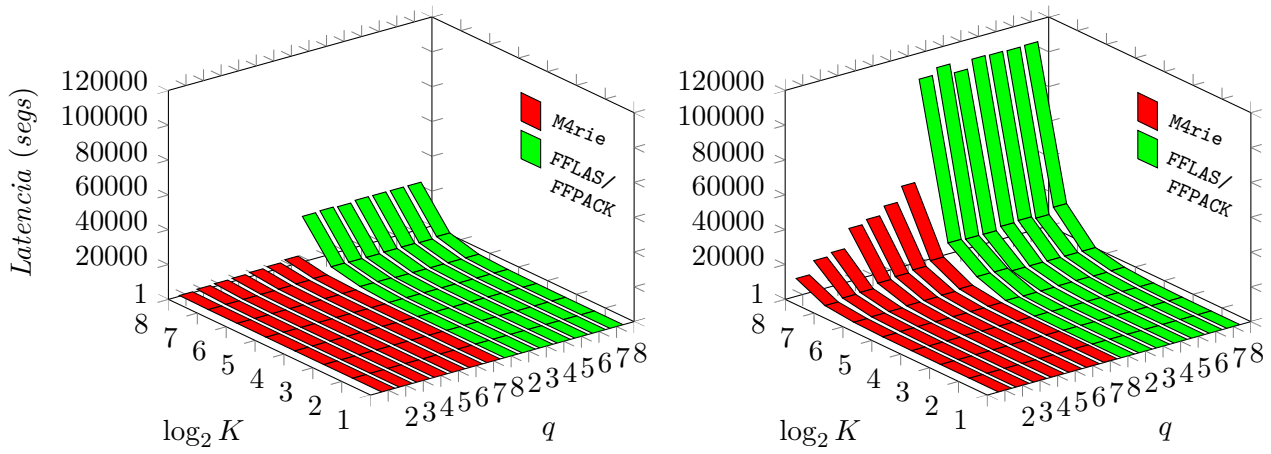
Figura 6.13: Tiempos de computación (normalizados) para el cálculo del rango y la inversa sobre un cuerpo binario ($q = 1$)

En el siguiente grupo de resultados (Figura 6.14), se extiende el análisis a cuerpos $GF(2^q)$, con $q \in [2, 8]$, limitando la comparativa a las librerías `M4rie` y `FFLAS/FFPACK` (`IT++` no soporta operaciones para cuerpos con $q > 1$ y el coste de `Matlab` se encuentra varios órdenes de magnitud por encima del resto). Se vuelve apreciar el mejor rendimiento ofrecido por `M4rie` que, a pesar de mostrar una clara dependencia con el orden del cuerpo extendido (al contrario que `FFLAS/FFPACK`), mantiene unos tiempos claramente inferiores. Se pone nuevamente de manifiesto la mayor complejidad del cálculo de la matriz inversa.

A continuación, se repite el experimento en tres equipos con prestaciones muy diferentes (sus principales características se resumen en la Tabla 6.2); la Figura 6.15 recoge los tiempos medios *absolutos* para los cálculos de rango e inversa.

Tabla 6.2: Especificaciones de los equipos utilizados

Equipo	***Año***	**Microprocesador**	***Frec. (GHz)***	***RAM (GB)***
A	2012	Intel Core i7-3520m	2.90	8
B	2011	Intel Core i5-3317-U	1.70	8
C	2004	Intel Pentium M	1.86	1



(a) Rango (b) Inversa

Figura 6.14: Tiempos de computación (normalizados) para el cálculo del rango y la inversa sobre cuerpos binarios de orden $q > 1$

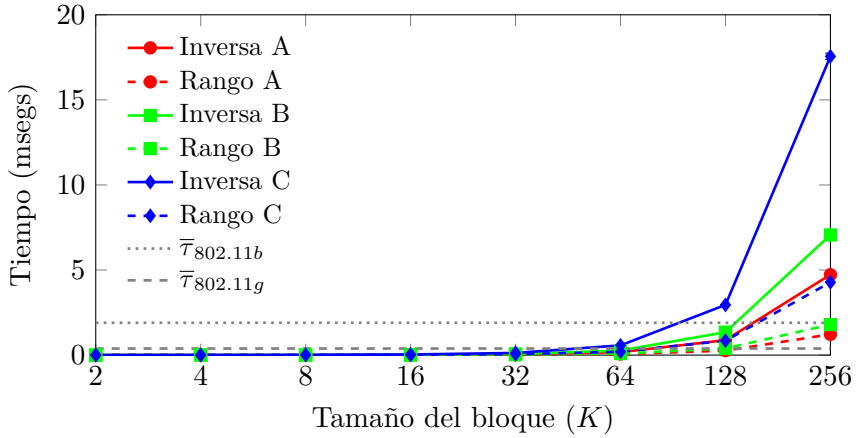


Figura 6.15: Comparativa del tiempo de cálculo sobre diferentes máquinas

No obstante, el resultado más importante del análisis anterior no es la comparativa entre estas curvas. Junto a éstas aparecen representados los retardos medios en una transmisión punto a punto sobre enlaces *IEEE 802.11b* ($\approx 2mseg$) e *IEEE 802.11g* ($\approx 400\mu seg$), tiempos que establecen la diferencia entre eventos de recepción consecutivos. Así, si el tiempo invertido en realizar cualquiera de los cálculos es mayor, la entidad *NC* ralentizará el procesado de los vectores, encolando los mensajes en el *buffer* del nivel de enlace. Es importante destacar que el cálculo del rango se efectúa tras cada recepción correcta; por el contrario, la inversa se realiza una única vez por bloque. A la vista de los resultados, cuanto mayor sea la capacidad de la tecnología, mayor será la restricción con respecto al tiempo máximo permitido para ejecutar las operaciones pertinentes, por lo que sería bueno disponer de un conocimiento previo (e.g. nivel físico utilizado, tiempos de cómputo de cada equipo, tamaño del bloque, orden del cuerpo, requerimientos de la aplicación, etc.)

para poder mantener un rendimiento óptimo en cada caso, pudiendo plantearse incluso una modificación dinámica en función de las necesidades.

En cualquier caso, este efecto no sería visible en el entorno de `ns-3`, ya que al ser un simulador por eventos discretos, los retardos asociados a estas operaciones algebraicas (externas a la plataforma) no se tendrán en cuenta en el tiempo de procesado.

6.5 Simulación y Resultados

En esta sección se presentarán los resultados más relevantes en el análisis del protocolo *intra-flujo*, llevado a cabo sobre el simulador `ns-3`. Recuérdese que su objetivo es la provisión de un servicio fiable sobre enlaces inalámbricos. Por lo tanto, se comparará el comportamiento del esquema propuesto con el de *TCP*, comprobando si la combinación de un proceso de codificación aleatoria unido a *UDP* puede mejorar el comportamiento de la solución de nivel de transporte por excelencia, *TCP*.

Uno de los pilares fundamentales del *NC* es la inclusión de una capa de inteligencia adicional en los nodos intermedios, que les permite modificar la información a su paso. Se incluirá en el análisis una versión limitada de la solución completa *intra-flujo RLNC*, en la que sólo los nodos origen realizan combinaciones lineales de los paquetes que atraviesan la entidad *NC*, alternativa definida como *RLSC*.

Serán tres las configuraciones a comparar en este apartado. Además, los resultados se presentarán en tres fases, en función de las principales propiedades de la solución propuesta (codificación en la fuente, recodificación en los nodos intermedios + escucha oportunista y encaminamiento oportunista).

- En un primer análisis se considerarán dos nodos que se comunican directamente entre sí a través de un enlace inalámbrico. Se determinarán los umbrales del rendimiento del protocolo *intra-flujo*, corroborando la caracterización analítica que se presentó en la Sección 6.4.
- En el siguiente punto se extenderá el análisis a un escenario de tres nodos (con una estructura similar al de *Alice y Bob* descrito en el Capítulo 4), en el que la fuente se apoya en un nodo intermedio para comunicarse con el destino. En este caso se dejará abierta la posibilidad de que el nodo destino pueda escuchar (gracias al modo *promiscuo*) los mensajes transmitidos directamente por la fuente, lo que permitirá estudiar los beneficios asociados a la recombinación en los nodos intermedios.
- Finalmente, se analizarán los posibles beneficios del empleo de mecanismos de encaminamiento oportunista. Partiendo del escenario del punto anterior, se desplegará un número variable de potenciales nodos *forwarders* (que retransmitirán la información en base a una serie de criterios establecidos dentro de la propia entidad *NC*),

cuantificando las diferencias entre las diferentes estrategias de reenvío, cuyas bases se definirán más adelante.

La Tabla 6.3 recoge los principales parámetros utilizados durante las campañas de simulación.

Tabla 6.3: Parámetros de la simulación

Parámetro	**Valor**
Nivel físico	*IEEE 802.11b* (11 *Mbps*)
Errores	Modelo *nativo*/*HMP*
Transmisiones por paquete	1, 4, 7
RTS/CTS	Deshabilitado
Tamaño del bloque (K)	2, 4, 8, 16, 32, 64, 128, 256
Orden del cuerpo $GF(2^q)$	$q \in [1, 8]$
Capa de transporte	*UDP* (*NC*)/*TCP*
Aplicación	$\left\lceil \frac{10000}{K} \right\rceil$ paquetes
Tasa binaria (aplicación)	Constante (Condiciones de saturación)
Tráfico	*Unicast*
Payload	1500 *bytes* en nivel *IP*
Número de simulaciones	100 realizaciones/punto

Se mantienen los parámetros utilizados durante toda la memoria (i.e. enlaces *IEEE 802.11b*, tráfico *unicast*, etc.); en este caso sólo se utiliza una conexión de datos (de S_1 a D_1), volviendo a utilizar una tasa superior a la capacidad del medio físico. El nivel de aplicación genera un número de paquetes que sea múltiplo del tamaño del bloque, para asegurar un número "exacto" de bloques completos. Con respecto al tamaño de los paquetes, el *payload* será el máximo permitido sin recurrir a la fragmentación en los niveles inferiores, *MTU* = 1500 octetos; para el caso de *TCP* tendrán un tamaño fijo de 1460 *bytes*, mientras que para las alternativas de *NC* se utilizará la base de un datagrama *UDP*, al que se le restará el tamaño de la cabecera *NC intra-flujo*, cuya longitud se calcula a partir de la expresión $1472 - \left(9 + \left\lceil \frac{K \cdot q}{8} \right\rceil\right)$ octetos.

Para modelar la presencia de errores en el canal se volverá a recurrir al modelo *nativo* del simulador, aunque también se repetirá el experimento en algunas configuraciones con el modelo *HMP*, para observar el impacto de las ráfagas sobre estos esquemas de codificación. Teniendo en cuenta que la información de un paquete individual pierde sentido, la utilización de las retransmisiones a nivel *MAC* no aporta ningún beneficio. Para comprobar esta premisa, se modificará el valor de este atributo en la capa de enlace, analizando las consecuencias.

Respecto a la configuración del nivel *NC*, se utilizarán bloques de hasta 256 paquetes, valor máximo permitido por el octeto que establece el valor de K en la cabecera del protocolo. Además, se fijará un máximo para el orden de los cuerpos de Galois de $q = 8$,

que asegura que la mayoría de las combinaciones entre paquetes nativos dé lugar a vectores linealmente independientes. Como se adelantó en la caracterización de las librerías algebraicas estudiadas, la campaña de simulación se llevará a cabo en los módulos `M4ri/M4rie`.

Finalmente, según un procedimiento *Monte Carlo*, cada experimento se repetirá 100 veces para cada punto de configuración, representándose (siempre que sea interesante) el valor medio y el intervalo de confianza del 95 % de los valores observados. Dentro de los resultados se volverá a estudiar el *throughput* como métrica principal del rendimiento, poniendo especial atención en otros estadísticos, como la latencia o los factores de penalización del proceso de codificación.

6.5.1. Caracterización del rendimiento

Como punto de partida para analizar el comportamiento de los mecanismos de codificación/decodificación, se utilizará el escenario que se muestra en la Figura 6.16. El enlace $S_1 \rightarrow D_1$ será ideal inicialmente, para después manipular su calidad, habilitando la aparición de errores.

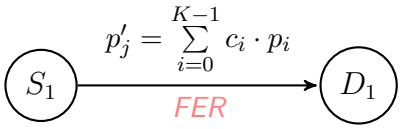
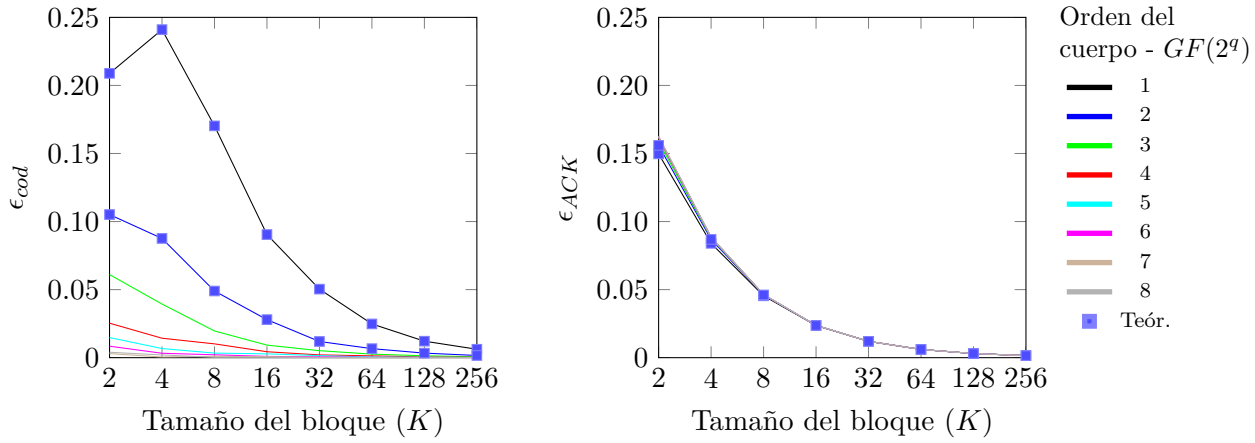


Figura 6.16: Escenario básico para la caracterización del protocolo *NC intra-flujo*

El primero de los aspectos a evaluar se corresponde con los factores de penalización que se introdujeron en la Sección 6.4, ϵ_{cod} y ϵ_{ACK}, que caracterizan la sobrecarga del esquema de codificación *RLC*. Como se ha descrito anteriormente, el nodo S_1 enviará aleatoriamente K' combinaciones lineales de los paquetes correspondientes al bloque i-ésimo, de la forma $p'_i = \sum_{i=0}^{K-1} c_i \cdot p_i$, hasta que finalmente el destino reciba K vectores linealmente independientes (recuérdese que $K^{'} \geq K$). Al no implementar ningún tipo de control en la fase de generación de los coeficientes y confiar toda la operación a un generador de números aleatorios, es posible que algunas de las combinaciones no aporten información adicional, lo que dará lugar a descartar el paquete, situación equivalente a una pérdida producida en el canal. En el otro extremo, tras la correcta decodificación de un bloque completo, el receptor envía un mensaje de reconocimiento, produciendo una sobrecarga que deberá ser tenida en cuenta. Es preciso destacar que, junto a las curvas que representan los valores obtenidos empíricamente, se representan las cifras calculadas a partir de las expresiones teóricas presentadas en la Sección 6.4, para comprobar la validez de ambos enfoques.

En la Figura 6.17 se muestran los resultados correspondientes a estas métricas. En la primera (Figura 6.17a) se observa la influencia de la transmisión de vectores linealmente dependientes. Se aprecia que al incrementar el tamaño del bloque o el orden del cuerpo se reduce el porcentaje de paquetes redundantes, siendo inferior al 1 % en configuraciones con

$K \geq 64$ y $q \geq 2$. Resulta especialmente anómalo el valor obtenido para $K = 1$ y $q = 1$, ya que el factor ϵ_{cod} rompe con la tendencia decreciente del resto de la gráfica; la razón está en las pocas alternativas al elegir los coeficientes aleatorios (4 en total), siendo uno de ellos nulo. El simple hecho de impedir la transmisión de este vector reduce la penalización asociada a su envío. A medida que los atributos crecen, la probabilidad de generar uno de estos vectores nulos (i.e. $\frac{1}{K \cdot 2^q}$) disminuye rápidamente.



(a) Penalización por combinaciones lineales

(b) Penalización por reconocimientos (ACKs)

Figura 6.17: Factores de penalización de una transmisión *RLC*

En cuanto a la ocupación del canal por parte de los reconocimientos, representada en la Figura 6.17b, se ve que cuanto mayor es el tamaño del bloque, menor será la frecuencia de envío de los *ACKs*, y menor será la sobrecarga correspondiente. Se observa además que no hay ninguna dependencia con el orden del cuerpo. En cualquier caso, la penalización es inferior a 0.01 para $K \geq 64$.

Una vez cuantificada la sobrecarga asociada a las operaciones propias del nivel *NC*, resultará sencillo extender el análisis al rendimiento medido en la capa de aplicación del nodo D_1. Utilizando la Ecuación (6.12) para calcular el *throughput* a partir de la capacidad máxima del canal para una transmisión *UDP* y de los coeficientes ϵ_{cod} y ϵ_{ACK}, la Figura 6.18 representa los resultados para un canal *sin errores*, donde el rendimiento se mueve entre los 3.99 *Mbps* para $K = 2$ y $q = 1$ hasta los 5.87 *Mbps* obtenidos con $K = 256$ y $q = 1$.

Hay que tener en cuenta en este punto que el tamaño de la cabecera del protocolo *NC* es directamente proporcional a K y q, pudiendo alcanzar hasta los 256 octetos. Teniendo en cuenta que el *payload* máximo es de 1463 *bytes* (descontando los 9 *bytes* correspondientes a la parte fija de la cabecera *NC*), es posible que la sobrecarga inducida al transportar los vectores de coeficientes provoque una reducción del rendimiento final, como se aprecia en la figura para bloques grandes y cuerpos de Galois de orden superior que 3.

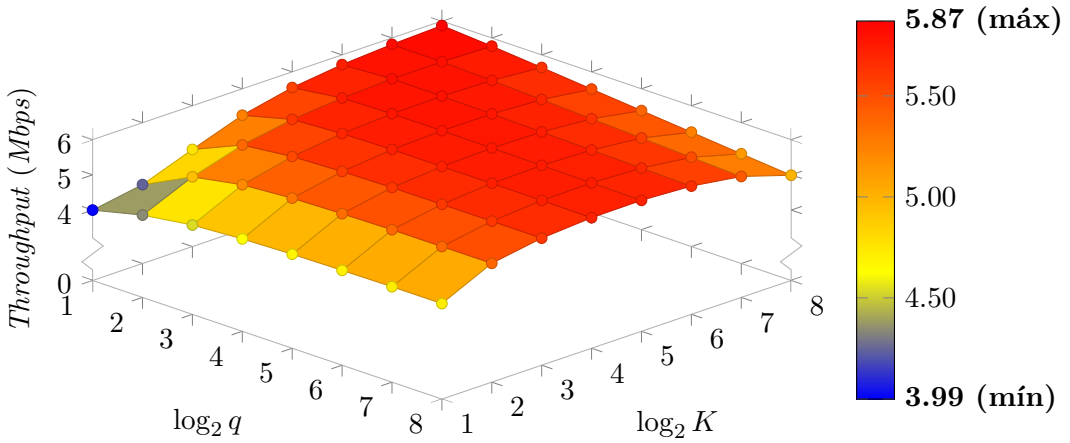


Figura 6.18: Representación 3D del *throughput* sobre un enlace *IEEE 802.11b* ideal

La Figura 6.19 utiliza una representación bidimensional, facilitando visualizar la variación del rendimiento con la configuración de los atributos de codificación, comparándolo al mismo tiempo con el *throughput* máximo de *TCP* sobre un enlace punto a punto *IEEE 802.11b*. Parece evidente que la utilización de bloques pequeños resulta perjudicial, con valores inferiores incluso a los del propio *TCP*. A medida que se incrementan K y q, la penalización causada por el uso del *RLC* se reduce paulatinamente, por lo que la pendiente de crecimiento de *throughput* se estabiliza. Además, cuando $q > 1$, la sobrecarga asociada a la cabecera invierte la tendencia, reduciéndose incluso el rendimiento, aunque el tamaño del bloque crezca. El valor máximo del *throughput* se alcanza al utilizar un cuerpo binario con bloques de 256 paquetes. Sin embargo, a partir de este punto se utilizará una configuración con $K = 64$, ya que los tiempos de cálculo garantizan una ejecución significativamente más rápida de las simulaciones.

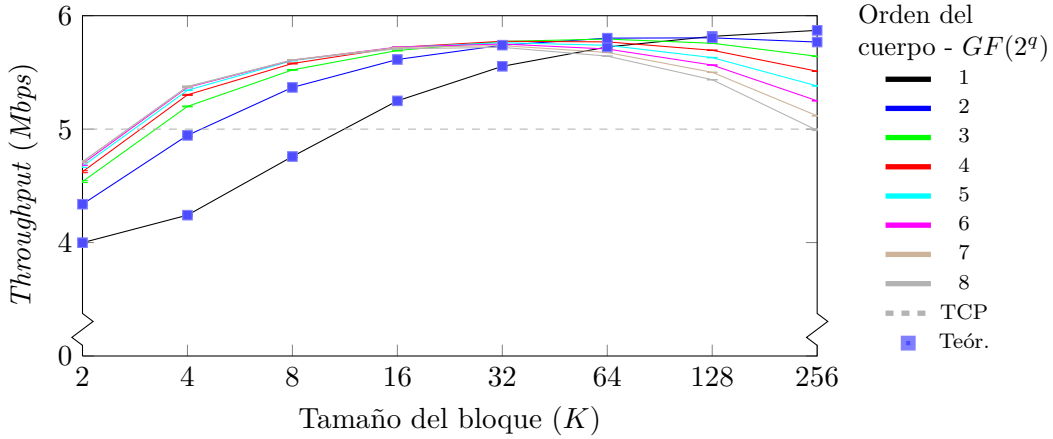
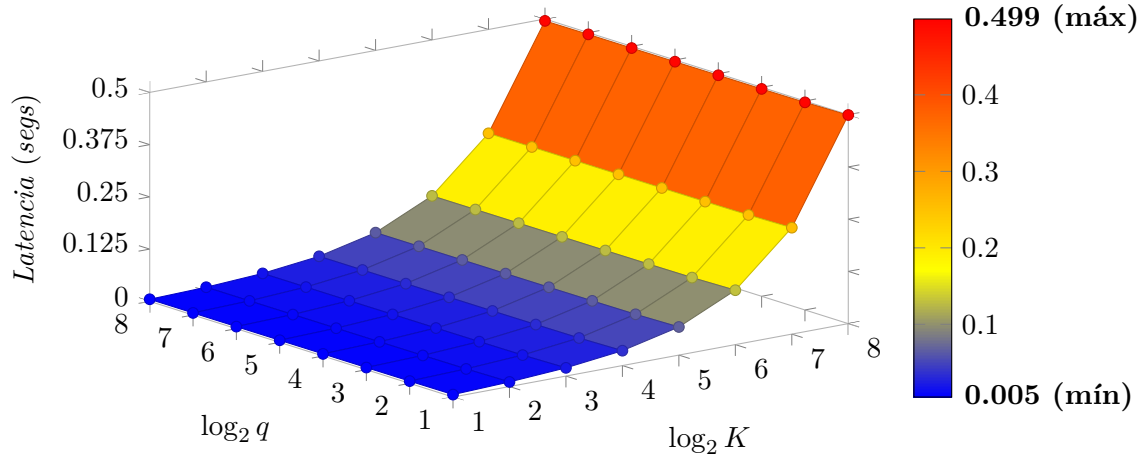


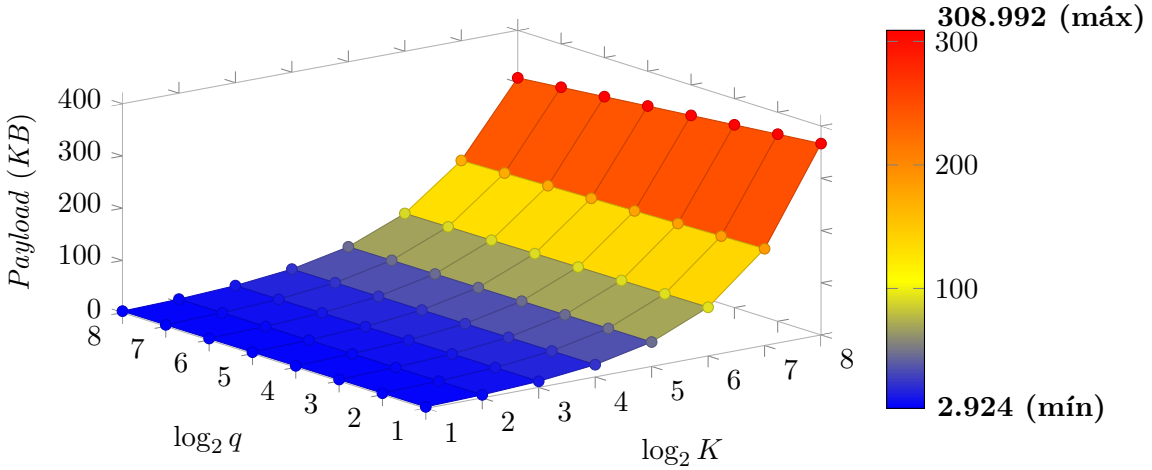
Figura 6.19: Representación 2D del *throughput* sobre un enlace *IEEE 802.11b* ideal

Teniendo en cuenta que la información no sube al nivel de aplicación de forma fluida tras la recepción individual de cada paquete, sino que los K mensajes nativos que componen

bloque lo harán de manera simultánea, una vez se haya decodificado con éxito. Así, cuanto más grandes sean los bloques, será necesario un mayor número de transmisiones para su decodificación y, por tanto, mayor será el tiempo entre dos bloques consecutivos. En la Figura 6.20 se pueden encontrar los valores medios de retardo (Figura 6.20a) y la cantidad de datos (*payload*) que sube a la aplicación tras la recuperación de un bloque completo (Figura 6.20b), donde el rango de información útil asciende desde los escasos 3 KB ($K = 2, q = 1$) hasta cantidades superiores a los 300 KB, cuando $K = 256$. Del mismo modo, se observa un descenso a medida que el orden del cuerpo crece, consecuencia de la sobrecarga del protocolo.



(a) Retardo



(b) *Payload*

Figura 6.20: Dualidad entre retardo y cantidad de información que le llega a la aplicación tras la decodificación de un bloque

Como el retardo puede ser un factor determinante para ciertos servicios, podría ser necesario establecer algún tipo de mecanismo que limite el tamaño de los bloques, para mantener esta latencia entre decodificaciones dentro de unos márgenes razonables.

Como último aspecto antes de dar el paso a otros escenarios más complejos, se configurará el enlace para analizar la influencia de errores de transmisión durante las conexiones. La Figura 6.21 representa la evolución del *throughput* en función de la probabilidad de error de trama en el enlace $S_1 \rightarrow D_1$, utilizando como base el modelo *Nativo* del simulador.

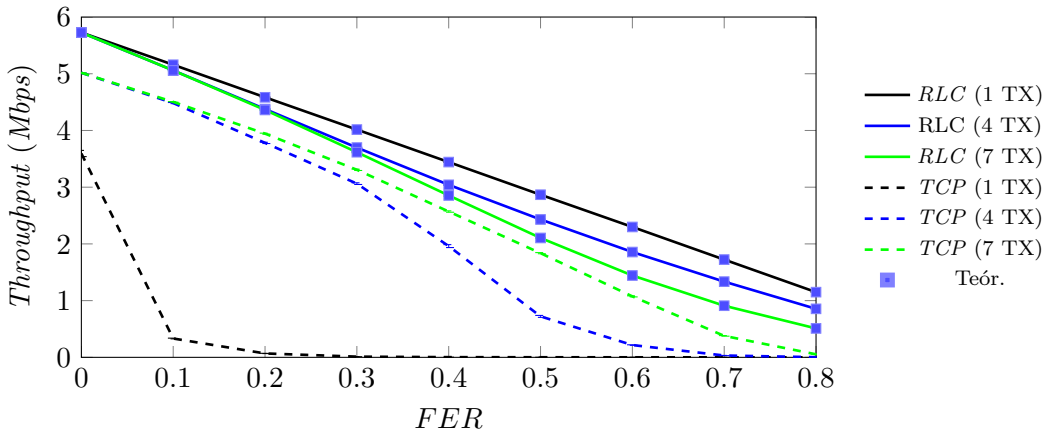


Figura 6.21: *Throughput* sobre un enlace con errores (modelo de canal *sin* memoria)

Las principales conclusiones que se pueden extraer de las curvas son las siguientes:

- En primer lugar, se ve que, a medida que la calidad del canal empeora, el *throughput* desciende en todas las configuraciones. Se ve, sin embargo, que el protocolo *intra-flujo* presenta un mejor rendimiento que *TCP*, pudiendo incluso ofrecer valores de rendimiento aceptables sobre canales con probabilidades de error de trama superiores al 70 %, cifras en las que una conexión *TCP* no puede finalizar correctamente.

- De nuevo, se observa un valor de *FER* a partir del cual el descenso del rendimiento de *TCP* crece a un ritmo mayor. Como se discutió en el Capítulo 2, la probabilidad de error de paquete en un canal sin memoria se puede aproximar con la expresión $\mathcal{P}_{paquete} = (\mathcal{P}_{trama})^N$, siendo N el número máximo de intentos de transmisión por paquete en el nivel de enlace. Por lo tanto, para que el efecto de esta pérdida de mensajes empiece a ser relevante (en [Katt08] se concluye de que un valor de un 2.5 % es suficiente para desestabilizar una conexión *TCP*), la probabilidad de error de trama deberá ser elevada: ≈ 0.4 para 4 transmisiones o ≈ 0.7 cuando se pueden hacer 7 intentos por paquete.

- En cuanto a la influencia del número máximo de transmisiones por paquete, se aprecia que, en el caso del protocolo *TCP*, el incremento de este parámetro sirve para reforzar la transmisión, recuperándose ante ráfagas cortas de errores, sin necesidad de recurrir

al reenvío de segmentos desde el nivel de transporte. Así, se ve que el esquema de retransmisiones del estándar *IEEE 802.11* es fundamental en este tipo de conexiones, como puede verse en la curva correspondiente a una transmisión por paquete, ya que un simple error equivale a la pérdida definitiva del segmento. Si por el contrario se habilitan más intentos, el descenso del *throughput* será menos acentuado, ya que se enmascara la pérdida de segmentos puntuales.

- Sin embargo, estas retransmisiones muestran un efecto inverso cuando se utiliza el protocolo *NC intra-flujo*. A medida que la probabilidad de error de trama aumenta, el mejor comportamiento se corresponde con la configuración sin retransmisiones en el nivel *MAC*, siendo la única de las curvas en la que se ve un comportamiento lineal, ajustándose a la Ecuación (6.15). La conclusión que se puede extraer a la vista de estas curvas es que *la solución óptima será deshabilitar este esquema*, lo que se hará en las simulaciones que se hagan a partir de este punto.

Se ha repetido el experimento cambiando a un modelo de canal a ráfagas (*HMP*). En la Figura 6.22 se muestra el *throughput* obtenido en las diferentes configuraciones en función de distancia entre los nodos S_1 y D_1. En primer lugar se comprueba el mayor grado de variabilidad en las medidas, recogido a través de un intervalo de confianza del 95 % mayor. Además, mientras que el efecto de las ráfagas de errores trae consigo graves consecuencias para el rendimiento de *TCP*, en el caso de la combinación *UDP/RLC* no se ve tan damnificado, presentando una dinámica similar a la vista en la Figura 6.21.

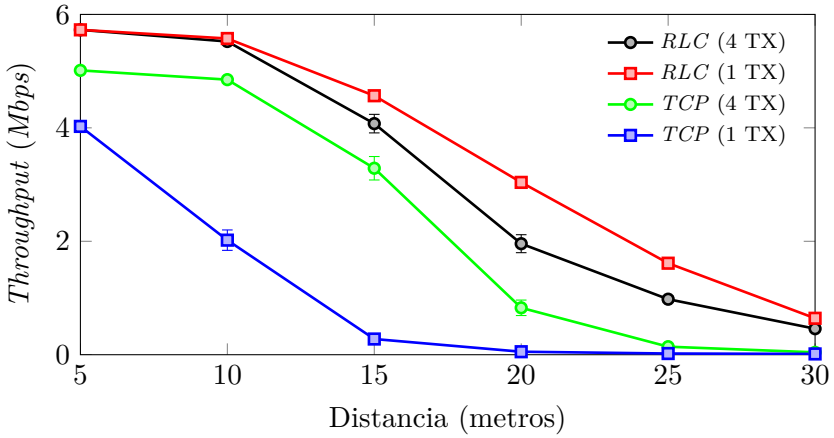
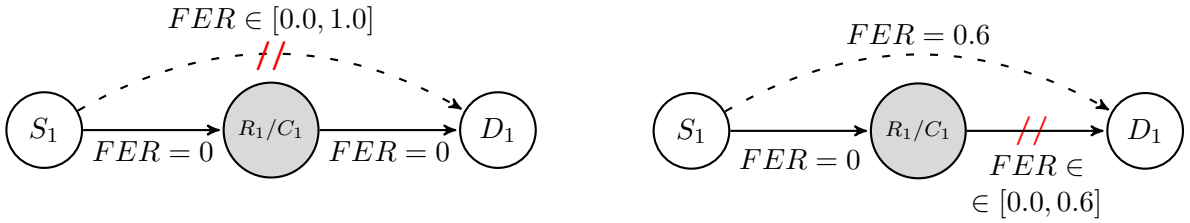


Figura 6.22: *Throughput* sobre un enlace con errores (modelo de canal *con* memoria)

6.5.2. Influencia de la recodificación

Una vez estudiado el comportamiento del mecanismo de codificación aleatorio *intra-flujo* extremo a extremo, se introduce en esta segunda fase la presencia de un nodo intermedio, que hará las veces de *router* entre origen y destino. Así, se pondrá especial interés en

resaltar las diferencias entre recombinar los paquetes codificados en los nodos intermedios (*RLNC*) o simplemente reenviar los mensajes de manera transparente (*RLSC*). Asimismo, se estudiarán dos circunstancias complementarias, como ilustra la Figura 6.23. En primer lugar se modificará la calidad del enlace (promiscuo) directo $S_1 \rightarrow D_1$, manteniendo el resto sin errores, tal y como muestra la Figura 6.23a. Posteriormente, se estudiará el efecto de los errores en el enlace $R_1/C_1 \rightarrow D_1$, como recoge la Figura 6.23b, manteniendo el enlace $S_1 \rightarrow D_1$ con una $FER = 0.6$.



(a) Influencia del enlace directo $S_1 \rightarrow D_1$

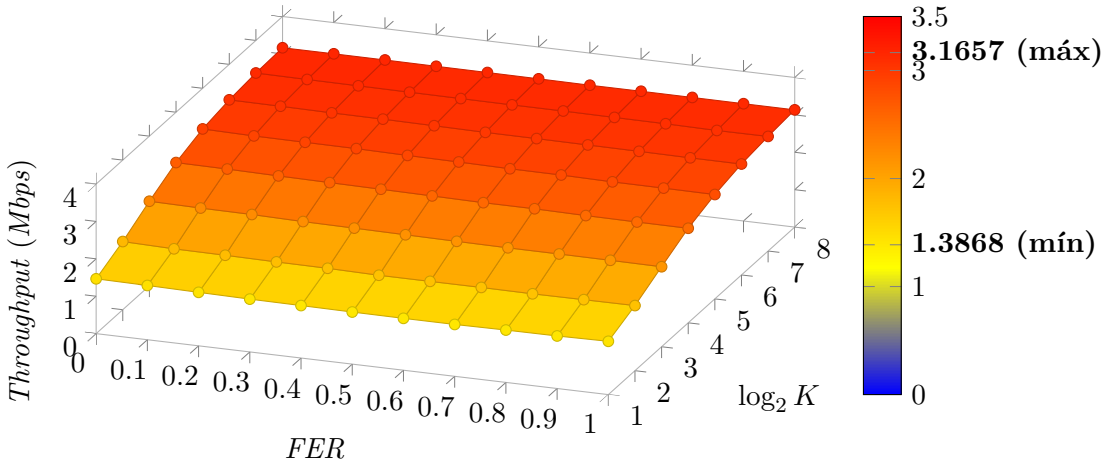
(b) Configuración alternativa

Figura 6.23: Escenarios de 3 nodos

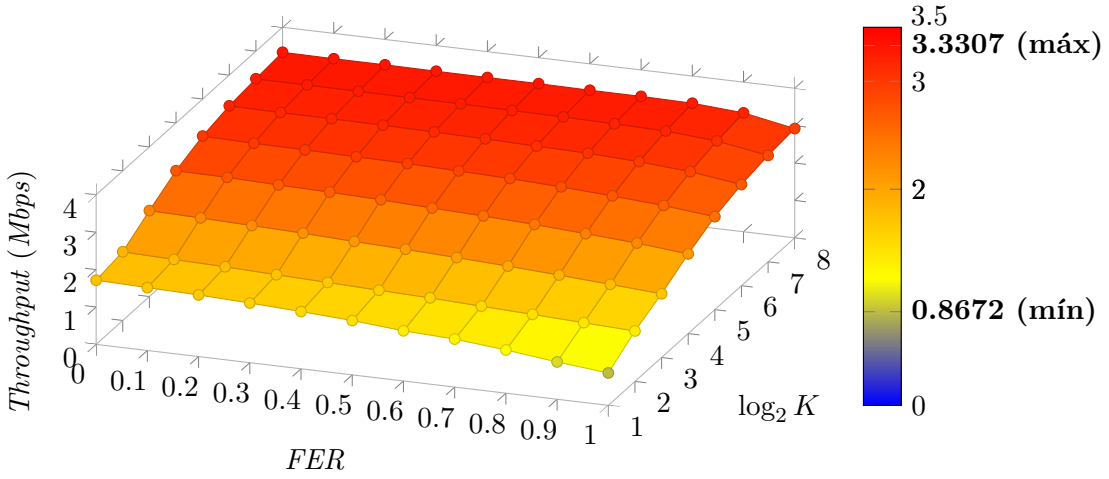
En todas las simulaciones que se realicen se eliminarán las retransmisiones en el nivel de enlace, utilizando bloques de 64 paquetes con coeficientes elegidos en el cuerpo binario $GF(2)$. Con respecto a *TCP*, se mantendrán las 3 retransmisiones *IEEE 802.11* por paquete.

Pasando ahora a describir los resultados correspondientes al primero de los escenarios, la Figura 6.24 representa la variación del *throughput* en función de la calidad del canal del enlace $S_1 \rightarrow D_1$. La gráfica superior (Figura 6.24a) se corresponde con el caso más sencillo, en el que el nodo R_1 se limita a redirigir los paquetes entre S_1 y D_1. A la vista de los resultados, se comprueba que la escucha oportunista no aporta beneficio alguno, manteniendo un comportamiento similar al estudiado en la Sección 6.5.1. En este caso, el nodo R_1 únicamente reenvía los mensajes que recibe, mientras que D_1 puede escuchar los paquetes enviados directamente por S_1. Así, cada vez que se reciba un paquete cualquiera p'_i sobre el enlace $S_1 \rightarrow D_1$, este último volverá a recibir esa misma combinación p'_i ($R_1 \rightarrow D_1$), por lo que será inmediatamente descartado. Se puede concluir, por tanto, que el número de paquetes codificados enviados en la red se va a mantener constante.

Por otro lado, el comportamiento al emplear las técnicas de recombinación es diferente, tal y como se muestra en la Figura 6.24b. En primer lugar, se comprueba que el valor del *throughput* máximo (3.3307 *Mbps* con $K = 256$) es un 5 % superior al observado en el caso anterior; no obstante, se puede apreciar una disminución del rendimiento cuando los valores de *FER* en el enlace $S_1 \rightarrow D_1$ son elevados (i.e. $FER \geq 0.8$), haciendo que el rendimiento mínimo baje hasta los 0.8672 *Mbps* con el tamaño de bloque más pequeño. En estas curvas intervienen tres factores: *(1)* la escucha oportunista permite recibir información linealmente independiente por varios caminos, ya que los coeficientes enviados por el nodo R_1 serán diferentes a los generados originalmente por S_1, aportando una mayor



(a) *RLSC*



(b) *RLNC*

Figura 6.24: Impacto de la escucha oportunista

variabilidad al sistema (lo que permite reducir el número de transmisiones). *(2)* La inteligencia adicional de C_1 le permite detectar paquetes con vectores linealmente dependientes, evitando su transmisión, que no aporta información adicional. *(3)* Además, se observa que cuando la calidad en este enlace es muy baja, el rendimiento global de la transmisión se ve perjudicado; la explicación es que cuando D_1 no es capaz de recibir las combinaciones lineales generadas por la fuente, utilizará aquellas enviadas por el nodo intermedio, que cuenta con una menor libertad para crear nuevos vectores, ya que opera con matrices incompletas.

Se puede concluir esta primera prueba afirmando que permitir a los nodos intermedios procesar los paquetes recibidos, transformando su contenido antes de enviarlos, se traduce en un efecto positivo, eliminando la propagación de paquetes no novedosos. Además, el

esquema se beneficia de la propiedad *broadcast* del medio inalámbrico, pasando de un encaminamiento *unipath* a otro en el que los destinos reciben diferentes combinaciones por múltiples caminos, lo que deriva en un descenso del número de transmisiones.

Pasando ahora a la segunda parte del análisis, se estudia ahora la influencia de los errores en el enlace $R_1/C_1 \rightarrow D_1$ (manteniendo una probabilidad de error de trama constante - 60 % - en el enlace directo $S_1 \rightarrow D_1$). En la Figura 6.25 se muestra el rendimiento observado en este segundo escenario, incluyendo las curvas pertenecientes a las técnicas de *RLNC* y *RLSC*, junto con el comportamiento de *TCP* según la ruta que utilice, existiendo dos alternativas: la que emplea dos saltos, $S_1 \rightarrow R_1/C_1 \rightarrow D_1$, o la que hace uso del enlace directo $S_1 \rightarrow D_1$.

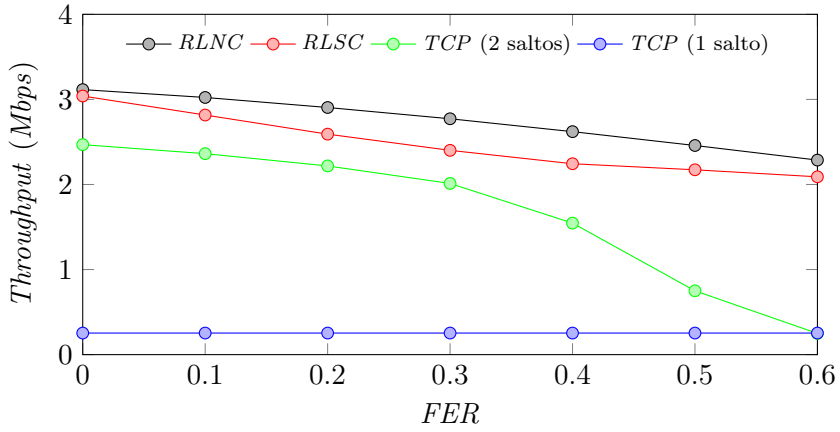


Figura 6.25: *Throughput* en un escenario de 3 nodos con errores (y escucha oportunista)

A la vista de los resultados, se puede apreciar como el rendimiento de las dos alternativas de *NC* presenta unas cifras similares al que se obtuvo cuando el enlace $R_1/C_1 \rightarrow D_1$ era ideal. En cuanto la calidad del canal se va deteriorando, las curvas de ambas soluciones se va separando, poniendo de manifiesto la mejora del grado adicional de aleatoriedad que induce el nodo C_1. En valores relativos, el uso de *RLNC* consigue mejorar hasta en un 20 % (para una *FER* del 30 %) el rendimiento observado con *RLSC*.

Con respecto al protocolo *TCP*, resulta evidente que su rendimiento es claramente inferior al de las técnicas de *NC*. En el caso de la ruta directa, se observa un *throughput* constante (≈ 0.25 *Mbps*), comprobándose que la utilización de una ruta más corta no es aconsejable si ésta experimenta una calidad claramente inferior a los enlaces de los caminos con mayor número de saltos. Para la ruta más larga ($S_1 \rightarrow R_1 \rightarrow D_1$), se ve una tendencia similar a la observada para un enlace punto a punto, manteniéndose siempre por debajo del *NC*. Como se ha discutido en repetidas ocasiones, a partir de una tasa de error cercana al 40 %, las ráfagas de errores serán lo suficientemente largas como para inducir la pérdida de segmentos de manera más continuada, penalizando el *throughput* de una forma mucho más agresiva.

6.5.3. Ventajas del encaminamiento oportunista

En esta tercera fase se extiende el análisis, incorporando la posibilidad de que sean varios los nodos los encargados de asegurar que la información llegue correctamente a su destino. Así, aprovechando la característica *pseudo-broadcast* del medio radio, será posible difundir un flujo de datos a través de múltiples caminos simultáneos. Si a esto se le suma además una solución de codificación como la proporcionada por el protocolo *NC intra-flujo*, la información percibida por los nodos finales no será necesariamente la misma que la generada en el origen, factor que contribuye a incrementar la probabilidad de recibir vectores con coeficientes linealmente independientes.

No obstante, al existir múltiples nodos que reciben y reenvían la información al mismo tiempo, se incrementa la contención del canal, lo que podría derivar en una bajada general del rendimiento.

Como escenario base para el análisis se utilizará la topología representada en la Figura 6.26, donde S_1 establece una conexión con D_1. De nuevo, la distancia entre ambos imposibilita el intercambio directo de mensajes. Como punto de apoyo, se dispone de un número variable de nodos intermedios R_i, que comparten el rol de potenciales *forwarders*. Su presencia no modificará el comportamiento de la combinación de *TCP* con un encaminamiento clásico *store-and-forward*, que seleccionará de antemano una única ruta.

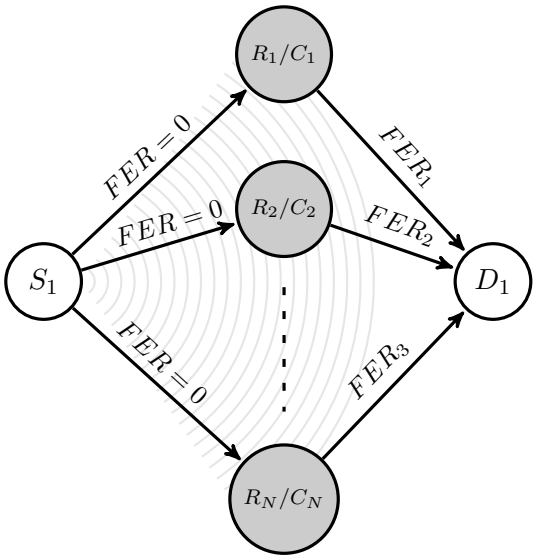


Figura 6.26: Escenario canónico para el estudio de las ventajas del encaminamiento oportunista

En el caso de utilizar un encaminamiento oportunista, habrá que decidir qué nodo C_i retransmitirá un paquete en un instante dado, para lo que se recurrirá a tres soluciones diferentes de coordinación, descritas a continuación:

1. El primero de ellos se corresponde con la situación más pesimista, en la que cualquier nodo que reciba en su entidad *NC* un paquete de datos cuya dirección *IP* destino no coincida con la del propio dispositivo, generará un reenvío inmediato. Se le denominará encaminamiento "**simple**".

2. Como una solución intermedia, se ha definido un mecanismo oportunista que reduce el número de transmisiones redundantes mediante un sencillo procedimiento: asumiendo que un nodo conoce de antemano el número de posibles *forwarders*, N, la probabilidad de que se produzca un reenvío al recibir un paquete (1 en el encaminamiento *simple*) se multiplicará por un factor $\lambda_i = \frac{1}{N}$. Esta solución asegura que, en promedio, sólo uno de los nodos intermedios envíe el paquete, reduciendo la contención causada por el acceso simultáneo de varios de ellos. A esta técnica se la ha asignado el sobrenombre de **equiprobable**.

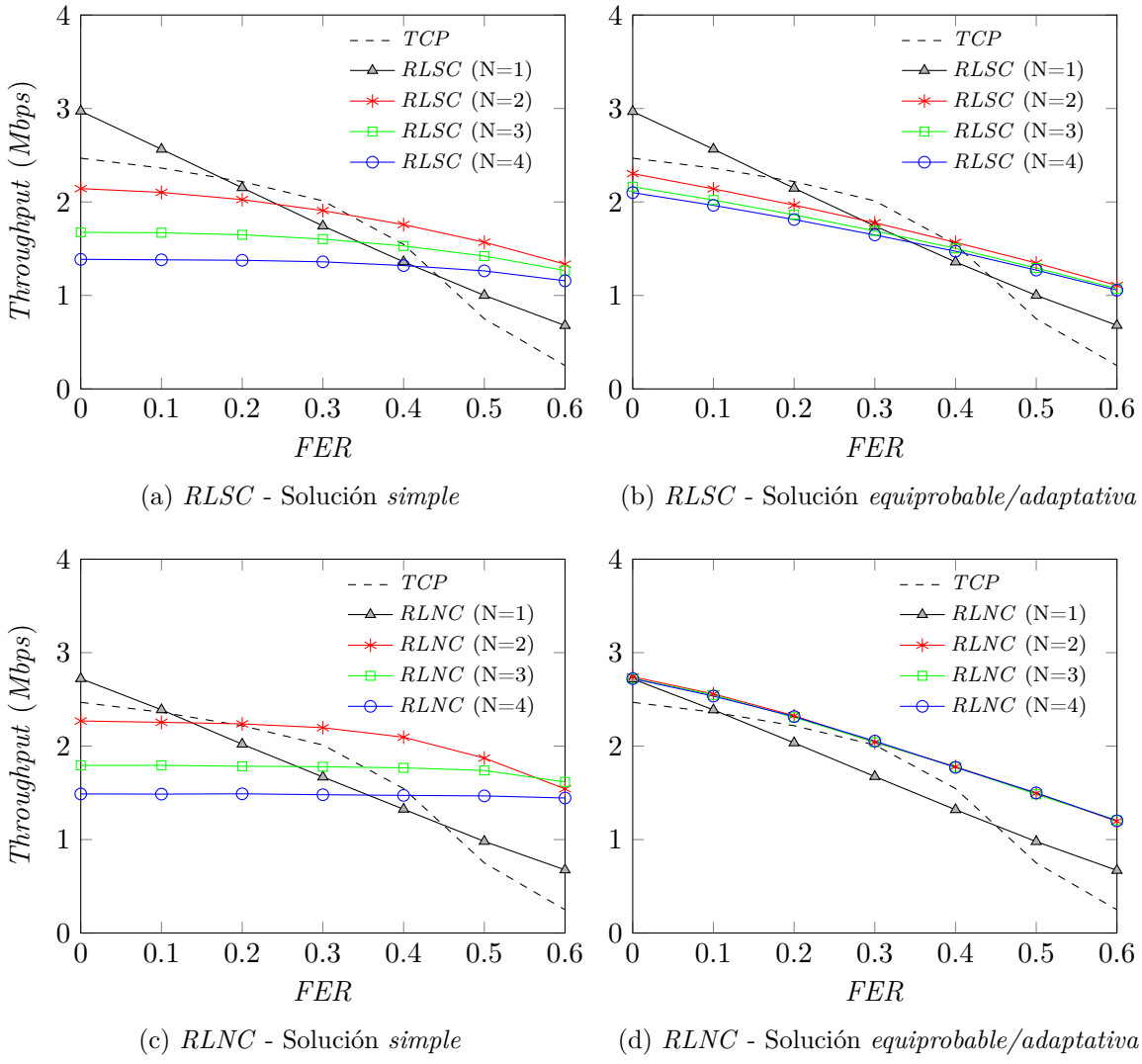
3. Por último, se añade una capa de inteligencia adicional al motor de decisión, que permite a los nodos adaptarse a la calidad del enlace. El principal inconveniente del modelo anterior es que no es capaz de otorgar ningún tipo de prioridad a aquellos nodos cuyos enlaces con el siguiente salto presenten unas mejores condiciones. Como solución, se monitoriza el nivel físico (*cross-layer*), de forma que cada nodo estima la probabilidad de error de trama en su enlace con el destino[17] para favorecer a aquellas transmisiones con una mayor probabilidad de éxito. A partir de estos valores se calculará la probabilidad λ_i de reenviar cada paquete recibido p' (o su versión recombinada p''), tal y como se ve en la Ecuación 6.16.

$$\lambda_i = \frac{1}{FER_i} \cdot \frac{1}{\sum_{j=1}^{N} \frac{1}{FER_j}} \tag{6.16}$$

   Cuando la probabilidad de error de trama es igual en todos los enlaces, $\lambda_i = \frac{1}{N}$, se llega a la expresión vista en el encaminamiento *equiprobable*. Como referencia, a este tercer esquema se le conocerá como **adaptativo**.

Se considera en primer lugar un escenario en el que el número de nodos intermedios R_i/C_i varía desde $i = 1$ hasta $i = 4$. Como se ve en la Figura 6.26, el primer enlace desde S_1 hacia los intermedios R_i/C_i es ideal, garantizando así que tengan la misma información (sus matrices de coeficientes $\mathcal{C}_i$ estarán sincronizadas en todo momento). Sin embargo, los enlaces finales $R_i/C_i \rightarrow D_1$ estarán configurados con una misma $FER \geq 0$, $FER_1 = FER_2 = FER_3 = FER_4$. En un primer análisis se averiguará: *(1)* la capacidad del encaminamiento oportunista para enmascarar las pérdidas en canales hostiles y *(2)* la eficacia de la coordinación en el reenvío por parte de los nodos intermedios. El comportamiento de las soluciones *equiprobable* y *adaptativa* será idéntico, ya que las tasas de error en todos los enlaces es la misma durante todo el proceso.

[17] En este caso, se conectarán las diferentes entidades *NC* con una instancia *monitora* que contiene los atributos que configuran la *FER* (constante) de cada uno de los enlaces.



(a) *RLSC* - Solución *simple*

(b) *RLSC* - Solución *equiprobable/adaptativa*

(c) *RLNC* - Solución *simple*

(d) *RLNC* - Solución *equiprobable/adaptativa*

Figura 6.27: Comparación de los diferentes esquemas *NC intra-flujo* sobre enlaces simétricos $R_i/C_i \rightarrow D_1$

En la Figura 6.27 se presentan los valores de *throughput* obtenidos para cada configuración. Lo primero que llama la atención es que el rendimiento obtenido por el protocolo *NC* en condiciones ideales con un único nodo intermedio es inferior cuando C_1 procesa los paquetes, recodificando su contenido (i.e. *RLNC*). Como se vio en la Figura 6.24, si D_1 no es capaz de recibir los mensajes codificados directamente de la fuente a través del enlace $S_1 \rightarrow D_1$, las ventajas asociadas a la recodificación desaparecen, ya que aumenta la probabilidad de que las recombinaciones generadas en los intermedios sean linealmentes dependientes. Además, la falta de un enlace directo origen-destino penaliza el rendimiento global, viéndose inferior al mostrado por *TCP* en un encaminamiento *store-and-forward*, como se aprecia cuando la *FER* se sitúa entre el 10 % y el 40 %. Como es lógico, el com-

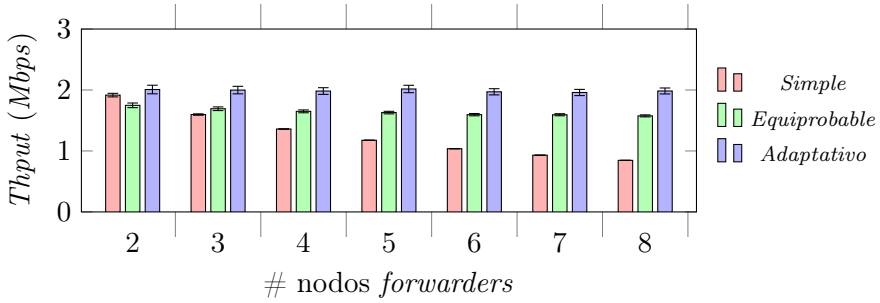
portamiento en todos estos casos es idéntico, ya que el nodo R_1/C_1 es el único elemento que se sitúa entre los dos extremos.

En cuanto al comportamiento general de los diferentes esquemas de encaminamiento oportunista ($N > 1$), no se observan grandes diferencias entre el *throughput* obtenido por *RLSC* y *RLNC* para la estrategia *simple* (Figuras 6.27a y 6.27b), aunque la segunda mejora levemente a la primera, a medida que la calidad del canal se va deteriorando, ya que la recodificación permite compensar de alguna manera el impacto de las pérdidas. Sin embargo, el aspecto más interesante es que, para valores de *FER* bajos, habrá casos en los que el uso de múltiples nodos *forwarders* penaliza el rendimiento, viéndose incluso inferior a la utilización de un encaminamiento clásico, sin retransmisiones oportunistas ($N = 1$). Esto se debe a la *contención adicional* causada por estas estaciones que, si no se gestiona adecuadamente, dará lugar a mensajes redundantes. Como puede verse, cuantas más estaciones intervengan en el proceso de reenvío, menor será el *throughput* percibido en el destino. Tratando de extraer el lado positivo de esto, se puede ver que el uso de varios nodos intermedios ofrece una mayor resistencia a las pérdidas, manteniendo el rendimiento constante hasta que la capacidad de enmascarar los errores se ve superada, lo que ayuda a mejorar el rendimiento que se obtiene con $N = 1$ en los canales con mayor *FER*. Reutilizando unos sencillos cálculos que se introdujeron en la Sección 6.3.2, se comprueba fácilmente que la probabilidad de recibir un paquete en este escenario, cuando todos los nodos intermedios replican el mensaje recibido, será de $(1-(1-FER)^N)$, siendo N el número de *forwarders* potenciales. Con esto se puede extrapolar que, a medida que N aumenta, menor será el número de transmisiones necesarias para asegurar que al menos un paquete original llegue al destino.

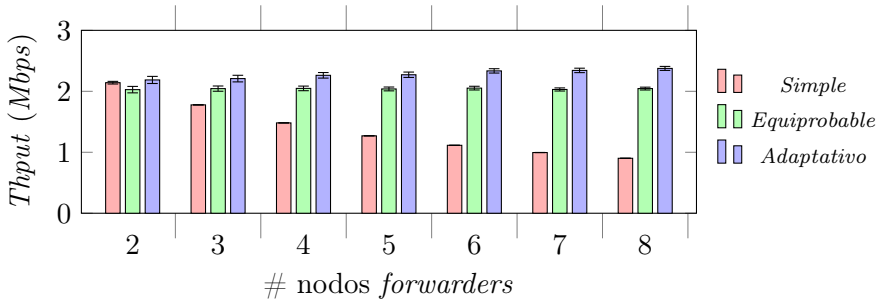
En lo que respecta a las versiones más avanzadas del encaminamiento oportunista (Figuras 6.27c y 6.27d), se verá más claramente la diferencia entre ambas técnicas de codificación, haciendo más palpable si cabe el beneficio de la recodificación en los nodos intermedios. Así, los valores de *throughput* se sitúan siempre por encima de los ofrecidos por *TCP* cuando haya solamente un nodo entre S_1 y D_1. En este caso, los paquetes generados por los nodos intermedios no son los mismos, aumentando la probabilidad de recibir vectores con coeficientes linealmente independientes. Por el contrario, en el esquema *RLSC* solamente se perciben mejoras para valores de *FER* superiores al 30 %. Sin embargo, para las condiciones de canal más hostiles ($FER > 0.4$), el hecho de reducir el número de transmisores redundantes causa un rendimiento inferior al visto para el encaminamiento simple, ya que se reduce la probabilidad de recibir los datos en el destino.

Finalmente, prestando atención a la curva que resume el comportamiento de *TCP* (que es la misma en todas las configuraciones), se vuelve a poner de manifiesto que, aunque mejore las prestaciones de alguna de las soluciones propuestas cuando la *FER* es baja, su rendimiento cae abruptamente para calidades peores ($FER \geq 0.4$), siendo sensiblemente peor que el del protocolo *NC* en cualquiera de sus configuraciones.

En un segundo análisis se modifica la configuración de las calidades en los enlaces $R_i/C_i \rightarrow D_1$, eligiendo un valor de *FER* aleatorio entre 0 y 0.6, independiente para cada uno de ellos. Además, se incrementará el número de nodos intermedios, estudiando escenarios que van desde los 2 hasta los 8 dispositivos. Con esta nueva configuración, las soluciones *equiprobable* y *adaptativa* dejarán de ofrecer las mismas probabilidades de transmisión λ_i.



(a) *Store-and-Forward*



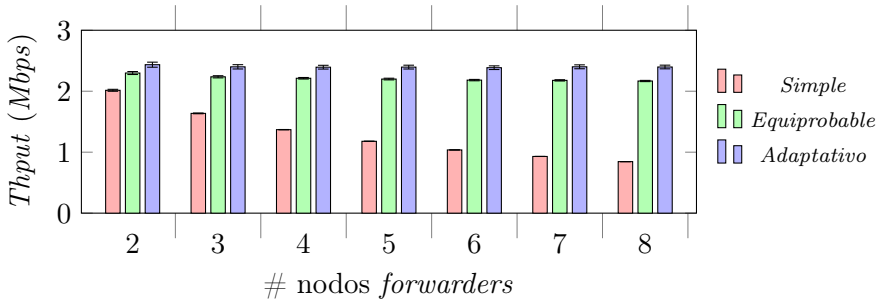
(b) *Network Coding*

Figura 6.28: *Throughput* medio en función del número de nodos intermedios

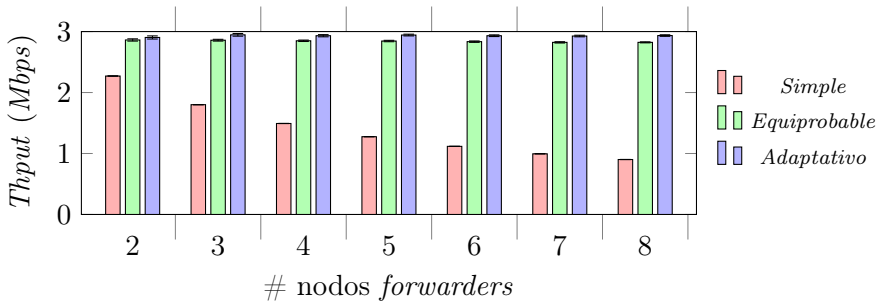
La Figura 6.28 representa el valor medio del *throughput* y el intervalo de confianza del 95 % en función del número de nodos intermedios R_i/C_i. A simple vista, se comprueba la ganancia proporcionada por las alternativas *probabilísticas* sobre la solución *simple*, cuyo rendimiento, además de ser inferior en todos los casos (salvo con $N = 2$), decrece rápidamente a medida que el número de nodos intermedios aumenta, consecuencia de una mayor contención. Por otra parte, la asignación de probabilidades de transmisión en los nodos intermedios consigue independizar el *throughput* medio, lo que permite concluir que existirá un umbral a partir del cual añadir nodos adicionales no va a reportar ningún beneficio. Además, la utilización de la información proporcionada por los niveles físicos (*cross-layer*) permite adaptarse a las condiciones del canal de una forma más dinámica, consiguiendo mantener el rendimiento por encima de los 2 *Mbps*. Por último, hay que destacar las prestaciones del esquema *RLNC*, que son superiores a las ofrecidas por una codificación exclusiva en la fuente.

Manteniendo el mismo escenario, se introduce la posibilidad de que D_1 escuche directamente los paquetes enviados por S_1. Así, se evaluará el beneficio conjunto que puede

aportar la combinación escucha/encaminamiento oportunista a una transmisión *NC intra-flujo*. Como en situaciones anteriores, la probabilidad de error en este enlace ($S_1 \rightarrow D_1$) se ha fijado en 0.6, presentando los resultados con esta nueva configuración en la Figura 6.29.



(a) *Store-and-Forward*



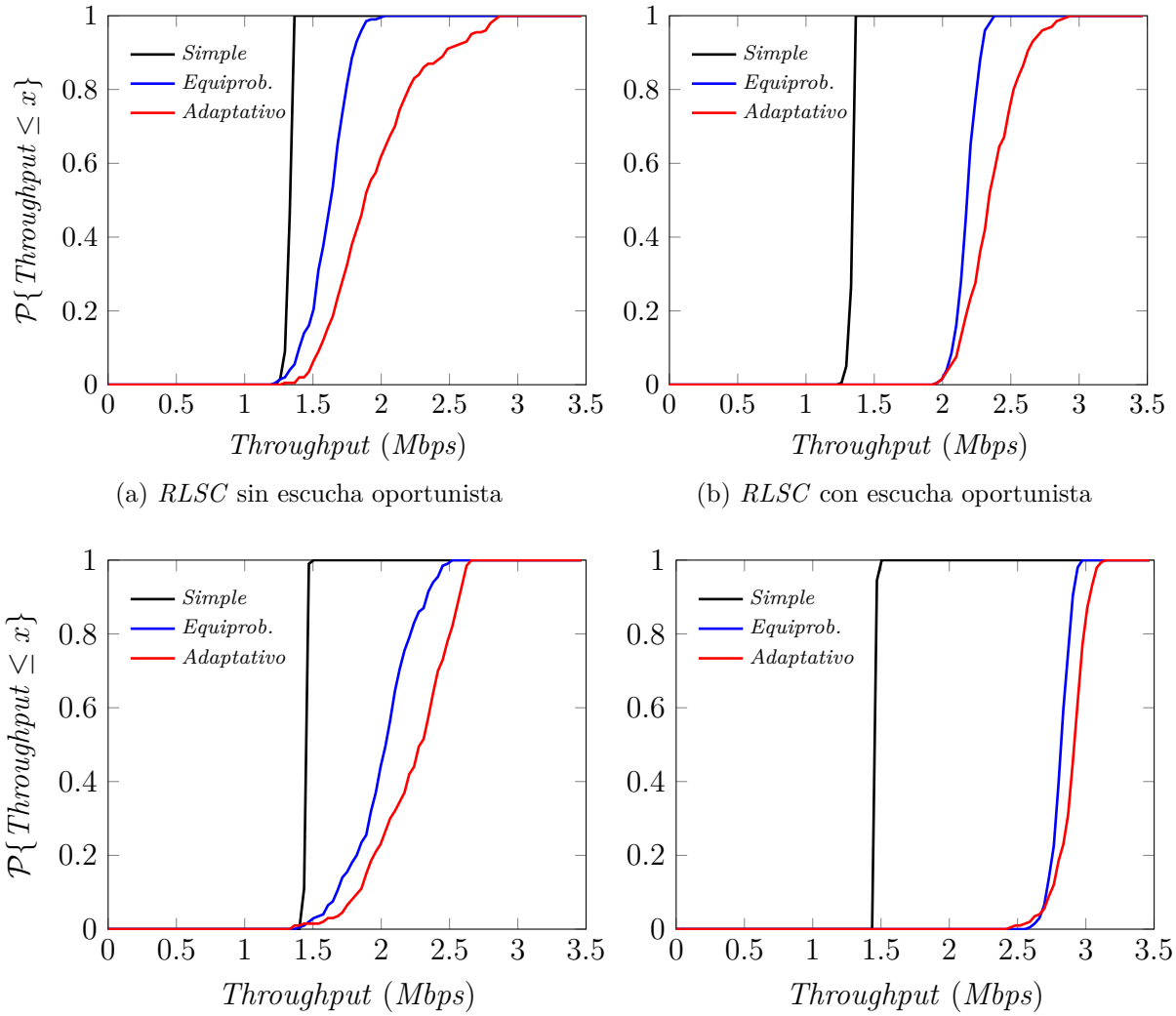
(b) *Network Coding*

Figura 6.29: *Throughput* medio en función del número de nodos intermedios (escucha oportunista habilitada)

Como ya se comentó en la Sección 6.5.2, la escucha directa en el destino, además de habilitar rutas alternativas más cortas, reducirá el número de transmisiones necesarias por parte de los nodos intermedios, con el consiguiente aumento del rendimiento. Además, mientras que el encaminamiento *simple* ofrece unos valores muy similares a los vistos en la figura anterior, las dos soluciones *probabilísticas* se benefician de la escucha oportunista para incrementar el valor del *throughput*. A la vista de los resultados, la operación conjunta de la escucha oportunista y las recombinaciones en los nodos intermedios aporta una clara mejoría con respecto al simple reenvío de los paquetes originales, alcanzando unas prestaciones cercanas a las que se observaron cuando el enlace $C_1 \rightarrow D_1$ era ideal (Figura 6.25), poniendo de manifiesto el potencial de este tipo de esquemas de transmisión.

Finalmente se representa, para completar el análisis, la función de distribución acumulada del *throughput* para cada una de las configuraciones anteriores, manteniendo las condiciones definidas en la etapa anterior (e.g. errores aleatorios, escucha oportunista en el enlace $S_1 \rightarrow D_1$ habilitada/deshabilitada, etc.). En este caso, se fijará el número de nodos intermedios ($N = 4$), aumentando a 200 simulaciones independientes por configura-

ción. Los resultados correspondientes a cada experimento se encuentran representados en la Figura 6.30.



(a) *RLSC* sin escucha oportunista

(b) *RLSC* con escucha oportunista

(c) *RLNC* sin escucha oportunista

(d) *RLNC* con escucha oportunista

Figura 6.30: Función de distribución del *throughput* en un escenario con 4 nodos intermedios

El primer aspecto a destacar es que el rendimiento con el encaminamiento *simple* es siempre inferior a 1.5 *Mbps*, con un comportamiento muy predecible, como también se vio en la Figura 6.27. Se trata por lo tanto de un claro ejemplo de configuración ineficiente, ya que al estar más enfocado a la corrección de errores que a la mejora del *throughput*, la redundancia que añade limita en gran medida el rendimiento alcanzado. En un punto intermedio se comprueba como la estrategia probabilística *equiprobable* supone una mejora notable, superando el umbral de 1.5 *Mbps* en todas las configuraciones y garantizando, incluso, un mínimo de 2.5 *Mbps* para la configuración más completa (*RLNC* con escucha oportunista, como se ve en la Figura 6.30d). Finalmente, la tercera de las curvas ilustra el

comportamiento del encaminamiento *adaptativo*, mostrándose en todo momento superior al de las otras dos alternativas. Únicamente en la configuración más completa (Figura 6.30d) el rendimiento de la solución *equiprobable* se acerca al visto en esta versión más avanzada; no obstante, este emparejamiento se debe a que ambas alternativas se encuentran próximas al umbral óptimo de *throughput* que se puede alcanzar con las condiciones del escenario estudiado.

6.6 Conclusiones y líneas futuras

En este Capítulo se presenta un protocolo *NC intra-flujo* que surge de la idea originalmente descrita en [Ho03a], que promueve la elección de coeficientes aleatorios a la hora de combinar la información de los paquetes. Este esquema simplificará enormemente las operaciones de codificación, recodificación y posterior decodificación, reduciendo drásticamente la complejidad de protocolos que adopten estos principios. Además, se ha optado por la combinación de un protocolo de transporte más simple, *UDP*, con el mencionado módulo de *RLC intra-flujo*. Como resultado, se ofrece un servicio que garantiza la recepción ordenada de todos los paquetes en el destino, equivalente por tanto al de *TCP*. Además, al tratarse de una solución enfocada a su empleo sobre redes inalámbricas, ha sido diseñada para explotar las ventajas inherentes a este medio de propagación, entre las que destaca su naturaleza *broadcast*, que promueve un nuevo concepto de encaminamiento que no se limita a una única ruta por conexión, existiendo varios caminos por los que la información puede llegar al receptor.

Se han propuesto dos estrategias: una en la que los nodos intermedios simplemente reenvían los paquetes hacia el siguiente salto (*RLSC*) y otra en la que procesan y recombinan los mensajes codificados antes de su reenvío (*RLNC*). Otro de los elementos fundamentales de la implementación llevada a cabo es el uso de una serie de soluciones *cross-layer*, que permiten conectar la capa *NC* con diferentes funciones situadas en los niveles inferiores. Gracias al empleo de estas técnicas se ha logrado conseguir una tasa de inyección óptima de paquetes codificados mediante la cual se evita la saturación del *buffer* del nivel de enlace. Además, tras la recepción de un mensaje de reconocimiento se borran selectivamente los paquetes restantes en el *buffer* del nivel *MAC*, para reducir al máximo el envío de información desfasada.

Uno de los elementos más problemáticos que se asocian a estos mecanismos es el coste operacional asociado a las tareas de codificación/recodificación/decodificación. En concreto, el esquema propuesto emplea el álgebra de cuerpos finitos de Galois sobre matrices de dimensión $K \times K$. Así, el cálculo del rango y de la matriz inversa pueden limitar el rendimiento del protocolo en un sistema real. Para poder hacer frente a este problema, se ha llevado a cabo una búsqueda exhaustiva de una herramienta externa que ofrezca una mayor eficiencia a la hora de ejecutar estas operaciones, concluyendo que la pareja de librerías `M4ri/M4rie` es la solución más atractiva.

Para evaluar las prestaciones ofrecidas por este protocolo en despliegues inalámbricas, se ha llevado a cabo un profundo análisis en base a simulaciones sobre la plataforma *ns-3*, donde se caracteriza el rendimiento del protocolo, en tres etapas diferenciadas.

En la primera de ellas se ha estudiado el rendimiento del protocolo en el escenario más básico posible, donde los nodo transmisor y receptor están conectados a través de un único enlace. Se ha caracterizado el efecto de los dos factores de penalización que introduce un esquema de codificación aleatoria: la recepción de transmisiones no útiles (los vectores que transportan son linealmente dependientes a los que ya tiene almacenados el nodo receptor) y el tiempo asociado a la transmisión de los reconocimientos del nivel *NC*. Los resultados han dejado claro que, cuanto mayor sea el tamaño del bloque (K) y el orden del cuerpo, $GF(2^q)$, menor será la influencia de estos factores. Por otro lado, transmitir bloques grandes causa un retardo más elevado entre decodificaciones consecutivas, así como un mayor coste computacional en las operaciones matriciales; además, los cuerpos de Galois de mayor orden necesitarían una sobrecarga mayor en la cabecera del protocolo, lo que podría disminuir el rendimiento. En situaciones ideales, se ha comprobado que el *throughput* máximo se consigue con bloques de 256 paquetes y coeficientes elegidos en un cuerpo binario $GF(2)$, obteniendo una mejora cercana al 20 % frente al *TCP* tradicional. Esta ventaja será mayor aún sobre canales hostiles, mientras que *TCP* no es capaz de soportar tasas de error superiores al 50 %, la combinación *UDP/RLC* mantiene unas cifras de rendimiento aceptables. Por último, se comprueba que el uso de retransmisiones de nivel de enlace no aporta beneficio alguno al rendimiento.

Posteriormente se ha incrementado la distancia entre los nodos, incorporando un elemento intermedio que reenvía los paquetes entre el origen y el destino; además, aprovechando la naturaleza *broadcast* del medio inalámbrico, el receptor podrá escuchar directamente los mensajes enviados por la fuente. Se ha comparado el rendimiento de las dos posibles operaciones de los nodos intermedios, comprobando que la redundancia generada cuando éstos solamente reenvían la información que escuchan es compensada cuando además recombinan los paquetes. Se ha observado que cualquiera de las estrategias analizadas, combinadas con la escucha oportunista en el destino, proporciona unas prestaciones muy superiore a las vistas al emplear *TCP* sobre una única ruta.

Para concluir con las simulaciones, se ha considerado un nuevo escenario en el que el número de nodos intermedios varía, derivando en la aparición de caminos alternativos entre el origen y el destino. Se han utilizado los principios de las técnicas de *encaminamiento oportunista*, que establecen unas probabilidades de transmisión en los nodos. A través de tres estrategias (*simple*, *equiprobable* y *adaptativa*), se ha comprobado que la utilización de múltiples rutas permite enmascarar las pérdidas producidas en el canal e incrementar el rendimiento global. Se ha comprobado además que el uso de un esquema *cross-layer* basado en la monitorización de la calidad del enlace puede adaptarse mejor a las condiciones cambiantes de la red. Además, los resultados ponen de manifiesto que la recombinación en los nodos intermedios proporciona una ganancia adicional de rendimiento con respecto a un esquema tradicional.

A continuación se enumeran las cuestiones que deberían afrontarse en el futuro.

- Sería interesante buscar alguna solución que permita reducir el tiempo computacional de cálculo del rango y de la matriz inversa, tomando como referencia algunos trabajos en los que se proponen técnicas más avanzadas, como el uso de ventanas deslizantes [Wu12b] o la decodificación inmediata [Li11b].

- También se podría reducir la sobrecarga adicional que implica incorporar el vector con los coeficientes utilizados para la codificación en la cabecera del protocolo. Las soluciones serían: *(1)* la utilización de generadores de números pseudo-aletorios en los nodos, transmitiendo exclusivamente la semilla utilizada en cada paquete [Kim12a]; *(2)* llevar a cabo un establecimiento previo de la conexión en el que se negocien los parámetros (i.e. tamaño del bloque, orden del cuerpo, etc.), permitiendo eliminar estos campos de los mensajes de datos.

- La memoria disponible en algunos dispositivos (por ejemplo, sensores) puede suponer también una limitación, siendo imposible disponer de *buffers* ilimitados. Para ello se podría utilizar el mecanismo descrito en la Sección 6.2.1, basado en el acceso directo al contenido de la capa de aplicación (evitando así tener la información duplicada en el nivel *NC*).

- Para llevar las técnicas de encaminamiento oportunista a la práctica, habrá que desarrollar un protocolo de señalización que permita difundir la información pertinente (número de vecinos, calidad de los enlaces, etc.) de manera dinámica.

- Gracias a estas técnicas de encaminamiento oportunista, será posible generalizar y caracterizar el rendimiento del esquema *RLNC* sobre topologías más complejas.

- También podría resultar interesante estudiar el posible impacto del uso de dispositivos con múltiples interfaces de red, permitiendo dividir el tráfico a través de diferentes caminos.

- Finalmente, se podría valorar la inclusión de algún tipo de mecanismo de control de flujo, que permitiera optimizar el rendimiento en función de las aplicaciones y del número de conexiones activas. La entidad *NC* podría así regular la tasa de inyección de paquetes hacia los niveles inferiores [Fu14], o modificar dinámicamente los parámetros de codificación (tamaño de bloque y orden del cuerpo), en función de los requerimientos concretos del sistema.

Merece la pena remarcar una vez más la política *open-source* que se ha seguido durante el desarrollo de la presente Tesis; poniéndose a disposición de la comunidad científica todo el código fuente, documentación y resultados que se han descrito en este Capítulo [Góme].

Referencias

[Adel09] M. Adeli y Huaping Liu. «Secure network coding with minimum overhead based on hash functions». En: *Communications Letters, IEEE* 13.12 (dic. de 2009), págs. 956-958. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2009.12.091648.

[Ahls00] R. Ahlswede, Ning Cai, S.-Y.R. Li y R.W. Yeung. «Network information flow». En: *Information Theory, IEEE Transactions on* 46.4 (jul. de 2000), págs. 1204-1216. ISSN: 0018-9448. DOI: 10.1109/18.850663.

[Aho74] Alfred V. Aho y John E. Hopcroft. *The Design and Analysis of Computer Algorithms*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1974. ISBN: 0201000296.

[Albr11] Martin R. Albretch. *The M4RIE library for dense linear algebra over small fields with even characteristic*. 2011.

[Alwi13] C. de Alwis, H.K. Arachchi, A. Fernando y A. Kondoz. «Towards minimising the coefficient vector overhead in random linear Network Coding». En: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. Mayo de 2013, págs. 5127-5131. DOI: 10.1109/ICASSP.2013.6638639.

[Amir95] E. Amir, H. Balakrishnan, S. Seshan y R.H. Katz. «Efficient TCP over networks with wireless links». En: *Hot Topics in Operating Systems, 1995. (HotOS-V), Proceedings., Fifth Workshop on*. Mayo de 1995, págs. 35-40. DOI: 10.1109/HOTOS.1995.513451.

[Bisw05] Sanjit Biswas y Robert Morris. «ExOR: opportunistic multi-hop routing for wireless networks». En: *SIGCOMM Comput. Commun. Rev.* 35.4 (ago. de 2005), págs. 133-144. ISSN: 0146-4833. DOI: 10.1145/1090191.1080108. URL: http://doi.acm.org/10.1145/1090191.1080108.

[Bouk14] Azzedine Boukerche y Amir Darehshoorzadeh. «Opportunistic Routing in Wireless Networks: Models, Algorithms, and Classifications». En: *ACM Comput. Surv.* 47.2 (nov. de 2014), 22:1-22:36. ISSN: 0360-0300. DOI: 10.1145/2635675. URL: http://doi.acm.org/10.1145/2635675.

[Byer98] John W. Byers, Michael Luby, Michael Mitzenmacher y Ashutosh Rege. «A Digital Fountain Approach to Reliable Distribution of Bulk Data». En: *SIGCOMM Comput. Commun. Rev.* 28.4 (oct. de 1998), págs. 56-67. ISSN: 0146-4833. DOI: 10.1145/285243.285258. URL: http://doi.acm.org/10.1145/285243.285258.

[Chac07] Szymon Chachulski, Michael Jennings, Sachin Katti y Dina Katabi. «Trading structure for randomness in wireless opportunistic routing». En: *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM '07. Kyoto, Japan: ACM, 2007, págs. 169-180. ISBN: 978-1-59593-713-1. DOI: 10.1145/1282380.1282400. URL: http://doi.acm.org/10.1145/1282380.1282400.

[Cham10] M.-L. Champel, K. Huguenin, A.-M. Kermarrec y N. Le Scouarnec. «LT Network Codes». En: *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*. Jun. de 2010, págs. 536-546. DOI: 10.1109/ICDCS.2010.14.

[Chen10] Chien-Chia Chen, Soon Y Oh, Phillip Tao, Mario Gerla y MY Sanadidi. «Pipeline network coding for multicast streams». En: *Proceedings of the 5th International Conference on Mobile Computing and Ubiquitous Networking (ICMU), Seattle, USA*. 2010.

[Chen11] Chien-Chia Chen, Clifford Chen, Soon Y. Oh, Joon-Sang Park, Mario Gerla y M.Y. Sanadidi. «ComboCoding: Combined intra-/inter-flow network coding for {TCP} over disruptive {MANETs}». En: *Journal of Advanced Research* 2.3 (2011). Special issue on Mobile Ad-Hoc Wireless Networks, págs. 241-252. ISSN: 2090-1232. DOI: http://dx.doi.org/10.1016/j.jare.2011.05.002. URL: http://www.sciencedirect.com/science/article/pii/S2090123211000671.

[Chi12] Yang Chi y D.P. Agrawal. «Murco: An opportunistic network coding framework in multi-radio networks». En: *Communications (ICC), 2012 IEEE International Conference on*. Jun. de 2012, págs. 5371-5375. DOI: 10.1109/ICC.2012.6363728.

[COMANDER] *COMANDER*. http://www.mc-comander.eu/. 2012.

[Cout03] Douglas S. J. de Couto, Daniel Aguayo, John Bicket y Robert Morris. «A High-throughput Path Metric for Multi-hop Wireless Routing». En: *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*. MobiCom '03. San Diego, CA, USA: ACM, 2003, págs. 134-146. ISBN: 1-58113-753-2. DOI: 10.1145/938985.939000. URL: http://doi.acm.org/10.1145/938985.939000.

[Dier08] T. Dierks y E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246 (Proposed Standard). Updated by RFCs 5746, 5878, 6176. Internet Engineering Task Force, ago. de 2008. URL: http://www.ietf.org/rfc/rfc5246.txt.

[Dima10] A.G. Dimakis, P.B. Godfrey, Y. Wu, M.J. Wainwright y K. Ramchandran. «Network Coding for Distributed Storage Systems». En: *Information Theory, IEEE Transactions on* 56.9 (sep. de 2010), págs. 4539-4551. ISSN: 0018-9448. DOI: 10.1109/TIT.2010.2054295.

[DIWINE] *Dense Cooperative Wireless Cloud Network - DIWINE).* http://diwine-project.eu. 2015.

[Duma] Jean-Guillaume Dumas, Pascal Giorgi y Clément Pernet. *FFPACK: Finite Field Linear Algebra Package.*

[Ferr09] Diogo Ferreira, Luísa Lima y João Barros. «NECO: NEtwork COding simulator.» En: *SimuTools*. Ed. por Olivier Dalle, Gabriel A. Wainer, L. Felipe Perrone y Giovanni Stea. ICST, 15 de mayo de 2009, pág. 52. ISBN: 978-963-9799-45-5. URL: http://dblp.uni-trier.de/db/conf/simutools/simutools2009.html#FerreiraLB09.

[FFLAS] *FFLAS-FFPACK. Finite Field Linear Algebra subroutines package.* http://www-ljk.imag.fr/membres/Jean-Guillaume.Dumas/FFLAS/index.html.

[Fitz10] Frank Fitzek, Morten V. Pedersen, Janus Heide y Muriel Medard. «Network Coding Applications and Implementations on Mobile Devices». En: *Proceedings of the 5th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. Association for Computing Machinery, 2010. ISBN: 978-1-4503-0278-4. DOI: 10.1145/1868612.1868628.

[Frag06] Christina Fragouli, Jean-Yves Le Boudec y Jörg Widmer. «Network Coding: An Instant Primer». En: *SIGCOMM Comput. Commun. Rev.* 36.1 (ene. de 2006), págs. 63-68. ISSN: 0146-4833. DOI: 10.1145/1111322.1111337. URL: http://doi.acm.org/10.1145/1111322.1111337.

[Fu14] A. Fu, P. Sadeghi y M. Medard. «Dynamic Rate Adaptation for Improved Throughput and Delay in Wireless Network Coded Broadcast». En: *Networking, IEEE/ACM Transactions on* 22.6 (dic. de 2014), págs. 1715-1728. ISSN: 1063-6692. DOI: 10.1109/TNET.2013.2292613.

[Gall63] Robert G. Gallager. «Low-Density Parity-Check Codes». Tesis doct. 1963.

[Góme] David Gómez, Eduardo Rodríguez, Mario Puente y Ramón Agüero. *Network Coding architecture source code and documentation (ns-3).* https://github.com/dgomezunican/network-coding-ns3.

[Guo11] Bin Guo, Hongkun Li, Chi Zhou y Yu Cheng. «Analysis of General Network Coding Conditions and Design of a Free-Ride-Oriented Routing Metric». En: *Vehicular Technology, IEEE Transactions on* 60.4 (mayo de 2011), págs. 1714-1727. ISSN: 0018-9545. DOI: 10.1109/TVT.2011.2121097.

[Hamm50] R. W. Hamming. «Error Detecting and Error Correcting Codes». English. En: *BELL SYSTEM TECHNICAL JOURNAL* 29.2 (1950), págs. 147-160. ISSN: 0005-8580.

[Hass10] S. Hassayoun, P. Maille y D. Ros. «On the Impact of Random Losses on TCP Performance in Coded Wireless Mesh Networks». En: *INFOCOM, 2010 Proceedings IEEE*. Mar. de 2010, págs. 1-9. DOI: 10.1109/INFCOM.2010.5462128.

[Hend12] T. Henderson, S. Floyd, A. Gurtov e Y. Nishida. *The NewReno Modification to TCP's Fast Recovery Algorithm*. RFC 6582 (Proposed Standard). Internet Engineering Task Force, abr. de 2012. URL: http://www.ietf.org/rfc/rfc6582.txt.

[Hers07] John Hershberger, Matthew Maxel y Subhash Suri. «Finding the K Shortest Simple Paths: A New Algorithm and Its Implementation». En: *ACM Trans. Algorithms* 3.4 (nov. de 2007). ISSN: 1549-6325. DOI: 10.1145/1290672.1290682. URL: http://doi.acm.org/10.1145/1290672.1290682.

[Ho03a] Tracey Ho, Ralf Koetter, Muriel Médard, David R. Karger y Michelle Effros. «The benefits of coding over routing in a randomized setting». En: *In Proceedings of 2003 IEEE International Symposium on Information Theory*. 2003.

[Ho03b] Tracey Ho, Muriel Médard, Jun Shi, Michelle Effros y David R. Karger. «On Randomized Network Coding». En: *In Proceedings of 41st Annual Allerton Conference on Communication, Control, and Computing*. 2003.

[Ho08] Tracey Ho y Desmond Lun. *Network Coding: An Introduction*. New York, NY, USA: Cambridge University Press, 2008. ISBN: 052187310X, 9780521873109.

[Hsu11] Che-Jung Hsu, Huey-Ing Liu y Winston K.G. Seah. «Opportunistic routing: A review and the challenges ahead». En: *Computer Networks* 55.15 (2011), págs. 3592-3603. ISSN: 1389-1286. DOI: http://dx.doi.org/10.1016/j.comnet.2011.06.021. URL: http://www.sciencedirect.com/science/article/pii/S1389128611002337.

[Huan08] Yong Huang, M. Ghaderi, D. Towsley y W. Gong. «TCP Performance in Coded Wireless Mesh Networks». En: *Proc. of IEEE SECON*. 2008, págs. 179-187. DOI: 10.1109/SAHCN.2008.31.

[Hund12] M. Hunderbøll, J. Ledet-Pedersen, J. Heide, M.V. Pedersen, S.A. Rein y F.H.P. Fitzek. «CATWOMAN: Implementation and Performance Evaluation of IEEE 802.11 Based Multi-Hop Networks Using Network Coding». En: *Vehicular Technology Conference (VTC Fall), 2012 IEEE*. 2012, págs. 1-5. DOI: 10.1109/VTCFall.2012.6399115.

[IRTF] *Network Coding IRTF Research Group*. http://trac.tools.ietf.org/group/irtf/trac/wiki/nwcrg.

[IT++] *IT++ Mathematical library*. http://itpp.sourceforge.net/.

[Jagg05] S. Jaggi, P. Sanders, P.A. Chou, M. Effros, S. Egner, K. Jain y L.M.G.M. Tolhuizen. «Polynomial time algorithms for multicast network code construction». En: *Information Theory, IEEE Transactions on* 51.6 (jun. de 2005), págs. 1973-1982. ISSN: 0018-9448. DOI: 10.1109/TIT.2005.847712.

[Katt07] Sachin Katti, Shyamnath Gollakota y Dina Katabi. «Embracing Wireless Interference: Analog Network Coding». En: *SIGCOMM Comput. Commun. Rev.* 37.4 (ago. de 2007), págs. 397-408. ISSN: 0146-4833. DOI: 10.1145/1282427.1282425. URL: http://doi.acm.org/10.1145/1282427.1282425.

[Katt08] S. Katti, H. Rahul, Wenjun Hu, D. Katabi, M. Médard y J. Crowcroft. «XORs in the Air: Practical Wireless Network Coding». En: *Networking, IEEE/ACM Transactions on* 16.3 (jun. de 2008), págs. 497-510. ISSN: 1063-6692. DOI: 10.1109/TNET.2008.923722.

[Kim12a] MinJi Kim, Jason Cloud, Ali ParandehGheibi, Leonardo Urbina, Kerim Fouli, Douglas J. Leith y Muriel Médard. «Network Coded TCP (CTCP)». En: *CoRR* abs/1212.2291 (2012).

[Kim12b] MinJi Kim, Thierry Klein, Emina Soljanin, João Barros y Muriel Médard. «Modeling Network Coded TCP: Analysis of Throughput and Energy Cost». En: *CoRR* abs/1208.3212 (2012).

[Kim12c] MinJi Kim, Ali ParandehGheibi, Leonardo Urbina y Muriel Médard. «CTCP: Coded TCP using Multiple Paths». En: *CoRR* abs/1212.1929 (2012). URL: http://arxiv.org/abs/1212.1929.

[Kodo] *Kodo Library documentation*. http://kodo-docs.steinwurf.com/en/latest/.

[Koet01] R. Koetter y M. Médard. «An algebraic approach to network coding». En: *Information Theory, 2001. Proceedings. 2001 IEEE International Symposium on*. 2001, pág. 104. DOI: 10.1109/ISIT.2001.935967.

[Krig13] J. Krigslund, J. Hansen, M. Hundeboll, F.H.P. Fitzek y T. Larsen. «CORE: COPE with MORE in Wireless Meshed Networks». En: *IEEE VTC2013-Spring: Cooperative Communication, Distributed MIMO and Relaying*. Dresden, Germany, jun. de 2013.

[Le10] Jilin Le, J.C.-S. Lui y Dah-Ming Chiu. «DCAR: Distributed Coding-Aware Routing in Wireless Networks». En: *Mobile Computing, IEEE Transactions on* 9.4 (abr. de 2010), págs. 596-608. ISSN: 1536-1233. DOI: 10.1109/TMC.2009.160.

[Li03] S.-Y.R. Li, R.W. Yeung y Ning Cai. «Linear network coding». En: *Information Theory, IEEE Transactions on* 49.2 (feb. de 2003), págs. 371-381. ISSN: 0018-9448. DOI: 10.1109/TIT.2002.807285.

[Li11a] Baochun Li y Di Niu. «Random Network Coding in Peer-to-Peer Networks: From Theory to Practice». En: *Proceedings of the IEEE* 99.3 (mar. de 2011), págs. 513-523. ISSN: 0018-9219. DOI: 10.1109/JPROC.2010.2091930.

[Li11b] Xiaohang Li, Chih-Chun Wang y Xiaojun Lin. «On The Capacity of Immediately-Decodable Coding Schemes for Wireless Stored-Video Broadcast with Hard Deadline Constraints». En: *Selected Areas in Communications, IEEE Journal on* 29.5 (mayo de 2011), págs. 1094-1105. ISSN: 0733-8716. DOI: 10.1109/JSAC.2011.110519.

[Li12] Jung-Shian Li y Kun-Hsuan Liu. «Network-coding-based Cache Policy for Loss Recovery Enhancement in Reliable Multicast». En: *Int. J. Netw. Manag.* 22.4 (jul. de 2012), págs. 330-345. ISSN: 1099-1190. DOI: 10.1002/nem.808. URL: http://dx.doi.org/10.1002/nem.808.

[Lint92] Jacobus H. van Lint y Richard M. Wilson. *A course in combinatorics.* Cambridge University Press, 1992, págs. I-XII, 1-530. ISBN: 978-0-521-41057-1.

[Liu10] Zheng Liu y Shudong Jin. «Diagnosing the limitations of network coding at transport layer». En: *Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on*. 2010, págs. 436-442. DOI: 10.1109/ISWPC.2010.5483720.

[Liu12] H. Liu e Y. Gu. «TCP with hop-oriented network coding in multi-radio multi-channel wireless mesh networks». En: *Networks, IET* 1.3 (sep. de 2012), págs. 171-180. ISSN: 2047-4954. DOI: 10.1049/iet-net.2012.0048.

[Luby02] Michael Luby. «LT Codes». En: *Proceedings of the 43rd Symposium on Foundations of Computer Science*. FOCS '02. Washington, DC, USA: IEEE Computer Society, 2002, págs. 271-. ISBN: 0-7695-1822-2. URL: http://dl.acm.org/citation.cfm?id=645413.652135.

[Luby97] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, Daniel A. Spielman y Volker Stemann. «Practical Loss-resilient Codes». En: *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*. STOC '97. New York, NY, USA: ACM, 1997, págs. 150-159. ISBN: 0-89791-888-6. DOI: 10.1145/258533.258573. URL: http://doi.acm.org/10.1145/258533.258573.

[Luca09] Daniel E. Lucani, Muriel Médard y Milica Stojanovic. «Random linear network coding for time-division duplexing: field size considerations». En: *Proceedings of the 28th IEEE conference on Global telecommunications*. GLOBECOM'09. Honolulu, Hawaii, USA: IEEE Press, 2009, págs. 4601-4606. ISBN: 978-1-4244-4147-1. URL: http://dl.acm.org/citation.cfm?id=1811982.1812146.

[Luca14a] Daniel E. Lucani, Morten Videbæk Pedersen, Janus Heide y Frank H. P. Fitzek. «Fulcrum Network Codes: A Code for Fluid Allocation of Complexity». En: *CoRR* abs/1404.6620 (2014). URL: http://arxiv.org/abs/1404.6620.

[Luca14b] D.E. Lucani, M.V. Pedersen, J. Heide y F.H.P. Fitzek. «Coping with the upcoming heterogeneity in 5G communications and storage using Fulcrum network codes». En: *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on*. Ago. de 2014, págs. 997-1001. DOI: 10.1109/ISWCS.2014.6933499.

[M4RI] *M4RI(e)- Linear Algebra over GF(2) and GF(2+)*. http://m4ri.sagemath.org/.

[Mass69] J.L. Massey. «Shift-register synthesis and BCH decoding». En: *Information Theory, IEEE Transactions on* 15.1 (ene. de 1969), págs. 122-127. ISSN: 0018-9448. DOI: 10.1109/TIT.1969.1054260.

[MatGF] *Matlab* - Toolbox *para operaciones en Cuerpos de Galois*. http://es.mathworks.com/help/comm/galois-field-computations.html.

[MURCO] *MURCO. A Network Coding Framework in Multi-Radio Networks (GitHub repository)*. https://github.com/yangchi/Murco. 2012.

[Nage10] T. Nage, F.R. Yu y M. St-Hilaire. «TCP-Aware Network Coding with Opportunistic Scheduling in Wireless Mobile Ad Hoc Networks». En: *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*. Ene. de 2010, págs. 1-5. DOI: 10.1109/CCNC.2010.5421777.

[Naze11] B. Nazer y M. Gastpar. «Reliable Physical Layer Network Coding». En: *Proceedings of the IEEE* 99.3 (mar. de 2011), págs. 438-460. ISSN: 0018-9219. DOI: 10.1109/JPROC.2010.2094170.

[Neum08] Axel Neumann, Corinna Aichele, Marek Lindner y Simon Wunderlich. *Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.)* Internet-Draft draft-wunderlich-openmesh-manet-routing-00. http://www.ietf.org/internet-drafts/draft-wunderlich-openmesh-manet-routing-00.txt. IETF Secretariat, abr. de 2008. URL: http://www.ietf.org/internet-drafts/draft-wunderlich-openmesh-manet-routing-00.txt.

[Papa82] Christos H. Papadimitriou y Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982. ISBN: 0-13-152462-3.

[Para10] A Parandehgheibi, A Ozdaglar, M. Effros y M. Médard. «Optimal reverse carpooling over wireless networks - a distributed optimization approach». En: *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*. Mar. de 2010, págs. 1-6. DOI: 10.1109/CISS.2010.5464771.

[Para13] A. Paramanathan, M.V. Pedersen, D.E. Lucani, F.H.P. Fitzek y M. Katz. «Lean and mean: network coding for commercial devices». En: *Wireless Communications, IEEE* 20.5 (oct. de 2013), págs. 54-61. ISSN: 1536-1284. DOI: 10.1109/MWC.2013.6664474.

[Part90] C. Partridge y R.M. Hinden. *Version 2 of the Reliable Data Protocol (RDP)*. RFC 1151 (Experimental). Internet Engineering Task Force, abr. de 1990. URL: http://www.ietf.org/rfc/rfc1151.txt.

[Pede13] M.V. Pedersen, J. Heide, P. Vingelmann y F.H.P. Fitzek. «Network Coding Over the $2^{32}-5$ Prime Field». En: *IEEE International Conference on Communications (ICC) - Symposium*. Budapest, Hungary, jun. de 2013.

[Pern01] Clément Pernet. *Implementation of Winograd's matrix multiplication over finite fields using ATLAS level 3 BLAS*. 2001.

[Pete72] W.W. Peterson y E.J. Weldon. *Error-correcting Codes*. MIT Press, 1972. ISBN: 9780262160391. URL: http://books.google.es/books?id=5kfwlFeklx0C.

[Pomp13] Mario Puente Pomposo. «Mejora del comportamiento del protocolo UDP sobre redes malladas inalámbricas multi-salto a través de técnicas de Network Coding». Tesis de lic. Universidad de Cantabria, 2013.

[Post81] J. Postel. *Transmission Control Protocol*. RFC 793 (INTERNET STANDARD). Updated by RFCs 1122, 3168, 6093, 6528. Internet Engineering Task Force, sep. de 1981. URL: http://www.ietf.org/rfc/rfc793.txt.

[QualDF] *Qualcomm Buys Online and Mobile Video Tech Provider Digital Fountain*. https://gigaom.com/2009/02/27/419-qualcomm-buys-online-and-mobile-video-tech-provider-digital-fountain/.

[QUIC] *Experimenting with QUIC*. http://blog.chromium.org/2013/06/experimenting-with-quic.html.

[Qure14] Jalaluddin Qureshi, Chuan Heng Foh y Jianfei Cai. «Online {XOR} packet coding: Efficient single-hop wireless multicasting with low decoding delay». En: *Computer Communications* 39 (2014). Research advances and standardization activities in {WLANs} Research advances and standardization activities in {WLANs}, págs. 65-77. ISSN: 0140-3664. DOI: http:

//dx.doi.org/10.1016/j.comcom.2013.09.006. URL: http://www.sciencedirect.com/science/article/pii/S0140366413002119.

[Reed60] I. S. Reed y G. Solomon. «Polynomial Codes Over Certain Finite Fields». En: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), págs. 300-304. URL: http://dx.doi.org/10.1137/0108018.

[SAGE] *SAGE - Open-source Mathematical Software System*. http://www.sagemath.org/.

[Samu08] P. Samuel David y A. Kumar. «Network coding for TCP throughput enhancement over a multi-hop wireless network». En: *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*. Ene. de 2008, págs. 224-233. DOI: 10.1109/COMSWA.2008.4554414.

[Sand03] Peter Sanders, Sebastian Egner y Ludo Tolhuizen. «Polynomial Time Algorithms for Network Information Flow». En: *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures*. SPAA '03. San Diego, California, USA: ACM, 2003, págs. 286-294. ISBN: 1-58113-661-7. DOI: 10.1145/777412.777464. URL: http://doi.acm.org/10.1145/777412.777464.

[Scal07] L. Scalia, F. Soldo y M. Gerla. «PiggyCode: A MAC Layer Network Coding Scheme to Improve TCP Performance Over Wireless Networks». En: *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*. Nov. de 2007, págs. 3672-3677. DOI: 10.1109/GLOCOM.2007.697.

[Sefe11] H. Seferoglu, A. Markopoulou y K.K. Ramakrishnan. «I2NC: Intra- and inter-session network coding for unicast flows in wireless networks». En: *INFOCOM, 2011 Proceedings IEEE*. 2011, págs. 1035-1043. DOI: 10.1109/INFCOM.2011.5934877.

[Seng10] C. Senger, S. Schober, Tong Mao y A. Zeh. «End-to-End algebraic network coding for wireless TCP/IP networks». En: *Telecommunications (ICT), 2010 IEEE 17th International Conference on*. Abr. de 2010, págs. 607-612. DOI: 10.1109/ICTEL.2010.5478841.

[Shok06] Amin Shokrollahi. «Raptor Codes». En: *IEEE/ACM Trans. Netw.* 14.SI (jun. de 2006), págs. 2551-2567. ISSN: 1063-6692. DOI: 10.1109/TIT.2006.874390. URL: http://dx.doi.org/10.1109/TIT.2006.874390.

[Sinh07] Rishi Sinha, Christos Papadopoulos y John Heidemann. *Internet Packet Size Distributions: Some Observations*. Inf. téc. ISI-TR-2007-643. Originally released October 2005 as web page http://netweb.usc.edu/~rsinha/pkt-sizes/. USC/Information Sciences Institute, mayo de 2007. URL: http://www.isi.edu/~johnh/PAPERS/Sinha07a.html.

[Soro09] S. Sorour y S. Valaee. «A network coded ARQ protocol for broadcast streaming over hybrid satellite systems». En: *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*. Sep. de 2009, págs. 1098-1102. DOI: 10.1109/PIMRC.2009.5449889.

[Sun09] Yue Sun, Ying Li y Xinmei Wang. «Cooperative hybrid-ARQ protocol with network coding». En: *Communications and Networking in China, 2009. ChinaCOM 2009. Fourth International Conference on*. Ago. de 2009, págs. 1-5. DOI: 10.1109/CHINACOM.2009.5339946.

[Sund08] J.K. Sundararajan, D. Shah y M. Medard. «ARQ for network coding». En: *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*. Jul. de 2008, págs. 1651-1655. DOI: 10.1109/ISIT.2008.4595268.

[Sund11] J.K. Sundararajan, D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher y J. Barros. «Network Coding Meets TCP: Theory and Implementation». En: *Proceedings of the IEEE* 99.3 (mar. de 2011), págs. 490-512. ISSN: 0018-9219. DOI: 10.1109/JPROC.2010.2093850.

[Swap13] B.T. Swapna, A. Eryilmaz y N.B. Shroff. «Throughput-Delay Analysis of Random Linear Network Coding for Wireless Broadcasting». En: *Information Theory, IEEE Transactions on* 59.10 (oct. de 2013), págs. 6328-6341. ISSN: 0018-9448. DOI: 10.1109/TIT.2013.2271895.

[Thom12] N. Thomos y P. Frossard. «Toward One Symbol Network Coding Vectors». En: *Communications Letters, IEEE* 16.11 (nov. de 2012), págs. 1860-1863. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2012.092812.121661.

[Tras06] Danail Traskov, Niranjan Ratnakar, Desmond S. Lun, Ralf Koetter y Muriel Médard. «Network coding for multiple unicasts: An approach based on linear optimization». En: *IEEE Int. Symp. Inf. Theory* (2006).

[Trul11] O. Trullols-Cruces, J.M. Barcelo-Ordinas y M. Fiore. «Exact Decoding Probability Under Random Linear Network Coding». En: *Communications Letters, IEEE* 15.1 (2011), págs. 67-69. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2010.110310.101480.

[Ving11] P. Vingelmann, F.H.P. Fitzek, M.V. Pedersen, J. Heide y H. Charaf. «Synchronized multimedia streaming on the iPhone platform with network coding». En: *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*. Ene. de 2011, págs. 875-879. DOI: 10.1109/CCNC.2011.5766632.

[VPNMTU] *MTU setting across VPN or L2TP Tunnel (Network Performance Issues)*. http://www.apalytics.com/mtu-across-vpn-l2tp-tunnel-network-performance-issues/.

[Wu12a] Fei Wu, Cunqing Hua, Hangguan Shan y Aiping Huang. «Moving Window Network Coding in Cooperative Multicast (v1)». En: *CoRR* abs/1209.3827 (2012). URL: http://arxiv.org/abs/1209.3827.

[Wu12b] Fei Wu, Cunqing Hua, Hangguan Shan y Aiping Huang. «MWNCast: Cooperative multicast based on moving window network coding». En: *Global Communications Conference (GLOBECOM), 2012 IEEE*. Dic. de 2012, págs. 138-144. DOI: 10.1109/GLOCOM.2012.6503103.

[Wu13] Tin-Yu Wu, S. Guizani, Wei-Tsong Lee y Po-Chang Huang. «An enhanced structure of layered forward error correction and interleaving for scalable video coding in wireless video delivery». En: *Wireless Communications, IEEE* 20.4 (ago. de 2013), págs. 146-152. ISSN: 1536-1284. DOI: 10.1109/MWC.2013.6590062.

[Wu15] Fei Wu, Cunqing Hua, Hangguan Shan y Aiping Huang. «Cooperative multicast with moving window network coding in wireless networks». En: *Ad Hoc Networks* 25, Part A (2015), págs. 213-227. ISSN: 1570-8705. DOI: http://dx.doi.org/10.1016/j.adhoc.2014.10.011. URL: http://www.sciencedirect.com/science/article/pii/S1570870514002273.

[Yan10] Yan Yan, Baoxian Zhang, Jun Zheng y Jian Ma. «CORE: a coding-aware opportunistic routing mechanism for wireless mesh networks [Accepted from Open Call]». En: *Wireless Communications, IEEE* 17.3 (jun. de 2010), págs. 96-103. ISSN: 1536-1284. DOI: 10.1109/MWC.2010.5490984.

[YANCI] *ns3-YANCI. Yet Another Network Coding Implementation*. https://github.com/yangchi/ns3-yanci. 2012.

[Yang14] Shenghao Yang y R.W. Yeung. «Batched Sparse Codes». En: *Information Theory, IEEE Transactions on* 60.9 (sep. de 2014), págs. 5322-5346. ISSN: 0018-9448. DOI: 10.1109/TIT.2014.2334315.

[Yen72] Jin Y. Yen. «Finding the Lengths of All Shortest Paths in N -Node Nonnegative-Distance Complete Networks Using 12N3 Additions and N3 Comparisons». En: *J. ACM* 19.3 (jul. de 1972), págs. 423-424. ISSN: 0004-5411. DOI: 10.1145/321707.321712. URL: http://doi.acm.org/10.1145/321707.321712.

[Yeun07] Raymond W. Yeung. «Avalanche: A Network Coding Analysis». En: *Communications in Information and Systems* (2007), págs. 353-358.

[Zhan06] Shengli Zhang, Soung-chang Liew y Patrick P. Lam. «Physical-layer network coding». En: *in ACM Mobicom 2006*. 2006.

[Zhan13] X. Zhang, G. Neglia, J. Kurose, D. Towsley y H. Wang. «Benefits of Network Coding for Unicast Application in Disruption-Tolerant Networks». En: *Networking, IEEE/ACM Transactions on* 21.5 (2013), págs. 1407-1420. ISSN: 1063-6692. DOI: 10.1109/TNET.2012.2224369.

[Zhao12] Xubo Zhao. «Notes on "Exact Decoding Probability Under Random Linear Network Coding"». En: *Communications Letters, IEEE* 16.5 (mayo de 2012), págs. 720-721. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2012.041112.112564.

[Zhen07] Zizhan Zheng y P. Sinha. «XBC: XOR-based buffer coding for reliable transmissions over wireless networks». En: *Broadband Communications, Networks and Systems, 2007. BROADNETS 2007. Fourth International Conference on*. Sep. de 2007, págs. 76-85. DOI: 10.1109/BROADNETS.2007.4550409.

7

Conclusiones y líneas futuras

Son muchos los interrogantes que aparecen al plantearse cómo serán las comunicaciones del futuro. Nadie duda sin embargo que las comunicaciones inalámbricas van a tomar un papel aún más protagonista. A pesar de la madurez del mercado, la aparición continua de nuevos dispositivos y servicios pone de manifiesto la gran importancia de estas tecnologías. Por otro lado, se puede decir que las perspectivas del protocolo *TCP*, como alternativa más extendida para el nivel de transporte, no son tan claras, a pesar de que todavía es la solución mayoritaria para muchas de las aplicaciones que se emplean en Internet. Uno de los motivos que pone en riesgo su dominio es que su comportamiento sobre este tipo de redes dista mucho de ser el adecuado. El principal problema se origina por sus mecanismos de control de congestión que, al no distinguir la causa de la pérdida de un segmento, interpretan de manera persistente siempre una situación de congestión, lo que causará una severa penalización del rendimiento de las transmisiones, al reducirse la tasa de envío en el nodo fuente, que podría llegar a estar inactivo durante largos intervalos de tiempo.

Partiendo de esta problemática, en esta Tesis se han propuesto una serie de soluciones que se podrían ver como complementos (o sustitutos) del protocolo *TCP*, para ser particularmente empleadas sobre redes malladas (multi-salto) inalámbricas. Antes de estudiar sus prestaciones, se ha llevado a cabo un análisis de una de las tecnologías con mayor presencia en las redes inalámbricas, *IEEE 802.11*. Esta caracterización se ha realizado con un doble objetivo; en primer lugar permite conocer cuáles son los límites del rendimiento que puede ofrecer y, además, se ha demostrado que los modelos de canal típicamente incluidos en simuladores de red no reflejan adecuadamente las particularidades del medio radio, especialmente el comportamiento *a ráfagas* que pone de manifiesto. Como alternativa, se han propuesto e implementado en el marco del simulador `ns-3` dos alternativas capaces de reflejar adecuadamente las características observadas sobre escenarios reales.

Posteriormente se ha presentado una implementación completa del protocolo *MPTCP*, que puede verse como una evolución natural de *TCP*. Su rasgo más característico es que divide una única sesión de transporte en múltiples conexiones simultáneas. Se ha evaluado su comportamiento, llevando a cabo varios experimentos bajo diferentes condiciones, en los que se ha demostrado que la multiplexación del tráfico da lugar a un mayor ancho

de banda agregado, además de ofrecer otras capacidades también interesantes, como la de adaptar la carga en función de la condición de los enlaces, favoreciendo el uso de aquellas conexiones más estables.

También se han diseñado e implementado dos aproximaciones diferentes de las denominadas técnicas de *NC*. En primer lugar, un esquema que combina paquetes pertenecientes a diferentes flujos, interactuando directamente con el protocolo *TCP*; se ha comprobado que, aunque ofrece una ganancia notable sobre escenarios con condiciones ideales, no se trata de una solución apropiada a medida que la calidad del medio radio empeora. Para solventar este problema, se ha propuesto una segunda alternativa, que combina paquetes pertenecientes a una misma conexión, basándose en el protocolo *UDP*, para proporcionar un servicio de entrega de datos confiable. El estudio de sus prestaciones ha puesto de manifiesto mejoras claras, con un comportamiento más robusto que la alternativa tradicional basada en *TCP* sobre canales hostiles.

En el resto del capítulo se presentan las principales aportaciones del trabajo realizado así como las líneas de investigación que quedan abiertas tras la realización de la Tesis, enfocadas principalmente en la interacción entre las diferentes partes que han sido sujeto de análisis.

7.1 Principales aportaciones

A continuación se resumen las principales aportaciones de este trabajo.

- En primer lugar, se han estudiado las prestaciones que se pueden alcanzar sobre un enlace *IEEE 802.11b*, tanto analíticamente como a través de una campaña de medidas. La caracterización teórica permite establecer la cota del rendimiento sobre esta tecnología, asumiendo canales libres de errores. Por otra parte, la caracterización empírica sirve para comprobar el efecto de la propagación inalámbrica, poniéndose de manifiesto un efecto memoria, de manera que se observa una correlación entre pérdidas consecutivas, lo que causa que los errores aparezcan en forma de ráfaga.

- Posteriormente, las trazas recogidas sobre el escenario real se han utilizado para modelar dos modelos de canal, que se han implementado en el marco del simulador `ns-3`: *HMP* y *BEAR*. El primero de ellos utiliza un proceso oculto de Markov, mientras que *BEAR* estima la relación señal a ruido percibida en los nodos receptores, basándose en tres contribuciones diferenciadas: propagación, y desvanecimientos lentos y rápidos.

- Con respecto al modelo *HMP*, se pueden destacar dos contribuciones específicas: se ha propuesto una aproximación novedosa para modelar las transiciones entre estados de la cadena de Markov, basándose en su duración temporal, lo que permite

desacoplar el proceso de las condiciones con las que fue "entrenado", otorgándole la flexibilidad necesaria para adaptarse a diferentes patrones de tráfico. Por otra parte, se ha presentado una solución para que el esquema *HMP* pueda modificar su comportamiento en función de la separación entre los nodos, solventando una de sus principales limitaciones.

- Se ha llevado a cabo una extensa campaña de simulación en la que se ha evaluado el comportamiento de los dos modelos de canal presentados, comparándolos con la alternativa básica que incorpora por defecto el simulador, utilizando tráfico *UDP* y *TCP*. Se ha visto que, al contrario que el modelo básico, tanto *HMP* como *BEAR* reflejan adecuadamente el comportamiento observado en las medidas reales, siendo capaces de reflejar los efectos producidos por las ráfagas de tramas con error, especialmente relevantes en el caso de *TCP*. Finalmente, se ha estudiado el coste computacional de cada uno de los modelos, en función del número de nodos desplegados en un escenario, demostrando que el modelo *HMP* presenta un compromiso adecuado entre sus prestaciones y complejidad.

- Se ha migrado una implementación previa del protocolo *MPTCP* a una versión más actual del simulador `ns-3`, evaluándose posteriormente su comportamiento sobre redes malladas inalámbricas. En un primer análisis, llevado a cabo sobre una topología canónica, se ha estudiado el comportamiento de diferentes mecanismos de control de congestión sobre un escenario sin errores, observando un beneficio notable al dividir la conexión entre varios subflujos

- El análisis se ha extendido, considerando canales con errores, y comparando el rendimiento con el observado con el protocolo *TCP*, comprobando que los mecanismos de control de congestión de *MPTCP* son capaces de favorecer aquellos caminos que presenten mejores condiciones. También se ha demostrado el beneficio, en términos de rendimiento, que se deriva de la utilización de interfaces de red completamente ortogonales.

- Una vez caracterizado el comportamiento de *MPTCP* sobre topologías sencillas, se han implementado tres algoritmos (*link*, *node* y *zone disjoint*) para obtener múltiples rutas sobre despliegues de red genéricos, que serán posteriormente utilizadas para cada uno de los subflujos creados durante el establecimiento de la conexión. Se ha analizado el comportamiento de dichos algoritmos, estudiando su capacidad para establecer rutas realmente ortogonales. Como último paso, se han utilizado los resultados del análisis anterior, para corroborar el rendimiento que ofrece el protocolo *MPTCP* sobre topologías de red más complejas.

- Se ha diseñado e implementado un protocolo *NC inter-flujo*, que otorga a los nodos intermedios la capacidad de combinar paquetes que pertenezcan a diferentes conexiones, aprovechando la naturaleza *broadcast* de los canales inalámbricos. Asimismo, se ha añadido la posibilidad de encapsular los reconocimientos *TCP* en las cabeceras propias de *NC*. También se ha incorporado una capa de inteligencia adicional,

capaz de monitorizar el estado de los flujos que atraviesan los nodos en tiempo real. Finalmente, se ha incluido un esquema novedoso de retransmisiones propietarias al nivel *NC*, que permite recuperar los segmentos perdidos, de manera transparente al protocolo *TCP*.

- Se ha caracterizado, a través de una extensa campaña de simulación sobre escenarios canónicos ideales, la influencia de los diferentes parámetros de configuración del protocolo, como los relacionados con los *buffers* de codificación y de decodificación. Los resultados han mostrado que, con una selección adecuada de estos atributos, se puede obtener una ganancia del 39 % en términos de *throughput*, con una reducción del 30 % de las transmisiones totales necesarias, al combinar la codificación de los segmentos de datos con la encapsulación de los *ACKs* de *TCP*.

- Se ha comprobado que estos resultados positivos no se mantienen cuando se utiliza el protocolo sobre canales no ideales, de tal manera que su rendimiento puede ser incluso peor que el observado al utilizar esquemas de transmisión tradicionales. Se ha estudiado la influencia de los errores de transmisión, en función del flujo (o flujos) sobre los que se producían.

- Se ha propuesto un algoritmo para identificar oportunidades de codificación sobre despliegues genéricos de red, utilizando la teoría de grafos. Se ha comprobado que sólo un porcentaje muy bajo de los escenarios (inferior al 50 %) cumple con los requisitos para que los esquemas de codificación *inter-flujo* sean viables; además las rutas generadas son significativamente más largas que las alternativas óptimas, lo que deriva en rendimientos inferiores.

- Utilizando los resultados del algoritmo anterior, se ha analizado (a través del simulador `ns-3`) el beneficio que se obtendría en aquellas situaciones en las que la utilización del protocolo *inter-flujo* fuera viable. Se ha comprobado que únicamente en aquellas situaciones en las que el algoritmo encuentra rutas de la misma longitud que las óptimas (o con una diferencia de un salto) se consigue mejorar el *throughput*. Por otro lado, el beneficio es muy bajo ($\approx 2\,\%$), y el porcentaje en el que se dan estas circunstancias es también muy reducido.

- Se ha diseñado e implementado un protocolo que, mediante la combinación de *UDP* y un esquema *NC intra-flujo*, ofrece un servicio que garantiza la correcta recepción de la información en el destino. Su operación comienza en el nodo fuente, que genera combinaciones lineales aleatorias en bloques de paquetes de tamaño fijo, que seguirán enviándose hasta que el receptor confirme que ha sido capaz de recuperar el contenido original. Teniendo en cuenta que se diseña con el objetivo de operar sobre redes inalámbricas, el esquema explota la diversidad espacial del medio radio, de manera que varios nodos escuchan los mensajes de manera simultánea. Además, se han utilizado diferentes soluciones *cross-layer*, comunicando el módulo *NC* con las capas inferiores, implementando procedimientos que mejoran las prestaciones del protocolo.

- El protocolo contempla dos modos de operación; en el primero únicamente las fuentes codifican paquetes (*RLSC*), mientras que en la segunda alternativa, *RLNC*, a los nodos intermedios se les otorga una inteligencia adicional, para que puedan procesar el contenido de los mensajes.

- Uno de los aspectos que en mayor medida pueden afectar al rendimiento de este protocolo es la complejidad de las operaciones necesarias, especialmente en el proceso de decodificación (cálculo de rango e inversa de matrices). Se ha realizado una comparativa de varias librerías especializadas en el álgebra de cuerpos finitos, para seleccionar aquella que ofrece unas mejores prestaciones.

- Posteriormente se ha estudiado, tanto teóricamente como sobre el simulador, el comportamiento del protocolo sobre un enlace. Se ha prestado especial atención al efecto de los dos factores (a nivel *NC*) que penalizan el rendimiento: la generación de vectores linealmente dependientes, ya que no aportan información, y la transmisión de confirmaciones tras la correcta decodificación de un bloque. Se ha comprobado que la utilización de tamaños de bloque y órdenes de los cuerpos de Galois elevados incrementa la sobrecarga de la cabecera del protocolo, penalizando el *throughput*. Además, a medida que el tamaño del bloque crece, también es superior el retardo entre la recepción de bloques consecutivos en la aplicación receptora. Tras un primer estudio en condiciones ideales, se han generado errores en el enlace, comprobando que la solución propuesta tiene una robustez notablemente superior a la ofrecida por una transmisión tradicional *TCP*.

- Posteriormente, se ha extendido el análisis, incrementando la distancia entre los nodos y situando otros intermedios entre ellos, que hacen las veces de *routers*. En este escenario se ha analizado la ganancia que aporta la escucha oportunista, que le permite al receptor recibir la información por dos caminos diferentes. Además, se ha constatado que el proceso de recombinación en los nodos intermedios introduce una mayor aleatoriedad, incrementando la probabilidad de recibir paquetes linealmente independientes y, por tanto, el rendimiento final.

- Para analizar los beneficios que puede aportar un encaminamiento oportunista se han implementado tres alternativas diferentes, comprobando la importancia de adaptar adecuadamente la tasa de generación de tráfico. Se tiene que llevar a cabo un compromiso entre el beneficio de una carga superior, que permite enmascarar la pérdida de información, y sus posibles desventajas, ya que incrementaría la probabilidad de recibir mensajes repetidos, así como la contención en el canal radio. Se ha vuelto a demostrar el beneficio del proceso de recodificación por parte de los nodos intermedios.

Destacar finalmente que algunos de los resultados obtenidos han sido publicados en diversos foros, lo que valida la calidad del trabajo realizado. En el Anexo A se enumeran todas las publicaciones derivadas de la Tesis, que se resumen en 2 revistas internacionales,

una de ellas con *JCR* (*Journal Citation Report*), 3 publicaciones en conferencias nacionales y 11 en internacionales; entre estas últimas se puede destacar la participación en el *workshop* anual de desarrolladores del simulador `ns-3`. En esta misma línea, reiterar nuevamente que la mayoría de las implementaciones llevadas a cabo se han puesto a disposición de la comunidad investigadora, a través de repositorios de código libre.

7.2 Líneas futuras de investigación

En las conclusiones de cada uno de los bloques que componen esta memoria ya se han ido adelantando las que se consideran las líneas de investigación más interesantes de entre las que han aparecido a raíz del trabajo que se ha realizado. Aunque en un primer momento parece que se trata de temáticas independientes entre sí, se encuentran numerosas e interesantes sinergias entre ellas. La Figura 7.1 resume cuáles son los aspectos a mejorar para cada una de las soluciones analizadas a lo largo de la Tesis y, además, representa unas zonas de convergencia, que resultaría muy interesante considerar en un futuro, que se describen de manera breve a continuación.

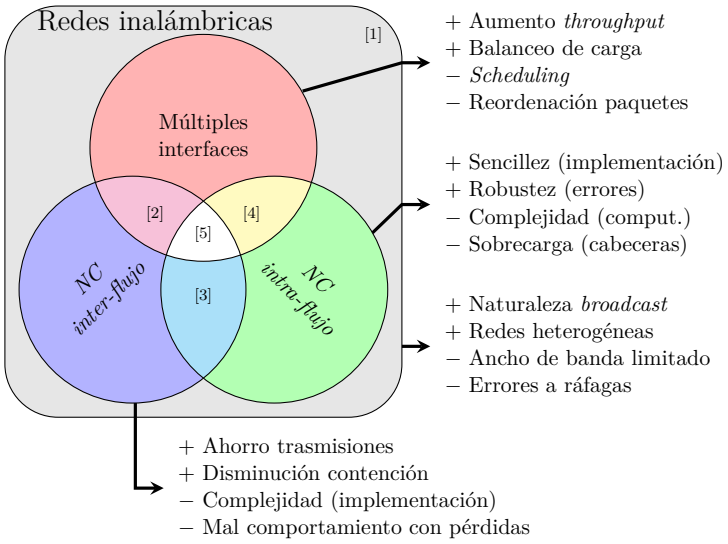


Figura 7.1: Áreas de investigación abiertas a partir de las soluciones presentadas en la Tesis

[1]. En primer lugar, se puede destacar que cualquier modificación o mejora del modelado de los canales inalámbricos no tendrá ningún impacto en el desarrollo del resto de las partes, pudiendo verse como un aspecto ortogonal. Aún así sería conveniente actualizar los modelos de canal (*HMP* y *BEAR*), adaptándolos a versiones más recientes del estándar *IEEE 802.11*. Para ello, se deberá llevar a cabo una campaña de experimentos sobre escenarios reales, para utilizar los resultados en la configuración de los modelos de canal. También resultará interesante estudiar de manera detallada

las prestaciones de los modelos de canal que incorpora el simulador `ns-3` en sus últimas versiones, ya que reflejan un comportamiento a ráfagas, comparándolos con las prestaciones de las dos propuestas realizadas en el marco de la Tesis.

[2]. En base a los resultados observados, parece que la operación conjunta de técnicas *NC inter-flujo* y soluciones multi-interfaz (por ejemplo, el protocolo *MPTCP*) sea la alternativa con menor potencial, ya que las limitaciones que se pusieron de manifiesto en el Capítulo 5 no desaparecerán por el hecho de emplear dispositivos con múltiples interfaces. Por otro lado, sí que resultará interesante analizar los posibles beneficios de integrar la versión avanzada del protocolo, ya que podría mitigar los efectos negativos asociados a los errores en la transmisión y a la desincronización entre los flujos.

[3]. Un aspecto que sí parece que podría aportar un beneficio más claro sería la combinación de los dos esquemas de *NC* que se han propuesto en la Tesis. Así, gracias a las propiedades del álgebra de cuerpos finitos del protocolo *intra-flujo*, la operación conjunta con la alternativa *inter-flujo* permitiría enmascarar las posibles pérdidas, ya que la información de un mensaje que se pierda se podría recuperar con las recepciones posteriores de paquetes codificados, sin necesidad de una retransmisión explícita. Además, la aleatoriedad que introducen las técnicas *RLC* reduce de manera considerable la complejidad de las operaciones de codificación y decodificación, lo que permite eliminar la necesidad de disponer de complejos motores de decisión.

Con respecto a la posible interacción entre los dos esquemas, sería necesario adaptar el protocolo inter-flujo para que su operación fuera independiente del protocolo *TCP*, procesando datagramas; la alternativa intra-flujo permanecería por tanto como el esquema de codificación principal.

[4]. Posiblemente la alternativa más directa podría ser el dividir el tráfico *RLNC* en múltiples subflujos, que además generaría varios beneficios. Gracias a la propiedad *rateless* de los códigos utilizados, se eliminan los problemas que aparecen al utilizar múltiples interfaces, como la gestión del tráfico entre los diferentes subflujos o la reordenación de los paquetes. Desde el punto de vista de implementación, a través de las técnicas *cross-layer* descritas en el Capítulo 6, el envío de paquetes a los niveles superiores se podría realizar sin la necesidad de introducir complejos planificadores, ya que la tasa de inyección se optimizaría en función de las condiciones del canal radio y, además, teniendo en cuenta las características de los códigos empleados, la recepción desordenada de los paquetes no tendría ningún efecto negativo.

[5]. La opción más ambiciosa e interesante sería el desarrollo de una solución integral, capaz de combinar, de una forma eficiente y práctica, la operación de las tres alternativas, aprovechando el grado de aleatoriedad que introduce el nodo fuente en el esquema *RLC*.

A

Publicaciones derivadas de la Tesis

A.1 Revistas internacionales

[1]. David Gómez, Ramón Agüero, Marta García-Arranz y Luis Muñoz. «On the use of Hidden Markov Processes and auto-regressive filters to incorporate indoor bursty wireless channels into network simulation platforms». English. En: *Wireless Networks* (2015), págs. 1-18. ISSN: 1022-0038. DOI: 10.1007/s11276-015-0909-0. URL: http://dx.doi.org/10.1007/s11276-015-0909-0

[2]. D. Gómez, P. Garrido, C. Rabadán, R. Agüero y L. Muñoz. «TCP Performance Enhancement over Wireless Mesh Networks by means of the Combination of Multi-RAT Devices and the MPTCP Protocol». En: *International Journal of Network Protocols and Algorithms* 6.3 (dic. de 2014), págs. 56-81. DOI: 10.5296/npa.v6i3.5387. URL: http://dx.doi.org/10.5296/npa.v6i3.5387

A.2 Conferencias nacionales

[3]. David Gómez, Ramón Agüero, Marta García-Arranz, Roberto Sanz y Luis Muñoz. «Hacia la norma IEEE 802.11n: caracterización experimental de las extensiones de capa MAC para la mejora del rendimiento en redes WLAN». En: *VIII Jornadas de Ingeniería Telemática (JITEL 2009)*. Universidad Politécnica de Cartagena, sep. de 2009, págs. 38-45. ISBN: 978-84-969-9727-1

[4]. P. Garrido, D. Gómez, R. Agüero y L. Muñoz. «Mejora del rendimiento de TCP en redes malladas inalámbricas con técnicas multi-camino: MPTCP». En: *XI Jornadas de Ingeniería Telemática (JITEL 2013)*. Universidad of Granada, oct. de 2013, págs. 69-76. ISBN: 978-84-616-5597-7

[5]. C. Rabadán, P. Garrido, D. Gómez y R. Agüero. «Algoritmos y técnicas multi-camino para la mejora del rendimiento de TCP sobre redes malladas inalámbricas». En: *XI*

Jornadas de Ingeniería Telemática (JITEL 2013). Universidad de Granada, oct. de 2013, págs. 77-84. ISBN: 978-84-616-5597-7

A.3 Conferencias internacionales

[5]. D. Gómez, S. Hassayoun, A. Herrero, R. Agüero y D. Ros. «Impact of network coding on TCP performance in wireless mesh networks». En: *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*. 2012, págs. 777-782. DOI: 10.1109/PIMRC.2012.6362888

[6]. David Gómez, Sofiane Hassayoun, Arnaldo Herrero, Ramón Agüero, David Ros y Marta García-Arranz. «On the Addition of a Network Coding Layer within an Open Connectivity Services Framework». English. En: *Mobile Networks and Management*. Ed. por Andreas Timm-Giel, John Strassner, Ramón Agüero, Susana Sargento y Kostas Pentikousis. Vol. 58. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2013, págs. 298-312. ISBN: 978-3-642-37934-5. DOI: 10.1007/978-3-642-37935-2_23. URL: http://dx.doi.org/10.1007/978-3-642-37935-2_23

[7]. D. Gómez, R. Agüero, M. García-Arranz y L. Muñoz. «On the modeling of a realistic wireless channel by means of a Hidden Markov Process». En: *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on*. 2012, págs. 397-402. DOI: 10.1109/WiMOB.2012.6379104

[8]. David Gómez, Ramón Agüero, Marta García-Arranz y Luis Muñoz. «Replication of the Bursty Behavior of Indoor WLAN Channels». En: *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*. SimuTools '13. Cannes, France: ICST (Institute for Computer Sciences, Social-Informatics y Telecommunications Engineering), 2013, págs. 219-226. ISBN: 978-1-4503-2464-9. URL: http://dl.acm.org/citation.cfm?id=2512734.2512764

[9]. David Gómez, Carlos Rabadán, Pablo Garrido y Ramón Agüero. «Multipath Algorithms and Strategies to Improve TCP Performance over Wireless Mesh Networks». English. En: *Mobile Networks and Management*. Ed. por Dirk Pesch, Andreas Timm-Giel, Ramón Agüero, Bernd-Ludwig Wenning y Kostas Pentikousis. Vol. 125. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer International Publishing, 2013, págs. 15-28. ISBN: 978-3-319-04276-3. DOI: 10.1007/978-3-319-04277-0_2. URL: http://dx.doi.org/10.1007/978-3-319-04277-0_2

[10]. David Gómez, Ramón Agüero, Marta García-Arranz y Luis Muñoz. «TCP performance over indoor bursty wireless channel models». En: *Wireless Days (WD), 2013 IFIP*. 2013, págs. 1-4. DOI: 10.1109/WD.2013.6686458

[11]. David Gómez, Eduardo Rodríguez, Ramón Agüero y Luis Muñoz. «Reliable Communications over Wireless Mesh Networks with Inter and Intra-flow Network Coding». En: *Proceedings of the 2014 Workshop on Ns-3*. WNS3 '14. Atlanta, Georgia: ACM, 2014, 4:1-4:8. ISBN: 978-1-4503-3003-9. DOI: 10.1145/2630777.2630781. URL: http://doi.acm.org/10.1145/2630777.2630781

[12]. D. Gómez, R. Agüero, M. García-Arranz y D. Ros. «TCP Acknowledgement Encapsulation in Coded Multi-Hop Wireless Networks». En: *Vehicular Technology Conference (VTC Spring), 2014 IEEE 79th*. Mayo de 2014, págs. 1-5. DOI: 10.1109/VTCSpring.2014.7023118

[13]. D. Gómez, E. Rodríguez, R. Agüero y L. Muñoz. «Reliable communications over lossy wireless channels by means of the combination of UDP and Random Linear Coding». En: *Computers and Communication (ISCC), 2014 IEEE Symposium on*. Jun. de 2014, págs. 1-6. DOI: 10.1109/ISCC.2014.6912516

[14]. Pablo Garrido, David Gómez, Francisco Santos y Ramón Agüero. «On the Feasibility of Inter-flow Network Coding Over Random Wireless Mesh Networks». English. En: *Mobile Networks and Management*. Ed. por Ramón Agüero, Thomas Zinner, Rossitza Goleva, Andreas Timm-Giel y Phuoc Tran-Gia. Vol. 141. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer International Publishing, 2015, págs. 259-274. ISBN: 978-3-319-16291-1. DOI: 10.1007/978-3-319-16292-8_19. URL: http://dx.doi.org/10.1007/978-3-319-16292-8_19

[15]. David Gómez, Pablo Garrido, Eduardo Rodríguez, Ramón Agüero y Luis Muñoz. «Enhanced opportunistic random linear source/network coding with cross-layer techniques over wireless mesh networks». En: *2014 IFIP Wireless Days, WD 2014, Rio de Janeiro, Brazil, November 12-14, 2014*. 2014, págs. 1-4. DOI: 10.1109/WD.2014.7020842. URL: http://dx.doi.org/10.1109/WD.2014.7020842

A.4 Relación de las publicaciones con el contenido de la Tesis

A modo de resumen, la Tabla A.1 muestra una relación entre las publicaciones relacionadas y la parte de la Tesis sobre la que trata cada una de ellas.

Tabla A.1: Resumen de las publicaciones derivadas de la presente Tesis

Código	*Modelos de canal*	*MPTCP*	*NC inter-flujo*	*NC intra-flujo*
[WN2014]	✓			
[NPA2014]		✓		
[Jitel2009]	✓			
[Jitel2013_a]		✓		
[Jitel2013_b]		✓		
[PIMRC2012]			✓	
[MONAMI2012]			✓	
[WiMob2012]	✓			
[WNS3_2013]	✓			
[MONAMI2013]		✓		
[WD2013]	✓			
[WNS3_2014]			✓	✓
[VTC2014]			✓	
[ISCC2014]				✓
[MONAMI2014]			✓	
[WD2014]				✓

Lista de acrónimos

3GPP 3rd Generation Partnership Project.

AR Auto-Regressive.

ARQ Automatic Repeat Request.

BATMAN Better Approach To Mobile Adhoc Networking.

BEAR Bursty Error model based on an Auto-Regressive filter.

BER Bit Error Rate.

BLAS Basic Linear Algebra Subprograms.

BOWL Berlin Open Wireless Lab.

CATWOMAN Coding Applied To Wireless On Mobile Ad-hoc Networks.

CB Coding Buffer.

CCK Complementary Code Keying.

CRC Cyclic Redundancy Check.

CRM Coding-aware Routing Metric.

CSMA/CA Carrier Sense Multiple Access/Collision Avoidance.

CTCP Network Coded TCP.

CW Contention Window.

CWND Congestion Window.

D2D Device to Device.

DB Decoding Buffer.

DCAR Distributed Coding Aware Routing.

DCCP Data Congestion Control Protocol.

DCE Direct Code Execution.

DCF Distributed Control Function.

DIFS Distributed Inter-Frame Space.

DIWINE Dense Cooperative Wireless Cloud Network.

DQPSK Differential Quadrature Phase Shift Keying.

DRAM Dynamic Random Access Memory.

DSACK Duplicated Selective ACKnowledgement.

DSL Digital Suscriber Line.

DSN Data Sequence Number.

DSS Data Sequence Signal.

DTN Delay Tolerant Networks.

DVB-H Digital Video Broadcasting - Handheld.

DVB-IPTV Digital Video Broadcasting - Internet Protocol TV.

ECC Error-Correcting Code.

EFB Erroneous Frame Burst.

ETX Expected Transmission Count.

ExOR Extremely Opportunistic Routing.

FD Función de Distribución.

fdp función densidad de probabilidad.

FDR Frame Detection Rate.

FEC Forward Error Correction.

FER Frame Error Rate.

FF Fast Fading.

FIFO First In First Out.

FiWi Fiber Wireless.

FORM Free-ride Oriented Routing Metric.

FTP File Transfer Protocol.

GNU GNU's Not Unix!.

GPL Generic Public License.

HMP Hidden Markov Process.

I-TCP Indirect TCP.

IANA Internet Assigned Numbers Authority.

IEEE Institute of Electrical and Electronics Engineers.

IETF Internet Engineering Task Force.

IP Internet Protocol.

IRTF Internet Research Task Force.

LD Link Disjoint.

LDPC Low Density Parity Check.

LLC Logical Link Control.

LT Luby Transform.

LTE Long Term Evolution.

M2M Machine to Machine.

MAC Medium Access Control.

MIT Massachusetts Institute of Technology.

MORE MAC independent Opportunistic Routing.

MPEG-2 Moving Picture Experts Group 2.

MPTCP MultiPath TCP.

MSS Maximum Segment Size.

MTU Maximum Transfer Unit.

MURCO MUlti Radio with network COding.

NAT Network Address Translation.

NC Network Coding.

ND Node Disjoint.

NECO NEtwork COding simulator.

OFDM Orthogonal Frequency Division Multiplex.

OLIA Opportunistic Linked Increases Algorithm.

ORBIT Open-Access Research Testbed for Next-Generation Wireless Networks.

OSI Open System Interconnection.

OSPF Open Shortest Path First.

P2P Peer-to-Peer.

PER Packet Error Rate.

PLCP Physical Layer Convergence Protocol.

PNC Physical layer Network Coding.

RAPTOR Rapid Tornado.

RFC Request For Comments.

RIP Routing Information Protocol.

RLC Random Linear Coding.

RLNC Random Linear Network Coding.

RLSC Random Linear Source Coding.

RR Round Robin.

RTO Retransmission TimeOut.

RTS/CTS Request To Send/Clear To Send.

RTT Round Trip Time.

RUDP Reliable User Datagram Protocol.

SACK Selective Acknowledgement.

SCTP Stream Control Transmission Protocol.

SF Slow Fading.

SIFS Short Inter-Frame Space.

SIGTRAN SIGnalling TRANsport.

SNR Signal to Noise Ratio.

SSN Subflow Sequence Number.

TCP Transport Control Protocol.

TLS Transport Layer Security.

TLV Tipo-Longitud-Valor.

UDP User Datagram Protocol.

UMTS Universal Mobile Telecommunications System.

v.a. variable aleatoria.

WiFi Wireless Fidelity.

WiMAX Worldwide Interoperability for Microwave Access.

WLAN Wireless Local Area Network.

WMN Wireless Mesh Network.

WRR Weighted Round Robin.

YANCI Yet Another Network Coding Implementation.

ZD Zone Disjoint.