CODIFICACIÓN A NIVEL DE PAQUETE PARA COMUNICACIONES MÓVILES POR SATÉLITE

Memòria del Treball Final de Carrera
d'Enginyeria Tècnica de Telecomunicació,
especialitat Sistemes Electrònics

realitzat per

Rafael Montalbán Gutiérrez

i dirigit per

Gonzalo Seco Granados

Bellaterra, 12 de setembre de 2007

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, Gonzalo Seco Granados

Professor de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Rafael Montalbán Gutiérrez.

I per tal que consti firma la present.

Signat: .....

Bellaterra, 14 de setembre de 2007

RESUMEN

Una de las opciones contempladas para transmitir contenidos multimedia y proporcionar acceso a Internet a grupos de usuarios móviles es el uso de satélites. Las condiciones de propagación del canal móvil implican que de alguna manera deberemos garantizar la calidad de servicio. Esto cobra incluso más importancia si tenemos en cuenta que, en el caso de acceso a Internet, no se tiene la capacidad de asumir cierto porcentaje de pérdida de datos que tenemos, por ejemplo, en la transmisión de sonido o vídeo (rebajando la calidad).

Entre las principales alternativas estudiadas para esta clase de entornos está la inclusión de codificaciones a nivel paquete. El funcionamiento de esta técnica se basa en incluir en la transmisión paquetes redundantes, obtenidos mediante un determinado algoritmo. El receptor será capaz de recuperar la información original que se quería enviar, siempre y cuando haya recibido correctamente una cierta cantidad de paquetes, similar a la cantidad de paquetes originales. A este mecanismo se le conoce como Forward Error Correction (FEC) a nivel de paquete.

En esta memoria se valora brevemente las alternativas existentes y se explican algunas de las codificaciones para FEC más importantes. A continuación se realiza un estudio comparativo de algunas de ellas: las variantes de LDPC (Low Density Parity Check) conocidas como LDGM (Low Density Generator Matrix), y la codificación Raptor.

RESUM

Una de les opcions que es contemplen per transmetre continguts multimèdia i proporcionar accés a Internet a grups de usuaris mòbils és fer servir satèl·lits. Les condiciones de propagació del canal mòbil impliquen que d'una manera o altra haurem de garantir la qualitat de servei. Això té fins i tot més importància si tenim en compte que, en el cas d'accés a Internet, no es té la capacitat d'assumir cert percentatge de pèrdua de dades que tenim, per exemple, en la transmissió de so o vídeo (rebaixant la qualitat).

Entre les principals alternatives per a aquesta classe d'entorns es troba la inclusió de codificacions a nivell de paquet. El funcionament d'aquesta tècnica es basa en incloure a la transmissió paquets redundants, obtinguts mitjançant un determinat algoritme. El receptor podrà recuperar la informació original que es volia enviar, sempre que hagi rebut una certa quantitat de paquets, similar a la quantitat de paquets originals. A aquest mecanisme se'l coneix com Forward Error Correction (FEC) a nivell de paquet.

En aquesta memòria es valoren breument les alternatives existents i s'expliquen algunes de les codificacions per a FEC més importants. A continuació es realitza un estudi compartiu d'algunes d'elles: les variants de LDPC (Low Density Parity Check) conegudes com LDGM (Low Density Generator Matrix), i la codificació Raptor.

ABSTRACT

The use of satellites is one of the options considered to broadcast multimedia contents and to provide access to the Internet for groups of mobile users. The propagation conditions of mobile channel make necessary to guarantee the quality of service. This is even more important if we note that no data loss is allowed in the case of providing Internet access, while in sound or video transmissions it is (reducing the quality).

One of the main alternatives which is being studied for this environments is the inclusion of packet level codifications. The basic idea of this technique is to include redundant packets in the transmission. These redundant packets can be obtained using a codification algorithm. The receiver will be able to recover all the original information provided it has received a certain amount of packets which will be similar to the amount of original packets. This technique is known as packet level Forward Error Correction (FEC).

In this dissertation, the alternatives to FEC are briefly explained. Next, some of the most important packet level FEC codifications are explained. After that, a comparative study of some of them will be done, specifically the LDPC (Low Density Parity Check) variants known as LDGM (Low Density Generator Matrix) and the Raptor codification.

ÍNDICE

CAPÍTULO 1: Introducción

Desde hace años la transmisión vía satélite en la banda Ku (10-12GHz) se usa con gran éxito comercial para la transmisión de contenidos multimedia (audio y vídeo) a múltiples terminales fijos. La principal ventaja de este tipo de sistema es que permite prestar servicio a un número potencialmente ilimitado de usuarios situados dentro de la amplia zona de cobertura del satélite. Posteriormente también se ha usado para proporcionar acceso a Internet, encapsulando los paquetes IP dentro de paquetes de los estándares típicos de satélite (como por ejemplo el protocolo del ETSI[1] DVB-S/S2[2]), ya sea usando un canal de retorno terrestre, o más recientemente el canal de retorno vía satélite proporcionado por estándares de la misma familia, como el DVB-RCS[3].

Era sólo cuestión de tiempo que se pensará en extender el uso de los satélites para la transmisión de contenidos multimedia a terminales móviles, y también su uso para proporcionar acceso a Internet a los usuarios de estos terminales. La gran cobertura que ofrecen los satélites existentes en la banda Ku, que cubre la mayor parte de la superficie terrestre (océanos incluidos) hace que la idea sea comercialmente prometedora. El problema principal reside en que los terminales móviles concretos tienen grandes limitaciones de tamaño y potencia disponible; por ejemplo, seria imposible dotarlos de las antenas necesarias. En cambio, se cree que sería viable, tanto técnica como económicamente, usar los satélites para dotar de esta clase de servicios a grupos de usuarios móviles compartiendo una misma infraestructura; como, por ejemplo, podrían ser los medios de transporte colectivos (trenes, barcos, autobuses, aviones comerciales,…). En estos casos las limitaciones antes descritas desaparecen, ya que tanto el tamaño como la potencia disponible no suponen un problema.

Este nuevo uso de los satélites para este propósito plantea también nuevos problemas a resolver. Los estándares anteriormente mencionados fueron diseñados para transmisiones a terminales fijos y no están preparados para el entorno móvil. La aplicación directa de estos estándares no sería válida, ya que al ser móviles los terminales, tendremos los problemas tradicionales del canal móvil: efectos doppler,

[1] European Telecomunications Standards Institute
[2] Digital Video Broadcasting via Satellite / via Satellite (2nd Generation)
[3] Digital Video Broadcasting – Return Channel via Satellite

handovers más o menos frecuentes y fenómenos de desvanecimiento (*fading*) y ensombrecimiento (*shadowing*).

Por ejemplo, en [1] se describe el canal móvil con el que nos encontramos en el ámbito ferroviario. Si ignoramos túneles y estaciones subterráneas, donde se necesitaría el apoyo de una infraestructura terrestre para mantener el servicio (la cual no se estudia en este documento), nos enfrentamos básicamente a fenómenos de *fading* profundo (>10dB), debidos a los arcos de alimentación. Éstos se producen de forma periódica y su duración es variable, dependiente de la velocidad a la que circule el tren y del trazado por el que circule. Además existe una cierta atenuación constante debida a la catenaria (<2dB) y a los postes de sujeción de ésta (2-3dB), que no supone un gran problema al ser relativamente baja, y *shadowing* producido por obstáculos (arboles, edificios, puentes,…), cuya aparición es más o menos aleatoria. En la Ilustración 1 se muestran arcos de alimentación y catenarias, como ejemplo de obstáculos atenuantes en un canal móvil.

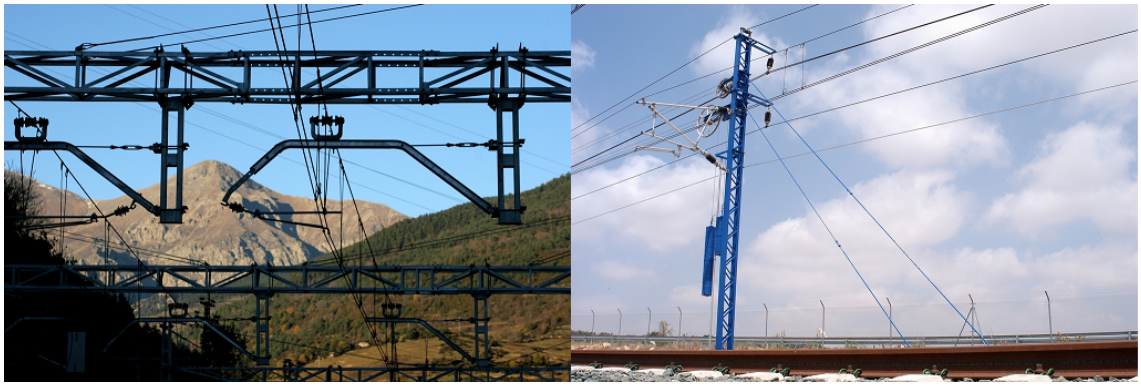

Ilustración 1 – Arcos de alimentación, postes de sujeción y catenarias

Todos estos problemas implican que nuevos estándares serán necesarios, o bien se deberán realizar modificaciones a los ya existentes. Estos estándares deberán añadir medidas para garantizar la calidad de servicio y la integridad de los datos enviados en el canal móvil. Además, se debe tener en cuenta que mientras que las transmisiones de audio y vídeo pueden permitir una cierta cantidad de pérdida de datos, con una consiguiente pérdida de calidad, el tráfico IP no lo permite. Tradicionalmente, en comunicaciones punto a punto se ha optado por el uso de ARQ[4], que consiste en una

[4] Automatic Repeat reQuest

petición automática de repetición; ya sea mediante una metodología de ACK[5], en la que se confirma la recepción de cada paquete (o grupo de paquetes) y si no se recibe confirmación de un paquete éste debe ser reenviado; o NACK[6], en la cuál si un paquete no se recibe se pide su reenvío. Esta aproximación ha demostrado ser óptima para las redes punto a punto tradicionales, pero en el escenario planteado, al menos por sí sola, no es la mejor opción. Esto se debe a que el canal de retorno es limitado y el tiempo de retorno elevado. Aparte de esto, es importante destacar que cuando se trata de difusión de contenidos multimedia (punto a multipunto, de un emisor a múltiples usuarios), es habitual que ni tan sólo dispongamos de un canal de retorno, y en el caso de disponer de uno las técnicas ARQ no son adecuadas porque implican muchas repeticiones; uno u otro usuario no habrá recibido correctamente la información y pedirá la repetición.

Si la alternativa tradicional no es adecuada para el entorno que nos planteamos, deberemos estudiar las alternativas existentes que no requieran del uso de un canal de retorno para garantizar una correcta recepción de la información. Entre otras, existen dos alternativas que se consideran adecuadas para este propósito en la transmisión vía satélite: el uso de FEC (Forward Error Correction) a nivel de paquete o el uso de diversidad. Estos dos métodos, adecuadamente aplicados, permiten reducir la probabilidad de que un paquete no se reciba (o no sea recuperable) en el terminal de destino.

Las técnicas de diversidad consisten en mejorar la fiabilidad de recepción de la señal usando 2 o más canales de comunicación diferentes. El hecho de usar múltiples canales, siempre y cuando estos sean estadísticamente diferentes, combinando adecuadamente la señal recibida por cada uno de ellos, permite aumentar la probabilidad de correcta recepción de la señal (e.g., no se producirá desvanecimiento en todos los canales simultáneamente). De entre las diferentes técnicas de diversidad se cree que se podrían ser una opción en éstos entornos la diversidad espacial y la diversidad temporal.

La diversidad espacial se basa en que la señal se propague por dos o más caminos estadísticamente diferentes. Esto se consigue mediante el uso de 2 o más antenas en

[5] ACKnoledgement
[6] Negative ACK

transmisión y/o recepción. Estas antenas deben estar situadas de manera que los caminos seguidos por la señal sean estadísticamente diferentes. Por ejemplo, en el caso comentado anteriormente, se podría situar dos antenas receptoras en el tren, a suficiente distancia entre ellas, de manera que se intentaría que como mínimo una de ellas en cada momento no sufriera los efectos atenuantes de los arcos de alimentación o el ensombrecimiento de obstáculos.

La diversidad temporal consiste en transmitir múltiples versiones de una misma señal en diferentes instantes de tiempo. Esta medida, aunque efectiva, supone el problema evidente de que la eficiencia espectral se reduce según aumentamos las retransmisiones de la información.

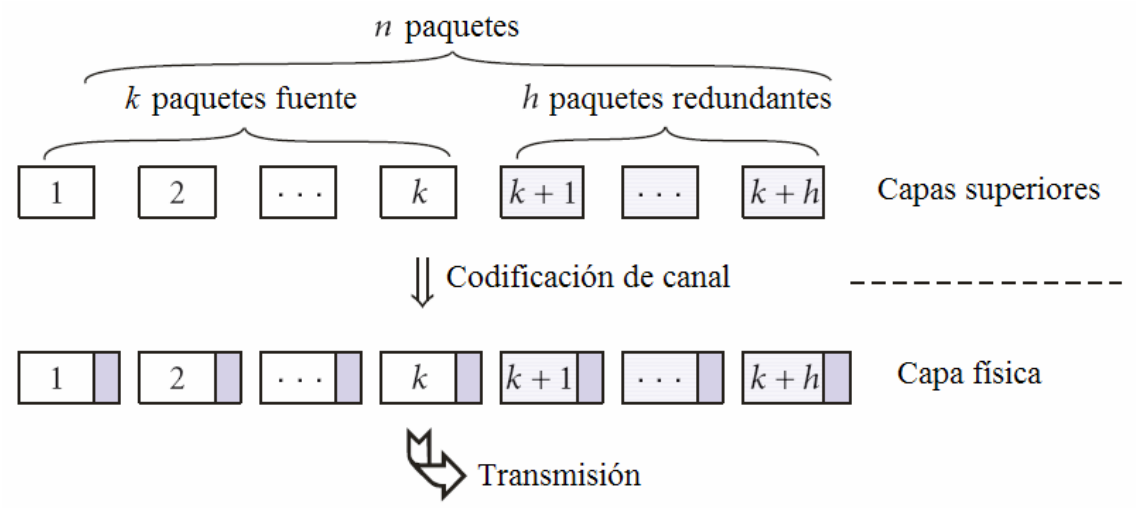

Ilustración 2 – FEC a nivel de paquete y codificación de canal

Por su parte el FEC consiste en aplicar una cierta codificación a la información que ha de ser enviada. Este proceso produce una cantidad de datos codificados superior a la original, que son los que finalmente se envían. El receptor será capaz de recuperar la información original que se quería enviar, siempre y cuando se haya recibido correctamente una cierta cantidad de datos, similar a la cantidad de datos originales. Estas codificaciones se pueden hacer a dos niveles diferentes: a nivel de bit (codificación de canal) o a nivel de paquete.

Estos dos niveles de codificación se muestran en la Ilustración 2. En el primer caso partimos de un paquete de longitud k bits, al cuál aplicamos una cierta codificación

obteniendo *n* bits (donde $n > k$, $n - k$ bits extra); se envían los n bits resultantes y si el receptor recibe correctamente como mínimo *k'* bits podrá recuperar los *k* originales que querían ser enviados. El funcionamiento a nivel de paquete es muy similar: tenemos *K* paquetes, de una longitud determinada, que codificamos obteniendo *N* paquetes de la misma longitud, los cuáles son los que enviamos. Si se reciben correctamente *K'* de los paquetes codificados se podrá recuperar los K paquetes originales que se iban a enviar. En ambas explicaciones *k'*/*K'* es igual a *k*/*K* o a una cantidad ligeramente superior, que dependerá de la codificación usada. La gran diferencia entre ambos niveles de codificación es básicamente la cantidad de datos que son protegidos por la codificación; en el primer caso se protege únicamente un paquete de la pérdida de unos cuantos bits, mientras que en el segundo caso se protege a *K* paquetes de la pérdida de un cierto número de paquetes.

En el entorno móvil se pueden producir periodos relativamente largos de desvanecimiento en los que puede que no recibamos correctamente los datos y que equivaldrán al tiempo de transmisión de una cierta cantidad de paquetes. Nos interesa, por tanto, estudiar las codificaciones a nivel de paquete. Éstas nos permitirán tiempos de protección de los datos suficientemente grandes en comparación con la longitud de los desvanecimientos, de manera que se pueda recuperar los paquetes que se pierdan debido a éstos.

En este proyecto se pretende estudiar diferentes tipos de codificación a nivel de paquete. En concreto se estudiarán en profundidad las codificaciones libres LDGM[7] y la codificación patentada DF Raptor, todas ellas codificaciones a nivel de paquete que ofrecen buenas prestaciones para grandes bloques de codificación. Las codificaciones LDGM, LDGM Staircase y LDGM Triangle son casos particulares de las codificaciónes LDPC[8], las cuáles han sido desarrolladas por el INRIA[9]. Las codificaciones Raptor, creadas por Amin Shokrollahi, están basadas en las codificaciones LT[10] inventadas por Michael Luby y están patentadas por la empresa Digital Fountain.

[7] Low Density Generator Matrix
[8] Low Density Parity Check
[9] Institut National de Recherche en Informatique et en Automatique
[10] Luby Transform

En está introducción se han presentado los problemas de la transmisión a terminales móviles vía satélite, las alternativas existentes para garantizar la calidad de servicio y el objetivo de este proyecto. En el capítulo 2, se realizará una explicación a nivel teórico de algunas de las codificaciones a nivel de paquete existentes. A continuación, en el capítulo 3 se presentará un análisis de diferentes aspectos de las codificaciones LDPC/LDGM y la codificación Raptor, y se realizará una comparación entre las diferentes codificaciones analizadas. Finalmente, el capítulo 4 resume los resultados del proyecto y presenta las conclusiones del mismo.

CAPÍTULO 2: Explicación Teórica

En este capítulo se pretende realizar una explicación de diferentes codificaciones que pueden ser aplicadas a nivel de paquete, sin entrar demasiado a fondo en los fundamentos matemáticos en que se basan cada una de ellas. Sin embargo, antes de empezar con la explicación de cada una de las codificaciones concretas, creo que será conveniente realizar una explicación de conceptos generales de la codificación a nivel de paquete y hacer una distinción entre las diferentes codificaciones que se explicarán. En esta explicación se usarán indistintamente los términos código y codificación.

En general, todos los códigos a nivel de paquete nos dan un número *n* de paquetes codificados a partir de *k* paquetes fuente. Definimos el ratio de codificación de la siguiente forma:

$$r = k/n$$

De forma equivalente se define el ratio de expansión FEC como:

$$fec\_ratio = 1/r = n/k$$

También de forma general, los decodificadores necesitan un cierto número de paquetes *k'* para ser capaces de realizar la recuperación de los paquetes fuente. *k'* es mayor o igual a *k* según la codificación.

Una codificación se considera sistemática cuando los *k* paquetes originales se encuentran entre los *n* paquetes codificados, y no sistemática cuando no es así. En general, se prefiere que un código sea sistemático, sobre todo para propósitos de fiabilidad parcial: si no recibimos paquetes suficientes para realizar la decodificación o ésta no tiene éxito, si usábamos una codificación sistemática, al menos tendremos algunos paquetes originales correctamente recibidos. En una transmisión de vídeo, por ejemplo, aunque no se disponga de la totalidad de la información, aún se podría reproducir, aunque reduciendo la calidad. En cambio si la codificación era no sistemática los paquetes que tendremos no tendrán ninguna utilidad.

En el caso de las codificaciones MDS[11], son necesarios exactamente *k* paquetes codificados para poder recuperar los *k* paquetes originales, lo que es óptimo desde el punto de vista de la cantidad de paquetes necesaria para realizar la decodificación. Lamentablemente, las codificaciones MDS existentes, como por ejemplo la Reed-Solomon, tienen una complejidad de cálculo de codificación y decodificación elevada, y ésta suele depender cuadráticamente del tamaño, lo que lo limita bastante. Por lo tanto las codificaciones prácticas para grandes bloques, en general, no serán MDS y podemos entender que introducen una ineficiencia de codificación que definimos como:

$$inef\_cod = \frac{número\ de\ paquetes\ necesarios\ para\ descodificar}{k} = \frac{k'}{k}$$

Esta ineficiencia será un número mayor o igual a 1, de manera que si es exactamente 1 la codificación será óptima y en el resto de los casos no lo será. Cuanto mayor sea *inef_cod* más ineficiente será la codificación.

Existe otra fuente de ineficiencia que aparece cuando un archivo u objeto es demasiado grande para ser codificado en un solo bloque, de manera que se segmenta en varios bloques. Si se usan bloques pequeños, se hace imprescindible que la transmisión de los paquetes se haga en orden pseudo-aleatorio, mediante un entrelazador. De esta manera evitamos que una ráfaga de errores grande en comparación con la longitud de un bloque nos impida la correcta decodificación de éste, y por extensión nos impida llegar a recibir correctamente el archivo entero. Pero al transmitir en orden aleatorio se produce el *coupon collector problem* (cuya traducción aproximada es “problema del coleccionista de cupones”): se reciben paquetes de bloques que ya han sido decodificados mientras el receptor espera por el último paquete que le falta en un bloque que aún no ha decodificado. Finalmente, estaremos recibiendo para algunos bloques cantidades de paquetes superiores a *k*, lo que significa que, a pesar de estar usando una codificación teóricamente óptima, ya no estaremos en una situación óptima. Definimos la siguiente ineficiencia debida al uso de codificadores de bloque pequeño para codificar objetos grandes, donde T será el total de paquetes que hemos recibido hasta decodificar todos los bloques, y B será el número de bloques:

$$inef\_coupon = \frac{T}{k \cdot B}$$

[11] Maximum Distance Separable

Una inteligente manera de evitar este problema cuando se usan codificaciones de bloque pequeño se muestra en [1] (similar a la usada en el estándar del ETSI DVB-H[12]), logrando codificar grandes cantidades de datos mediante codificaciones de bloque pequeño sin que ocurra este problema. Con esta técnica existen, sin embargo, otras limitaciones que se comentarán más adelante.

Por otro lado, los bloques grandes son deseables puesto que un paquete redundante solo puede permitir la recuperación de un paquete borrado en su bloque. Es obvio que la solución óptima es que un archivo concreto se codifique en un solo bloque. Esto sólo es posible si trabajamos con codificaciones que permitan tamaños de bloque muy grandes.

A continuación se explicará el funcionamiento básico de las codificaciones LDPC/LDGM, las codificaciones Raptor, y las Reed-Solomon. De estas codificaciones, la Reed-Solomon se considera de bloque pequeño, aunque existen algunas técnicas, como la que ya se ha comentado, que permiten mejorar la aplicación de esta clase de códigos a archivos más grandes. Algunas de estas técnicas se explicarán brevemente en la sección de este capítulo que se dedicará a Reed-Solomon. Por su parte, las codificaciones Raptor y las LDPC/LDGM son codificaciones de bloque grande.

A efectos del tipo de canal al que se pueden aplicar estas codificaciones, todas ellas pueden operar tanto en Canales Binarios Simétricos (BSC[13]) como en Canales de Borrado Binario (BEC[14]). En una canal BSC un bit llega al receptor, o bien correctamente, o bien erróneamente, es decir, con su valor invertido. Esto sucede normalmente en canales ruidosos. Por otro lado, en un canal BEC un bit, o llega bien, o se pierde en el canal y no se recibe. Un ejemplo típico de BEC, que no opera en flujos de bits, sino en flujos de paquetes es Internet. En este caso, la razón principal de las pérdidas de paquetes es la congestión en los *routers*, y el CRC disponible en la mayoría de capas físicas garantiza que un paquete recibido no es erróneo (si lo es, se detecta y se considera que se ha perdido). Aunque algunos de estos códigos se pueden aplicar en los dos tipos de canal su funcionamiento es mucho más sencillo en los Canales de Borrado

[12] Digital Video Broadcasting for Handled terminals
[13] Binary Symmetric Channel
[14] Binary Erasure Channel

de Paquetes (PEC[15]) como Internet. Ya que normalmente el efecto de los desvanecimientos será el borrado de los paquetes transmitidos durante la duración de los mismos, y en la capa en que se sitúa el FEC a nivel de paquete (Aplicación/Transporte) podemos suponer que los paquetes erróneos se considerarán borrados, modelaremos, en nuestro caso, el canal satélite como un PEC. En el resto de este documento supondremos siempre que si se ha recibido un paquete, éste es correcto.

2.1 Codificaciones LDPC y LDGM

Los códigos LDPC fueron introducidos por primera vez por Gallager en 1960 [4,5] y permanecieron prácticamente olvidados hasta que en 1995 fueron recuperados por MacKay y Neal [6]. Posteriormente los códigos LDPC han sido usados por Luby y Shokrollahi como base para el desarrollo de códigos como los Tornado, los LT y los Raptor.

Los LDPC son códigos bloque lineales sistemáticos con una matriz de comprobación de paridad (*Parity Check Matrix*) H muy dispersa, es decir, que la mayoría de sus elementos valen 0. La matriz H tiene dimensiones $h \times n$ (donde $h = n - k$) y se puede entender como un sistema de *h* ecuaciones, representadas por cada una de las filas de la matriz. Los *k* primeros elementos de cada fila representan a los paquetes fuente y los *h* siguientes, que completan los *n* que hay en cada fila, representan a los paquetes redundantes resultantes de la codificación. Si un elemento concreto tiene valor 1 quiere decir que el paquete al que representa interviene en la ecuación a la que corresponde la fila en la que esté. La ecuación correspondiente a cada fila será la siguiente: la suma XOR bit a bit de todos los paquetes que intervienen en esa ecuación debe ser igual a un paquete con todos los bits a 0.

Un ejemplo de esto se ve más claramente en la Ilustración 3, en la que se muestra una matriz H de dimensiones 12×16 ($k = 4$, $n = 16$, $h = 12$) que nos da un ratio de codificación $r = 1/4$, en este caso la ecuación correspondiente a la primera fila sería: $s_3 \oplus p_4 \oplus p_6 \oplus p_9 = 0$, donde s_i representa un paquete fuente y p_i representa un paquete redundante. Esta matriz se consideraría regular puesto que cada paquete interviene exactamente en el mismo número de ecuaciones (mismo número de

[15] Packet Erasure Channel

elementos a 1 en cada columna), es decir, todos los paquetes tienen peso $j = 3$. También cabe destacar que en la matriz de la figura no se ve con claridad la dispersión de ésta, lo que se debe al hecho de que sus dimensiones son muy pequeñas. En la práctica se suelen usar matrices de k = 10000 dónde el peso de cada paquete no suele superar $j = 10$ [2].

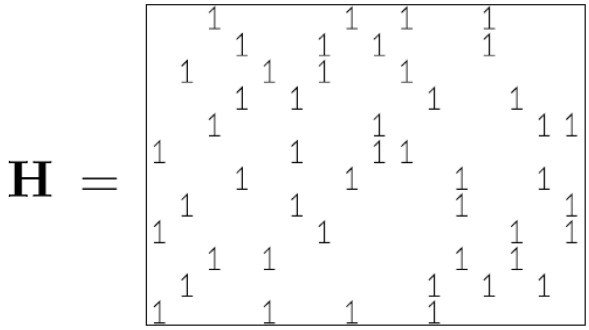

Ilustración 3 – Ejemplo de matriz LDPC (los elementos a 0 han sido omitidos)

El peso de cada una de las columnas es importante puesto que influye bastante en las cualidades del código. Este peso también suele ser llamado *left degree*. Al número de elementos que aparecen en cada ecuación, por su parte, se le llama *right degree*. En una matriz regular como la de la ilustración, tanto uno como otro son constantes para cualquier columna o fila. En este tipo de matrices la relación entre uno y otro será la siguiente:

$$right\_deg_{LDPC} = \frac{n \times left\_deg}{n-k}$$

Una forma dual de entender la codificación LDPC es mediante un grafo bipartito. El grafo bipartito correspondiente a la matriz de la Ilustración 3 se muestra en la Ilustración 4. En estos grafos existen dos clases de nodos: los nodos que representan a los paquetes (tanto fuentes como redundantes, representados en la figura por círculos), también llamados nodos mensaje, y los nodos que representan a las ecuaciones, también llamados nodos de comprobación. Las líneas de conexión entre los nodos equivalen a los unos de la matriz: una línea de unión entre un nodo mensaje y un nodo de comprobación significa que este paquete aparece en la ecuación correspondiente. En el

caso de ejemplo vemos como de cada nodo mensaje salen 3 líneas ($left\_deg = 3$) y que a cada nodo de comprobación llegan 4 líneas ($right\_deg = 4$).

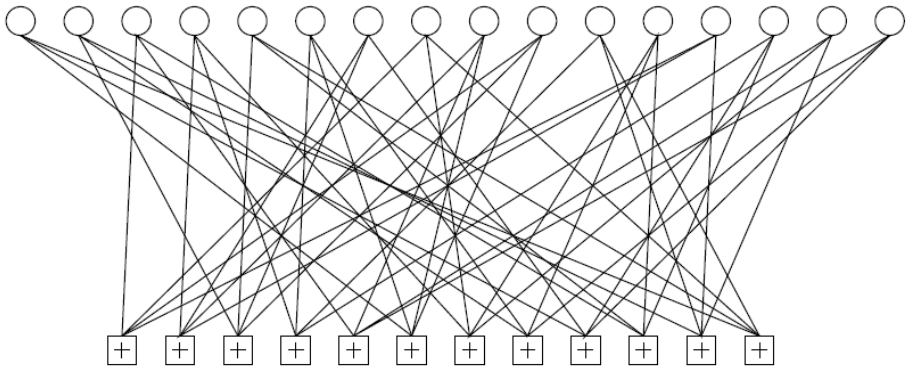

Ilustración 4 – Grafo bipartito de la matriz H de la Ilustración 3

Volviendo a la representación matricial, el proceso de codificación consistirá en resolver el sistema de ecuaciones lineales que representa la matriz. Esta es una tarea compleja que puede consumir bastante tiempo, por tanto, lo que normalmente se hace es producir una matriz generadora G, de manera que la codificación se realice multiplicando los paquetes fuente por esta matriz. Desafortunadamente la matriz G es una matriz densa (muchos elementos a 1) lo que hace que el proceso sea muy lento, siendo el tiempo de codificación proporcional a $(n-k)*k$ [3]. En el caso de la representación mediante grafo, podemos entender la codificación de la siguiente manera: si conocemos todos los nodos paquete conectados con un nodo de comprobación menos 1, podemos calcular el valor de éste como la suma XOR bit a bit de los paquetes conocidos.

Las codificaciones LDGM son casos particulares de códigos LDPC que tienen la propiedad de permitir una codificación rápida, de hecho muy rápida si la comparamos con el método de codificación LDPC tradicional. Esta rápida codificación es posible gracias a que la matriz H esta formada por una matriz $h \times k$ de tipo disperso (a la cuál llamaremos H'), concatenada con una matriz de dimensiones $h \times h$ (que llamaremos P). La matriz P será una matriz identidad (en el caso de LDGM), una matriz bidiagonal inferior o escalera (en el caso de LDGM Staircase) o una distribución pseudo-aleatoria de unos en la parte inferior de una matriz bidiagonal (en el caso de LDGM Triangle). Es decir tenemos una matriz LDPC en la que las columnas correspondientes a los paquetes redundantes han sido simplificadas muchísimo, lo que reduce las propiedades

recuperadoras de borrados del código, pero disminuye mucho la complejidad de codificación.

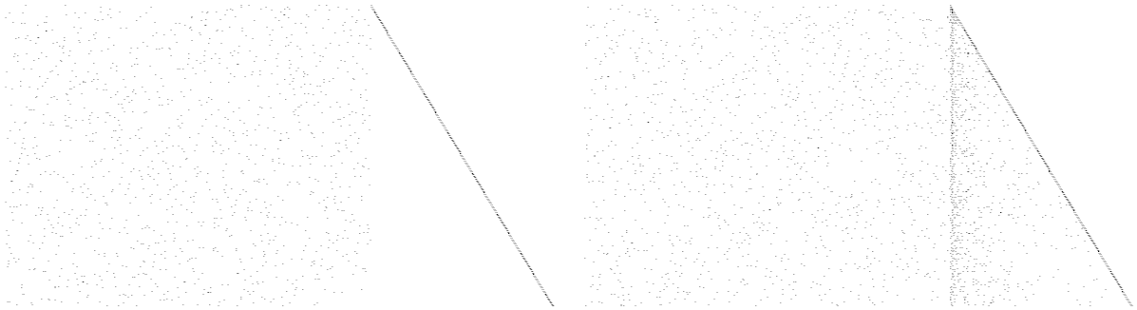

Ilustración 5 – Matriz LDGM Staircase y matriz LDGM Triangle (los elementos a 0 han sido omitidos)

La Ilustración 5 muestra dos ejemplos de matrices LDGM Staircase y Triangle. En la parte derecha de esta ilustración podemos observar la matriz bidiagonal en las *h* últimas columnas de la matriz LDGM Staircase. En la parte izquierda se muestra una matriz LDGM Triangle, siendo las últimas *h* columnas una matriz bidiagonal con una distribución pseudo-aleatoria de unos en el triángulo inferior. Se puede observar como la concentración de unos es mayor en las primeras columnas del triángulo, de manera que los primeros paquetes redundantes estarán más protegidos y darán más protección, al aparecer en más ecuaciones. Se puede encontrar el algoritmo de rellenado de la matriz bidiagonal, para obtener la matriz triangle en [2].

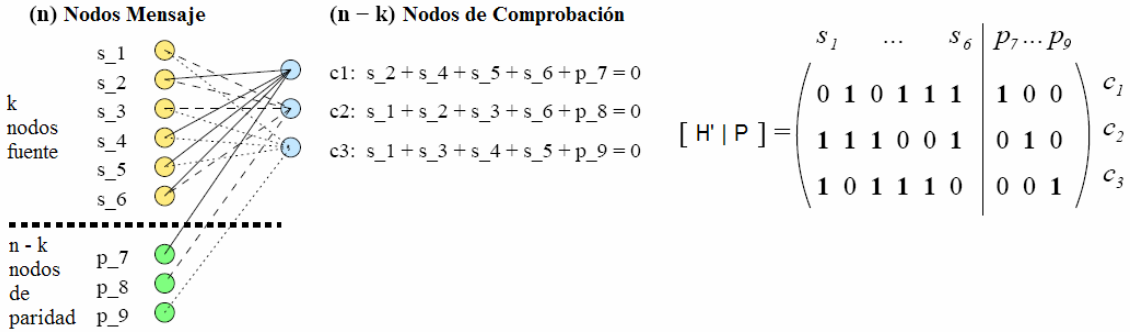

Ilustración 6 – Grafo bipartito y matriz de un código LDGM

La Ilustración 6 muestra un grafo bipartito junto con la matriz LDGM que le corresponde. En este caso vemos como dos líneas unen a cada nodo mensaje fuente a cada nodo de comprobación ($left\_deg = 2$ en H'), mientras que una sola línea une un nodo mensaje paridad con cada nodo de comprobación. Como se pude deducir, toda

matriz LDGM, será una matriz LDPC irregular; aunque podremos seguir hablando de *left degree*, refiriéndonos al número de veces que aparece cada paquete fuente en las ecuaciones, y hablaremos de un matriz LDGM regular cuando este número sea constante para todos los paquetes fuente.

La ventaja principal de un código LGDM sobre un LDPC tradicional es que, al estar el triángulo superior de P vacío (todos sus elementos son 0), podemos aplicar un algoritmo de codificación muy sencillo. El sistema equivalente a una matriz LDGM de cualquiera de los tres tipos es resoluble de manera trivial siguiendo el siguiente procedimiento: resolvemos por orden las ecuaciones representadas en la matriz de paridad, obteniendo de la primera el primer paquete redundante, de la segunda el segundo,… y así sucesivamente.

Por otro lado la decodificación de cualquier código LDPC, sea LDPC tradicional o LDGM, se puede realizar de manera sencilla siguiendo un algoritmo iterativo según se van recibiendo los paquetes. Esto es especialmente interesante puesto que como la decodificación se realiza mientras se reciben los paquetes, al recibir el último paquete necesario para acabar la decodificación el tiempo que se tarda en finalizarla es mínimo. El algoritmo de decodificación iterativo es el siguiente:

- Cada vez que se recibe un paquete su valor es sustituido en todas las ecuaciones en que parece.
- Se comprueba si en alguna de las ecuaciones sólo falta por conocer un paquete, si es así, se calcula su valor y si quedan ecuaciones por resolver, se sustituye el paquete que se ha calculado en ellas y se realiza otra iteración de este paso; si no se puede resolver ninguna ecuación, esperamos a la recepción de otro paquete.

La decodificación acaba cuando no queda ninguna incógnita en las ecuaciones.

2.2 Codificaciones Raptor

Las codificaciones Raptor son un tipo de codificaciones inventadas por Shokrollahi, de la clase “Fuente Digital” o codificaciones Fuente. Estas codificaciones están basadas en unas codificaciones de esta misma clase conocidas como LT, creadas por Luby.

Las codificaciones del tipo Fuente tienen la peculiaridad de que *n* no está fijado *a priori* y el codificador, teóricamente, puede generar un flujo infinito de paquetes redundantes diferentes. En un codificador de este tipo, nos entran *k* elementos (bits, símbolos o paquetes) a partir de los cuáles se calcula los elementos de salida. Los elementos redundantes son calculados como la suma XOR bit a bit de algunos de los elementos fuente, de forma similar a la manera en que los paquetes LDGM eran calculados. Se supone que de alguna manera el decodificador sabrá a partir de que elementos fuente fue obtenido un determinado elemento redundante. Esto en la práctica se puede conseguir mediante la adición de cabeceras en los paquetes que contengan la información suficiente para que el receptor pueda saber como se calculó ese paquete concreto. Otra opción podría ser existiera una cierta sincronización temporal entre emisor y receptor, de manera que el receptor pudiera predecir como fue generado cada paquete. Es importante remarcar que sin esta información un paquete obtenido de una fuente digital no sirve para nada.

En esta clase de codificaciones, entendemos por algoritmo de decodificación aquél que es capaz de recuperar los *k* elementos originales a partir de cualquier conjunto de *k'* elementos redundantes, con alta probabilidad. En general, en este tipo de codificaciones no se puede garantizar un 100% de probabilidad de éxito en la decodificación aunque hayamos recibido *k'* elementos. Para considerar que una codificación es una buena codificación Fuente, el valor de *k'* debe ser muy cercano a *k*, y el tiempo de decodificación debe ser más o menos linear respecto de *k*.

Los códigos LT, en los cuales se basan las codificaciones Raptor, funcionan de la siguiente a manera. Cada vez que un elemento redundante es generado en un codificador LT, una distribución de pesos es muestreada devolviendo un entero *d* que estará entre 1 y *k*. Entonces *d* elementos originales son elegidos de forma más o menos aleatoria, de manera que el elemento redundante será la suma XOR de éstos. Por lo tanto entenderemos el peso de un elemento redundante como el número de paquetes originales que fueron sumados para generarlo.

El problema principal que tienen los códigos LT es que al ser elegidos de manera pseudo-aleatoria los paquetes que se sumaran para generar cada paquete redundante, nos podemos encontrar con la situación de que habiendo recibido los paquetes necesarios

para poder decodificar, esta fracase, puesto que nada garantiza que todos los paquetes originales hayan sido suficientemente cubiertos. Esto se soluciona aumentando el peso medio de los paquetes redundantes, pero esto complica la codificación y la decodificación. En [8] y [9] se demuestra que si un código LT tiene un algoritmo de decodificación con una probabilidad de error que es como mucho inversamente polinómica en *k*, y si el algoritmo necesita *k'* símbolos redundantes para operar, entonces el peso medio de un paquete redundante debe ser al menos $c \cdot k \cdot \log(k)/k'$, para una cierta constante *c* que estará determinada por *k* y por la probabilidad aceptable de fallo en la decodificación. Por lo que en el caso deseable en que *k'* es muy cercana a *k*, la codificación LT necesita un peso medio proporcional a $\log(k)$. Es importante remarcar que existen distribuciones de peso que consiguen este límite con un decodificador bastante rápido. Estas distribuciones se muestran en [10].

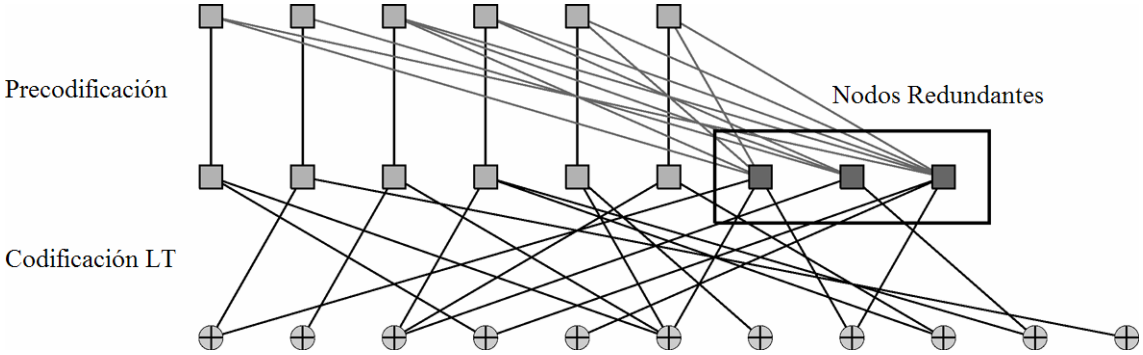

Ilustración 7 – Grafo de una codificación Raptor

La idea básica tras las codificaciones Raptor es relajar la condición necesaria en cuanto a peso medio de la codificación LT; digamos que hasta el punto de que únicamente una fracción constante de elementos entrantes en el codificador LT sea recuperable. Con esta aproximación aparece evidentemente un problema: necesitamos recuperar todos los elementos originales y no sólo una fracción constante. Esto se soluciona añadiendo una codificación previa al codificador LT, a la cual llamaremos precodificación. Esta precodificación garantizará que a partir de la fracción constante de elementos intermedios (elementos resultantes de la precodificación que serán la entrada del codificador LT), los elementos originales serán recuperables. Esta idea se puede ver de forma gráfica en la Ilustración 7. Los elementos originales son precodificados mediante una codificación del tipo LDPC. Después se les aplica la codificación LT, y

ahora ya no hará falta que ésta pueda garantizar la recuperación de todos los nodos intermedios: con una cierta fracción la precodificación ya podrá recuperar todos los paquetes originales. Gracias a esto, aunque codificación y decodificación parecen haberse complicado con un paso extra, en realidad se simplifica la complejidad de las codificaciones a usar.

En el caso de los códigos Raptor no se acostumbra a usar el término ineficiencia, sino que en su lugar, normalmente, se habla de *overhead*. Simplemente se debe tener en cuenta que:

$$overhead = \varepsilon = inef\_cod - 1$$

En [8] se muestra una clase de códigos Raptor universales que para un número de paquetes de entrada k y un $\varepsilon > 0$, pueden producir infinitos paquetes codificados de manera que con sólo $k(1+\varepsilon)$ de estos paquetes, se pueden recuperar los k paquetes originales que se quería enviar. La complejidad de codificación para un paquete de éste tipo sería proporcional a $\log(1/\varepsilon)$ y la complejidad de decodificación de los k paquetes originales sería $k \cdot \log(1/\varepsilon)$, para una cierta probabilidad de fallo en la decodificación. La probabilidad de fallo deseada en la decodificación también influye en la complejidad del código, cuanto más pequeña se desee más complejo será. Es decir, podemos conseguir codificaciones de muy baja ineficiencia y probabilidad de fallo, pero cuanto menor se desee que sean, más complicado será codificar y decodificar. Para más información al respecto se recomienda la lectura de [8].

Es importante destacar que las técnicas de codificación descritas en este apartado implican que la codificación que se realizará no será sistemática, puesto que se enviará exclusivamente paquetes redundantes, entre los cuales no habrá necesariamente ninguno de los paquetes originales. Por las razones que ya se han comentado en la introducción de este capítulo se prefieren los códigos sistemáticos, por lo que este es un inconveniente a tener en cuenta. La idea inmediata de transmitir los elementos originales antes de enviar los redundantes, prueba rápidamente no ser adecuada. Puesto que estos elementos no son producto de la codificación no podrían ser usados en la recuperación de los elementos borrados y la ineficiencia sería muy grande. Si se desea una codificación Raptor sistemática, la mejor opción es realizar una precodificación no

sistemática y modificar el codificador LT de manera que los primeros *k* paquetes codificados que produzca sean los *k* paquetes originales. Esto supone un aumento en la complejidad de codificación y decodificación, pero es asumible [8].

2.3 Codificaciones Reed-Solomon

Las codificaciones Reed-Solomon o RS se diferencian de las que hemos explicado en los apartados anteriores en que son consideradas codificaciones de bloque pequeño. Esto se debe a que las codificaciones RS están intrínsicamente limitas por el uso de campos Galois de longitud finita.

El tamaño del campo finito es igual a $q-1=2^m-1$, que a su vez es el máximo teórico de símbolos codificados, *n*, que podremos tener. Por otra parte, *m*, es el tamaño en bits de cada uno de los símbolos, tanto los de entrada como los resultantes de la codificación. Debido a limitaciones matemáticas a nivel de complejidad de cálculo (está crece cuadráticamente respecto de *n*), *m* raramente podrá valer más de 16, siendo mucho más común $m=8$; razón por la cuál prescindimos del término paquete. Con un sencillo cálculo podemos ver que las longitudes máximas de los bloques de codificación serán:

$$L(m=8)=(q-1)\cdot m\cdot r=\left(2^8-1\right)\cdot 8\cdot r=2040\cdot r \text{ bits}=255\cdot r \text{ bytes}$$

$$L(m=16)=(q-1)\cdot m\cdot r=\left(2^{16}-1\right)\cdot 16\cdot r=1048560\cdot r \text{ bits}=131070\cdot r \text{ bytes}$$

Donde *r* es el ratio de codificación y por lo tanto será más pequeño que 1. Vemos que, incluso en el caso de usar $m=16$ en el que la complejidad de cálculo ya hace a los codificadores casi impracticables, la longitud de bloque máxima que nos podrá dar no supera los 131 KBytes, lo que confirma la condición de bloque pequeño.

A pesar de estos inconvenientes en cuanto a tamaño de bloque, hay que recordar que las codificaciones RS son del tipo *Maximum Distance Separable*, lo que significa que $k=k'$, es decir, es necesaria exactamente la misma cantidad de paquetes codificados que paquetes originales teníamos, para llevar a cabo la decodificación. Por lo tanto en ese aspecto son óptimos y merecen ser tenidos en cuenta.

En uno de los apartados de [12] se realiza un estudio para un caso determinado de codificador RS ($n=192$, $k=128$, $r=2/3$) y un codificador LDGM del mismo ratio. En el se demuestra que la ineficiencia debida al *coupon collector problem* que sufre el RS, fenómeno que fue explicado anteriormente, iguala a la ineficiencia intrínseca del LDGM para un valor de bloque de codificación 1200 KB. Cabe destacar, además, que en este artículo se usaban LDGM normales, siendo (como se verá en el capítulo de análisis) más eficientes los LDGM Staircase y Triangle, por lo que probablemente para estos el valor sea incluso más bajo.

Por otra parte, existen algunas técnicas que permiten mejorar la aplicación de esta clase de códigos a bloques más grandes. En [1] por ejemplo se propone la solución mostrada en la Ilustración 8, la cual es muy parecida a la usada en el estándar DVB-H. Se propone colocar los paquetes a enviar, en este caso IP, por columnas en una matriz (en la que cada elemento es un byte) rellenando hasta 192 columnas. Después se realiza una codificación RS de $k=192$ y $n=255$, ampliando de esta manera la matriz hasta las 255 columnas, donde 64 columnas serán redundantes. Después se procede a enviar los paquetes IP uno a uno, encapsulados en el protocolo inferior, y las columnas de símbolos redundantes, a modo de paquetes redundantes, encapsuladas en este mismo protocolo. En recepción se procede a realizar el proceso inverso, rellenando la matriz con los paquetes recibidos correctamente. Si en menos de 64 columnas no se ha borrado algún paquete o se ha recibido erróneamente (en éste caso se considera como si se hubiese borrado) se podrá recuperar la matriz original. El tamaño del bloque con este sistema se ajusta aumentando la longitud de las columnas. El problema reside en que si aumentamos demasiado la longitud de ésta, aumentaremos la probabilidad de columna errónea, hasta el punto de que se pierdan por esta causa más de 64 columnas. Por otra parte, si no tenemos un tamaño de bloque suficientemente grande podrá haber ráfagas de borrados de más de 64 columnas. Este problema no aparece en una codificación que sea realmente de bloque grande, puesto que no tendremos limitaciones en cuanto a tamaño de bloque, excepto la obvia que nos impone el retardo que introduce el aplicar la codificación.

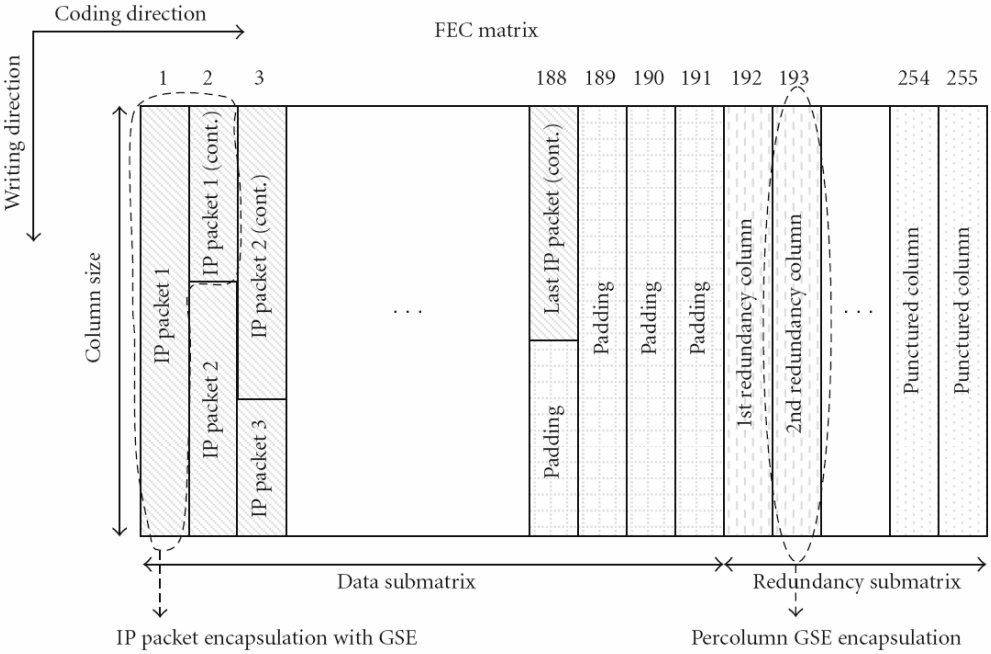

Ilustración 8 – Escritura de los paquetes por columnas y codificación RS por filas

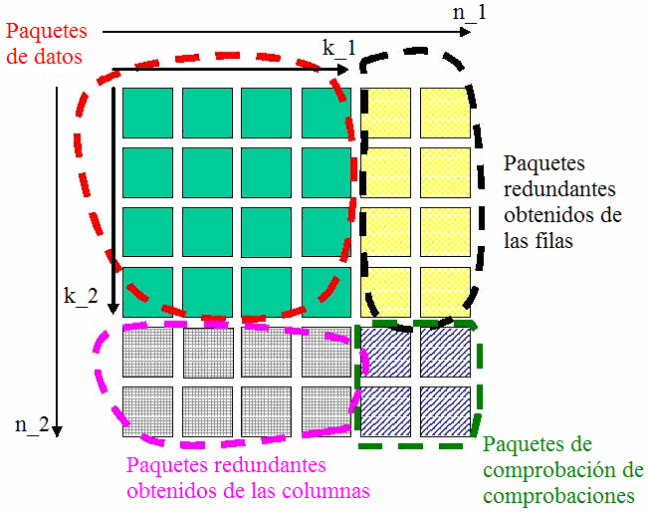

Ilustración 9 – Aplicación de la codificación RS en 2D

Otra alternativa que se explica en [13] es la aplicación de codificación RS en dos dimensiones. Esta idea se muestra en la Ilustración 9. Siguiendo este método podemos aplicar una codificación RS de bloque pequeño a un bloque bastante más grande. Por ejemplo, si usamos un codificador RS donde $k = 192$ y $n = 255$ podemos conseguir un bloque resultante en el cual $k_{total} = k^2 = 36864$ y $n_{total} = n^2 = 65025$. Sin embargo, si recordamos que cada posición de esta matriz será de 1 byte, no se puede decir que el bloque resultante sea demasiado grande, aunque supone una mejora.

Las conclusiones a las que se llega en el artículo es que la codificación RS en 2D se comporta como es de esperar en una codificación de bloque grande y es más eficiente en cuanto a consumo de ancho de banda que el RS tradicional. Además puede ser interesante para algunas aplicaciones la idea que se propone, según la cual mediante el uso de un solo codificador RS capaz de cambiar entre codificación 2D y 1D, no necesitaríamos tener una codificación determinada para bloques grandes y otra para bloques pequeños.

CAPÍTULO 3: Análisis de codificaciones a nivel de paquete

En este capítulo se analizarán algunas de las codificaciones a nivel de paquete explicadas anteriormente. En el caso de LDPC/LDGM los datos que se muestran han sido obtenidos mediante simulación. En el caso de Raptor, debido a problemas técnicos, los datos que se muestran han sido extraídos de los artículos citados. Así mismo se compararán estas codificaciones entre ellas a tenor de los resultados mostrados.

3.1 Características de la simulación

En el CD adjunto a la memoria se encuentran los distintos programas y funciones usadas para realizar las simulaciones y algunas gráficas que ilustran las próximas secciones. De los programas en C++ se incluyen los códigos fuente y los binarios para plataformas x86 Windows y GNU/Linux. Las funciones y *scripts* de Matlab se incluyen en forma de ficheros ".m".

Para la realización de las simulaciones de las codificaciones LDPC/LDGM se han escrito toda una serie de funciones en Matlab. Estas funciones implementan, entre otros, un codificador, un decodificador, un generador de matrices H y un simulador. Lamentablemente, a pesar de que éstas funcionan correctamente y pueden ser usadas para simulaciones puntuales, para los tamaños de matriz que nos interesan su comportamiento es excesivamente lento. Por lo tanto para realizar las simulaciones de estas codificaciones se ha optado por el código C++ libre del INRIA que implementa una librería y distintos simuladores del comportamiento de los LDPC/LDGM. A este código se le han realizado algunas modificaciones para facilitar el uso de los resultados de interés.

En el caso de las simulaciones de las codificaciones Raptor, a pesar de que se pretendía implementar un simulador basado en lo que se explica en [14], [15] y [16], debido a problemas con la programación en C++ se ha tenido que optar por los resultados existentes en la literatura. Sin embargo, en el CD adjunto se incluye una versión incompleta del código fuente escrito.

Como referencia para los tiempos de codificación y decodificación de LDPC/LDGM, se debe tener en cuenta que la plataforma sobre la que se han realizado

las simulaciones se trata de un AMD Turion 64 X2 @1,60GHz con 1GB de memoria RAM. Las simulaciones se han realizado sobre el sistema operativo Ubuntu Linux 7.04.

3.2 Codificaciones LDPC/LDGM

A no ser que se indique lo contrario, las simulaciones que se mostrarán en esta sección han sido realizadas para un número de paquetes de entrada *k* igual a 10000 y con un tamaño fijado para cada uno de los paquetes de 1kB, de manera que el bloque de codificación serán 10MB, lo que se considera representativo de lo que sería un bloque de codificación grande.

Para encontrar resultados más completos de esta clase de codificaciones que los que se mostrarán, se recomienda la lectura de [2], artículo que ha sido escrito por los creadores de estos códigos.

3.2.1. Codificaciones LDGM

En la Ilustración 10 podemos ver el comportamiento de codificaciones LDGM con *left degree* 3, de diferentes ratios de expansión FEC, en función de la probabilidad de pérdida de paquete.

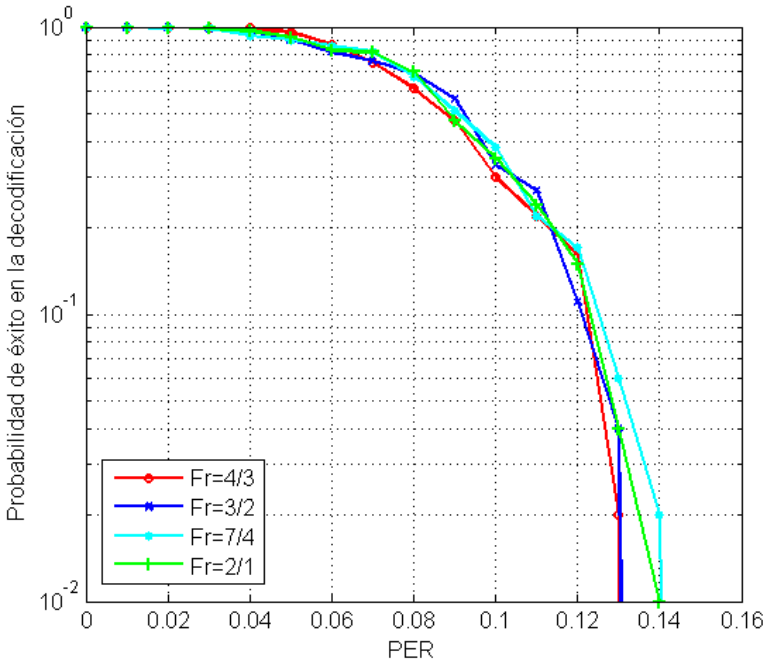

Ilustración 10 – Probabilidad de éxito en la decodificación en función de la probabilidad de pérdida de paquete para diferentes códigos LDGM

Puede resultar extraño el hecho de que a pesar de aumentar el ratio de expansión los resultados sean prácticamente iguales. Esto se debe a que la ineficiencia para estas codificaciones crece si aumentamos el ratio de expansión. En el caso de LDGM con *left degree* 3 este incremento es muy pronunciado, tal como se deduce de la Ilustración 10 y como se muestra en la Ilustración 11. Es importante notar que si aumentásemos el *left degree*, podríamos obtener resultados de ineficiencia mejores (en las ecuaciones aparecen más paquetes, por lo que un paquete redundante podrá recuperar más paquetes), pero esto se haría al coste de disminuir la dispersión de la matriz generadora, aumentado de forma importante los tiempos de codificación y decodificación.

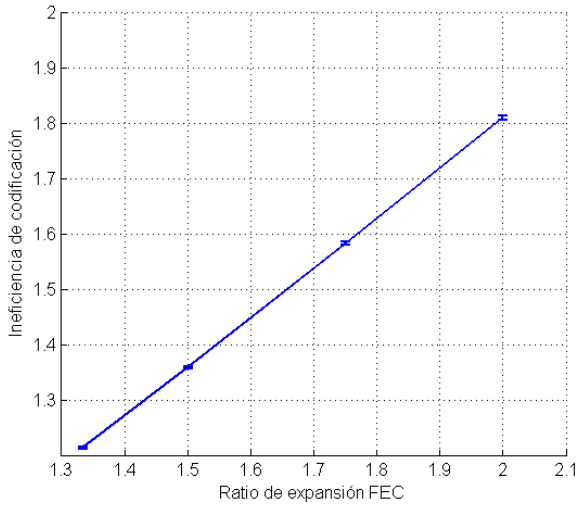

Ilustración 11 – Ineficiencia de codificación en función del ratio de expansión FEC para códigos LDGM de *left degree* = 3, los intervalos corresponden a una confianza del 99%

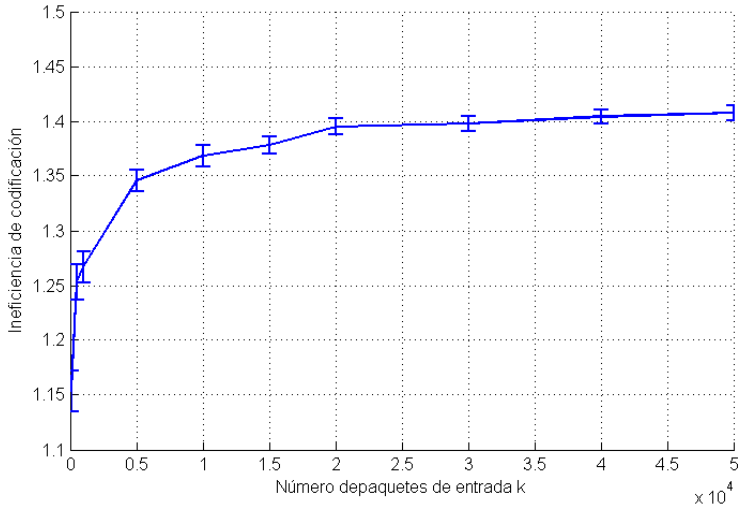

Ilustración 12 – Ineficiencia de codificación en función del número de paquetes de entrada para códigos LDGM de *left degree* = 3 y ratio de expansión FEC de 1.5, los intervalos corresponden a una confianza del 99%

También se ha estudiado la evolución de la ineficiencia en función del número de paquetes de entrada *k*. Es importante remarcar que mientras que la ineficiencia, tal como se muestra en la Ilustración 12, varía en función del número de paquetes de entrada, no depende en absoluto del tamaño de estos, que sólo influirá en la cantidad de cálculos que requiera cada suma de paquetes, pero no en ningún aspecto del funcionamiento de la codificación en sí.

En la Ilustración 14 y la Ilustración 15 se muestran los tiempos de codificación y decodificación medios para diferentes ratios de codificación. Según aumenta el ratio de expansión FEC, y por tanto $h = n - k$, aumenta el número de ecuaciones que serán representadas en la matriz. Como es de esperar, esto supone, también, un mayor número de paquetes a calcular, produciéndose un aumento de los tiempos de codificación y mayor numero de ecuaciones con las que trabajar en el proceso de decodificación.

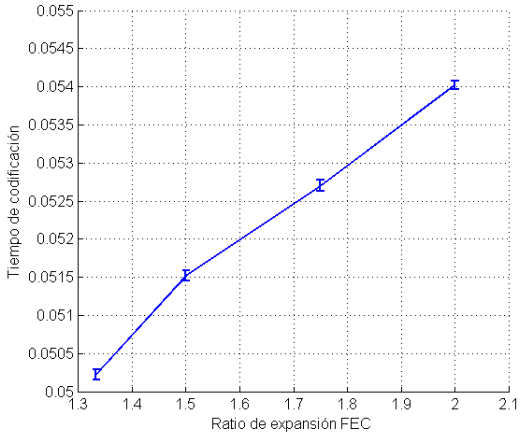

Ilustración 13 – Tiempo de codificación en función del ratio de expansión para códigos LDGM de *left degree* = 3, los intervalos corresponden a una confianza del 99%

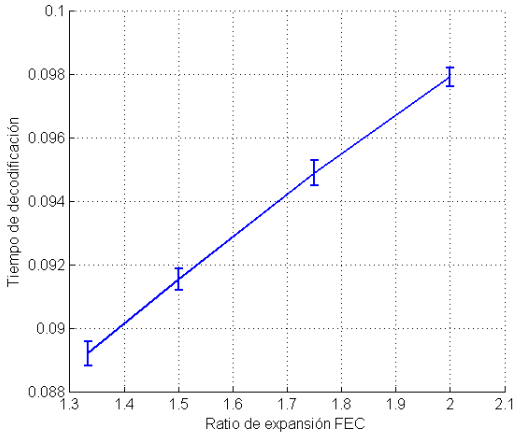

Ilustración 14 – Tiempo de codificación en función del ratio de expansión para códigos LDGM de *left degree* = 3, los intervalos corresponden a una confianza del 99%

3.2.2. Codificaciones LDGM Staircase

En la Ilustración 15 podemos ver el comportamiento de codificaciones LDGM Staircase, de diferentes ratios de expansión FEC, en función de la probabilidad de pérdida de paquete. A diferencia de lo que pasaba en el caso de las codificaciones LDGM normales, en este caso se observa un comportamiento normal, de manera que si añadimos más redundancia, tendremos más probabilidad de que el bloque se reciba correctamente.

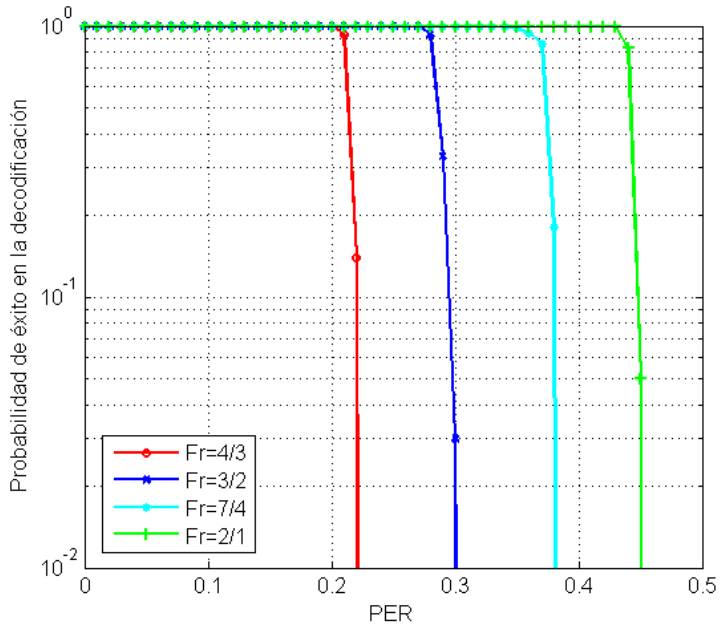

Ilustración 15 – Probabilidad de éxito en la decodificación en función de la probabilidad de pérdida de paquete para diferentes códigos LDGM Staircase

La ineficiencia de codificación en función del ratio de expansión se comporta tal como se muestra en la Ilustración 16.

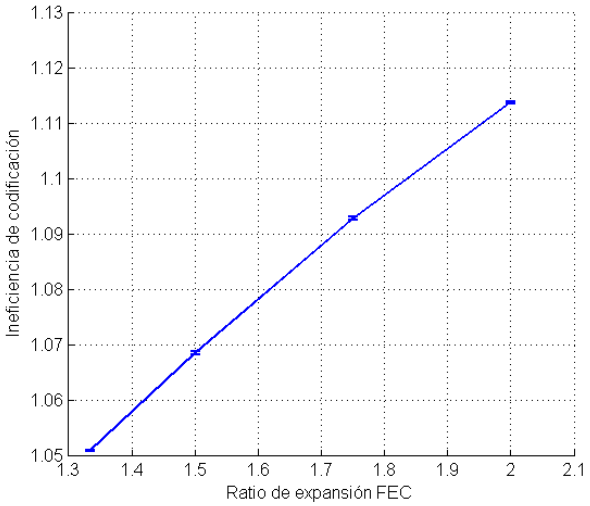

Ilustración 16 – Ineficiencia de codificación en función del ratio de expansión FEC para códigos LDGM Staircase, los intervalos corresponden a una confianza del 99%

El comportamiento de la ineficiencia respecto al número de paquetes de entrada se muestra en Ilustración 17. La ineficiencia, en este caso es alta para *k* pequeña, pero

según aumentamos *k* se reduce rápidamente, tendiendo asintóticamente a un valor final bastante bajo.

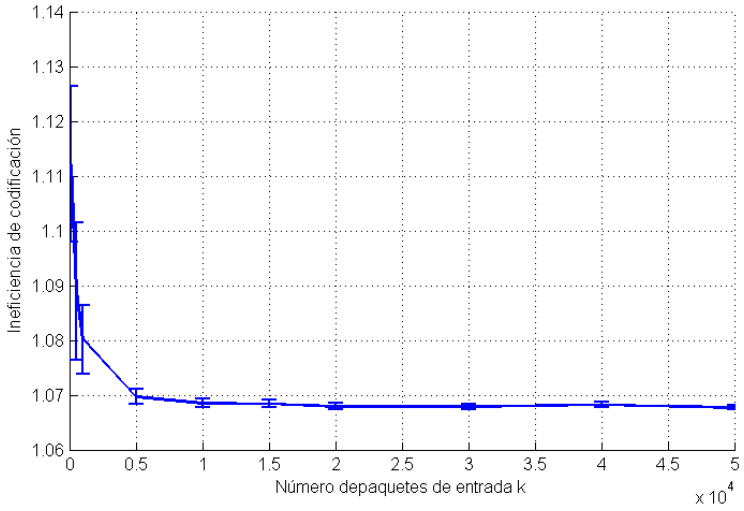

Ilustración 17 – Ineficiencia de codificación en función del número de paquetes de entrada para códigos LDGM Staircase y ratio de expansión FEC de 1.5, los intervalos corresponden a una confianza del 99%

Por su parte los tiempos de codificación y decodificación se comportan según lo esperado, tal como se muestra en la Ilustración 18 y la Ilustración 19.

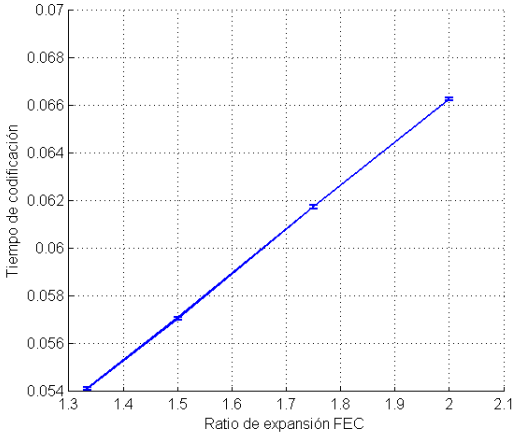

Ilustración 18 – Tiempo de codificación en función del ratio de expansión para códigos LDGM Staircase, los intervalos corresponden a una confianza del 99%

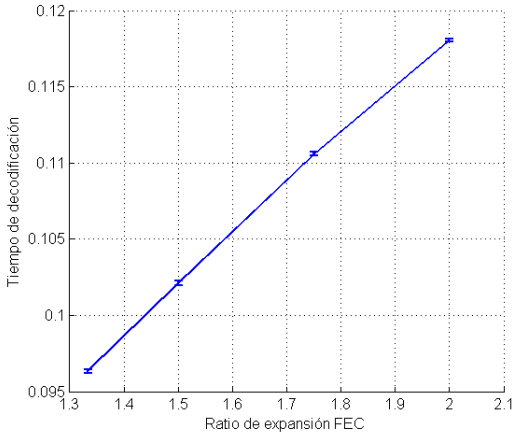

Ilustración 19 – Tiempo de decodificación en función del ratio de expansión para códigos LDGM Staircase, los intervalos corresponden a una confianza del 99%

3.2.3. Codificaciones LDGM Triangle

En la Ilustración 20 podemos ver el comportamiento de codificaciones LDGM Triangle, de diferentes ratios de expansión FEC, en función de la probabilidad de

pérdida de paquete. En este caso se observa un comportamiento normal, de manera que si añadimos más redundancia, tendremos más probabilidad de que el bloque se reciba correctamente.

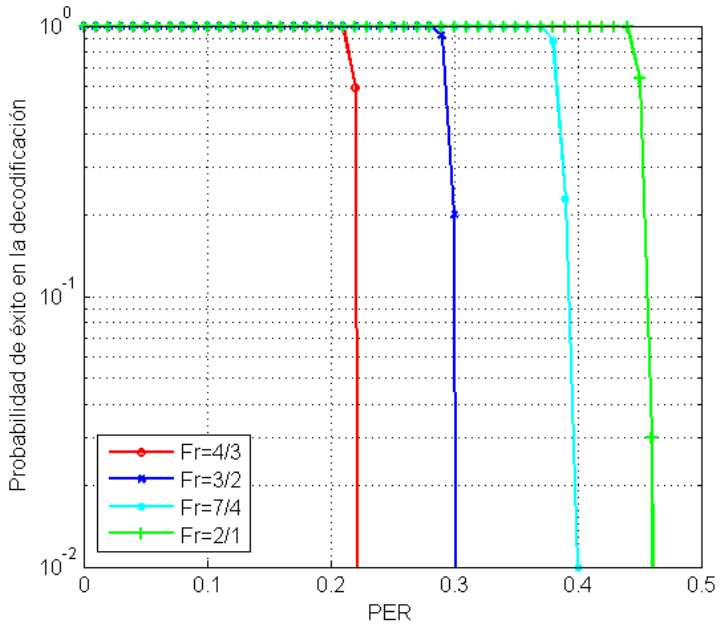

Ilustración 20 – Probabilidad de éxito en la decodificación en función de la probabilidad de pérdida de paquete para diferentes códigos LDGM Triangle

La ineficiencia de codificación en función del ratio de expansión se comporta tal como se muestra en la Ilustración 21.

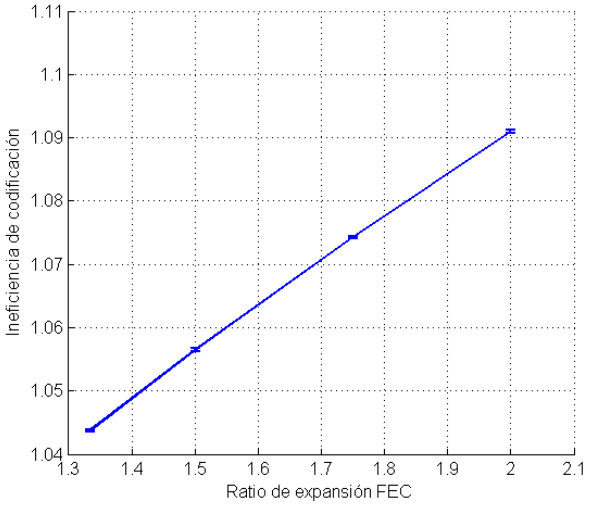

Ilustración 21 – Ineficiencia de codificación en función del ratio de expansión FEC para códigos LDGM Triangle, los intervalos corresponden a una confianza del 99%

El comportamiento de la ineficiencia respecto al número de paquetes de entrada es muy parecido al de LDGM Staircase. Éste se muestra en Ilustración 22.

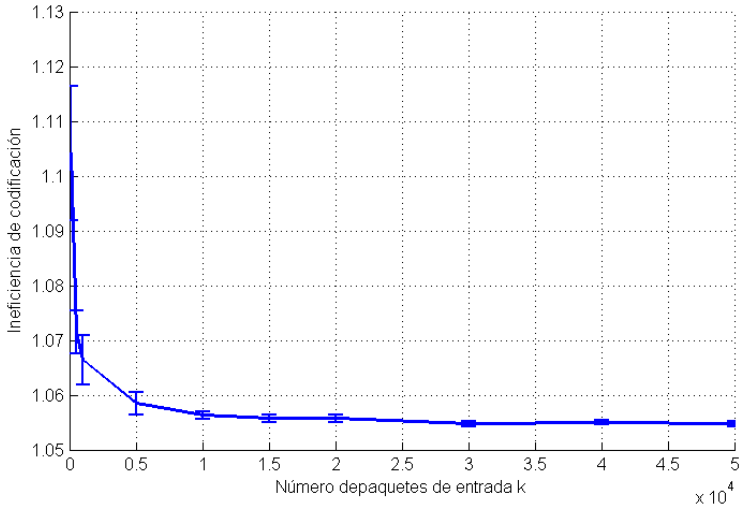

Ilustración 22 – Ineficiencia de codificación en función del número de paquetes de entrada para códigos LDGM Triangle y ratio de expansión FEC de 1.5, los intervalos corresponden a una confianza del 99%

Finalmente, los tiempos de codificación se muestran en la Ilustración 23 y la Ilustración 24.

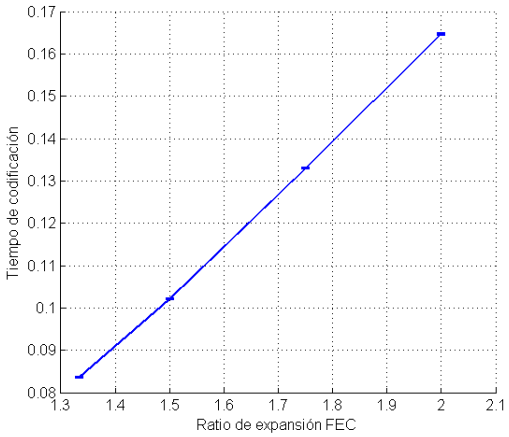

Ilustración 23 – Tiempo de codificación en función del ratio de expansión para códigos LDGM Triangle, los intervalos corresponden a una confianza del 99%

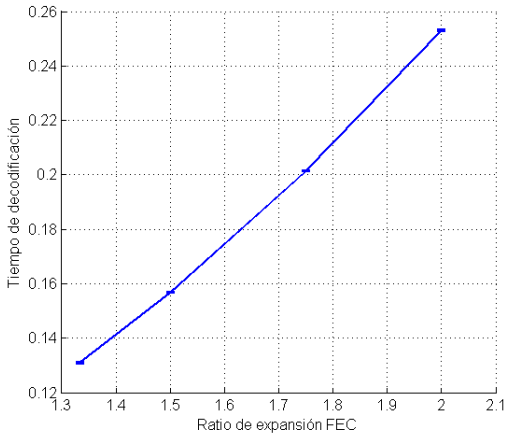

Ilustración 24 – Tiempo de decodificación en función del ratio de expansión para códigos LDGM Triangle, los intervalos corresponden a una confianza del 99%

3.2.4. Conclusiones

Mientras que las simulaciones muestran resultados muy interesantes para las codificaciones LDGM Staircase y LDGM Triangle, en el caso de LDGM estándar, éstos no lo son tanto. Tal como se muestra en la Ilustración 25, la ineficiencia de LDGM es mucho más grande que la de las otras dos variantes, y además, respecto al ratio de expansión aumenta mucho más rápido.

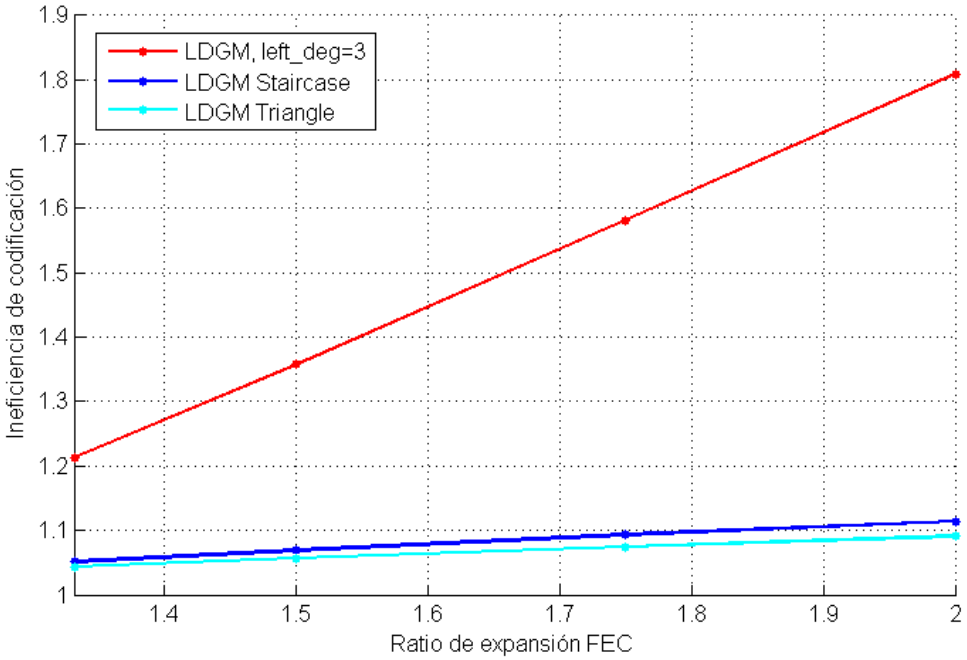

Ilustración 25 – Evolución de la ineficiencia de codificación en función del ratio de expansión FEC para LDGM con *left degree* = 3, LDGM Staircase y LDGM Triangle

Además, como se puede ver en la Ilustración 26, el comportamiento de LDGM respecto el número de paquetes de entrada tampoco es nada bueno, ya que al contrario que LDGM Staircase y LDGM Triangle la ineficiencia, ya siendo muy superior, tiende a aumentar, mientras que la de éstos tiende a disminuir.

Se podrían obtener codificaciones LDGM más eficientes aumentando el *left degree* pero, como se ha explicado anteriormente, esto aumenta de forma considerable los tiempos de codificación y decodificación (al hacerse la matriz menos dispersa). Además, aún así, no llega a igualar en prestaciones a LDGM Staircase y Triangle, y sigue manteniendo una pendiente respecto al ratio de expansión FEC igual de elevada, tal como se muestra en [2].

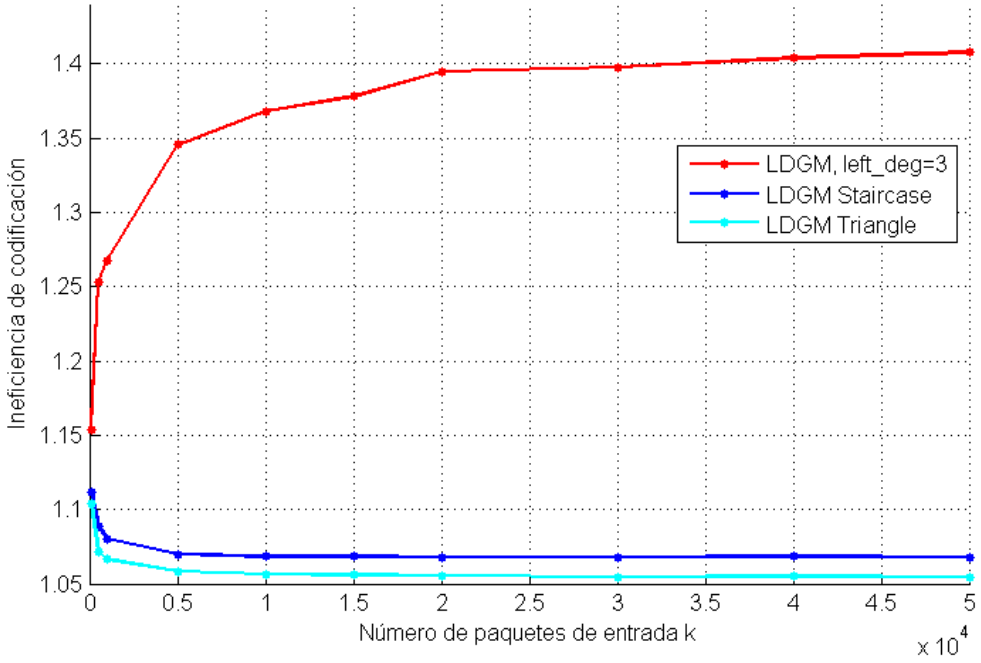

Ilustración 26 – Evolución de la ineficiencia de codificación en función del número de paquetes de entrada para LDGM con *left degree* = 3, LDGM Staircase y LDGM Triangle

La conclusión con la que debemos quedarnos por tanto es que LDGM Staircase y LDGM Triangle muestran resultados competitivos y mejoran al LDGM estándar. En cuánto a LDGM Staircase y LDGM Triangle, el segundo demuestra ser ligeramente más eficiente para los ratios de expansión estudiados. Sin embargo si estudiamos con detenimiento las ilustraciones 18, 19, 23 y 24, vemos que LDGM Staircase es sensiblemente más rápido tanto en codificación como en decodificación. Podemos concluir que si estamos realmente interesados en una ineficiencia tan baja como sea posible deberíamos optar por LDGM Triangle, pero si tenemos interés en un compromiso entre buenos resultados a nivel de ineficiencia, y tiempos de codificación y decodificación bajos, deberíamos optar por LDGM Staircase.

3.3 Codificaciones Raptor

Los resultados que se mostrarán a continuación han sido extraídos de diversas fuentes, las cuales se citan junto a los datos. Se debe tener en cuenta que existen diversas implementaciones diferentes para los códigos Raptor, algunas muy optimizadas y otras no tanto. Al contrario que las codificaciones LDPC/LDGM, las cuales están determinadas por la matriz generadora y sus prestaciones están bien definidas, las

prestaciones de las codificadores Raptor son muy variables dependiendo de lo optimizado que sea el diseño de la precodificación y la distribución de pesos del codificador LT. Además, el algoritmo de decodificación puede tener un gran impacto en las prestaciones.

3.3.1. Resultados

Como ya se comentó en la explicación teórica, en el caso de los códigos fuente como el Raptor es imposible garantizar una probabilidad de fallo de la decodificación igual a 0, debido a la pseudo-aleatoriedad en la generación de los paquetes. Sin embargo, la ineficiencia de un código Raptor se suele fijar para un valor determinado para el cual la probabilidad de fallo de la decodificación sea baja, digamos 10^{-3} o 10^{-4}. Esto no significa que un cierto porcentaje de las decodificaciones vayan a fallar y perdamos necesariamente los datos del bloque que estuvieramos decodificando, sino que simplemente podremos esperar por un paquete más (o uno cuántos más) y volver a intentar la decodificación.

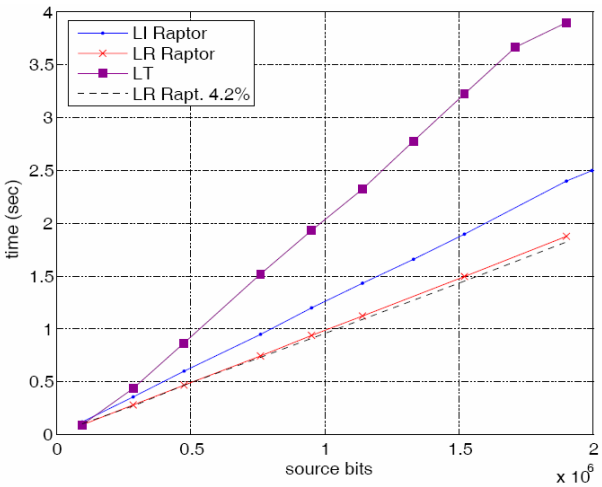

Ilustración 27 – Tiempos de codificación en función del número de paquetes de entrada para diferentes codificaciones Raptor y una codificación LT; LT, LI/LR Raptor ε = 0.12, LR Raptor ε = 0.042

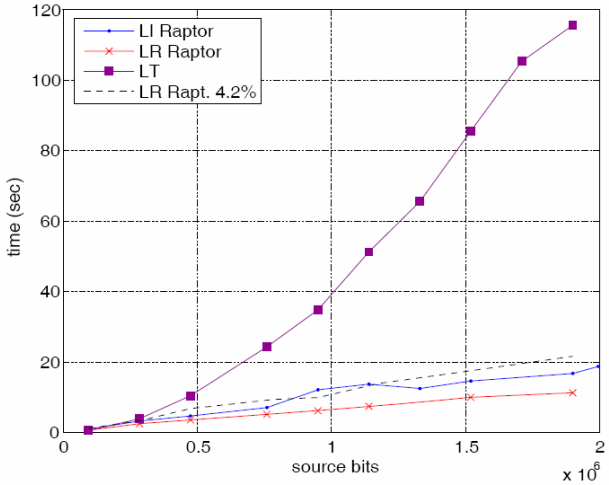

Ilustración 28 – Tiempos de decodificación en función del número de paquetes de entrada para diferentes codificaciones Raptor y una codificación LT; LT, LI/LR Raptor ε = 0.12, LR Raptor ε = 0.042

En la Ilustración 27 y la Ilustración 28, ambas extraídas de [9], se muestran los tiempos de codificación y decodificación para distintas codificaciones Raptor en función del número *k* de paquetes de entrada, con un *overhead* ε constante. En ellas se puede ver claramente como los tiempos de codificación y decodificación, y por lo tanto la complejidad de estos procesos, son lineales respecto de *k*. Aunque se escapa del

propósito de este proyecto, es interesante comentar que la codificación LT, los resultados de la cual que se muestra en las ilustraciones anteriores, fue diseñada con prestaciones similares a los Raptor representados. Como se puede ver, a pesar de los pasos extras que incluyen los codificadores Raptor, éstos son bastante más rápidos, a parte que destaca el hecho que el tiempo de decodificación no es lineal respecto de *k*.

Por los datos mostrados en [8] sabemos que los resultados de las codificaciones Raptor para *k* bastante grande son muy buenos. En la tabla de la Ilustración 29, que ha sido extraída de este mismo documento, se muestran los *overheads* para diferentes *k* grandes, junto con las distribuciones de peso usadas para el codificador LT y el peso medio de los paquetes. En este documento también se menciona el hecho de que cuando *k* es pequeña aparecen ciertos problemas para diseñar codificaciones eficientes que no sean demasiado complejas.

k	65536	80000	100000	120000
Ω_1	0.007969	0.007544	0.006495	0.004807
Ω_2	0.493570	0.493610	0.495044	0.496472
Ω_3	0.166220	0.166458	0.168010	0.166912
Ω_4	0.072646	0.071243	0.067900	0.073374
Ω_5	0.082558	0.084913	0.089209	0.082206
Ω_8	0.056058	0.049633	0.041731	0.057471
Ω_9	0.037229	0.043365	0.050162	0.035951
Ω_{18}				0.001167
Ω_{19}	0.055590	0.045231	0.038837	0.054305
Ω_{20}		0.010157	0.015537	
Ω_{65}	0.025023			0.018235
Ω_{66}	0.003135	0.010479	0.016298	0.009100
Ω_{67}		0.017365	0.010777	
ε	0.038	0.035	0.028	0.02
a	5.87	5.91	5.85	5.83

Ilustración 29 – Distribuciones de peso para diferentes codificaciones Raptor, ε es el *overhead* y *a* el peso medio de un símbolo de salida

Sin embargo la codificación Raptor diseñada para el estándar MBMS[16] del 3GPP[17], se ha diseñado especialmente para ser eficiente para números de paquetes de entrada relativamente pequeños y requerir poca capacidad de proceso. En [17] se afirma que la probabilidad de fallo en la decodificación para estos códigos, habiendo recibido m paquetes, se puede modelar (para $k > 200$) como:

$$P_f = \begin{cases} 1 & \text{si } m < k, \\ 0.85 \cdot 0.567^{m-k} & \text{si } m \geq k \end{cases}$$

La ecuación modela muy bien los resultados reales, tal como muestra la Ilustración 30, extraída de [17].

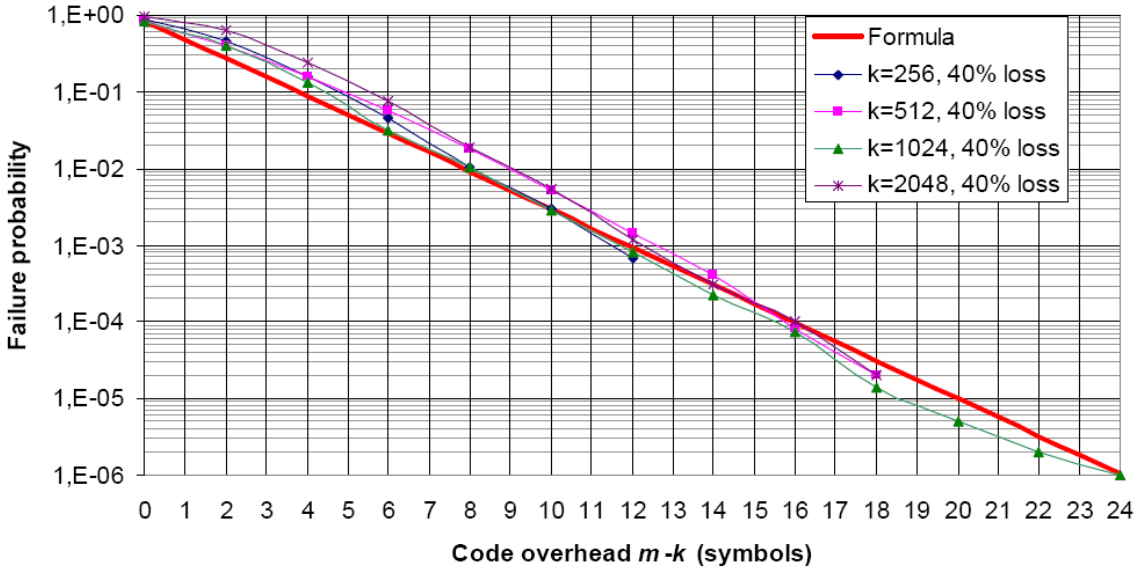

Ilustración 30 – Probabilidad de fallo en la decodificación en función de la cantidad extra de paquetes recibidos por encima de *k*, se muestra la aproximación que nos da la fórmula y datos reales para diferentes *k*

A partir de esta ecuación podemos de deducir lo siguiente:

$$m - k = z$$

$$inef\_cod = m/k = (z+k)/k \Rightarrow z = k\,(inef\_cod - 1)$$

$$P_f = 0.85 \cdot 0.567^z \Rightarrow z = \log_{0.567}\left(P_f / 0.85\right)$$

$$\boxed{inef\_cod = \frac{1}{k} \cdot \log_{0.567}\left(P_f / 0.85\right) + 1, \text{ si } k > 200 \text{ y } m \geq k}$$

[16] Multimedia Broadcast Multicast Service
[17] 3rd Generation Partnertship Project

Ahora si fijamos una *k* y una probabilidad de fallo en la decodificación podremos tener la ineficiencia de codificación aproximada. En la Ilustración 31 se muestran la curva de ineficiencia aproximada en función de *k* para una $P_f = 10^{-3}$.

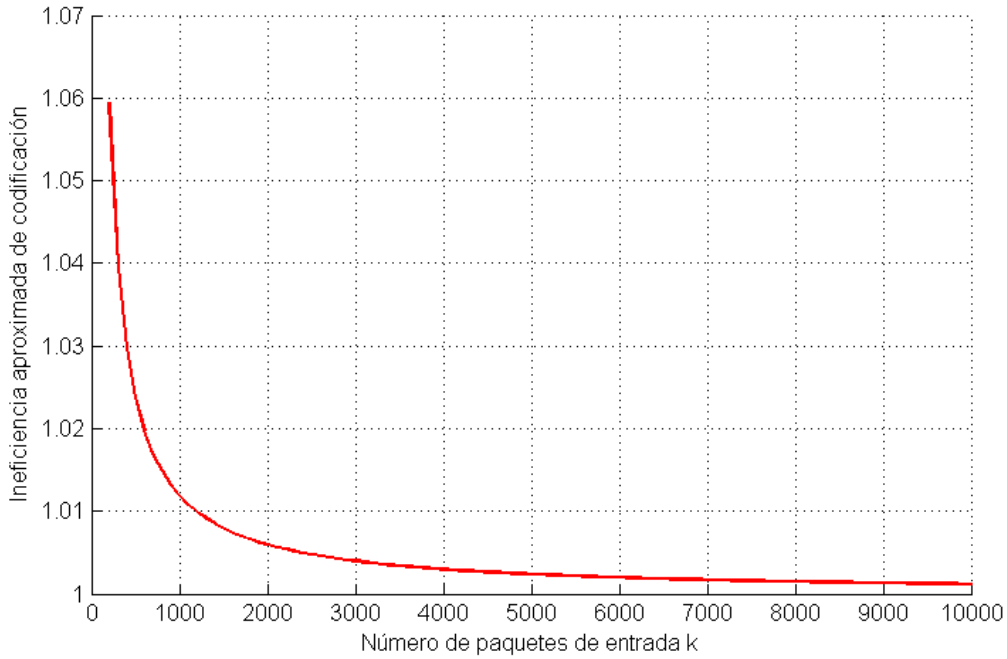

Ilustración 31 – Ineficiencia de codificación aproximada mediante la fórmula para diferentes números de paquetes de entrada y una probabilidad de fallo de decodificación de 10^{-3}

Como se puede observar, la ineficiencia disminuye rápidamente tendiendo asintóticamente hacia $inef\_cod = 1$.

3.3.2. Conclusiones

En los resultados de la literatura se muestra que las codificaciones Raptor dan unas prestaciones excelentes, siempre y cuando se diseñe adecuadamente la precodificación y los parámetros del codificador LT. Tanto para *k* relativamente pequeña, como para *k* grande, existen codificadores adecuados que pueden trabajar con baja probabilidad de fallo e ineficiencias muy bajas. Además en el caso de fallo de decodificación, éste no supone un grave problema puesto que con uno o más paquetes extra se puede volver a intentar la decodificación, sin que esto suponga un gran incremento en la ineficiencia.

3.4 Comparación de las codificaciones LDGM/LDPC y Raptor

El parámetro básico que nos indica la calidad de una codificación de bloque grande es la ineficiencia de codificación. En este aspecto la Ilustración 32 habla por si sola.

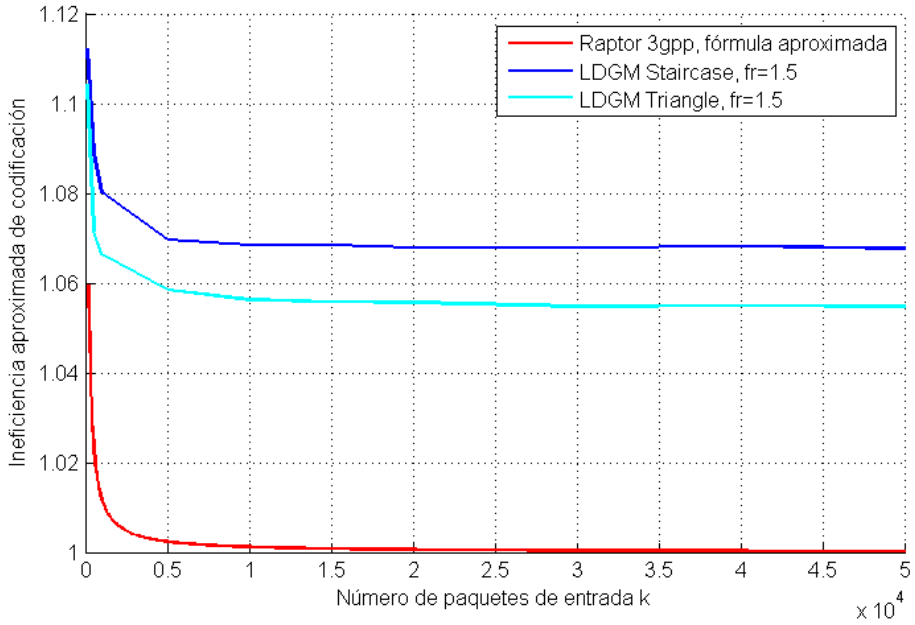

Ilustración 32 – Ineficiencia de codificación en función del número de paquetes de entrada para la codificación Raptor diseñada para el 3GPP y las codificaciones LDGM Staircase y Triangle, para un ratio de expansión FEC 1,5

Las codificaciones Raptor están muy por encima de las LDPC/LDGM en cuanto a eficiencia, aunque se debe tener en cuenta que los resultados de éstas tampoco son malos para este ratio de expansión. No hay que olvidar que la ineficiencia de las LDPC/LDGM crece con el ratio de expansión FEC, mientras que la ineficiencia Raptor es totalmente constante (Ilustración 33).

De hecho en el caso de Raptor no tiene sentido hablar de ratio de expansión, ya que simplemente podemos decidir el número de paquetes redundantes sobre la marcha. Ésta es también una ventaja de las codificaciones Raptor sobre las LDPC/LDGM. En el caso de LDPC/LDGM el número de paquetes resultantes de la codificación *n* está prefijado y no existe ninguna manera para incrementar ese número posteriormente, excepto crear una nueva matriz generadora y empezar todo de nuevo. Pero la eficiencia, de esta manera, es pésima, ya que los paquetes redundantes ya enviados no servirán en

la decodificación del nuevo bloque. Por el contrario, los códigos Raptor pueden enviar simplemente más paquetes si los enviados no son suficientes.

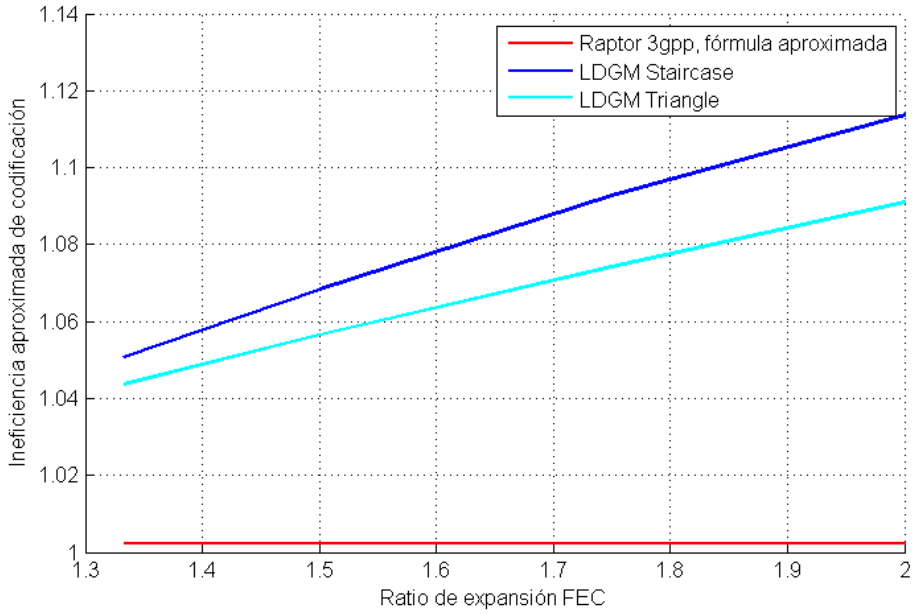

Ilustración 33 - Ineficiencia de codificación en función de expansión FEC, para *k* = 10000, de la codificación Raptor diseñada para el 3GPP y las codificaciones LDGM Staircase y Triangle

Por otra parte, en el caso de LDGM Triangle (este problema no aparece para Staircase) nos vemos obligados a guardar todos los paquetes redundantes que se van generando. Esto se debe a que durante la codificación pueden ser necesarios paquetes anteriormente calculados para calcular un nuevo paquete. Esto supone un consumo extra de memoria en el emisor, que para ratios de expansión grandes, puede ser importante. Para los Raptor en cambio sólo hace falta almacenar los paquetes fuente.

A pesar de estas limitaciones, LDGM Staircase y Triangle son codificaciones buenas con muchas aplicaciones potenciales. Además superan a las Raptor en cuanto a decodificación, puesto que el procesado de ésta se realiza poco a poco según se reciben los paquetes, y el proceso de decodificación Raptor no se puede comenzar hasta haber recibido los paquetes necesarios.

Finalmente, la conclusión a la que se llega es que las codificaciones Raptor superan en prácticamente todo a las codificaciones LDPC/LDGM. Sin embargo no

debemos olvidar el hecho de que las codificaciones LDPC/LDGM son totalmente libres, mientras que las Raptor, al estar patentada, sólo las puede usar Digital Fountain (o aquellos que puedan pagar a esta empresa).

CAPÍTULO 4: Conclusiones

En este proyecto se ha realizado un estudio de diferentes codificaciones que podrían ser aplicadas a nivel de paquete, para comunicaciones móviles por satélite. También se presentado brevemente algunas alternativas a la codificación, en concreto, la diversidad espacial y temporal.

El estudio se ha centrado en aquellas codificaciones que permiten bloques de codificación grandes, de manera que los tiempos de protección sean suficientemente elevados para los fenómenos que suelen aparecer en el canal móvil. Concretamente se han estudiado detalladamente las codificaciones libres LDPC/LDGM y las codificaciones patentadas Raptor.

Después de un análisis detallado de las prestaciones de estos tipos de codificación se llega a la conclusión clara de que Raptor supera ampliamente a LDPC/LDGM en la mayoría de aspectos y sería la opción más adecuada y eficiente. Sin embargo los resultados de LDPC/LDGM no son tan malos y se trata de buenos códigos para bloque grande. Además no se debe olvidar que son totalmente libres, mientras que los Raptor están patentados por Digital Fountain.

Temas de estudio futuro podrían ser los códigos Online [18], que son fuentes digitales parecidas a los códigos Raptor y están libres de patentes, y unos nuevos códigos del tipo MDS [19] que pueden ser aplicados a bloques de codificación más grandes que los MDS tradicionales.

BIBLIOGRAFÍA

[1] Stefano Cioni, Cristina Párraga Niebla, Gonzalo Seco Granados, Sandro Scalise, Alessandro Vanelli-Coralli, and María Angeles Vázquez Castro, "Advanced Fade Countermeasures for DVB-S2 Systems in Railway Scenarios," EURASIP Journal on Wireless Communications and Networking, vol. 2007, Article ID 49718, 17 pages, 2007. doi:10.1155/2007/49718

[2] Vincent Roca , Christoph Neumann, "Design, Evaluation and Comparison of Four Large Block FEC Codecs, LDPC, LDGM, LDGM Staircase and LDGM Triangle, plus a Reed-Solomon Small Block FEC Codec", INRIA Rhône-Alpes, Planète Research Team, Rapport de recherche N° 5225, June 9, 2004, 24 pages

[3] Vincent Roca, Zainab Khallouf, Julien Laboure, "Design and Evaluation of a Low Density Generator Matrix (LDGM) Large Block FEC Codec", INRIA Rhône-Alpes, Planète project, France

[4] R. G. Gallager, "Low density parity check codes", PhD thesis, Massachussets Institute of Technology, 1960

[5] R. G. Gallager, "Low density parity check codes", IEEE Transactions on Information Theory, 8(1), Jan. 1962

[6] D. MacKay, R. Neal, "Good codes based on very sparse matrices", 5th IAM Conference: Cryptography and Coding, LNCS No. 1025, 1995.

[7] David J.C. MacKay, "Information Theory, Inference, and Learning Algorithms", Cambridge University Press, 2003

[8] Amin Shokrollahi, "Raptor Codes", Laboratoire d'algorithmique, Laboratoire des mathematiques algorithmiques, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland, September 19, 2005

[9] Pasquale Cataldi, Miquel Pedròs Shatarski, Marco Grangetto, Enrico Magli, "Implementation and performance evaluation of LT and Raptor codes for multimedia applications", Proc. of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'06), 0-7695-2745-0/06

[10] Michael Luby, "LT-codes," Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science (FOCS), Vancouver, BC, Canada, Nov. 2002, 10 pages, pp. 271–280

[11] Vicent Roca, "Reed-Solomon Forward Error Correction (FEC)", Internet-Draft, work in progress

[12] Rosario Firrincieli, Gianni Albertazzi, Giovanni E. Corazza and Alessandro Vanelli-Coralli, "Packet layer FEC for SDMB services", DEIS/ARCES, University of Bologna

[13] M. Chipeta, M. Karaliopoulos, L. Fan, B.G. Evans, "On 2-D Reed-Solomon packet-level FEC and its use in SDMB", Mobile Communications Research Group, Centre for Communications Systems Research, University of Surrey

[14] Michael Luby, "Raptor encoder specification", 3GPP SA4-PSM SWG#6, Tdoc S4-AHP139, October 11-13 2004, Newbury UK, Agenda Item: 4.4.1

[15] Michael Luby, "Raptor decoder specification", 3GPP SA4-PSM SWG#6, Tdoc S4-AHP140, October 11-13 2004, Newbury UK, Agenda Item: 4.4.1

[16] Michael Luby, "Raptor systematic code specification", 3GPP SA4-PSM SWG#6, Tdoc S4-AHP141, October 11-13 2004, Newbury UK, Agenda Item: 4.4.1

[17] Michael Luby, Mark Watson, Tiago Gasiba, Thomas Stockhammer, "Mobile Data Broadcasting over MBMS, Tradeoffs in Forward Error Correction", MUM'06, December 4-6, 2006, Stanford, CA, USA, Copyright 2006 ACM 1-59593-607- 60612

[18] P.Maymounkov, "Online Codes", Research Report TR2002-833, New York University, Nov. 2002

[19] Jérôme Lacan, Jérôme Fimes, "A Construction of Matrices With No Singular Square Submatrices", ENSICA / TéSA, Département de Mathématiques Appliquées et d'Informatique

El autor,

Firmado: Rafael Montalbán Gutiérrez

RESUMEN

Una de las opciones contempladas para transmitir contenidos multimedia y proporcionar acceso a Internet a grupos de usuarios móviles es el uso de satélites. Las condiciones de propagación del canal móvil implican que de alguna manera deberemos garantizar la calidad de servicio. Esto cobra incluso más importancia si tenemos en cuenta que, en el caso de acceso a Internet, no se tiene la capacidad de asumir cierto porcentaje de pérdida de datos que tenemos, por ejemplo, en la transmisión de sonido o vídeo (rebajando la calidad).

Entre las principales alternativas estudiadas para esta clase de entornos está la inclusión de codificaciones a nivel paquete. El funcionamiento de esta técnica se basa en incluir en la transmisión paquetes redundantes, obtenidos mediante un determinado algoritmo. El receptor será capaz de recuperar la información original que se quería enviar, siempre y cuando haya recibido correctamente una cierta cantidad de paquetes, similar a la cantidad de paquetes originales. A este mecanismo se le conoce como Forward Error Correction (FEC) a nivel de paquete.

En esta memoria se valora brevemente las alternativas existentes y se explican algunas de las codificaciones para FEC más importantes. A continuación se realiza un estudio comparativo de algunas de ellas: las variantes de LDPC (Low Density Parity Check) conocidas como LDGM (Low Density Generator Matrix), y la codificación Raptor.

RESUM

Una de les opcions que es contemplen per transmetre continguts multimèdia i proporcionar accés a Internet a grups de usuaris mòbils és fer servir satèl·lits. Les condiciones de propagació del canal mòbil impliquen que d'una manera o altra haurem de garantir la qualitat de servei. Això té fins i tot més importància si tenim en compte que, en el cas d'accés a Internet, no es té la capacitat d'assumir cert percentatge de pèrdua de dades que tenim, per exemple, en la transmissió de so o vídeo (rebaixant la qualitat).

Entre les principals alternatives per a aquesta classe d'entorns es troba la inclusió de codificacions a nivell de paquet. El funcionament d'aquesta tècnica es basa en incloure a la transmissió paquets redundants, obtinguts mitjançant un determinat algoritme. El receptor podrà recuperar la informació original que es volia enviar, sempre que hagi rebut una certa quantitat de paquets, similar a la quantitat de paquets originals. A aquest mecanisme se'l coneix com Forward Error Correction (FEC) a nivell de paquet.

En aquesta memòria es valoren breument les alternatives existents i s'expliquen algunes de les codificacions per a FEC més importants. A continuació es realitza un estudi compartiu d'algunes d'elles: les variants de LDPC (Low Density Parity Check) conegudes com LDGM (Low Density Generator Matrix), i la codificació Raptor.

ABSTRACT

The use of satellites is one of the options considered to broadcast multimedia contents and to provide access to the Internet for groups of mobile users. The propagation conditions of mobile channel make necessary to guarantee the quality of service. This is even more important if we note that no data loss is allowed in the case of providing Internet access, while in sound or video transmissions it is (reducing the quality).

One of the main alternatives which is being studied for this environments is the inclusion of packet level codifications. The basic idea of this technique is to include redundant packets in the transmission. These redundant packets can be obtained using a codification algorithm. The receiver will be able to recover all the original information provided it has received a certain amount of packets which will be similar to the amount of original packets. This technique is known as packet level Forward Error Correction (FEC).

In this dissertation, the alternatives to FEC are briefly explained. Next, some of the most important packet level FEC codifications are explained. After that, a comparative study of some of them will be done, specifically the LDPC (Low Density Parity Check) variants known as LDGM (Low Density Generator Matrix) and the Raptor codification.