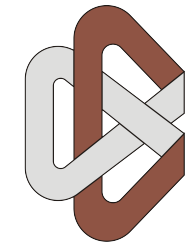


CIMAT

CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS AC

Transmisión Robusta de Multimedia usando FEC.

T E S I S

EN OPCIÓN AL TÍTULO DE:

Maestría en Ciencias con Especialización

en Computación y Matemáticas Industriales.

PRESENTA:

GILBERTO MÉNDEZ GARCÍA

DIRECTOR DE TESIS:

DR. ROGELIO HASIMOTO BELTRÁN

Guanajuato, Guanajuato, México

Diciembre de 2012

Dedicatoria

A Dios, por haberme permitido llegar hasta este punto y haberme dado salud para lograr mis objetivos, además de su infinita bondad y amor. *A mis padres,* la señora *Inelda García Trinidad* y el señor *Gilberto Méndez Torres*, por el apoyo incondicional que siempre me ofrecieron y por la motivación que me inculcaron en el estudio. Como un testimonio de cariño y eterno agradecimiento por mi existencia, valores morales y formación profesional. Por que sin escatimar esfuerzo alguno, han sacrificado una parte de su vida para formarme y por que nunca podre pagar todos sus desvelos ni aun con las riquezas. Por lo que soy y por todo el tiempo que les robé pensando en mí... Gracias con amor y respeto.

A mi hermana la *L.A.E. Maria Isabel Méndez García* quien siempre me a apoyado incondicionalmente.

A mi novia la *L.D. Landy Karolina Aguilar Uicab* quien siempre estuvo apoyándome en este transcurso de mi vida.

A mis maestros, por sus enseñanzas, tanto en el ámbito académico como en el personal; me llevo lo mejor de cada uno de ustedes, y no los defraudaré.

A todos ustedes les dedico este documento,
y les agradezco por todo su apoyo...

Agradecimientos

Quiero agradecer a mi asesor de tesis, el *Dr. Rogelio Hasimoto Beltrán* por su ayuda y orientación para la realización de este trabajo, por su paciencia y por haber puesto su confianza en mi.

También quiero agradecer al CONACYT, por haber brindado los recursos necesarios durante los dos años de la maestría.

Resumen

La transmisión de vídeo en tiempo real es hoy día uno de los eventos mas comunes en el Internet; sitios como YOUTUBE, NETFLIX, Telemedicina, Teleconferencia, IPTV (Televisión Digital por Internet), E-learning, Turismo Virtual etc., son claros ejemplos de la necesidad actual de las comunicaciones de vídeo en tiempo real. Como usuarios, deseamos que la transmisión/recepción de vídeo se realice con la mejor calidad posible y sin pérdidas de información durante la transmisión. Debido a esto, hay una amplia área de investigación en esquemas de protección para la transmisión de la información basadas en FEC (Forward Error Correction), ARQ (Automatic Repeat reQuest) y/o una combinación de estas. El sistema FEC es una técnica para el control y reparación de errores de transmisión de datos donde el emisor adiciona información redundante a los datos originales, de modo que si algunos de los datos originales se pierde, este puede ser reconstruido en el recibidor sin comunicación adicional. La desventaja importante de FEC es que aumenta la cantidad de información transmitida. Por otro lado, ARQ se basa en la retransmisión de la información pérdida durante la comunicación previa solicitud del recibidor. En condiciones de tráfico elevado y pérdida de información, los protocolos ARQ podrían empeorar las condiciones de la red debido a la retransmisión de un gran número de paquetes, incrementando las pérdidas de información.

En el presente trabajo, el interés se centra en generar un esquema de protección que

aproveche las ventajas de ambas técnicas (FEC y ARQ). El objetivo es lograr un sistema de "conjuntos jerárquicos" que ayuden a recuperar información perdida durante la transmisión y/o disminuir el efecto de pérdida de información sobre la calidad resultante del vídeo sin saturar la red de transmisión. Para lograr esto, se desarrolló un método eficaz basado en los códigos LDPC (Low-density parity-check) que subsana algunos de los problemas encontrados en estos métodos, en particular su aplicabilidad a transmisiones en tiempo real, mantiene los paquetes originales en el proceso de codificación, codificación no aleatoria (evita la transmisión de la matriz de codificación) y permite la retransmisión óptima de paquetes (sin duplicidad) para la decodificación de la información recibida. Los resultados muestran la superioridad del esquema propuesto bajo condiciones de congestionamiento alto de la red de transmisión.

Índice general

Índice de figuras

Índice de cuadros

1

Introducción

Resumen

En este capítulo abarcamos los principales problemas que existen en la red, el esquema de envió de información, los métodos que existen para combatirlos y como han estado evolucionando con el paso de los años. También se comenta sobre los canales por donde viaja la información para la transmisión de datos, por ejemplo, el canal de borrado binario (BEC) y los canales de borrado simétricos (BES), así como el canal Gaussiano. Es de aclarar que nuestro principal objetivo, en esta tesis, es trabajar alrededor de los problemas ocasionados por los canales de borrado binarios (BEC).

§ 1.1 Errores de transmisión

La transmisión de imagen y vídeo a través de redes no confiables pueden tener consecuencias catastróficas, si no se toman medidas que prevengan o reparen este problema.

Desde hace ya varios años, nuestra tecnología se ha incrementado notoriamente, el creciente número de usuarios de Internet, los navegadores de Word Wide Web (WWW) y servicios de videoconferencia, han ocasionado que los errores puedan presentarse

fácilmente en la información transmitida por medio de paquetes perdidos debido a la congestión de la red y/o errores de bits que corrompen considerablemente la calidad de la información recibida. Desde luego, cada vez más usuarios usamos el Internet para escuchar música, ver o transmitir vídeos ya sean en tiempo real o almacenados, de igual manera somos cada vez más los usuarios que guardamos información en servidores y deseamos ver en línea la transmisión de algún evento en vivo con la mejor calidad posible, sin pérdida de información y en el menor tiempo posible, sin embargo, demasiada información es producida en periodos muy cortos de tiempo, causando la saturación del Ruteador.

Cuando se pierden paquetes en el proceso de la transmisión, pueden ocurrir severos problemas, La figura 1.1, muestra un sistema simplificado de comunicación digital, donde se codifican los datos de imagen, entonces es transmitida a través de un canal con pérdida (es decir, Internet), y finalmente decodificada en el receptor.

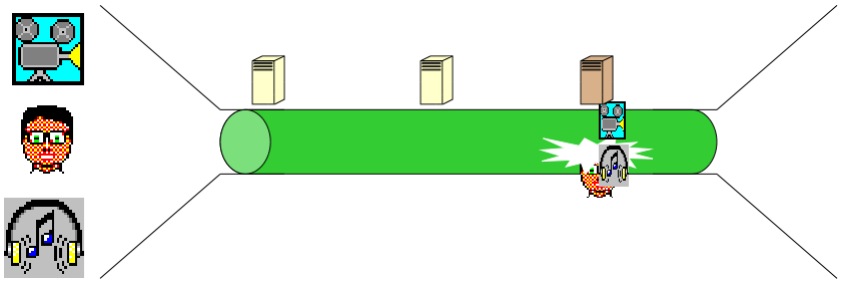


Figura 1.1: Perdida de información transmitida en el Internet.

Los efectos de errores de la red en la imagen recibida se representan en forma de pérdidas de bloques debido a errores de bits y a secuencias de pérdidas de bloques debido a errores de sincronización y/o paquetes perdidos. Para evitar o minimizar el efecto de los errores en la información recibida se han introducido diferentes esquemas de protección como la codificación de canal, post-procesamiento de imagen/vídeo, escalabilidad, protección de la red, etc., y que de alguna manera tienen que ser añadidos

al esquema presentado en la figura 1.1.

Este esquema general para proporcionar protección y recuperación de errores todavía está bajo investigación extensa y es una de las principales preocupaciones para los futuros estándares de compresión de imagen/vídeo tales como JPEG-2000, H.264 y MPEG-4.

Las señales experimentan una pérdida de intensidad debido a la atenuación y la transmisión se deteriora debido al ruido. En realidad, los errores de canal se suelen producir a ráfagas.

Figura 1.2: Perdida de información en una imagen recibida.

Técnicas que han demostrado proporcionar robustez a la transmisión a través de redes con pérdida es la transmisión escalable. En particular, la transmisión de vídeo escalable es el tipo de codificación de vídeo que nos permite decodificar el original a distintas resoluciones y tasas de imágenes por segundo dependiendo del potencial hardware del destino.

Así, un vídeo situado en un servidor determinado, podría ser descargado en su resolución original por el usuario de un PC con una conexión de banda ancha a Internet y por otra parte el usuario de un móvil con capacidad para visualizar vídeos podría descargarlo con una resolución adaptada a la pantalla de su móvil.

La transmisión de vídeo escalable se realiza mediante la división del vídeo original en distintas capas (capa base y capas de mejora). La suma de estas capas nos permitiría

decodificar el vídeo con la calidad original pero si queremos una menor calidad solo tenemos que tomar menos capas. La capa base, como su propio nombre indica, es indispensable mientras que el número de capas de mejora nos determinará la calidad del vídeo final.

Es por ello que necesitamos mejores mecanismos en la transmisión de datos en el Internet, hoy en día existen algoritmos que realizan este trabajo y lo realizan bastante bien, sin embargo, no es suficiente para combatir la perdida de información y el retardo, que esto ocasiona principalmente en comunicaciones de vídeo en tiempo real.

La situación general, a groso modo, es la siguiente. Supongamos que queremos enviar un mensaje. Este es enviado por un canal de comunicación, cuyas características dependen de la naturaleza del mensaje a ser enviado (i.e. sonido, imagen, datos). En general, hay que hacer una traducción entre el mensaje original (o palabra fuente) x y el tipo de mensaje c que el canal está capacitado para enviar (palabras código). Este proceso se llama codificación. Una vez codificado el mensaje lo enviamos a través del canal, y nuestro intermediario (el receptor) recibe un mensaje codificado (palabra recibida) posiblemente erróneo, ya que en todo proceso de comunicación hay ruido e interferencias. El mensaje recibido c' es traducido nuevamente a términos originales x', es decir, es decodificado. Todo el proceso se resume en el siguiente esquema

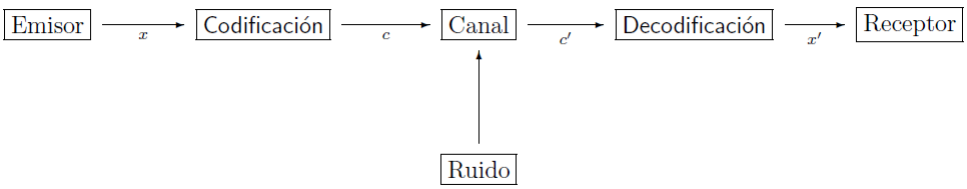


Figura 1.3: Esquema para el envío de información.

Sus componentes son los siguientes:

Emisor: es un proceso que genera símbolos pertenecientes a un alfabeto finito, en for-

ma discreta, o valores reales, en forma continua. Los símbolos o los valores generados son de interés para un receptor.

Codificación: es un mecanismo que opera sobre la salida de la fuente para ponerla en una forma adecuada a la transmisión.

Canal de transmisión: es un modelo del medio usado para la transmisión de los mensajes. En particular, puede modelar un cable eléctrico, una banda de frecuencias de radio, un rayo de luz, una superficie magnética, etc. El ruido se presenta dependiendo de la naturaleza del material usado para el canal, produciendo errores en la información enviada.

Decodificador: es un mecanismo que normalmente realiza la operación inversa del codificador, intentando recuperar en la forma más exacta posible el mensaje originalmente emitido por la fuente.

En general, $x' \neq x$ y es deseable que este error sea detectado (lo cual permite pedir una retransmisión del mensaje) y en lo posible corregido.

§ 1.2 Canales

Como se mencionó anteriormente, toda información que enviamos de un ordenador a otro es dividida en paquetes que viajan de manera independiente hasta su destino. El protocolo que realiza dicha acción es TCP/IP (*Transmisión Control Protocol/Internet Protocol*); TCP se encarga de fragmentar y unir los paquetes, mientras que el IP tiene como misión hacer llegar los paquetes de información a su destino correcto.

Entre las principales alternativas estudiadas para esta clase de entornos esta 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 transmitida, siempre y cuando se reciban 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. A continuación hacemos mención de los principales modelos de canales por donde viaja la información.

1.2.1 Canal Gaussiano

El canal AWGN (Additive White Gaussian Noise) es de gran importancia práctica porque permite modelar varios medios físicos incluyendo enlaces de radio y satélites.

En el caso general, tanto la salida como la entrada toman valores continuos y de tiempo discreto. La distribución condicional de y dado x es la distribución Gaussinana:

$$P(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-(x-y)^2/2\sigma^2} \tag{1.2.1}$$

En [14, pag 185] se puede ver que algunos canales de tiempo-continuos son equivalentes a los canales de tiempo-discreto. Si limitamos la entrada de un canal AWGN a dos valores posibles, por ejemplo $\{\pm 1\}$, entonces se dice que el canal es Gaussiano de entrada binaria (Binary Input Additive White Gaussian Noise, BIAWGN). La capacidad de este último canal es un poco inferior a la del canal AWGN (para más información, consultar [18]).

1.2.2 Canales Binarios Simétricos (BSC)

Un BSC se puede ver como un canal BIAWGN cuantizado, es decir que si el valor a la salida es mayor que 0 entonces la salida se considera 0 y si no se considera 1. Por

lo tanto, en una canal BSC un bit llega al receptor de manera correcta o incorrecta, es decir, con su valor invertido. Esto sucede normalmente en canales con mucho ruido.

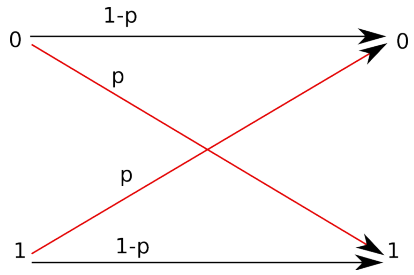


Figura 1.4: Canal Binario Simétrico con probabilidad p de que exista un cambio de bit.

1.2.3 Canal de Borrado Binario (BEC)

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 el Internet.

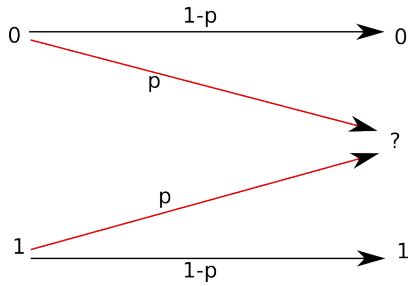


Figura 1.5: Canal de Borrado Binario con probabilidad p de que exista un borrado de bit.

En el caso de Internet la razón principal de las pérdidas de paquetes es la congestión en los routers, y el CRC[1] disponible en la mayoría de capas físicas garantiza que

[1] El algoritmo de detección de errores más utilizado en la práctica se basa en lo que se conoce como códigos polinómicos (también llamados códigos de redundancia cíclica o CRC, Cyclic Redundancy

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 de Paquetes (Packet Erasure Channel, PEC) 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.

Check). La idea básica es la misma que en el caso de los bits de paridad: añadir a los datos a transmitir unos bits adicionales cuyo valor se calcula a partir de los datos; la trama así construida se envía, y el receptor separa los bits de datos de la parte CRC; a partir de los datos recalcula el CRC y compara con el valor recibido; si ambos no coinciden se supone que ha habido un error y se pide retransmisión.

2

Sistemas para el control de errores

Resumen

En este capítulo abarcamos las principales opciones que existen para reparar errores que ocurren en los canales de comunicación. Actualmente existen tres sistemas para controlar o reparar errores; el primero es ARQ (Automatic Repeat-reQuest), el segundo es FEC (Forward Error Correction) y el tercero son los sistemas basados en la mezcla de estos dos últimos, los cuales son conocidos como sistemas híbridos ARQ.

La comunicación confiable por un canal se consigue gracias a la codificación para el control de errores. Un canal de comunicación, como ya se mencionó anteriormente, es el medio físico a través del cual ocurre la comunicación siendo afectado por algún tipo de ruido a nivel de bit o perdida de paquetes. Debido a el ruido existente en los canales es que se recurre al empleo de la codificación para el control de errores (o codificación de canal, como se le conoce) en un sistema de comunicación. Gracias a la codificación de canal es posible diseñar sistemas de comunicación que brinden una comunicación tanto rápida como confiable.

Hay dos maneras de evitar los errores en los canales de comunicación:

Retransmisión: Cuando no se está operando en tiempo real puede ser útil pedir el reenvío íntegro de las partes que se presumen erróneas o dañadas. Éste es posiblemente el método más seguro de corrección de errores aunque raramente es el método más eficiente. Es el caso por ejemplo el sistema ARQ (Automatic Repeat-reQuest), donde el receptor al detectar un error solicita la repetición automática de todo el mensaje.

Corrección de errores hacia adelante: Conocido también como *Forward Error Correction* (FEC), es el único esquema de corrección de errores que detecta y corrige los errores de transmisión en la recepción, sin pedir la retransmisión del mensaje enviado. Este tipo de sistema codifica la información agregando redundancia y así conseguir, bajo algún algoritmo, la información original a pesar que existan perdidas de determinado número de paquetes.

Los sistemas basados en ARQ son de utilidad en situaciones donde el número de usuarios compartiendo información es pequeño, en aplicaciones no interactivas, cuando predominan las pérdidas en enlaces compartidos de los árboles multicast, o cuando la probabilidad de pérdidas no es homogénea. Los sistemas FEC, en cambio, son interesantes cuando existen grupos grandes, predominan las pérdidas individuales, la probabilidad de pérdidas es homogénea o los buffers de recepción son limitados.

§ 2.1 Forward Error Correction (FEC)

El sistema FEC es una técnica usada en telecomunicaciones para el control y reparación de errores en la transmisión de datos sobre canales de comunicación ruidosos o pocos fiables. La idea central de un sistema FEC está basada en la transmisión de información redundante junto con los datos originales, de modo que si algunos de los datos originales se pierde, puede ser reconstruido usando la información redundante. Una desventaja importante de FEC es que aumenta la cantidad de información a transmitir,

que puede conducir a retrasos y la degradación de rendimiento de la red [27]. Se ha encontrado que el potencial de FEC para recuperarse de los errores depende en gran medida del comportamiento de la pérdida de la red. Si las pérdidas están altamente correlacionados, FEC será mucho menos eficaz que cuando las pérdidas se dispersan de manera uniforme.

El sistema FEC es aplicado en situaciones donde las retransmisiones son costosas o imposibles, tales como cuando se está transmitiendo a múltiples receptores en multicast. FEC es muy usado en dispositivos de almacenamiento para la recuperación de datos dañados.

Debido a la información redundante generado por los sistemas FEC los datos originales no son necesariamente los datos a enviar. Una codificación se considera **codificación sistemática**, cuando las palabras de código se mantienen intactos después de pasar por un codificador FEC, de lo contrario se dice que es una **codificación no sistemática**. 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 usamos 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 podrá reproducir, aunque reduciendo la calidad, en cambio si la codificación era no sistemática los paquetes que tendremos no tendrán ninguna utilidad.

Existen dos categorías principales de códigos FEC : los códigos de bloque y los códigos convolucionales.

Códigos de bloque: trabajan en bloques de tamaño fijo (paquetes) de bits o símbolos de un tamaño predeterminado. Los códigos de bloques pueden ser decodificados en tiempo polinomial que dependen a su longitud de bloque.

Los códigos de bloque pueden ser lineales o no lineales. Un código lineal es un código de corrección tal que cualquier combinación lineal de palabras de código es también una palabra de código, y puede representarse como un producto de matrices. Los códigos lineales se denominan también **códigos de comprobación de paridad**, pues la palabra de código se obtiene a partir de sumas módulo dos (o el espacio donde se esté trabajando) de las palabras de código. Un código de este tipo queda completamente caracterizado por una matriz generadora G (ver capítulo 4).

Ejemplos de códigos de bloque son códigos Reed-Solomon, códigos Hamming, códigos de Hadamard, códigos de expansión, códigos de Golay y códigos Reed-Muller, códigos LDPC. Estos ejemplos también pertenecen a la clase de códigos lineales, y por lo tanto son llamados **códigos de bloque lineales**.

Códigos convolucionales: trabajan en flujos de bits o con símbolos de longitud arbitraria. Con mayor frecuencia se decodifica con el algoritmo de Viterbi, aunque a veces otros algoritmos son utilizados. Los códigos convolucionales se utilizan ampliamente en numerosas aplicaciones con el fin de lograr la transferencia fiable de datos, incluyendo vídeo digital, la radio, la comunicación móvil, y la comunicación por satélite. Estos códigos se aplican a menudo en una vinculación con un código hard-decision, sobre todo Reed Solomon. Antes de turbo códigos, tales construcciones fueron las más eficientes, más próximo a los límites de Shannon.

§ 2.2 Automatic Repeat reQuest (ARQ)

En la mayoría de los sistemas de comunicación digitales, siempre que los eventos de error se producen en los mensajes transmitidos, algunas acciones deben ser tomadas

para corregir estos eventos. En algunas aplicaciones donde existe un enlace de dos vías de comunicación entre el emisor y el receptor, el receptor puede informar al remitente que el mensaje se ha recibido con error y, por tanto, pedir una repetición de ese mensaje. En principio, el procedimiento se puede repetir tantas veces como sea necesario hasta que el mensaje se recibe sin errores. Un sistema de control de errores en los que los mensajes recibidos erróneamente son simplemente retransmitidos se llama repetición automática de la solicitud (Automatic Repeat Request (ARQ)).

En el caso de aplicaciones de tiempo no real la retransmisión de información incorrecta o pérdida se puede confiar a técnicas ARQ que se han demostrado como poco apropiadas para servicios de tiempo real o con requerimientos de baja latencia por el elevado retardo que introducen las retransmisiones. ARQ ofrece dos variantes y ambas requieren que emisor y receptor intercambien algún tipo de información de estado por lo que incurren en retardos para el nodo receptor, implosión[1] en el nodo emisor y excesivo overhead en la red.

- En la primera variante de ARQ el receptor devuelve mensajes de confirmación positiva (ACK) incluso cuando ha recibido correctamente los datos. Este es el mecanismo tradicionalmente usado para aportar fiabilidad en las transmisiones unicast. Para poder implementar este protocolo es necesario que las tramas o paquetes que va a procesar tengan un número de secuencia que sirva para identificarlas en la recepción y en el proceso de retransmisión. Este planteamiento responde claramente al modelo de protocolos de la capa de enlace (también de transporte) orientados a conexión y con acuse de recibo que ofrecen fiabilidad a la capa de red.

[1] La implosión es el efecto negativo que experimentan las fuentes emisoras de tráfico cuando deben atender las solicitudes de retransmisión de paquetes perdidos o erróneos. En el caso de las transferencias multipunto la implosión genera importantes problemas.

- En la segunda variante de ARQ el receptor devuelve mensajes de acuse de recibo negativos (NAK) sólo cuando se han producido errores o pérdidas de datos. En este caso no se sobrecarga al emisor ni a la red con acuses de recibo innecesarios, ya que sólo se generarán cuando el receptor detecta problemas y solicita la retransmisión de la palabra código perdida o errónea. Desde luego, para poder implementarlo es necesario que las tramas dispongan de un número de secuencia que sirva de referencia para las retransmisiones.

Existen tres sistemas comunes de ARQ:

- Parada y espera (Stop&Wait ARQ):

  El sistema stop-and-wait ARQ, el transmisor envía una palabra de código al receptor y espera una respuesta positiva (ACK) o negativa (NAK) de este último. Un ACK recibido por el transmisor significa que no se detectaron errores en la transmisión y por lo tanto se envía la siguiente palabra de código. Por el contrario, un NAK recibido por el transmisor implica que hubo errores en la transmisión y por lo tanto debe reenviarse la palabra de código que fue transmitida.

  Este sistema es simple y se utiliza en muchos sistemas de comunicación de datos, sin embargo, este esquema es inherentemente ineficiente a causa del tiempo de inactividad de espera de un acuse de recibo de cada palabra de código transmitida. A menos que la longitud del código n es extremadamente larga, la fracción de tiempo de inactividad puede ser grande, sin embargo, utilizando una longitud de bloque muy largo no proporciona una solución, ya que la probabilidad de que un bloque contiene errores aumenta rápidamente con la longitud del bloque.

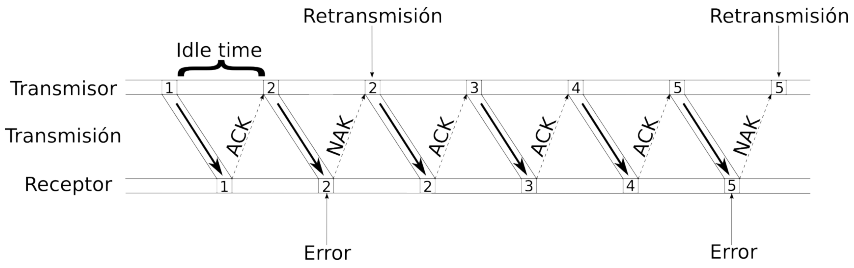


Figura 2.1: Protocol Stop&Wait ARQ.

- Retroceso N (Go-Back-N ARQ):

  En un sistema go-back-N ARQ, las palabras de código se transmiten continuamente. El transmisor no espera un acuse después de enviar una palabra de código; tan pronto como se haya completado el envío de una, comienza el envío de la palabra de código siguiente. El acuse de recibo de una palabra de código llega después de una demora de ida y vuelta, que se define como el intervalo de tiempo entre la transmisión de palabra de código y la recepción de un acuse de recibo de la palabra de código. Durante este intervalo, N-1 palabras de código también han sido transmitidos. Cuando se recibe un NAK, el emisor retransmite las palabras de código desde la que fue detectada errónea, es decir, vuelve a enviar la palabra de código y N-1 siguientes palabras en clave que se transmitieron durante el tiempo de ida y vuelta (es decir, el emisor retrocede y vuelve a enviar N palabras de código).

- Repetición selectiva (selective-repeat ARQ):

  En un sistema selectivo de repetición ARQ, las palabras en clave también se transmiten de forma continua, sin embargo, el transmisor reenvía (o repite) sólo las palabras de código que tienen un acuse negativo (correspondiente NAK). Debido a que las palabras de código normalmente debe ser entregado al usuario en el orden correcto, un búfer debe ser proporcionada en el receptor para almacenar los

vectores sin errores recibidos después de un vector detectado con errores. Cuando la primera palabra con acuse negativo reconocido se recibe correctamente, el receptor libera los vectores libres de errores recibidos en orden consecutivo hasta que se encuentre el siguiente vector erróneamente recibió. El buffer de recepción debe proporcionar suficiente espacio: de lo contrario, el búfer puede desbordarse, y se pueden perder datos. El selective-repeat ARQ es la más eficiente entre los tres esquemas básicos de ARQ, sin embargo, está es la más compleja de implementar.

En los sistemas de ARQ, el receptor debe realizar sólo un procedimiento de detección de error en los mensajes recibidos, sin tratar de corregir los errores. En consecuencia se tiene un código de detección de errores, bajo la forma específica de redundancia o símbolos de verificación de paridad, los cuales deben ser añadidos a la secuencia de los mensajes transmitidos.

A medida que la capacidad de los códigos de detección de errores aumenta, el número de símbolos redundantes añadidos también debe ser aumentado. Claramente, con tal sistema, un mensaje recibido se entrega al usuario siempre y cuando el recibidor no detecta ningún tipo de error. Dado que la codificación de detección de errores es muy sencilla, potente y robusta, los sistemas de ARQ constituyen un método sencillo y eficaz para facilitar la transferencia de alta fiabilidad de los mensajes desde la fuente hasta el usuario sobre una variedad de canales de transmisión. Por lo tanto, los sistemas ARQ están ampliamente utilizados en sistemas de comunicación de datos que son altamente sensibles a los errores, tales como en las comunicaciones de ordenador a ordenador.

Durante una gran congestión, cuando las tasa de pérdida de paquetes son altos, los protocolos basados en ARQ podrían empeorar la congestión debido a la retransmisión de un gran número de paquetes cada vez que se pierde un solo paquete (Davis, 1996).

Por lo tanto, los métodos como el Forward Error Control (FEC) y la ocultación de errores (CE) para la transmisión de imagen/vídeo representan una muy buena solución. Proporcionan una respuesta rápida (muy útil para aplicaciones en tiempo real como la WWW y videoconferencia respectivamente) mediante la eliminación de los retrasos de retransmisión en los gastos de una cierta degradación agradable en la imagen recibida. Además de esto, evitando la retransmisión tiene la ventaja de reducir los requisitos de espacio de amortiguación en la máquina de envío.

Los esquemas de codificación más empleados por los sistemas ARQ son los códigos de paridad y los códigos de redundancia cíclica (CRC - Cyclic Redundancy Check codes), de los cuales puede encontrarse un análisis detallado en [21][15].

§ 2.3 Esquemas híbridos (FEC/ARQ)

La mayor ventaja que tiene ARQ sobre FEC es que la detección de errores requiere un equipo de decodificación mucho menos complejo que el que se requiere para la corrección de errores. Otra desventaja que presenta FEC con respecto a ARQ es que con FEC puede darse el caso de que en la secuencia decodificada binaria aun contenga algunos errores que serían ya incorregibles. Sin embargo, en canales donde la probabilidad de error es alta, emplear un sistema ARQ implicaría la necesidad de retransmitir muchas veces una palabra de código, lo que limitaría de manera considerable la rapidez del sistema.

Comparando los sistemas para el control de errores FEC y ARQ, observamos que ARQ es simple y provee sistemas de alta fiabilidad; Además, los sistemas ARQ tienen un severo retroceso – Los rendimientos disminuyen rápidamente con la creciente tasa de error del canal. Los sistemas que utilizan FEC mantienen el rendimiento constante (igual a la tasa de código $R = k/n$) independientemente de la tasa de error de canal,

sin embargo, los sistemas de FEC tienen dos inconvenientes; en primer lugar, cuando un vector recibido se detecta un error, este debe ser decodificado y el mensaje decodificado debe ser entregado al usuario, independientemente de si es correcto o incorrecto. Porque la probabilidad de un error de decodificación es mucho más grande que la probabilidad de un error no detectado, esto es difícil para lograr un sistema FEC con alta fiabilidad. En segundo lugar, para obtener un sistema de alta fiabilidad y capaz de corregir una gran colección de patrones de error, se debe usar un código potente. Esto hace que la decodificación sea difícil de implementar y costoso. Por estas razones ARQ frecuentemente es preferible por encima de FEC para el control de error en los datos de sistemas de comunicación, como en redes de packet-switching de datos y en redes informáticas de comunicación, sin embargo, en los sistemas de comunicación (o de almacenamiento de datos) donde los canales de retorno no están disponibles o la retransmisión no es posible por alguna razón, FEC es la única opción.

Los inconvenientes en ambos sistemas de ARQ y FEC pueden ser superados si dos esquemas de control de error se combinan adecuadamente, este tipo de combinación se conoce como híbrido ARQ (hybrid-ARQ) y a pesar de que no se ha implementado en muchos sistemas, podría considerarse como la estrategia de control de errores más eficiente de acuerdo con [22].

Un sistema híbrido ARQ consta de un subsistema de FEC contenida en un sistema ARQ. La función del subsistema FEC es reducir la frecuencia de la retransmisión por la corrección de los patrones de error que se producen con mayor frecuencia. Esto aumenta el rendimiento del sistema. Cuando un patrón de error de menor frecuencia se produce y se detecta, el receptor realiza una solicitud de retransmisión en lugar de pasar el mensaje decodificado al usuario. Esto aumenta la fiabilidad del sistema como de un sistema FEC solo y un alto rendimiento como en un sistema con sólo ARQ. Además,

dado que el decodificador está diseñado para corregir una pequeña colección de patrones de error, que puede ser simple. El esquema FEC se puede incorporar en cualquiera de los tres esquemas ARQ básicos.

Tradicionalmente, la mayoría de investigaciones en materia de protocolos multicast fiables se han centrado en la recuperación de errores mediante técnicas ARQ. Sin embargo, el inconveniente de este método es su escasa escalabilidad cuando el número de componentes del grupo multicast empieza a crecer. Para solventar este inconveniente se han propuesto en la literatura diversas técnicas para aportar a ARQ la escalabilidad que le falta, entre ellas las dos siguientes [20]: establecimiento de asincronía entre receptores con la intención de evitar la implosión y, por otro lado, la aplicación de jerarquías a los árboles de distribución multicast. No obstante, la falta de escalabilidad hace que ARQ sea usado en protocolos unicast y se ha propuesto FEC para incrementar la fiabilidad en las comunicaciones multipunto.

3

Códigos de bloque lineal

Para poder codificar un objeto dentro de un código de bloque C, primero será necesario dividir la secuencia de bits del objeto en bloques de igual longitud, denotaremos con k al número de bloques (mejor conocidos como paquetes fuente). Para codificarlos es necesario mapearlos dentro de una palabra de código (*codeword*) el cual consiste de n paquetes ($n > k$) de misma longitud al de los paquetes fuente. Los $n - k$ paquetes adicionales son conocidos como paquetes de chequeo de paridad (*parity-check*), y son los paquetes de redundancia agregados para la detección y corrección de errores.

Un código de bloque es definido por un par de parámetros (n, k), donde k es la longitud del mensaje fuente y n es la longitud del mensaje codificado. La tasa de codificación es k/n y cada mensaje fuente está asociado a uno y solo un mensaje codificado. A continuación se dan definiciones y propiedades básicas respecto a los códigos de bloque.

§ 3.1 Definiciones básicas

Definición 3.1.1 (Esquema de codificación)**.** Un esquema de codificación es una función biyectiva:

$$f : W \to C \subseteq X^n.$$

Definición 3.1.2 (Esquema de decisión)**.** Un esquema de decisión es una función suprayectiva:

$$g : Y^n \to W.$$

Definición 3.1.3 (Código de bloque)**.** Sea $X = \{x_0, x_1, ..., x_{q-1}\}$ llamado un **alfabeto de código** y sea X^n el conjunto de todos los vectores de longitud n sobre X. Cualquier subconjunto no vació $C \subseteq X^n$ es llamado un $q-$ario código de bloque. q es llamado el **radio** del código y cada vector en C es llamado un palabra de código (*codeword*). Si $C \subseteq X^n$ contiene M codewords, entonces se dice que C tiene longitud n y tamaño M y es un $(n, log_q M)-$code

Los códigos de bloque están diseñados bajo campos finitos o bien **Galois Fields** ($\mathbb{F}_q$, donde $q = p$ o bien $q = p^m$ con p primo y $m \in \mathbb{Z}$) debido a las propiedades algebraicas que ofrecen estos campos. Denotaremos en lo siguiente $X = \mathbb{F}_q$. En la práctica p suele ser 2, así que los campos en los cuales se diseñan los bloques de código son $GF(2)$, $GF(4)$, $GF(8)$, $GF(16)$,.... Los algoritmos del RS son considerados de bloque pequeño debido a que trabajan en campos con $m \leq 8$, los LDPC pueden trabajar en campos donde $m \geq 8$. En la tabla 3.1 podemos observar diferentes mensajes fuente con su respectivo mensaje codificado sobre el campo $\mathbb{F}_2$, como se puede observar en este ejemplo, las paquetes son de tamaño un bit.

Mensaje fuente (*Message Word*)	Mensaje codificado (*codeword*)
$m_0 = (0000)$	$c_0 = (0000000)$
$m_1 = (1000)$	$c_1 = (1000110)$
$m_2 = (0100)$	$c_2 = (0100011)$
$m_3 = (1100)$	$c_3 = (1100101)$
$m_4 = (0010)$	$c_4 = (0010111)$
$m_5 = (1010)$	$c_5 = (1010001)$
$m_6 = (0110)$	$c_6 = (0110100)$
$m_7 = (1110)$	$c_7 = (1110010)$
$m_8 = (0001)$	$c_8 = (0001101)$
$m_9 = (1001)$	$c_9 = (1001011)$
$m_{10} = (0101)$	$c_{10} = (0101110)$
$m_{11} = (1101)$	$c_{11} = (1101000)$
$m_{12} = (0011)$	$c_{12} = (0011010)$
$m_{13} = (1011)$	$c_{13} = (1011100)$
$m_{14} = (0111)$	$c_{14} = (0111001)$
$m_{15} = (1111)$	$c_{15} = (1111111)$

Cuadro 3.1: Mensajes codificados de un $(7, 4)$ código de bloque.

§ 3.2 Códigos de bloque lineal

Definición 3.2.1 (Código lineal)**.** Un código de bloque C de longitud n se dice que es lineal si, y sólo si sus q^k codewords forman un subespacio vectorial de $\mathbb{F}_q^n$. La dimensión k de C es la dimensión de C como un $\mathbb{F}_q$-espacio vectorial.

La tabla 3.1 es un ejemplo de un código de bloque lineal. Existen dos propiedades importantes para un código bloque lineal, los cuales son:

1. Cualesquiera dos codeword's de C es otro codeword en C:

$$c_i + c_j = c_k, \tag{3.2.1}$$

donde c_i, c_j y $c_k \in C$.

2. Existe un conjunto de k codeword's en C los cuales son linealmente independientes, es decir, cada codeword en C es una combinación lineal de estos k codeword's.

$$c = x_0 g_0 + x_1 g_1 + ... + x_{k-1} g_{k-1}, \tag{3.2.2}$$

donde $x_i \in \mathbb{F}_q$, $g_i \in C$. A dicho conjunto se le define como base generadora de C.

§ 3.3 Matriz generadora y de paridad

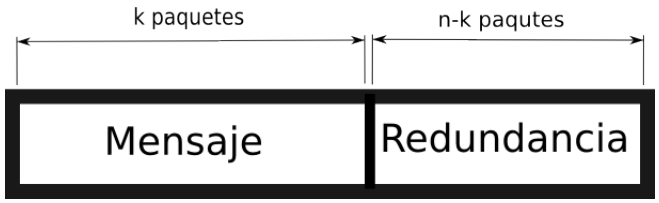
Definición 3.3.1 (Matriz generadora)**.** Sea C un $(n,k)-$código lineal. Una matriz $k \times n$ cuyos renglones forman una base de C en $\mathbb{F}_q^n$ es llamado una matriz generadora.

$$G = \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{pmatrix} = \begin{pmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ & & \vdots & \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{pmatrix} \tag{3.3.1}$$

Ejemplo 3.3.2. Un Hamming $(7,4)-$code tiene como matriz generadora:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{3.3.2}$$

Teniendo una matriz generadora G, ahora podemos dar un mapeo lineal $F^k \to F^n$ vía $x \mapsto xG$ el cual representara al mensaje codificado. Dado un código de bloque $C \subseteq \mathbb{F}_q^n$, seleccionamos una base del subespacio C, del cual formamos la correspondiente



Cuadro 3.2: Estructura sistemática de un codeword.

matriz generadora G, luego dado un mensaje fuente $x = (x_1, x_2, ..., x_k)$ su respectivo codeword es codificado como $c = (c_1, c_2, ..., c_n) = xG$.

$$c = x_0 g_0 + x_1 g_1 + ... + x_{k-1} g_{k-1} = (x_0, x_1, ..., x_{k-1}) \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{pmatrix} = xG \tag{3.3.3}$$

Una propiedad deseable de un código es que tenga una estructura sistemática en los codeword's, el cual consiste en tener en los primeros k lugares la parte del mensaje fuente y luego tener la parte redundante. En un código sistemático, la matriz generadora G está compuesta por una matriz identidad I_k de $k \times k$ y una matriz P de $k \times (n-k)$: $G = (I_k|P)$. Dado G podemos obtener la forma sistemática por medio de operaciones elementales. La tabla 3.1 es un ejemplo de un código sistemático. Para cada código lineal $C \subseteq \mathbb{F}_q^n$ podemos asociarle un compañero: puesto que C tiene una matriz generadora G cuyos renglones definen una base linealmente independiente y ya que C es un subespacio de $\mathbb{F}_q^n$ entonces podemos calcular su subespacio ortogonal $C^\perp$, el cual es llamado el código dual de C cuya dimensión será $n-k$. Si C es un código lineal (n,k), entonces $C^\perp$ es un código lineal $(n, n-k)$. Redefiniremos la notación para $C^\perp$ como H, entonces tendremos que

$$G * H^T = 0.$$

Esto se resume en el siguiente teorema:

Teorema 3.3.3. *Para cualquier matriz $k \times n$ G sobre $GF(q)$ con k renglones linealmente independientes, existe una matriz de $(n-k) \times n$ H sobre $GF(q)$ con $n-k$ renglones linealmente independientes tal que cualquier renglón g_i en G y cualquier renglón h_i de H, $g_i \cdot h_i = 0$. El espacio renglón de G es el espacio nulo de H y viceversa.*

Definición 3.3.4 (Matriz de chequeo de paridad)**.** Dado G una matriz generadora de C, se define H como la *matriz de chequeo de paridad*. Si G esta en forma sistemática ($G = (I_k|P)$) entonces $H = (P^T|I_{n-k})$.

Ejemplo 3.3.5. Un Hamming $(7,4)-$code tiene como matriz de chequeo de paridad:

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{3.3.4}$$

§ 3.4 Rendimiento de los códigos de bloque lineal

Cuanto más distintas sean las palabras del código, mejor es la capacidad de discriminación del código, es decir de detección y corrección de errores. Esta diferencia se mide como una distancia de Hamming y es una cota mínima para la capacidad de detección y corrección de errores.

Definición 3.4.1 (Distancia de Hamming)**.** Sean $x = (x_1, x_2, ..., x_n)$, $y = (y_1, y_2, ..., y_n)$ dos codewords en $\mathbb{F}_q$, entonces se define la **distancia de Hamming** como:

$$d_H(x,y) = \#\{i|x_i \neq y_i\}, \tag{3.4.1}$$

es decir, el número de lugares en el cual las dos palabras código son diferentes.

Proposición 3.4.2. *La distancia de Hamming d_H define una métrica sobre $\mathbb{F}_q$.*

Definición 3.4.3 (Mínima distancia)**.** La mínima distancia $d(C)$ de un código de bloque C es la distancia de Hamming más pequeña entre los codewords:

$$d(C) = \text{mín}\{d_H(x,y) |\ x,y \in C\} \tag{3.4.2}$$

Proposición 3.4.4. *Un código de bloque C con distancia mínima $d = d(C)$ puede detectar $d-1$ errores y corregir $\lfloor \frac{d-1}{2} \rfloor$. Además, estos límites son los mejores posibles.*

Definición 3.4.5 (Peso de Hamming)**.** Sea $x \in \mathbb{F}_q^n$, se define el peso de x ($w(x)$) como el número de coordenadas no cero. El peso mínimo de un código de bloque es el mínimo peso de sus codewords no cero.

Observe que la distancia de Hamming de x, y es el peso de su diferencia $x - y$.

Proposición 3.4.6. *La mínima distancia de un código lineal es su mínimo peso.*

Como conclusión puede ser mostrado que la mínima distancia de un código lineal es igual al mínimo número de columnas de la matriz de chequeo de paridad tales que sumen cero.

Ejemplo 3.4.7. Por ejemplo el código $(7,4)$ tiene distancia mínima 3 puesto que la suma de las columnas 3, 4 y 7 suman cero y es el numero más pequeño en el cual ocurre.

$$\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

4

Códigos correctores de errores FEC

Resumen

En este capítulo abarcamos el estado de arte relacionado a los códigos correctores de errores en los sistemas de comunicación. Comentamos sobre las aplicaciones y su importancia de utilización, así como también, los principales métodos de bloques de código como lo son: los Hamming codes entre otros y los cuales dieron pie a algoritmos más eficientes: como el Reed-Solomon codes (el más notable debido a su radio de codificación), Turbo codes y Low-Density Parity-Check codes (LDPC), estos últimos dos algoritmos relativamente nuevos que pueden proveer casi eficiencia óptima.

Supongamos que queremos enviar un vídeo en el Internet, dicho vídeo es particionado por algún protocolo en k paquetes, dichos paquetes se pueden enviar directamente, sin embargo, debido al ruido que existe en la red muchos paquetes podrían perderse, causando que la información no llegue completa y en consecuencia el vídeo no tenga la misma calidad. Para combatir estos problemas que suelen ocurrir en el Internet y evitar en lo menos posible la retransmisión del mensaje original, se han creado muchos métodos los cuales consisten en agregar información redundante ($n - k$) a la información original (k), de tal manera que la cantidad de información a enviar después de agregar

la redundancia es n, dicha información redundante tiene como objetivo, lograr la recuperación de toda la información original, a pasar que el receptor solo haya recibido una cierta cantidad de información ($k + \epsilon < n$ con ϵ pequeña).

§ 4.1 Estado del arte de los códigos correctores de errores

Entre los códigos descubiertos a lo largo de los años se encuentran: los códigos de Hamming, códigos convolucionales, el algoritmo Viterbi, los códigos Bose-Chaudhuri-Hocquenghem (BCH), los códigos Reed-Solomon (RS), la modulación codificada Trellis (TCM) hasta la aparición de los turbo códigos en los años de la década de los 90s y el redescubrimiento de los códigos de chequeo de paridad de baja densidad (LDPC) por D. J. C. Mackay and R. M. Neal en el año de 1995 [5].

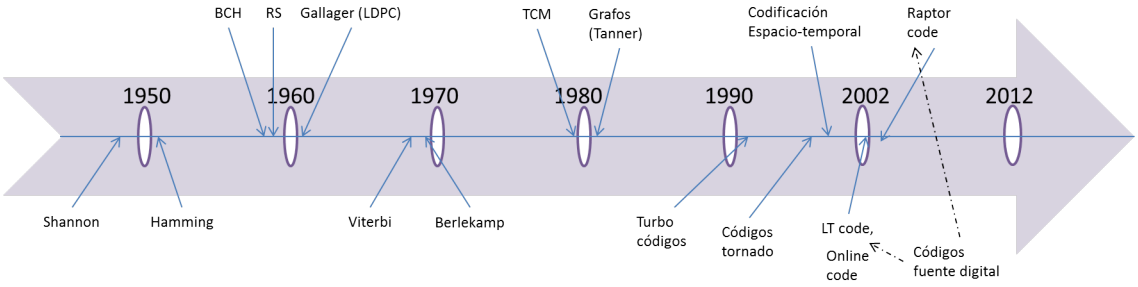


Figura 4.1: Evolución de los algoritmos FEC.

En el caso de las codificaciones *Maximum Distance Separable (MDS)*, 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 dramáticamente 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 se define como:

$$inef\_ratio = \frac{num\ pkg\ para\ decodificar}{k}.$$

Esta ineficiencia ser 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.

Otro parámetro usado en los códigos correctores de errores es la *tasa de codificación* (o tasa de información), definido como la proporción de datos que es útil (no redundantes). Esto es, si la tasa de codificación es k/n, para cada k bits de información útil, el codificador genera n bits de datos en total, de los cuales $n - k$ son redundantes.

Evento	Año	Referencias
Shannon's Theorem	1948	[19]
1948 fue el año en que quedó marcado el nacimiento de la teoría de la información. En ese año, Claude E. Shannon publicó un artículo en el cual abarco temas sobre los límites de la transmisión confiable de datos a través de canales no fiables y métodos sobre la forma de alcanzar estos límites. Entre otras cosas, este documento formaliza el concepto de información, y estableció límites para la cantidad máxima de información que puede ser transmitida a través de canales no fiables.		
Hamming code	1950	[24]
Los códigos usados en este artículo son llamados *códigos sistemáticos*. Un código sistemático puede ser definido como códigos en los cuales cada símbolo de código tiene exactamente n dígitos binarios, donde k dígitos son asociados con la información mientras los otros $h = n - k$ dígitos son usados para la detección y corrección de errores. Esto produce una *redundancia* R definido como el radio de los números de dígitos binarios usados para el mínimo número necesario para transmitir la misma información, esto es, $R = n/k$.		
BCH code	1959	[32][33][34]
Las siglas BCH son debido a las iniciales de los nombres de sus inventores. Los códigos BCH forman parte de corrección de errores de códigos cíclicos que se construyen utilizando campos finitos. Los códigos BCH fueron inventados en 1959 por Hocquenghem, y de forma independiente en 1960 por Bose y Ray-Chaudhuri.		

RS code	1960	[35]
En 1960, Irving Reed y Gus Salomón publicaron un artículo en Journal of the Society for Industrial and Applied Mathematics. En este artículo se describe un nuevo tipo de códigos de corrección de errores que ahora se llaman códigos Reed-Solomon (RS). Estos códigos tienen un gran poder y utilidad, y hoy se encuentra en muchas aplicaciones desde reproductores de discos compactos hasta en aplicaciones para trabajos en el espacio profundo [50]. El Reed-Solomon es un algoritmo que necesita la misma cantidad de paquetes originales para realizar la decodificación, este es un gran punto a favor, sin embargo, este algoritmo tiene una gran desventaja, debido a que trabaja sobre los campos de Galois, se restringe a codificar bajo el tamaño del campo; usualmente F es de tamaño 255 o bien 65, 536, correspondientes a paquetes de 8 o 16 bits, algo también a destacar es que los algoritmos estándar de Reed-Solomon requieren de un tiempo cuadrático para la decodificación, por ultimo estos algoritmos son considerados codificaciones de bloque pequeño debido a la dimensión sobre el cual pueden trabajar [9].		
Gallager code (LDPC)	1960	[12][13]
Robert G. Gallager recibió el grado de Ingeniero Electricista en la Universidad de Pennsylvania en 1953, y los grados de SM y Sc.D. en ingeniería eléctrica del Instituto de Tecnología de Massachusetts en 1957 y 1960, respectivamente. De 1953 a 1956, estuvo en los Bell Telephone Laboratories y luego en U.S. Signal Corps. Ha sido miembro del cuerpo docente en el MIT desde 1960, se convirtió en Co-Director del Laboratorio de Sistemas de Información y Decisión en 1986, y Fujitsu profesor en 1988. Su título actual es Profesor Emérito [51]. En su trabajo sobre los LDPC no mostró alguna relación entre grafos y matrices de paridad, relación que se le da años más tarde.		

Viterbi	1967	[36][37]
El algoritmo Viterbi fue propuesto por Andrew Viterbi en 1967 como un decodificador para los códigos convolucionles sobre líneas de comunicación digital.		
Berlekamp	1969	[38][39]
Es un algoritmo propuesto por Elwyn Berlekamp para la decodificación de los códigos Bose-Chaudhuri-Hocquenghem (BCH). Este algoritmo de decodificación consiguió que los códigos Reed-Solomon sean muy eficientes.		
TCM code	1976	[40]
Este algoritmo fue inventado por Gottfried Ungerboeck trabajando para IBM en la década de 1970's, sin embargo fue en 1976 cuando se describió por primera vez en un artículo en conferencia. Pasó desapercibido hasta que se publicó una nueva exposición detallada en 1982.		
Tanner graph	1981	[41]
R. Michael Tanner, introdujo la relación de grafos bipartitos y matrices de paridad. La idea básica es usar los grafos para estructurar las ecuaciones que definen el código de una manera que facilita la codificación y decodificación. En su trabajo realizo esta relación de grafos con los códigos LDPC y BCH. Un grafo bipartito es un conjunto de vértices $V = V_1 \cup V_2$, donde cada línea de conexión está formado por un vértice de V_1 y un vértice de V_2. Los grafos mejoraron el desempeño y eficiencia de los códigos.		

Turbo codes	1993	[17][25]
En el año de 1993 se realizó en Genova Suiza la conferencia internacional anual en comunicaciones. En este evento Claude Berrou, Alain Glavieux y Punya Thitimajshima presentaron la invención de un nuevo esquema de corrección de errores. Este esquema presentó mejores resultados que las técnicas actualmente existentes y una mucha mejor ganancia en la codificación [25]. Fue mostrada una gráfica en simulación en la cual su desempeño fue cercano a la capacidad teórica del canal. Ésta presentó una probabilidad de error de 10^{-5} para un valor de SNR de 0.7 dB. Estos resultados inspiraron a investigadores para proponer nuevos esquemas de codificación, mejoras y simplificación de los códigos existentes.		
LDPC (Mackay & Neal)	1995	[5]
Los códigos LDPC también conocidos como códigos de Gallager en honor a Robert G. Gallager que desarrolló el concepto de LDPC en su tesis doctoral en el MIT en 1960. Debido a la gran dificultad de aplicación cuando se inventaron, los códigos LDPC fueron olvidados. Con la aparición de los turbo códigos en 1993 usados para comunicaciones por satélite, en la década de los 90 los códigos LDPC fueron redescubiertos he iniciaron una revolución en los códigos correctores de errores pues han dejado ver que son una gran herramienta para la creación de nuevos códigos FEC.		
Tornado codes	1997	[42][47][48]
Los códigos de Tornado son códigos de borrado de bloques basados en grafos bipartitos irregulares dispersos. Dado un canal borrado con probabilidad p de pérdida, se puede corregir hasta $p(1-e)$ errores. Pueden ser codificado y decodificado en un tiempo proporcional a $Nlog(1/e)$. Por lo tanto los códigos de Tornado han sido diseñados principalmente para acelerar los códigos de borrado a través de Internet [42]. Los códigos de tornado son parte de una idea inicial (Códigos de fuente digital) generada por la empresa Digital Fountain.		

STC	1998	[43][52][53]
Los códigos espacio-temporal son un enfoque altamente eficientes del ancho de banda para la señalización dentro de la comunicación inalámbrica que se aprovecha de la dimensión espacial mediante la transmisión de una serie de flujos de datos utilizando múltiples antenas conjuntamente localizadas [52].		
Fountaine code	2002	[4][6][8][9][11]
Los códigos de fuente digital surgieron como una abstracción de la codificación de borrado, cambiaron el paradigma donde el usuario tenía que recibir los paquetes ordenados a uno donde el usuario solo debería recibir una cierta cantidad de paquetes. La idea principal de un código de fuente digital, se puede explicar basándonos de cómo funciona una fuente de agua: cuando usted llena su taza de la fuente, no le importa que gotas caen de la fuente, porque solo desea que su taza llena lo suficiente para saciar su sed. Con una fuente digital, el cliente obtiene los paquetes codificados a partir de uno o más servidores, y una vez que se obtienen suficientes paquetes, el cliente puede reconstruir el archivo original. ¿Qué paquetes se obtienen? es algo que no debería importar.[9] En [6] códigos de fuente lineales aleatorios, códigos LT, y los códigos raptor son revisados, donde los costos de cómputo de los mejores códigos de fuente son asombrosamente pequeña, escalando de forma lineal con el tamaño del archivo.		
LT codes	2002	[10]
Los códigos LT (Luby Transform) inventados por Luby en 1998, son la primera realización de una clase de códigos de borrado que Michael Luby llama códigos de borrado universales. Son la primera clase de los códigos de fuente digital: el número de paquetes de codificación que puede ser generado a partir de los datos es potencialmente ilimitado. Michael Luby y Jay Goldin fundaron de la empresa Digital Fountain en 1998, la primera empresa cuyo negocio se basa en los códigos de grafos-sparse.		

Online codes	2002	[44]
Son una clase de códigos casi-óptimos para canales de pérdidas. Los códigos en línea tienen codificación/decodificación en tiempo lineal, están basados en grafos bipartitos sparse similares a los códigos Tornado, pero con un par de propiedades nuevas: localmente codificable y rateless-ness. *Localmente codificable* implica que cada bloque de la codificación de un mensaje puede ser calculado independientemente de los otros en un tiempo constante. *Rateless-ness* es la propiedad de que cada mensaje tiene una codificación de prácticamente tamaño infinito.		
Raptor codes	2003	[1][2][45]
En [1] se introducen los códigos Raptor, los cuales son una extensión de los códigos LT con codificación y decodificación en tiempo lineal. Se exhibe una clase de códigos Raptor tal que para un entero k dado, y cualquier $\epsilon > 0$, los códigos Raptor producen una cadena infinita de paquetes tales que con cualquier subconjunto de paquetes de tamaño $k(1-\epsilon)$ será suficiente para recuperar los k paquetes originales con una alta probabilidad. Cada paquete codificado es generado usando $O(\log(1/\epsilon))$ operaciones y los paquetes originales son recuperados de una colección con $O(k\log(1/\epsilon))$ operaciones.		

En la época actual los esfuerzos se enfocan a resolver varios problemas en las implementaciones de códigos correctores de errores entre los cuales se encuentran: disminuir la complejidad en la decodificación, mejorar la ganancia en la codificación, asociar los métodos de codificación a las técnicas actuales de comunicación, entre otros.

Varias empresas dedicadas al ambiente de comunicaciones, buscan nuevas mejoras en los códigos existentes, de tal manera que puedan ofrecer un mejor servicio con la más alta calidad en la trasmisión de información.

Cada año se realiza el *Simposium internacional de turbo códigos y procesamiento ite-*

rativo de información, el cual tiene como objetivo dar a conocer el estado actual y las investigaciones realizadas de los métodos iterativos y de su aplicación en la teoría de la codificación de la información y de las comunicaciones digitales. Las dos principales técnicas de codificación actuales (turbo códigos y LDPC) son estudiadas así como variantes y métodos iterativos de codificación [26].

§ 4.2 LDPC (Low-density parity-check)

Los códigos LDPC han marcado un gran cambio en los algoritmos de sistemas FEC, estos algoritmos han dado lugar a la generación de nuevos códigos FEC (los algoritmos de fuente digital), sin embargo no han quedado atrás y siguen siendo un caso de gran investigación. A diferencia de los turbo códigos que son más aplicados a los medios inalámbricos, los códigos LDPC tienen una gran área de aplicación en las redes de comunicación (como lo es el Internet). A continuación abarcamos aspectos generales sobre estos algoritmos.

4.2.1 Grafos de Tanner y Matrices de chequeo de paridad

Un Grafo bipartito G se denomina al grafo cuyos vértices se pueden separar en dos subconjuntos disjuntos $V_1(G)$ y $V_2(G)$ y las líneas siempre unen vértices de un subconjunto con vértices de otro subconjunto. Además se tiene que:

- $V_1(G) \cup V_2(G) = V(G)$.
- La intersección de $V_1(G)$ y $V_2(G)$ es vacío.
- Para todos los puntos x_1, x_2 en $V_1(G)$ y para todos los puntos y_1, y_2 en $V_2(G)$, no existe línea alguna $x = (x_1, x_2)$, ni $x = (y_1, y_2)$.

Un grafo bipartito en el cual todos los elementos de V_1 están unidos con todos los elementos de V_2 se denomina *grafo bipartito completo*, ver figura 4.2.

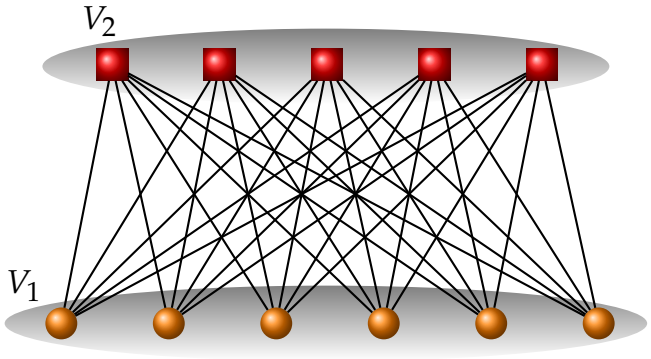


Figura 4.2: Grafo bipartito completo $K_{5,6}$

Se define como *grado del nodo* al número de líneas que llegan a conectarlo. Un grafo bipartito se dice que es *regular* si todos los nodos de cada conjunto tienen respectivamente el mismo grado (ver figura 4.4(a)), de lo contrario se dice que es *irregular*.

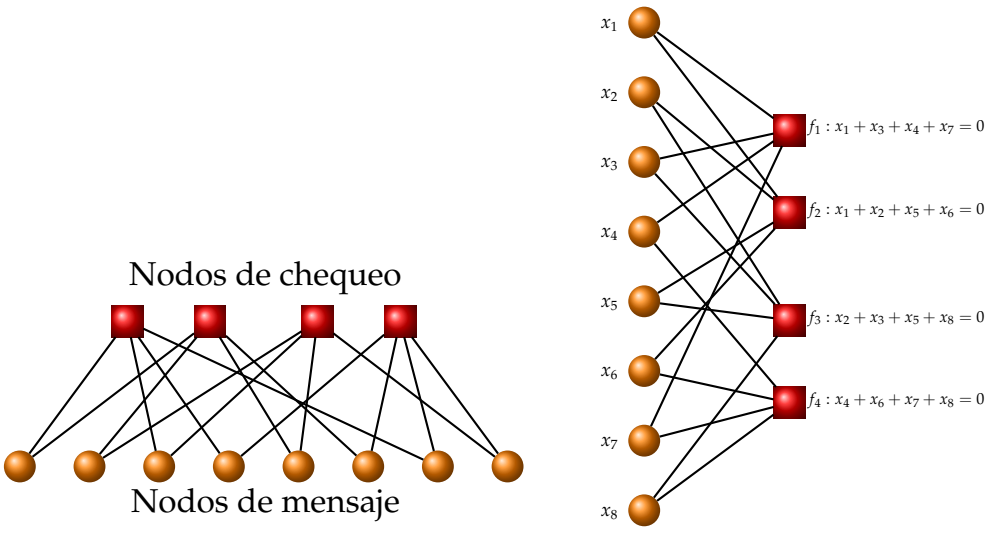
Grafos de Tanner regulares e irregulares

R. Michael Tanner, introdujo la relación entre una matriz de paridad de los LDPC (códigos Gallager), con los grafos bipartitos (ver [41]). Los códigos que uso Gallager se denominan regulares, esto debido a que la representación de su respectivo grafo es un grafo bipartito regular. Un *código de Gallager* (l, c)*-regular* es un código de chequeo de paridad de baja densidad en el que todas las columnas de H (matriz de paridad, ver [12]) tiene el mismo grado l y cada renglón tiene el mismo grado c (ver matriz 4.3), de lo contrario diremos que es un código de Gallager irregular. Un código LDPC tiene asociado un grafo de Tanner regular/irregular en el cual los grados se eligen de acuerdo a alguna distribución.

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{4.2.1}$$

Figura 4.3: Matriz de chequeo de paridad.

En la actualidad, los códigos LDPC son códigos lineales obtenidos de grafos bipartitos sparse. Supongamos que G es un grafo con n nodos izquierdos (llamados *nodos de mensaje*) y r nodos derechos (llamados *nodos de chequeo*) ver figura 4.4(a) y dimensión a lo más nxr en la siguiente manera: las n coordenadas de la palabra código son asociados con los n nodos de mensaje. Los códigos de palabra de esos vectores $(x_1, ..., x_n)$ tal que para todos los nodos de chequeo, la suma de las posiciones de vecinos entre los nodos de mensaje es cero, ver figura 4.4(b).



(a) Grafo bipartito $(2, 4)-$regular. (b) Sistema de ecuaciones.

Figura 4.4: Representación de un grafo de Tanner de un código LDPC.

A diferencia de Gallager que primero obtenía una matriz generadora de código, con la aparición de la relación con los grafos, ahora primero se construye el grafo bipartito sparse, luego se procede a representar la matriz de paridad y así obtener la matriz generadora de código. El comportamiento de los códigos LDPC puede mejorarse significativamente al permitir que los nodos tengan diferentes grados, así como otras mejoras estructurales. Es por ello que en los algoritmos actualidad existen códigos que están basados en grafos de Tanner irregulares.

4.2.2 Códigos LDPC

En 1993, se inventó un sistema de codificación iterativa: los códigos Turbo [17]. A la luz de este descubrimiento, los códigos LDPC fueron redescubiertos por MacKay y Neal [5] en 1995. Luego de ese redescubrimiento, han surgido y siguen surgiendo una cantidad importante de trabajos relacionados con los LDPC. En la actualidad son parte importante de los últimos códigos para la corrección de errores, denominados *códigos de fuente digital*. 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. A continuación damos una breve descripción de los códigos LDPC.

Un código LDPC es definido como el espacio nulo de una matriz rala H, denominada *matriz de chequeo de paridad*. Actualmente es representado por un grafo bipartito, usualmente llamado *grafos de Tanner*, en el cual los nodos de mensaje corresponden a las palabras código y los nodos de chequeo corresponden a las restricciones de chequeo de paridad. Con respecto a la matriz H, esta tiene dimensiones $r \times n$ donde $r = n - k$ y representa el número de ecuaciones, las k primeras columnas representan la información original con $n - k$ paquetes de redundancia o bien también conocidos como paquetes de paridad, es importante mencionar que la operación que se realiza es XOR

(operación bit a bit).

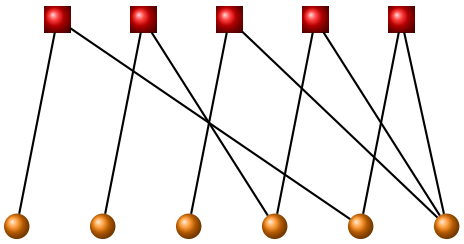
La representación matricial del grafo se obtiene viendo la matriz de adyacencia del grafo: Sea H una matriz binaria $r \times n$ en el cual la entrada (i, j) es 1 si y solo si el $i-$ésimo nodo de chequeo está conectado al $j-$ésimo nodo de mensaje en el grafo. Entonces el código LDPC definido por el grafo es el conjunto de vectores $x = (x_1, ..., x_n)$ tales que $Hx^T = 0$. La matriz H es llamado *matriz de chequeo de paridad*. Por el contrario, cualquier $(r \times n)-$matriz binaria da lugar a un grafo bipartito entre los n nodos de mensaje y los r nodos de chequeo, y el código definido como el espacio nulo de H es precisamente el código asociado a este grafo. Por lo tanto, tenemos que un código lineal binario tiene una representación dada por un grafo bipartito. Sin embargo no todos los códigos lineales binarios tienen una representación por un grafo bipartito ralo.

Es muy importante que la matriz asociada al grafo sea rala, si esto pasa entonces decimos que tenemos un código de chequeo de paridad de baja densidad (LDPC). Y la propiedad de ser rala es la clave para permitir crear algoritmos eficientes para la codificación y decodificación de un código LDPC. Mas general, por ejemplo, el código LDPC $(l, c)-$regular es un código lineal binario tal que cada nodo de mensaje tiene grado l y cada nodo de chequeo tiene grado c, ahora bien, el número de conexiones o links en el grafo de Tanner es ln, donde n es la longitud del código. Con forme n incrementa su valor, el número de conexiones en el grafo de Tanner crece linealmente con respecto a n.

A continuación mostramos una matriz de chequeo de paridad y su respectivo grafo representativo: teniendo en cuenta que las columnas representan los nodos de mensaje y los renglones representan los nodos de chequeo.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

(a) Matriz de paridad



(b) Grafo bipartito irregular.

Figura 4.5: Matriz de paridad y grafo respectivo.

Construcción de un LDPC

Un código LDPC es construido por medio de la matriz de chequeo de paridad H o bien por medio del grafo de Tanner, el principal requerimiento es que sea ralo. Existen diferentes métodos para conseguir esto, Gallager y Mackay han propuesto sus propios métodos.

Codificación

La forma de codificación es usando las columnas de la matriz H, cada columna nos informa que paquetes se van a sumar, por ejemplo: supongamos que tenemos 5 paquetes, cada uno de longitud de un bit, sea $msg = \{0,1,0,1,1\}$ el mensaje a enviar, ahora bien codificando el mensaje usando la matriz de paridad 4.5(a) obtenemos $msg\_cod = \{0,1,0,0,1,0\}$, donde las primeras tres palabras códigos codificados son iguales a los originales, la cuarta palabra de código queda como la suma de los paquetes 2 y 4, la quinta palabra de código queda como la suma de los paquetes 1 y 5, y la sexta palabra de código queda como la suma de los paquetes $3, 4$ y 5. En este ejemplo podemos observar que obtenemos un paquete de redundancia.

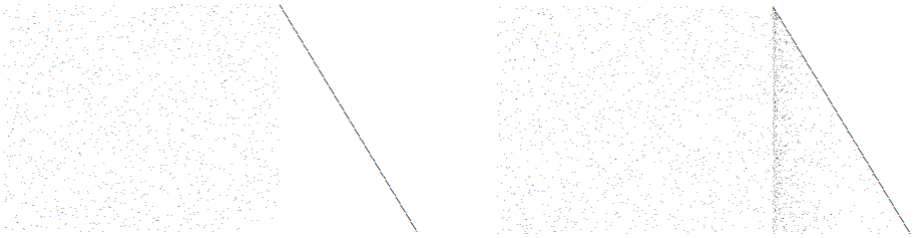
El ejemplo anterior es la forma más simple de codificar un mensaje, sin embargo, existen varias formas para decodificar, desde la forma de usar la matriz generadora G, el problema que sucede con esto, es que la complejidad crece en $O(n^2)$ para longitudes de bloques grandes, es por ello de la importancia de las matrices ralas. Los algoritmos más utilizados son los iterativos y están basados con respecto a la estructura de la matriz de chequeo de paridad o bien a la estructura de los grafos de Tanner.

Decodificación

El algoritmo utilizado para descodificar códigos de LDPC se descubrió varias veces de manera independiente y como una cuestión de hecho viene con diferentes nombres. Los más comunes son el *belief propagation algorithm*, el *message passing algorithm* y el *sum-product algorithm*. En [54] se muestra la explicación de este algoritmo, usando dos variantes, una muy simple usando *hard-decision* para luego extenderlo usando *soft-decision* el cual es más general con mejores resultados de decodificación.

4.2.3 Low Density Generator Matrix (LDGM)

Los códigos LDGM son un caso particular de los códigos LDPC, con la propiedad de tener una codificación más rápida, en comparación con los códigos LDPC tradicionales. La estructura de la matriz de paridad H tiene la siguiente forma $H = [H'\ P]$ donde H' es de dimensión $r \times k$. Con respecto a la matriz P: es la matriz identidad con dimensión $r \times r$ en el caso de LDGM; es una matriz bidiagonal inferior en el caso de LDGM Staircase; es una triangular inferior con distribución aleatoria de unos en el caso de LDGM Triangle. Para más detalle sobre este tipo de matrices ver [55].



(a) LDGM staircase (b) LDGM triangle

Figura 4.6: Ejemplo de matriz de paridad LDGM staircase y triangle (k=400,n=600). Extraído de [55].

§ 4.3 Códigos de fuente digital

En la actualidad los algoritmos más eficientes son los algoritmos de fuente digital, recibieron este nombre debido a la idea en que trabajan: pensando en una fuente de agua en el cual solo necesitamos llenar un bote con agua para estar satisfechos, dicha fuente arroja una gran cantidad de gotas (paquetes) que son capturados en el recipiente o recipientes (receptor o receptores). Estos algoritmos fueron diseñados a partir de los algoritmos LDPC, su principal característica es que crean una "codificación infinita" y son creados en tiempo real con una alta probabilidad cada vez más alta de decodificación conforme se reciben más paquetes. La idea fundamental es que con un código de fuente digital, todos los paquetes que llegan al destino son útiles, siempre y cuando no sean paquetes duplicados. En esta sección explicaremos y comentaremos algunas de las principales características de estos algoritmos. Claros ejemplos de algoritmos de fuente son: Tornado codes, LT codes y Raptor codes.

En los códigos de fuente digital se diseñan matrices aleatorias linealmente independiente cada n pasos, donde n es el total de los k paquetes originales y los paquetes de redundancia, el receptor necesita cualesquier $k + \epsilon$ paquetes para proceder a recuperar

toda la información.

Los códigos de fuente digital son algoritmos muy eficientes, surgieron a partir de los códigos LDPC, le siguieron los LDGM para luego crear los Tornado codes (no son estrictamente un código de fuente digital, sin embargo son los precursores de los LT codes), le siguieron los LT codes (la primera implementación de esta nueva clase de códigos de fuente digital) y ahora tenemos a los Raptor codes, este último se encuentra bajo patente.

Un código *rateless* genera tantos paquetes codificados como sean necesarios y adapta su tasa de codificación dinámicamente. Cualquier cantidad suficientemente grande de paquetes codificados permite recuperar los datos en el receptor. Los códigos LT (Luby, 2002) fueron los primeros códigos rateless eficientes, capaces de lograr de manera asintótica la capacidad del canal BEC, pero presentan un error de base cuando se acota la complejidad en la decodificación. El esquema rateless puede prácticamente ser implementado con códigos de fuente digital, por lo que se dice que son una familia de códigos rateless.

Un codificador de fuente digital

Cada paquete codificado t_n se genera a partir del fichero fuente $s_1, s_2, s_3, s_4, ..., s_k$ de la forma siguiente:

1. Se escoge de manera aleatoria el grado d_n del paquete de una distribución de grado $\rho(d)$; la elección adecuada de ρ depende del tamaño K del fichero fuente.

2. Escoger, de forma uniforme y aleatoria, d_n paquetes de entrada distintos. Igualar t_n a la suma binaria a nivel de bit, módulo 2 de esos d_n paquetes. Esta suma puede realizarse sucesivamente mediante sumas XOR de los paquetes.

El decodificador necesita conocer el grado de cada paquete que se recibe, y a qué paquetes fuente está conectada en el grafo. Esta información se puede comunicar al decodificador de varias maneras. Por ejemplo, si el transmisor y el receptor tienen sincronizados los relojes, puede usar generadores de número pseudo-aleatorios idénticos, alimentados por el reloj, para escoger el grado aleatorio y cada conjunto de conexiones. De forma alternativa, el transmisor podría escoger una llave aleatoria, τ_n, con la que el grado y las conexiones quedan determinados mediante un proceso pseudo-aleatorio, y enviar esta llave en la cabecera del paquete. En la medida que el tamaño del paquete sea mucho mayor que el tamaño de la llave, ésta sólo introducirá un pequeño overhead.

Decodificador

Decodificar un código con un gráfico de baja densidad es especialmente sencillo en el caso de un canal de borradura. La función del decodificador es recuperar s de $t = H \cdot s$, donde H es la matriz asociada al gráfico. El camino simple para intentar resolver este problema es mediante paso de mensajes. Se puede entender el algoritmo de decodificación como el algoritmo de suma-producto, pero todos los mensajes o son completamente inciertos o completamente ciertos. La premisa de mensaje incierto significa que el paquete s_k puede tener cualquier valor con la misma probabilidad. La premisa mensajes cierto significa que el paquete s_k tiene un valor en particular con probabilidad 1.

Esta simplicidad de los mensajes permite una descripción sencilla del proceso de decodificación. Los paquetes codificados $\{t_n\}$ los denominamos nodos de comprobación. Así el proceso de decodificación consiste:

1. Encontrar un nodo de comprobación t_n que esté conectado sólo a un paquete fuente s_k. (si no existe tal nodo de comprobación, el algoritmo se para en este

punto, y no es capaz de recuperar todos los paquetes fuente).

a Asignar $s_k = t_n$

b Sumar s_k a todos los nodos $t_{n'}$ que están conectados a s_k: 1. $t_{n'} := t_{n'} + s_k$ para todo n' tal que $G_{n',k} = 1$

c Eliminar todas las conexiones al paquete fuente s_k.

2. Repetir (1) hasta que todos los s_k sean determinados.

Veamos con un ejemplo cómo funciona el decodificador: continuando con el ejemplo de la matriz de paridad 4.5(a), supongamos que el receptor obtiene los siguientes códigos de mensaje $msg\_rev = \{0, 1, 0, 0, 1, \_\}$, es decir, perdimos el ultimo código de mensaje; procedamos a la recuperación ayudándonos de su respectivo grafo:

4.7(a) Inicializamos los códigos de palabra en sus respectivos lugares correctos y borramos las conexiones que salen del nodo de mensaje que perdimos.

4.7(b) Procedemos a buscar un nodo de mensaje que tenga grado uno y tomamos su valor colocando el respectivo valor en el nodo de chequeo con el cual tiene una conexión para luego realizar operaciones XOR con los nodos de mensaje al que está conectado este nodo de chequeo, luego procedemos a borrar dichas conexiones.

4.7(c) Continuamos con el siguiente nodo de mensaje que tenga grado uno y realizamos el procedimiento que se realizó en el inciso 4.7(b), de esta manera vamos obteniendo un decremento en los grados de los nodos de mensaje, como podemos ver en el inciso 4.7(d).

4.7(d) Procedemos a buscar nodos de mensaje con grado uno para enviar un mensaje a sus respectivos nodos de mensaje, informándoles del valor que tienen. Finalizaremos con valores en los nodos de chequeo, dichos valores serán la información original que se le haba enviado al receptor. Es importante notar que siempre se necesita un nodo mensaje con grado uno para proceder, de lo contrario el método falla.

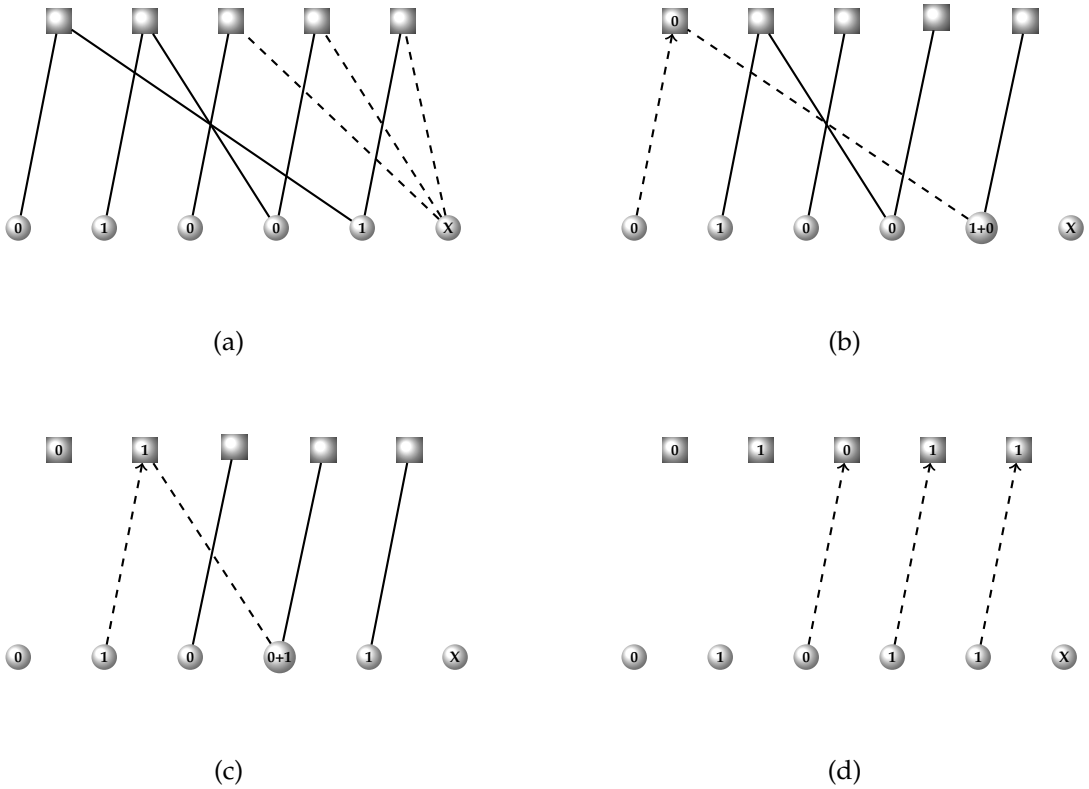


Figura 4.7: Esquema de recuperación de información usado por los códigos de fuente.

Observemos que usando la matriz de paridad 4.5(a), podemos perder cualquier nodo de mensaje y aun así obtendremos toda la información. Bajo esta matriz de paridad, solo necesitamos cuidar que nos llegue 5 nodos mensaje y as asegurar la recuperación.

4.3.1 Raptor Codes

Los códigos Raptor, introducidos (Shokrollahi, 2006) como una extensión de los códigos LT, se construyen a partir de la concatenación de un código LT y un código bloque con una tasa elevada llamado precódigo. Estos códigos han sido estudiados para el canal BEC, y son utilizados en la capa de aplicación como un protocolo eficiente para multicast (V7.6.0, 2007).

Los códigos Raptor mejoran la fiabilidad del sistema, al mismo tiempo que permiten un amplio grado de libertad en la elección de los parámetros de transmisión. Éstos son códigos fuente y, por tanto, se pueden generar tantos paquetes codificados por el codificador como se deseen al vuelo desde los paquetes fuente de un bloque de datos. El decodificador es capaz de recuperar el bloque de datos fuente de cualquier conjunto de los paquetes codificados sólo necesitando un número un poco mayor que el número de paquetes fuente. Como resultado de ello, los códigos Raptor operan casi como un código fuente ideal, el cual requeriría sólo el número exacto de paquetes fuente para la recuperación.

Como extensión de los códigos LT, los códigos Raptor son una clase de códigos fuente con coste constante de codificación y decodificación lineal. En comparación los códigos LT, los códigos Raptor alcanzan su superioridad computacional a expensas de un overhead asintóticamente mayor, aunque en la mayoría de los ajustes prácticos superan a los códigos LT en todos los aspectos. De hecho, para un overhead constante ? se pueden construir familias de códigos Raptor con coste de codificación $O(log(1/\epsilon))$, coste de decodificación $O(k \cdot log(1/\epsilon))$ y una probabilidad de error de decodificación que decae asintóticamente de forma inversa al polinomio en k (Shokrollahi, 2006).

Los códigos Raptor logran sus resultados utilizando una idea simple: la fuente x es precodificada utilizando un código lineal C, de dimensión K y un bloque de lon-

gitud n. La codificación de x con C produce un vector $z = (z_1, z_2, z_3, ..., z_m)$, llamados paquetes de entrada. A menudo, se utiliza una codificación para C, en cuyo caso $z = (x_1, x_2, x_3, ..., x_k, z_1, z_2, z_3, ..., z_{n-k})$, donde $z_1, z_2, z_3, ..., z_{m-k}$ son paquetes redundantes. Después, un código LT adecuadamente elegido se aplica a la z para crear paquetes de salida $y_1, y_2, ...$ Un código Raptor puede ser caracterizado por sus parámetros $(C, k, \Omega(x))$.

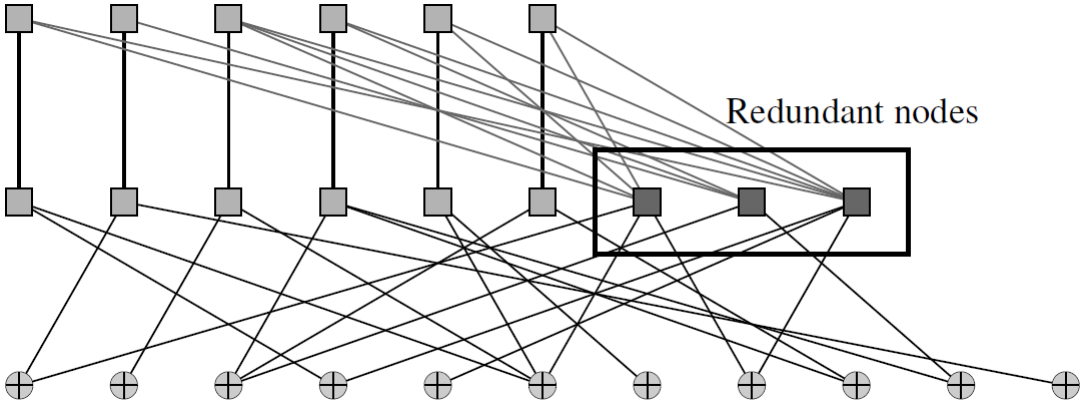


Figura 4.8: Raptor codes: Los símbolos de entrada son adjuntados con paquetes de redundancia (cuadros negros) en el caso de una precodificación sistemática.

§ 4.4 Parámetros y mediciones de desempeño

A continuación se introducen parámetros para medir el desempeño de los códigos FEC. Datos extraídos de [59].

- *FEC expansión ratio*: la tasa de código (*code rate*), definido como $R = k/n$ donde n es el número total de paquetes creados y el radio de expansión definido como el inverso del *code rate* ($fec_{ratio} = 1/R = n/k$), para este parámetro se tiene que cuanto mayor sea, mas paquetes de paridad son creados. Las tasas de código más usados en la literatura son $1/3$, $1/2$ y $2/3$. Los códigos FEC capaces de producir

un número infinito de paquetes de paridad tienen una tasa igual a cero y son llamados *rate-less*.

El máximo valor para la tasa de codificación es 1 cuando no hay redundancia agregada (esto es cuando el mensaje no se codifica). El desempeño de la codificación y la tasa de codificación son dos factores opuestos. Como se añade más redundancia, la capacidad de corrección de error es fortalecido pero la tasa de codificación baja. Un buen código debe maximizar el rendimiento de corrección de errores mientras se mantiene la tasa de codificación cercana a 1.

- tamaño del objeto: Si se considera paquetes de 1024 bytes de longitud (por ejemplo) entonces se toma como referencia la cantidad de paquetes y no la cantidad de bytes del objeto. Con los códigos LDPC/LDGM se codifica con un simple bloque que consiste de k paquetes, parámetro con el que trabajan los códigos FEC. En el caso de RSE, el objeto es dividido en varios bloques, siendo k el tamaño de cada bloque y en consecuencia la longitud de los paquetes es menor al de un simple bloque.

Global decoding inefficiency ratio: la ineficiencia de decodificación ($inef_{ratio}$) puede ser causado por el factor de que el sistema FEC no sea MDS (por ejemplo: LDPC/LDGM, códigos de fuente digital), ya que se deben recibir más de k paquetes para decodificar un bloque de tamaño k completo (esto también es conocido como *overhead factor* en [58]). Otro factor que produce esto es la necesidad de dividir objetos grandes en bloques pequeños, un ejemplo de esto es RSE.

En ambos casos se tiene que $inef_{ratio} * (numero\ de\ paquetes\ fuente)$, con $inef_{ratio} \leq 1$, debe ser recibido en orden para decodificar todo(s) el/los bloque(s).

$$inef_{ratio} = \frac{Numero\ de\ paquetes\ requeridos\ para\ decodificar}{numero\ de\ paquetes\ fuente} \tag{4.4.1}$$

Decoding inefficiency ratio confidence intervals: Considerando solo el promedio de $inef_{ratio}$ no es suficiente ya que se presentan algunas veces valores grandes, por lo tanto se suele medir varios intervalos de confianza para valores más grandes que el promedio y de la misma manera para valores más pequeños que el promedio.

Encoding/decoding times and bandwidths: La medición del tiempo de inicialización y el tiempo total de codificación (respectivamente decodificación) permiten medir el máximo ancho de banda sostenible. Más precisamente, dos codificaciones de ancho de banda pueden ser consideradas, una medición global de la codificación de ancho de banda:

$$globlal\_encoding\_bw = \frac{n * pkt\_size\_in\_bits)}{time} \tag{4.4.2}$$

y una medición parcial de la codificación del ancho de banda el cual solo toma en cuenta el número de paquetes de paridad producidos:

$$partial\_encoding\_bw = \frac{(n-k) * pkt\_size\_in\_bits)}{time} \tag{4.4.3}$$

La primera medición indica si el codec puede soportar una tasa de transmisión dada utilizando un sistema FEC sistemático (por default tanto paquetes fuente como de paridad son enviados), mientras que el segundo se refiere a lo que actualmente es producido por el sistema FEC.

Maximum Memory Requirements: Se suele medir durante la codificación y decodifiación, esto por la necesidad de almacenar los códigos fuente y los paquetes de paridad, también el buffer de almacenamiento de los codec's o bien la matriz de paridad que usan los LDPC/LDGM.

5

Método propuesto

Resumen

En este capítulo mostramos un código de fuente-hibrido-ARQ (FHARQ), en el sentido que se solicitaran tantos paquetes redundantes como paquetes perdidos se produzcan en Internet. El esquema está diseñado para generar una gran cantidad de paquetes redundantes, y para ser usado en streaming, aunque no se descarta la utilización en la transmisión de objetos ya almacenados (por ello se realiza algunas simulaciones con el objetivo de transmitir de objetos grandes). Es importante mencionar que los paquetes de redundancia se crearan al vuelo, con forme el receptor lo requiera. En cuanto al tipo de perdidas, este esquema es diseñado con respecto a una determinada cantidad de perdida en ráfagas (principal problema que ocurre en el Internet).

El método consiste en crear conjuntos a partir de k paquetes, estos conjuntos son codificados generando los paquetes de redundancia, cada conjunto tendrá paquetes codificados de conjuntos anteriores los cuales a su vez ayudaran a recuperar paquetes que se pierdan en el Internet. El método de decodificación consiste de dos etapas, la primera es recuperar toda la información posible y determinar que paquetes son necesarios para recuperar toda la información; en la segunda etapa se considera una retransmisión de los paquetes de redundancia necesarios para una decodificación completa.

Como se ha comentado en los capítulos anteriores, existen varios algoritmos para la recuperación de información en los canales de borradura, desde los códigos de Hamming hasta los códigos Raptor, estos últimos presentando muy buenos resultados en el proceso de codificación y decodificación. Desde luego estos códigos FEC también son usados para la transmisión de vídeo en tiempo real (los códigos Raptor son usados en los servicios de streaming 3GPP o DVB [63]), a pesar de ser códigos que no se comportan nada bien para objetos pequeños. Algo de lo que queremos evitar con respecto a los códigos de fuente digital es la generación aleatoria de la matriz de chequeo de paridad, debido a que de alguna manera hay que informarle al receptor con qué tipo de matriz se está codificando. El esquema que a continuación expondremos usa una matriz de chequeo de paridad fija, evitando el problema de sincronización entre el emisor y el receptor. Otros códigos para la trasmisión de vídeo en tiempo real son considerados en [64].

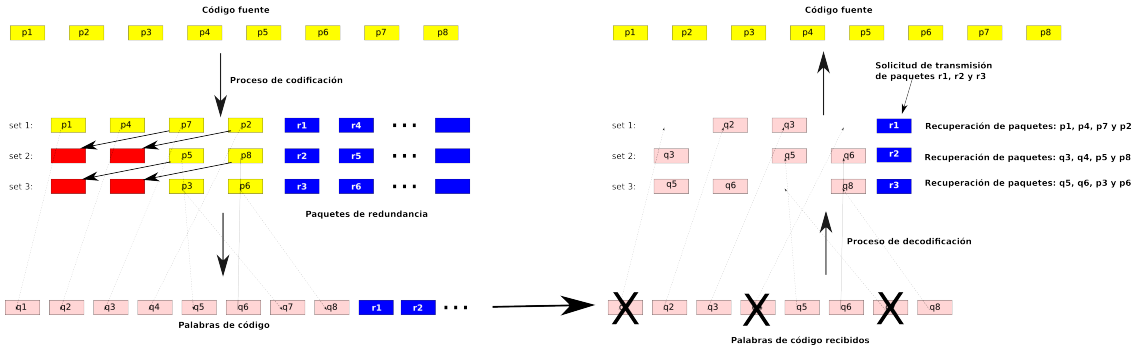


Figura 5.1: Esquema de codificación y decodificación del método FHARQ.

Cuando se envían los paquetes a través del Internet, estos paquetes llegan al recibidor con un retardo alto o bien algunos se pierden en el camino y los decodificadores trabajan de acuerdo al algoritmo que tengan implementado para subsanar este problema. Por ejemplo en los códigos Raptor, esperan cualesquiera $k + \epsilon$ (ϵ decrece conforme la tasa de codificación se acerca a cero) paquetes para recuperar la información,

sin embargo, al momento de codificar llega a ocurrir que el proceso de decodificación no funciono debido a que el grafo se llega a ciclar, es decir, los nodos de mensaje llagan a quedarse con grados mayor que dos. Además, por ser un código de fuente, como se dijo anteriormente, la matriz se va creando de manera aleatoria de tal manera que no será necesariamente invertible, en consecuencia tendrá que esperar por paquetes que le conviertan en una matriz invertible. Son muy eficientes en el proceso de decodificación debido al tipo de grafo con el que se trabaja (grafos bipartitos irregulares) y al tipo de operaciones que realizan (XOR), algunos trabajan con matrices sistemáticos o bien no sistemáticos. La idea de este nuevo esquema es trabajar con grafos bipartitos irregulares que sean MDS ya definidos y bien conocidos tanto por el emisor como por el receptor, o bien usando un sistema Reed-Solomon.

§ 5.1 Motivación

¿Cuál es la idea? la idea principal es obtener un esquema de **codificación jerárquica**, una codificación donde paquetes faltantes puedan ser recuperados a partir de otros conjuntos que estén organizados de alguna manera jerárquica. Como ejemplo pensemos en un fractal cuya estructura básica fragmentada o irregular, se repite a diferentes escalas. Como ejemplo tenemos al *conjunto de Cantor* (ver figura 5.2), el primer conjunto es el intervalo $[0, 1]$ (k paquetes fuente), luego cada nivel del conjunto de Cantor es subdividido en 3 partes descartando el segmento central (estos representan ciertos paquetes), así que podemos considerar sus niveles de particionado como conjuntos que queramos proteger.

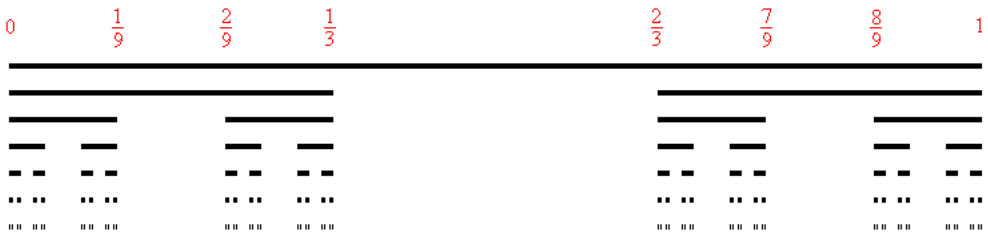


Figura 5.2: El conjunto de Cantor está en biyección con el segmento $[0, 1]$.

Aunque la idea de protección jerárquica mostrada anteriormente es más general, en esta tesis se trabajó con una protección jerárquica cuyos conjuntos tienen el mismo tamaño. El proceso de codificación en cada conjunto es el mismo que se realiza con los sistemas FEC existentes: se tienen k paquetes fuente y un codificador para obtener los paquetes de redundancia. Así que explicaremos una fase antes, la creación de los conjuntos.

5.1.1 Sistema jerárquico

En los canales de transmisión, como lo es el Internet, las pérdidas de paquetes ocurren en ráfagas (*burst*), debido a ello debemos tener un esquema robusto a este tipo de pérdida. Para combatir este problema, necesitaremos que los conjuntos tengan paquetes no continuos, para conseguir este objetivo, procederemos a desordenar los k paquetes fuente y proceder a construir los conjuntos.

Dado k paquetes fuente $S = \{1, 2, 3, ..., k-1, k\}$ ordenado, desordenaremos a S por medio de una distancia d,

$$S = \{1, 1+d, 1+2d, 1+3d, ..., 2, 2+d, 2+2d, 2+3d, ..., 3, 3+d, ...\},$$

donde $i + jd \leq k$, $i, j \in \{1, 2, 3, ...\}$.

Ejemplo 5.1.1. Consideremos a $S = \{1,2,3,4,5,6,7,8,9,10\}$ y sea $d = 4$, entonces tenemos

$$S = \{1,5,9,2,6,10,3,7,4,8\},$$

como el conjunto desordenado.

Lo siguiente es definir el tamaño de los conjuntos, en general el tamaño será definido por la distancia elegida d y excepto el primer conjunto todos tendrán $d/2$ paquetes del conjunto anterior más $d/2$ paquetes nuevos.

Ejemplo 5.1.2. Sea $S = \{1,5,9,2,6,10,3,7,4,8\}$ el conjunto desordenado y sea $d = 4$, entonces tenemos

$$\{1, \quad 5, \quad 9, \quad 2\}$$
$$\{9, \quad 2, \quad 6, \quad 10\}$$
$$\{6, \quad 10, \quad 3, \quad 7\}$$
$$\{3, \quad 7, \quad 4, \quad 8\}.$$

Ejemplo 5.1.3. A continuación se muestran diferentes perdidas en ráfaga de longitud a lo más d. Supongamos que los paquetes fuente son enviados de forma secuencial y que cada conjunto está protegido con un paquete de redundancia.

{_, 5, 9, _}	{1, _, 9, _}	{1, _, 9, 2}
{9, _, 6, 10}	{9, _, 6, 10}	{9, 2, _, 10}
{6, 10, _, 7}	{6, 10, _, 7}	{_, 10, _, 7}
{_, 7, _, 8}	{_, 7, _, 8}	{_, 7, _, 8}
a)	b)	c)
{1, _, 9, 2}	{1, _, 9, 2}	{1, 5, _, 2}
{9, 2, _, 10}	{9, 2, _, 10}	{9, 2, _, 10}
{_, 10, 3, _}	{_, 10, 3, _}	{_, 10, 3, _}
{3, _, _, 8}	{3, _, 4, _}	{3, _, 4, _}
d)	e)	f)
{1, 5, _, 2}	{1, 5, _, 2}	{1, 5, _, 2}
{_, 2, 6, _}	{_, 2, 6, _}	{_, 2, 6, _}
{6, _, 3, _}	{6, _, 3, 7}	{6, _, 3, 7}
{3, _, 4, _}	{3, 7, 4, _}	{3, 7, 4, 8}
g)	h)	i)

En cada uno de los incisos se muestran perdidas continuas marcadas con "_"; en

a) $\{1,2,3,4\}$,	d) $\{4,5,6,7\}$,	g) $\{7,8,9,10\}$,
b) $\{2,3,4,5\}$,	e) $\{5,6,7,8\}$,	h) $\{8,9,10\}$,
c) $\{3,4,5,6\}$,	f) $\{6,7,8,9\}$,	i) $\{9,10\}$.

En cada uno de los ejemplos se necesitan al menos k paquetes para iniciar una decodificación, supondremos que cada conjunto solo puede perder un paquete, así que necesitara un paquete de redundancia en caso de perder un paquete. Para el inciso a), de los conjuntos 2 y 3, se observa que son recuperables los paquetes 2 y 3 respectivamente y estos dan lugar a recuperar el paquete 1 y 4 en los respectivos conjuntos.

Ejemplo 5.1.4. Sea $S = \{1, 2, 3, ..., k = 50\}$ y $d = 16$ entonces se tiene los siguientes conjuntos para una tasa de codificación de $r = 2/3$:

Conjuntos jerárquicos

1	17	33	49	16	32	48	15	31	47	14	30	46	13	29	45	51
31	47	14	30	46	13	29	45	12	28	44	11	27	43	10	26	52
12	28	44	11	27	43	10	26	42	9	25	41	8	24	40	7	53
42	9	25	41	8	24	40	7	23	39	6	22	38	5	21	37	54
23	39	6	22	38	5	21	37	4	20	36	3	19	35	2	18	55
4	20	36	3	19	35	2	18	34	50	17	33	49	16	32	48	56
34	50	17	33	49	16	32	48	15	31	47	14	30	46	13	29	57
15	31	47	14	30	46	13	29	45	12	28	44	11	27	43	10	58
45	12	28	44	11	27	43	10	26	42	9	25	41	8	24	40	59
26	42	9	25	41	8	24	40	7	23	39	6	22	38	5	21	60
7	23	39	6	22	38	5	21	37	4	20	36	3	19	35	2	61
37	4	20	36	3	19	35	2	18	34	50	17	33	49	16	32	62
18	34	50	17	33	49	16	32	48	15	31	47	14	30	46	13	63
48	15	31	47	14	30	46	13	29	45	12	28	44	11	27	43	64
29	45	12	28	44	11	27	43	10	26	42	9	25	41	8	24	65
10	26	42	9	25	41	8	24	40	7	23	39	6	22	38	5	66
40	7	23	39	6	22	38	5	21	37	4	20	36	3	19	35	67
21	37	4	20	36	3	19	35	2	18	34	50	17	33	49	16	68
2	18	34	50	17	33	49	16	32	48	15	31	47	14	30	46	69
32	48	15	31	47	14	30	46	13	29	45	12	28	44	11	27	70
13	29	45	12	28	44	11	27	43	10	26	42	9	25	41	8	71
43	10	26	42	9	25	41	8	24	40	7	23	39	6	22	38	72
24	40	7	23	39	6	22	38	5	21	37	4	20	36	3	19	73
5	21	37	4	20	36	3	19	35	2	18	34	50	17	33	49	74
35	2	18	34	50	17	33	49	16	32	48	15	31	47	14	30	75

La última columna representan los paquetes de redundancia, un paquete por cada conjunto, el número de conjuntos, en este caso, quedo determinado por la tasa de codificación (r) o bien por la razón de expansión ($1/r$), aunque esto puede ser manejable ya que se puede usar un sistema FEC existente para generar más de un paquete de redundancia en cada conjunto y así disminuir el número de conjuntos o bien realizar un aumento en la razón de expansión.

§ 5.2 Codificación

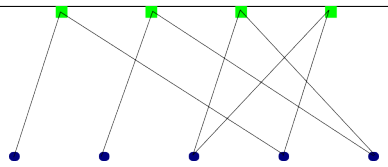
Sin pérdida de generalidad, consideremos 9 paquetes $p_1, p_2, ..., p_9$ de longitud l en cada paquete, ahora bien, desordenemos a estos paquetes de acuerdo a la distancia $d = 3$, entonces tenemos $p_1, p_4, p_7, p_2, p_5, p_8, p_3, p_6, p_9$ y ahora crearemos los conjuntos, en este caso, consideraremos a cada conjunto de tamaño 4 de tal manera que los primeros 2 paquetes serán paquetes codificados del conjunto anterior:

$$\begin{aligned} 1: &\quad \{1,4,7,2\} \\ 2: &\quad \{7,2,5,8\} \\ 3: &\quad \{5,8,3,6\} \\ 4: &\quad \{3,6,9,1\} \end{aligned} \tag{5.2.1}$$

Estos conjuntos serán nuestro principal objetivo a proteger, para ello usaremos una matriz linealmente independiente, con el cual codificaremos a cada conjunto y tanto el codificador como el decodificador sabrán cual es la matriz, sin tener que enviarla.

En este proceso es de principal importancia usar algún método FEC existente: por ejemplo usar códigos LDPC/LDGM o Reed-Solomon. Un punto importante a tener en cuenta, el esquema es más robusto bajo un método MDS debido a que los paquetes de redundancia solo sirven al conjunto respectivo y no a los demás, así que Reed-Solomon es una muy buena elección, otra es que la dimensionalidad decrece con respecto a los k paquetes y esto hace accesible la utilización de los Reed-Solomon ya que están diseñados para bloques de longitud corta. En esta tesis se usara una **simple matriz de chequeo de paridad** el cual más adelante comentaremos a detalle.

A continuación mostramos una matriz de chequeo de paridad, el cual usaremos para proteger los paquetes de cada conjunto:

$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$	
Matriz G	Grafo bipartito irregular respectivo

De acuerdo a la matriz generadora G, procederemos a codificar cada conjunto; El primer conjunto es $S_1 = \{p_1, p_4, p_7, p_2\}$, luego su codificación es

$$\begin{aligned} S_1G &= \{p_1, p_4, p_7 + p_2, p_1 + p_2, p_4 + p_7\} \\ &= \{q_1, q_4, q_7, q_2, q_{r_1}\}. \end{aligned} \tag{5.2.2}$$

A partir del segundo conjunto, son creados usando parte del conjunto anterior y paquetes fuente aun no codificados: el conjunto 2 está compuesto por los índices $\{7, 2, 5, 8\}$, luego los primeros dos índices corresponden a índices del conjunto anterior, mientras que los últimos índices no pertenecen a índices que ya hayan sido codificados, así que, $S_2 = \{q_7, q_2, p_5, p_8\}$, luego su codificación es:

$$\begin{aligned} S_2G &= \{q_7, q_2, p_5 + p_8, q_7 + p_8, q_2 + p_5\} \\ &= \{q'_7, q'_2, q_5, q_8, q_{r_2}\}. \end{aligned} \tag{5.2.3}$$

El tercer conjunto está compuesto por los índices $\{5, 8, 3, 6\}$, luego $S_3 = \{q_5, q_8, p_3, p_6\}$, así que su codificación es:

$$\begin{aligned} S_3G &= \{q_5, q_8, p_3 + p_6, q_5 + p_6, q_8 + p_3\} \\ &= \{q'_5, q'_8, q_3, q_6, q_{r_3}\}. \end{aligned} \tag{5.2.4}$$

El cuarto conjunto está compuesto por los índices $\{3, 6, 9, 1\}$, luego $S_4 = \{q_3, q_6, p_9, q_1\}$, así que su codificación es:

$$\begin{aligned} S_4G &= \{q_3, q_6, p_9 + q_1, q_3 + q_1, q_6 + p_9\} \\ &= \{q'_3, q'_6, q_9, q'_1, q_{r_4}\}. \end{aligned} \tag{5.2.5}$$

Resumiendo obtenemos los siguientes conjuntos después de la codificación:

$$\begin{aligned} C_1 : &\quad \{q_1, q_4, q_7, q_2, q_{r_1}\} \\ C_2 : &\quad \{q'_7, q'_2, q_5, q_8, q_{r_2}\} \\ C_3 : &\quad \{q'_5, q'_8, q_3, q_6, q_{r_3}\} \\ C_4 : &\quad \{q'_3, q'_6, q_9, q'_1, q_{r_4}\}. \end{aligned} \tag{5.2.6}$$

Observemos que por cada conjunto tenemos un paquete de redundancia y que los paquetes que nos interesan son aquellos que se codificaron directamente de los paquetes originales; al final de la protección de los conjuntos, reordenamos los paquetes codificados.

$$msg\_cod = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{r_1}, q_{r_2}, q_{r_3}, q_{r_4}\} \tag{5.2.7}$$

Esta forma de codificación es debido a que el primer objetivo es que se puedan compartir paquetes recuperados en diferentes conjuntos y el segundo objetivo que se puedan usar sistemas FEC tanto sistemáticos como no sistemáticos. La codificación explicada anteriormente es para sistemas FEC no sistemáticos. La principal ventaja de este nuevo esquema es que se comporta más robusto con un sistema sistemático simple que con un no sistemático. Por ultimo observemos que si la forma de codificar es sistemática, entonces siempre se tendrá que las q_i y q'_i serán algún p_i donde $i \in \{1, ..., k\}$.

Resumiendo el proceso de codificación, se presenta el siguiente algoritmo.

Algorithm 1 Codificación

Require: $r, k, d, G, msg = \{p_1, p_2, ..., p_k\}$
$matPkg = crearConjuntoJerarquico(k, d, r)$
$[M, N] = size(matPkg)$
$S_1 = \{p_{matPkg(1,1)}, p_{matPkg(1,2)}, \cdots, p_{matPkg(1,N)}\}$
Codificar el conjunto S_i usando G y obtener:
$Q_1 = \{q_{matPkg(1,1)}, q_{matPkg(1,2)}, \cdots, q_{matPkg(1,N)}, q_{1,r_{1,N+1}}, q_{1,r_{1,N+2}}, \cdots\}$
for $i = 1 \rightarrow M$ **do**
  $S_i = \{q_{matPkg(i,1)}, q_{matPkg(i,2)}, \cdots, q_{matPkg(i,N)}\}$
  for $j = 1 \rightarrow N$ **do**
    if $q_{matPkg(i,j)} \notin Q_{i-1}$ **then**
      $q_{matPkg(i,j)} = p_{matPkg(i,j)}$
    end if
  end for
  Codificar el conjunto S_i usando G y obtener
  $Q_i = Q_{i-1} \cup \{q_{matPkg(i,1)}, q_{matPkg(i,2)}, \cdots, q_{matPkg(i,M)}, q_{i,r_{i,M+1}}, q_{i,r_{i,M+2}}, \cdots\}$
end for

El orden de la transmisión de los paquetes es enviar los primeros q_i paquetes codificados hasta que el receptor reciba k paquetes y si no hay perdida, recuperar toda la información sin ningún problema. En caso de perdidas, se procede a recuperar todos los paquetes que están al alcance y luego, si es necesario, pedir retransmisión de los paquetes necesarios para proceder a la recuperación de la información completa.

§ 5.3 Decodificación

Buscamos realizar el mínimo número de retransmisiones cuando no podamos recuperar algún conjunto, así como también que las retransmisiones sean solo de paquetes redundantes, esto con el objetivo de no saturar el buffer del transmisor y también para que sea un esquema robusto en la transmisión de *streaming*. Para el ejemplo en el que estamos, consideraremos retransmisiones no solo de paquetes de redundancia, sino también de los propios paquetes codificados.

Regresando al ejemplo, si en el segundo conjunto (ver conjuntos 5.2.6) perdemos los paquetes q_5 y q_8 no podremos recuperarlo de cualquiera de los otros conjuntos, por lo tanto pediremos que retransmitan el paquete q_5 o bien q_8, recordemos que para este ejemplo solo podemos perder un paquete en cada conjunto, así que supondremos que ya contamos con su respectivo paquete de redundancia (q_{r_2}).

Como podemos observar, usamos la matriz de paridad G y la estructura jerárquica de los conjuntos para codificar y recuperar información cuando sea posible. La manera en que decodificaremos ya fue explicada anteriormente (ver proceso de decodificación página 47), sin embargo mostraremos como modificar la matriz de paridad cuando tengamos paquetes auxiliares de otros conjuntos.

Es importante hacer notar que los paquetes recibidos serán usados para la recuperación en diferentes conjuntos, sin embargo para que la decodificación tenga éxito, los paquetes que se utilicen deben ser aquellos que se usaron en la codificación y procurando modificar en la matriz de paridad G, su respectiva columna como una columna canónica (tiene un 1 en la posición i, donde i es la posición del paquete recuperado en el respectivo conjunto).

5.3.1 Ejemplo de proceso de decodificación sin pérdidas de paquetes

Ahora veremos el proceso de decodificación considerando que se recibieron los primeros k paquetes de *msg_cod* (ver 5.2.7). En primera instancia tenemos

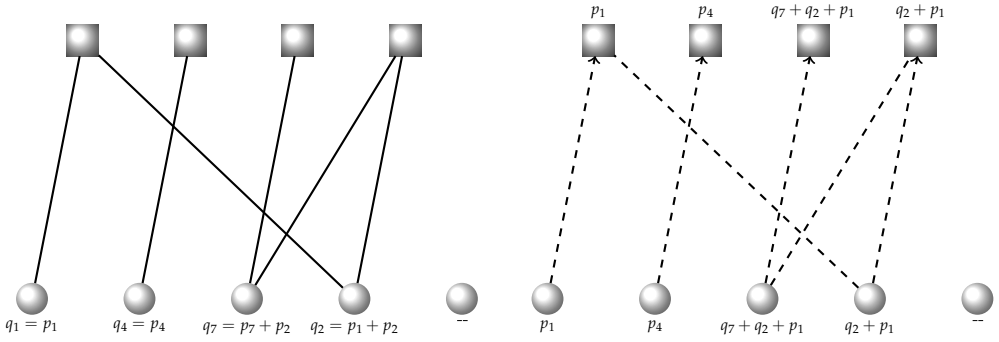
$$\begin{aligned} C_1 &: \{\ q_1,\ q_4,\ q_7,\ q_2,\ \_\} \\ C_2 &: \{\ \_,\ \_,\ q_5,\ q_8,\ \_\} \\ C_3 &: \{\ \_,\ \_,\ q_3,\ q_6,\ \_\} \\ C_4 &: \{\ \_,\ \_,\ q_9,\ \_,\ \_\}. \end{aligned} \tag{5.3.1}$$

Debido a que conocemos la estructura de los conjuntos jerárquicos y la estructura de la matriz de chequeo de paridad, para este ejemplo se tiene que:

$$\begin{aligned} C_1 &: \{ \quad q_1, \quad q_4, \quad q_7, \quad q_2, \quad \_\} \\ C_2 &: \{ \quad \mathbf{q_7}, \quad \mathbf{q_2}, \quad q_5, \quad q_8, \quad \_\} \\ C_3 &: \{ \quad \mathbf{q_5}, \quad \mathbf{q_8}, \quad q_3, \quad q_6, \quad \_\} \\ C_4 &: \{ \quad \mathbf{q_3}, \quad \mathbf{q_6}, \quad q_9, \quad \_, \quad \_\}. \end{aligned} \tag{5.3.2}$$

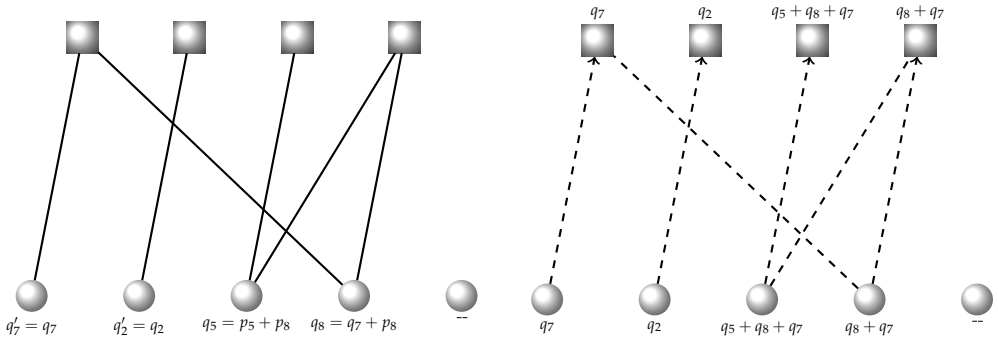
Sin embargo para el conjunto C_4, en el penúltimo paquete se utilizó q_1, pero el resultado de la codificación para ese lugar resulto ser $q_1 + q_3$, diferencia que se tiene con los dos primeros paquetes de cada conjunto, aunque este no es un obstáculo para detener la decodificación, ya que se conoce cuál es el paquete en cuestión, bastara con retirar el enlace de q_3 hacia este nodo, lo que se reduce a dejar la respectiva columna de la matriz G en una columna canónica.

Recuperación del primer conjunto:



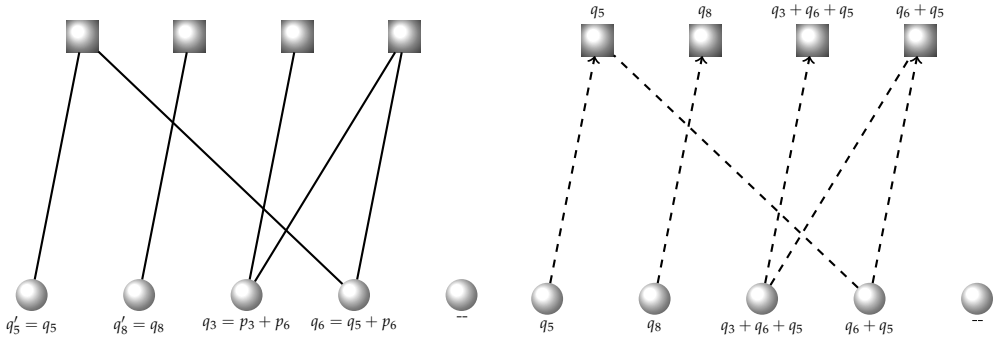
Cuadro 5.1: Estado inicial y final de la decodificación. Para finalizar este proceso, observemos que $q_7 + q_2 + p_1 = p_7 + p_2 + p_1 + p_2 + p_1 = p_7$ y $q_2 + p_1 = p_1 + p_2 + p_1 = p_2$, por lo tanto hemos recuperado el conjunto $\{p_1, p_2, p_4, p_7\}$.

Recuperación del segundo conjunto:



Cuadro 5.2: Estado inicial y final de la decodificación. Para finalizar este proceso, observemos que $q_5 + q_8 + q_7 = p_5 + p_8 + q_7 + p_8 + q_7 = p_5$ y $q_8 + q_7 = q_7 + p_8 + q_7 = p_8$, por lo tanto hemos recuperado el conjunto $\{q_7, q_2, p_5, p_8\}$.

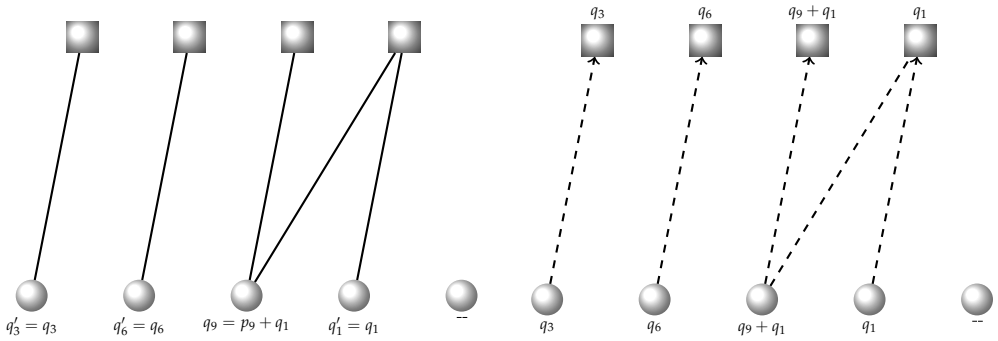
Recuperación del tercer conjunto:



Cuadro 5.3: Estado inicial y final de la decodificación. Para finalizar este proceso, observemos que $q_3 + q_6 + q_5 = p_3 + p_6 + q_5 + p_6 + q_5 = p_3$ y $q_6 + q_5 = q_5 + p_6 + q_5 = p_6$, por lo tanto hemos recuperado el conjunto $\{q_5, q_8, p_3, p_6\}$.

Recuperación del cuarto conjunto:

En esta fase, el cuarto paquete q_1' no es conocido, debido a que no se envió, sin embargo por la fase de codificación, se sabe que se utilizó el paquete q_1 y dicho paquete si fue recibido, y puesto que el grado de este nodo es mayor que uno, modificaremos la respectiva columna de la matriz de paridad (cuarta columna) para no sumarle nada a esta posición, debido a que tenemos el paquete recuperado. En las fases anteriores no se realizó este procedimiento debido a que el grado de respectivos nodos era uno.



Cuadro 5.4: Estado inicial y final de la decodificación. Para finalizar este proceso, observemos que $q_9 + q_1 = p_9 + q_1 + q_1 = p_9$, por lo tanto hemos recuperado el conjunto $\{q_3, q_6, p_9, q_1\}$.

En cada conjunto se recuperan paquetes codificados y paquetes fuente, situación totalmente diferente si se usa un código sistemático, ya que en cada conjunto siempre se estarán recuperando paquetes fuente, he aquí la importancia y diferencia entre trabajar con un código sistemático (por ejemplo Reed-Solomon) o un no sistemático (LDPC/LDGM).

5.3.2 Ejemplo de proceso de decodificación con pérdidas de paquetes

Supongamos que perdemos los paquetes q_1, q_3, q_5, q_8 y q_9, entonces procedemos a buscar en que conjuntos podemos recuperar información:

Conjuntos	pkg lost by set
$\{$--, $q_4, q_7, q_2, q_{r_1}\}$	1
$\{q'_7, q'_2$, --, --, $q_{r_2}\}$	2
$\{$--, --, --, $q_6, q_{r_3}\}$	3
$\{$--, q'_6, q_9, --, $q_{r_4}\}$	3

Recuperaremos información directamente de los conjuntos donde falte uno o ningún paquete, en este caso, podemos recuperar información del primer conjunto, luego ac-

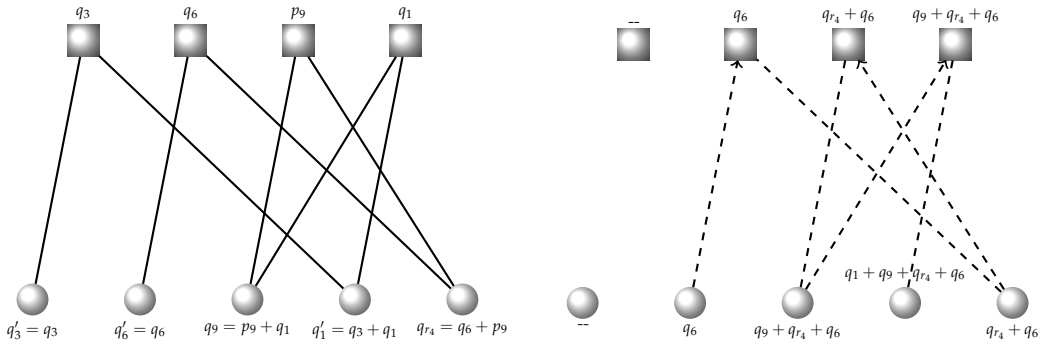
tualizaremos nuestro número de paquetes perdidos por conjunto, aplicando el procedimiento de decodificación ya antes mencionado, obtenemos que $pkg\_decod = \{p_1, p_2, p_4, p_7\}$ y nuestros conjuntos actualizados son:

Conjuntos	pkg lost by set
$\{q_1, q_4, q_7, q_2, q_{r_1}\}$	-1
$\{q_7, q_2, \_\_, \_\_, q_{r_2}\}$	2
$\{\_\_, \_\_, \_\_, q_6, q_{r_3}\}$	3
$\{\_\_, q_6, q_9, \_\_, q_{r_4}\}$	3

Ahora bien, el primer conjunto ya no nos sirve, debido que ya obtuvimos información del (por ello le colocamos un -1); Sin embargo, los demás conjuntos no se modificaron, esto provocara una retransmisión de paquetes, pero a diferencia de otros métodos, este solo necesita pedir 2 paquete del conjunto $\{q_5, q_8, q_3\}$ (o en su caso, 2 paquetes de redundancia que no se han enviado aun para este conjunto), si solicitemos la retransmisión de los paquetes p'_5 y p'_3, entonces tendríamos:

Conjuntos	pkg lost by set
$\{q_1, q_4, q_7, q_2, p_{r_1}\}$	-1
$\{q_7, q_2, q_5, \_\_, p_{r_2}\}$	1
$\{q_5, \_\_, q_3, q_6, p_{r_3}\}$	1
$\{q_3, q_6, q_9, q_1, p_{r_4}\}$	0

Observar que bajo esta estructura, no es posible utilizar el paquete q_1 en el último conjunto para recuperar el paquete q_3, esto debido a que el nodo variable correspondiente a q_3 depende del nodo variable $q'_1 = p_1 + q_3$:



Cuadro 5.5: Proceso de codificación y actual proceso utilizando el nodo variable correspondiente al paquete q_1 auxiliar de la decodificación. Para finalizar, observemos que $q_1 + q_9 + q_{r_4} + q_6 = q_1 + p_9 + q_1 + q_6 + p_9 + q_6 = 0$, por lo tanto el método de decodificación falla en estas circunstancias.

Procedemos a recuperar información de conjunto 2:

$$pkg\_decod = \{p_1, p_2, p_4, p_5, p_7, p_8\}.$$

Procedemos a recuperar información de conjunto 3:

$$pkg\_decod = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}.$$

Notar que en este último conjunto, recuperamos los paquetes q_5, q_8, p_3 y p_6 a diferencia de la información recuperada en el conjunto 2 (q_7, q_2, p_5 y p_8).

Para el ultimo conjunto, nos interesa recuperar el paquete p_9, pero observemos dos cosas; primero que usaremos a q_1, recordando que cuando se codifico se obtuvo q'_1 paquete que no fue enviado (aunque podría considerarse como paquete de redundancia y ser enviado).

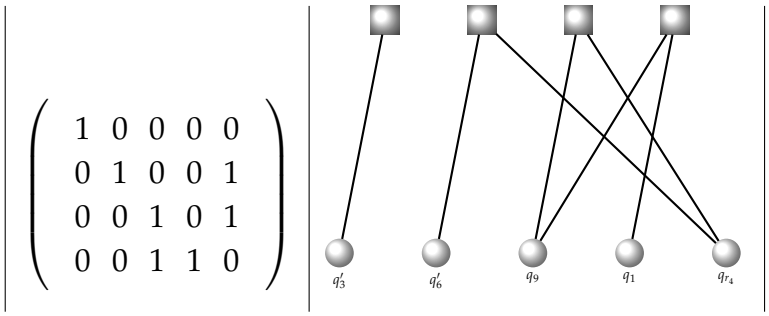


Figura 5.3: Matriz de paridad modificada y Grafo respectivo.

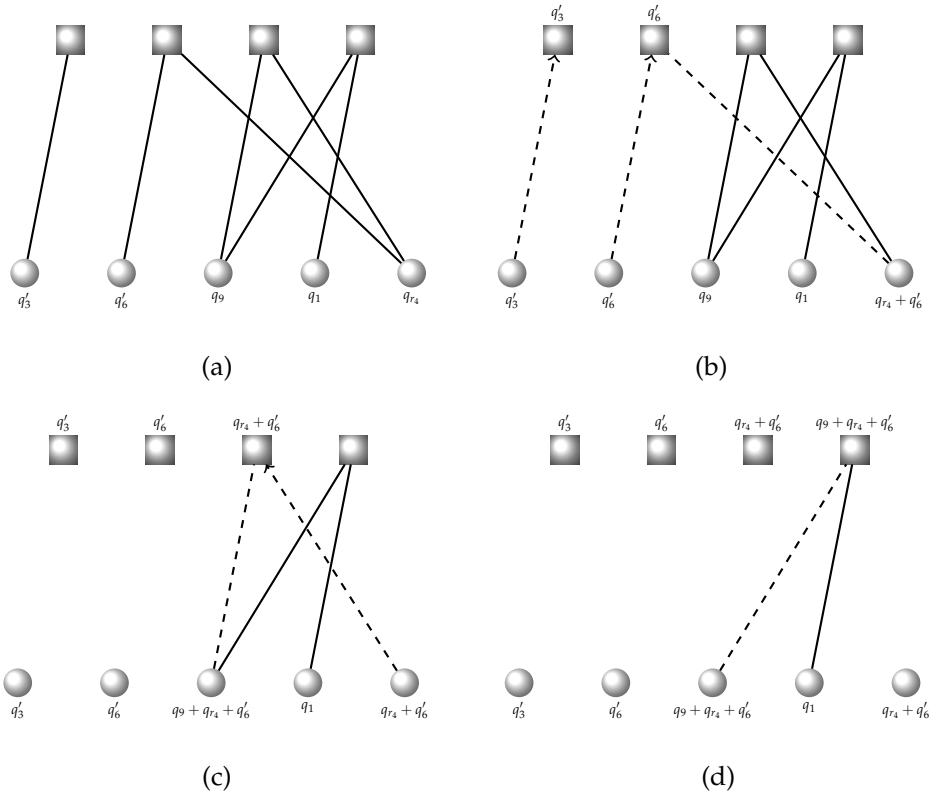


Figura 5.4: Proceso de recuperación usando el último conjunto.

Realicemos las últimas cuentas para corroborar que recuperamos la información del paquete original: $q_{r_4} + q'_6 = (q_6 + q_9) + q_6 = p_9$. Por último, notemos que $q_{r_4} + q'_6 + q_9 =$

$p_1 = q_1$. Por lo tanto:

$$pkg\_decod = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}.$$

Es importante tener una matriz de chequeo de paridad muy bien estructurada para no pasar por problemas como el que se explicó anteriormente. Un esquema más robusto y simple es el **simple chequeo de paridad** cuya matriz es:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

esta respectiva matriz y con la que se trabajó en el ejemplo generan el mismo espacio, sin embargo una es sistemática mientras que la otra es no sistemática respectivamente. Con la matriz sistemática nos evitamos problemas que anteriormente se mencionaron, aunque es importante tener en cuenta que este tipo de estructura solo nos generara un paquete de redundancia en cada conjunto jerárquico. Códigos FEC sistemáticos existentes como el Reed-Solomon o LDGM Sataircase/Triangle son buenas propuestas para generar más de un paquete de redundancia en cada conjunto jerárquico.

6

Resultados

Resumen

A continuación se realizaron varias pruebas comparando los resultados con métodos y pruebas existentes. La principal métrica que se uso es inef_ratio el cual nos mide la cantidad extra de paquetes necesarios para una decodificación exitosa. Nos guiamos principalmente de tres pruebas que se han realizado en los últimos años (2005 [57], 2008 [59] y 2009 [60]) y en los cuales se trabajaron para objetos grandes. La programación del algoritmo fue realizado en MATLAB 2011a, las pruebas se realizaron en una maquina con Ubuntu 10.04, 2 Gb de RAM, procesador Intel Pentium Dual core de 1.73 GHz.

Realizaremos dos tipos de retransmisiones para el esquema propuesto: El primero será el número de retransmisiones realizadas a partir del momento en recibir el último paquete fuente y el segundo será el número de retransmisiones realizadas a partir de recibir cualesquiera k paquetes. Siendo este último más ineficiente en el aspecto de la decodificación, debido a que los paquetes de redundancia no protegen a cualesquier paquetes.

Si se considera una tasa de perdida grande y el objeto es grande, se produciría un incremento notable en el número de retransmisiones, llegando a saturar el buffer del transmisor. Es por ello que se analizaron 3 pruebas considerando alguno de los dos

tipos de retransmisión:

1. Modelos de transmisión: en estas pruebas se consideró el tipo dos del esquema propuesto, esto con el objetivo de analizar la tendencia de retransmisiones donde existen diferentes tasas de pérdida de paquetes.

2. Tasa de código fijo y diferente tamaños de objeto: en esta prueba se utilizó el tipo uno del esquema propuesto, esto debido a que se desea dejar en claro que no importa el tamaño del objeto las retransmisiones necesarias serán óptimas para la recuperación de la información.

3. Tamaño de objeto fijo y diferentes tasas de codificación: Para estas pruebas se consideró el tipo dos del esquema propuesto, con el objetivo de analizar la tendencia del número de retransmisiones necesarias bajo el supuesto de tener más conjuntos jerárquicos.

6.0.3 Modelo de Gilbert

El modelo de perdida usado en [57] es un proceso de Markov de dos estados y fue desarrollado por Gilbert (*Modelo simplificado de Gilbert* ver [61]), y generalizado por Elliott (ver [62]). El modelo caracteriza una secuencia de errores generados por un canal de transmisión de datos. Como se puede observar en la figura 6.1, el modelo está compuesto por un estado *"good" (G)*, donde no ocurren perdida de paquetes y el cual tiene probabilidad P_G y un estado *"bad" (B)*, donde los paquetes se pierden, el cual tiene probabilidad P_B. p indica la probabilidad de pasar del estado G al estado B y q la probabilidad de pasar del estado B al estado G. La probabilidad de tener un error en

un canal de Gilbert está dada por la expresión:

$$p_{global} = P_G \left(\frac{q}{p+q} \right) + P_B \left(\frac{p}{p+q} \right), \tag{6.0.1}$$

Modelo de Gilbert simplificado

Una simplificación muy común del modelo de Gilbert, es tomar $P_G = 0$ y $P_B = 1$. Por lo tanto, teniendo p y q se puede calcular la probabilidad global de pérdida de paquete dada por:

$$p_{global} = \frac{p}{p+q}, \tag{6.0.2}$$

esta probabilidad está representada en la gráfica 6.2.

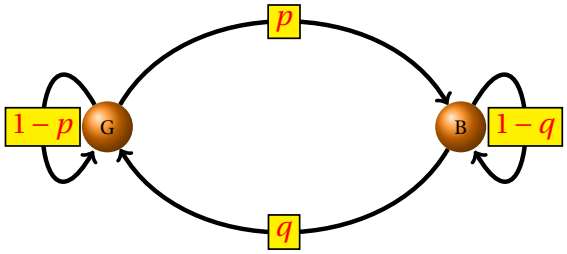


Figura 6.1: Modelo de pérdida de Markov con 2 estados.

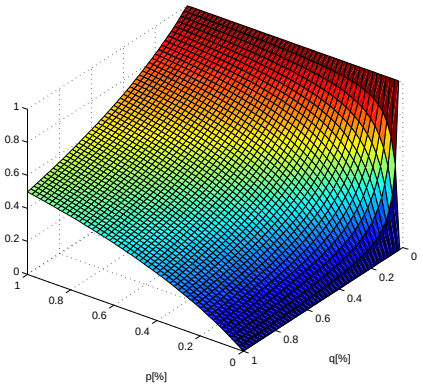


Figura 6.2: Probabilidad de error global.

6.0.4 Modelos de transmisión

En [57] se analizan tres tipos de códigos FEC: LDGM Staircase, LDGM Triangle y Reed-Solomon, los dos primeros considerados códigos de bloque grande, mientras que el Reed-Solomon es considerado de bloque pequeño debido a las restricciones de los tiempos de codificación/decodificación con respecto al campo de Galois. En este trabajo se realizaron pruebas de rendimiento a los códigos mencionados de acuerdo a la programación con respecto en cómo se envían los paquetes a través de un canal.

Se realizaron 100 simulaciones con respecto al promedio de ineficiencia de codificación (*inef_ratio*), usando una razón de expansión FEC de 1,5 (equivalente a una tasa de codificación de $2/3$). El objeto está compuesto de $20,000$ paquetes, cada paquete de 1024 bytes. Se realizó un estudio del impacto del canal variando las probabilidades p y q en el intervalo $[0, 1]$: se consideraron 14×14 respectivamente.

Se analizaron los primeros 4 modelos de los analizados en [57]:

Modelo 1: Enviar secuencialmente los paquetes fuente y luego secuencialmente los paquetes de redundancia.

Modelo 2: Enviar secuencialmente los paquetes fuente y luego aleatoriamente los paquetes de redundancia.

Modelo 3: Enviar secuencialmente los paquetes de redundancia y luego aleatoriamente los paquetes fuente.

Modelo 4: Enviar todo aleatoriamente.

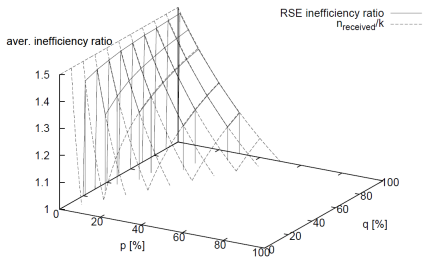
Modelo 5: Enviar los paquetes intercalados (solo para el Reed-Solomon).

Modelo 6: Enviar aleatoriamente unos cuantos paquetes fuente y luego enviar todos los paquetes de redundancia.

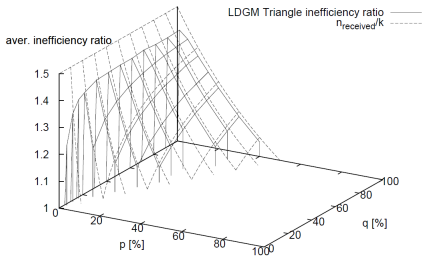
Es importante aclarar que a diferencia de los métodos FEC analizados en la literatura, los cuales trabajan sobre canales de una vía. El esquema propuesto trabaja en un canal de dos vías: el esquema detiene la transmisión cuando llegan k paquetes y luego realiza retransmisiones de paquetes específicos para lograr la decodificación exitosa, por lo tanto, los paquetes excedentes son las retransmisiones ejecutadas por el algoritmo. Por ultimo las retransmisiones se suponen libres de pérdidas.

Modelo 1

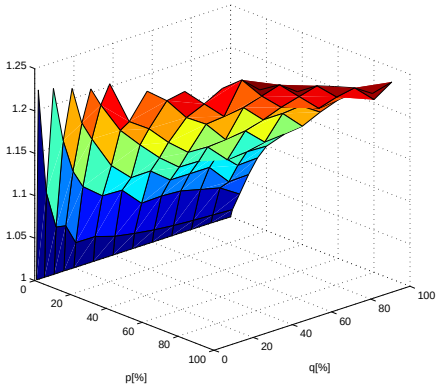
En este modelo, primero se envían los paquetes fuente de manera secuencial y luego los paquetes de redundancia, de manera secuencial también. En un estado sin perdida ($p = 0$) la ineficiencia de decodificación es 1 para todos los códigos. Sin embargo bajo un estado de perdida los códigos FEC logran decodificar la información al final de la transmisión. Contrario a lo que sucede con el esquema propuesto, donde la ineficiencia de decodificación crece conforme la probabilidad de perdida aumenta, esto se debe a la acumulación de paquetes que no se usan o dejan de servir para la recuperación de un conjunto.



(a) RSE (extraído de [57])



(b) LDGM Triangle (extraído de [57])

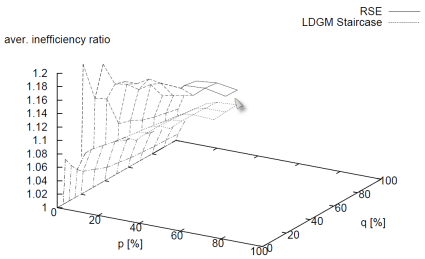


(c) Esquema propuesto

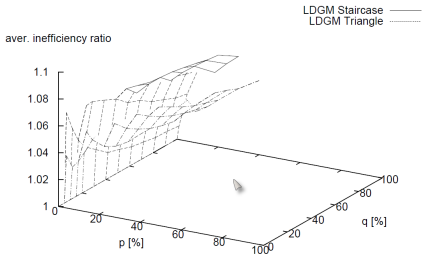
Figura 6.3: Análisis del modelo 1.

Modelo 2

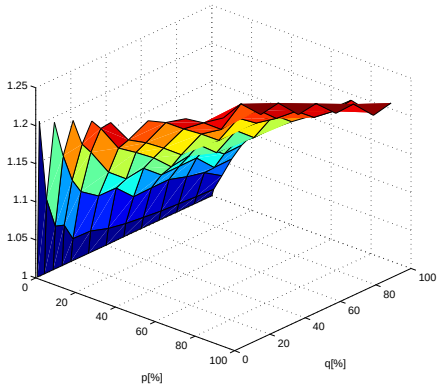
En este modelo se envían los paquetes fuente secuencialmente y luego se envían los paquetes de redundancia en un orden aleatorio. El esquema propuesto muestra una ligera mejoría con respecto al modelo 1. Sin embargo los códigos FEC muestran una notable mejoría en comparación al modelo anterior.



(a) RSE y LDGM Staircase (extraído de [57])



(b) LDGM Triangle y LDGM Staircase (extraído de [57])

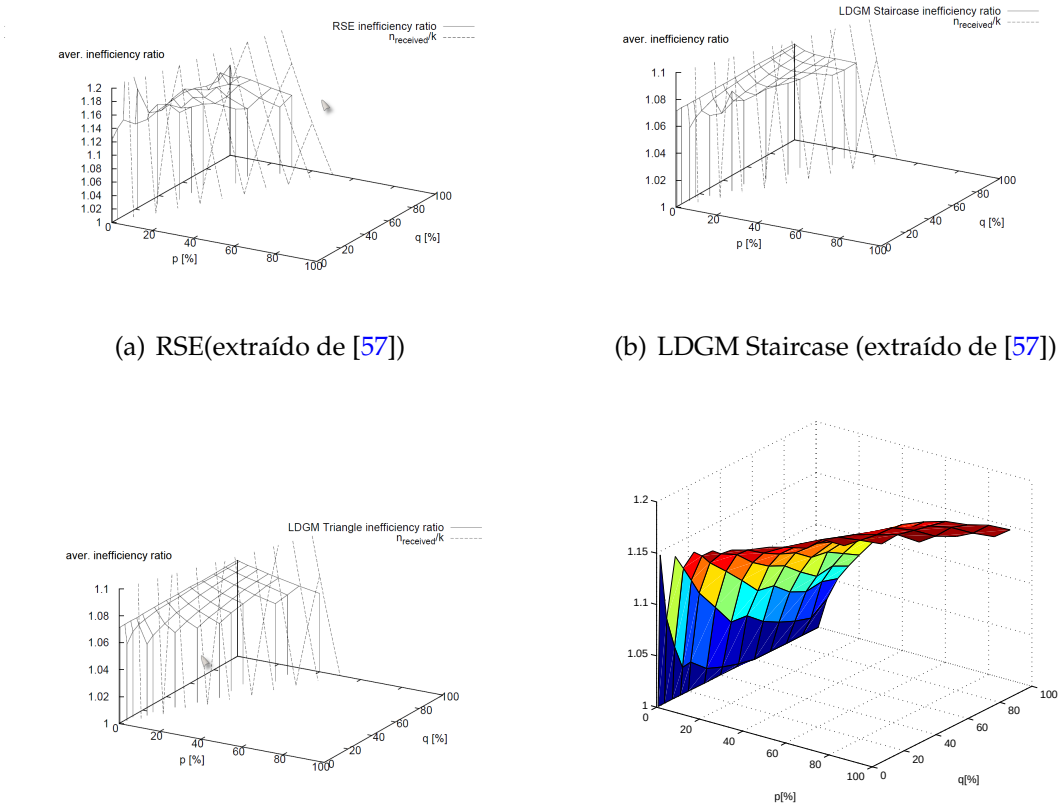


(c) Esquema propuesto

Figura 6.4: Análisis del modelo 2.

Modelo 3

Ahora se considera enviar primero los paquetes de paridad de manera secuencial y luego enviar los paquetes fuente de manera aleatoria. En este modelo se logra una mejoría en el esquema propuesto, y a pesar que no sigue siendo mejor que los esquemas FEC, se presenta una ventaja cuando no hay perdida, puesto que bajo este esquema los métodos FEC necesitan más de k paquetes en un estado sin perdida.



(a) RSE(extraído de [57])

(b) LDGM Staircase (extraído de [57])

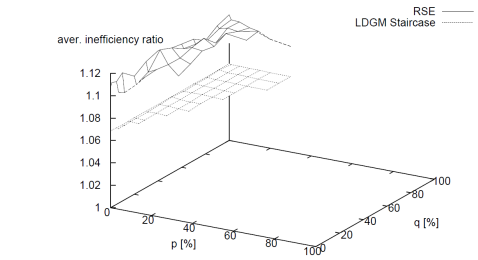
(c) LDGM Triangle (extraído de [57])

(d) Esquema propuesto

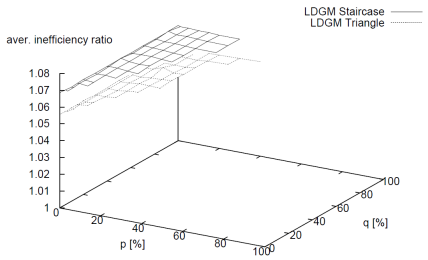
Figura 6.5: Análisis del modelo 3.

Modelo 4

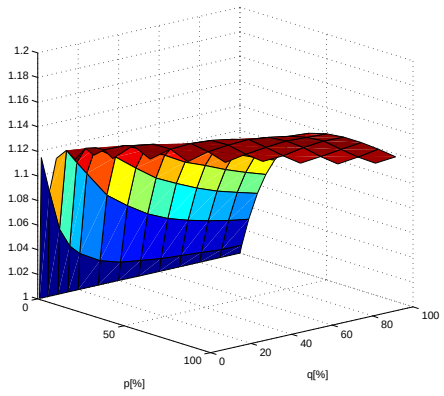
En este modelo, todos los paquetes se envían aleatoriamente y como se puede observar el esquema propuesto presenta la mejor ineficiencia de decodificación. Aunque los métodos FEC se mantienen en un valor casi fijo de la ineficiencia, el método va incrementando con forme crece la probabilidad de perdida y se mantiene en una ineficiencia de $\approx 1{,}12$.



(a) RSE y LDGM Staircase (extraído de [57])



(b) LDGM Staircase y LDGM Triangle(extraído de [57])



(c) Esquema propuesto

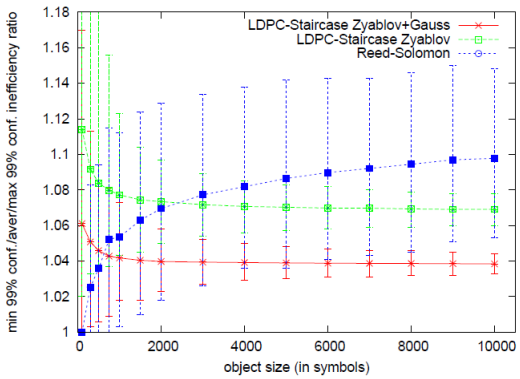
Figura 6.6: Análisis del modelo 4.

La principal ventaja que presenta el esquema propuesto es que va aumentando las retransmisiones con respecto a la probabilidad de perdida. La principal desventaja es que trabaja en un canal de dos vías a diferencia de los métodos FEC analizados en [57].

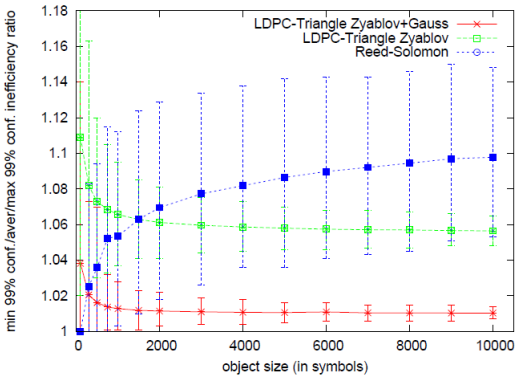
6.0.5 Tasa de código fijo y diferentes tamaños de objeto

El test realizado en [59] tiene los siguientes parámetros: la tasa de codificación es fijado a 2/3 y el tamaño de objeto es incrementado gradualmente hasta un tamaño de

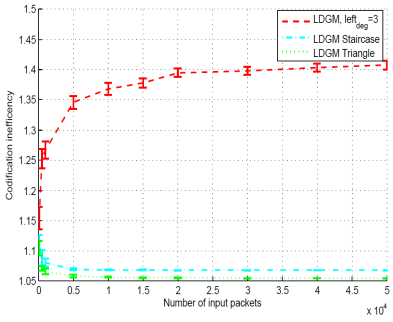
10,000 paquetes, el objetivo es evaluar los desempeños asintóticos:



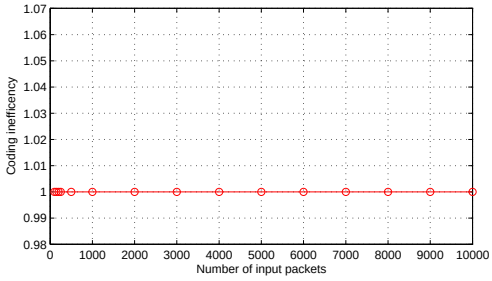
(a) LDPC-staircase (extraído de [59])



(b) LDPC-triangle (extraído de [59])



(c) LDGM, LDGM Staircase and LDGM Triangle (extraído de [60])



(d) Esquema propuesto

Figura 6.7: Comparación de diferentes esquemas de protección.

El esquema propuesto presenta una ineficiencia óptima, el cual no es afectado conforme la cantidad de paquetes aumenta. Sin embargo. Este nivel de ineficiencia de debe a que el esquema propuesto inicia las solicitudes de paquetes redundantes conforme se recibe el último paquete. En la tabla 6.1, se pueden observar porcentajes de paquetes redundantes necesarios para una decodificación exitosa:

Object size	100	500	1,000	2,000	5,000	10,000
LDPC-Staircase Gauss	6.10 %	4.59 %	4.17 %	3.97 %	3.89 %	3.84 %
LDPC-Triangle Gauss	3.82 %	1.62 %	1.28 %	1.14 %	1.06 %	1.03 %
Reed-Solomon	0 %	3.58 %	5.35 %	6.95 %	8.65 %	9.79 %
Esquema propuesto	0 %	0 %	0 %	0 %	0 %	0 %

Cuadro 6.1: Promedio de razón de ineficiencia en función del tamaño del objeto para una tasa de codificación de $rate = 2/3$ (extraído de [59]).

6.0.6 Tamaño de objeto fijo y diferentes tasas de codificación

El tamaño del objeto fue seleccionado usando $k = 10,000$ paquetes ($\approx$ 10 MBytes) y la tasa de codificación se redujo gradualmente. El objetivo es evaluar el comportamiento de los códigos a medida que se acercan a los códigos rateless.

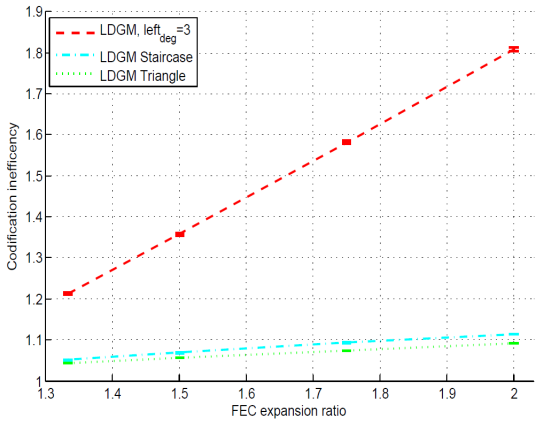
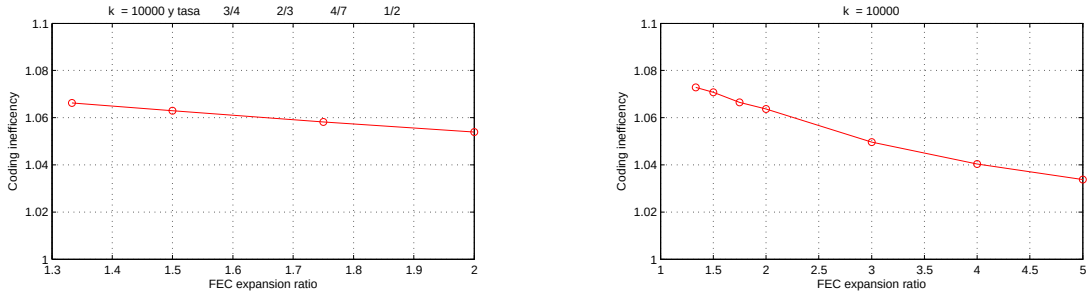


Figura 6.8: Coding inefficiency of LDGM, LDGM Staircase and LDGM Triangle as a function of FEC expansion ratio, when $k = 10000$ (extraído de [60]).

La figura 6.8 muestra la ineficiencia de decodificación de los códigos LDGM para 4 tasas de codificación, los cuales se muestra como crece conforme la tasa de codificación decrece, por otro lado, en las figuras 6.9(a) y 6.9(b) se pueden observar que la ineficien-

cia de decodificación del esquema propuesto se comporta de manera contraria, puesto que decrece conforme la tasa de decodificación decrece, es decir, el esquema propuesto tiene un comportamiento como un código rateless. Este es una importante ventaja con respecto a los códigos LDGM.



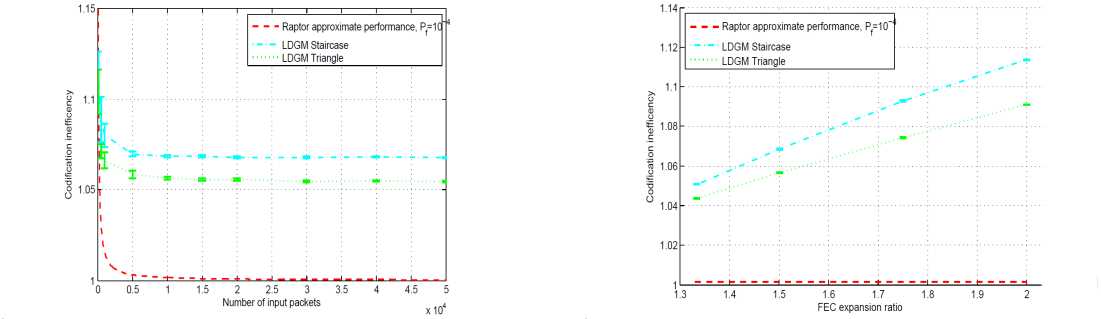
(a) Esquema propuesto usando 4 tasas de codificación

(b) Esquema propuesto usando 7 tasas de codificación

Figura 6.9: Desempeño del esquema propuesto.

Rendimiento de los códigos Raptor

En un código FEC Raptor del tipo **3GPP (Third Generation Partnership Project) MBMS ((Multimedia Broadcast Multicast Service)**, como se puede verificar en la figura 6.10(b), la ineficiencia de decodificación del código raptor es muy cerrado a 1 y no es afectado por la tasa de codificación usada.



(a) Codification inefficiency of LDGM Staircase, LDGM Triangle and 3GPP MBMS Raptor as a function of k, when FEC expansion ratio is 1.5 for the LDGM codes

(b) Codification inefficiency of LDGM Staircase, LDGM Triangle and 3GPP MBMS Raptor as a function of FEC expansion ratio, when k = 10000

Figura 6.10: Gráficas extraídos de [60]

Cuando $k = 10,000$ paquetes la ineficiencia del Raptor es de 1,002 mientras que LDGM Triangle es de 1,055 y 1,068 para el código LDGM Staircase. De igual manera se observa en la figura 6.10(a) que con forme el tamaño del objeto es más grande, la ineficiencia del código Raptor se aproxima a uno.

7

Conclusiones

Considerando una transmisión de objetos grandes, el modelo de transmisión que mostró mejor rendimiento es el modelo 4, el cual envía toda la información de manera aleatoria y fue con dicho modelo, que se trabajó en las simulaciones del esquema propuesto, sin embargo, considerando que el esquema es sistemático y que tiene un mejor comportamiento para objetos pequeños (como es el caso de la transmisión de vídeo en tiempo real), una buena elección para transmitir la información es usando el modelo 2, sin embargo esto quedara definido por la tasa de perdida de paquetes, debido a que el modelo 4 es más robusto para tasa de perdidas grandes.

Definitivamente el esquema propuesto demostró tener una buena ineficiencia bajo tasas pequeñas, independientemente del tipo de retransmisión usado, el cual es una ventaja importante frente a los códigos FEC, sin embargo, como ya se comentó antes, el método trabaja en un canal de dos vías, pues realiza retransmisiones que pueden ser costosas frente a *streaming* en múltiples usuarios. Una manera de combatir este problema es, no solo proteger los paquetes fuente, sino también proteger los paquetes redundantes que se van creando, esto con el fin de volver utilizables aquellos paquetes que en el proceso de la decodificación no son usados debido a que el conjunto jerárquico

respectivo está completo.

En esta tesis observamos el desempeño de los códigos FEC más utilizados en los últimos años, y también el rendimiento de los códigos Raptor, el cual no se logró superar (si consideramos objetos grandes) puesto que presentan una ineficiencia bastante optima, sin embargo esta ineficiencia optima se presenta conforme el bloque de código se vuelve más grande, a pesar de ello el esquema propuesto logra una ineficiencia bastante buena y estable, considerando que la creación no es aleatoria, como en el caso de los códigos Raptor. La ventaja que presenta el esquema propuesto sobre los Sistemas FEC es la robustez bajo objetos pequeños, por lo cual es un buen método para la transmisión de multimedia.

Los códigos Reed-Solomon son totalmente factibles para ser usados en el esquema propuesto y esto lo volvería un método para grandes bloques, además que le brindaría al esquema la creación de más paquetes de redundancia en cada conjunto jerárquico, consiguiendo así tasas más bajas.

Como trabajo a futuro queda implementar del algoritmo en C, y así analizar los tiempos de codificación y decodificación, así como también la ineficiencia que se produce en las retransmisiones de paquetes. De igual manera, el análisis de la puesta en marcha del esquema para el fin en que fueron creados (transmitir vídeo en tiempo real).

Bibliografía

[1] A. Shokrollahi, *Raptor Codes*, 2006, Transactions on Information Theory (IEEE) 52 (6): 2551-2567.

[2] A. Shokrollahi, *Raptor codes*. Digital Fountain Technical Report, June 2003.

[3] A. Shokrollahi. *Codes and graphs*. In STACS 2000 (invited talk), LNCS No. 1770,pages 112, 2000.

[4] D. J C MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, ISBN 0-521-64298-1, 2003.

[5] D. J C MacKay, R. M. Neal. *Good codes based on very sparse matrices, en Cryptography and Coding*. 5th IMA Conference, C. Boyd, No. 1025 in Lecture Notes in Computer Science, Springer, Berlin, pp. 100-111, 1995.

[6] D. J C Mackay, *Fountain codes*. 2005, Imagine 152, 1062-1068. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1561992.

[7] G. Demay, V. Rathi, L. K. Rasmussen, *Rate distortion bounds for binary erasure source using sparse graph code*, in Proc. Data Compression Conf., Snowbird, UT, Mar. 2010.

[8] J.W. Byers, M. Luby, M. Mitzenmacher, A. Rege. *A Digital Fountain Approach to Reliable Distribution of Bulk Data*. Proceedings of ACM SIGCOMM '98, pages 56-67, Vancouver, September 1998.

[9] M. Mitzenmacher. *Digital Fountains: A Survey and Look Forward*. 2004, Proc. 2004 IEEE Information Theory Workshop (ITW).

[10] M. Luby. *LT Codes*. 2002, Proceedings of the IEEE Symposium on the Foundations of Computer Science: 271-280.

[11] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman: *Efficient erasure correcting codes*, IEEE Trans. Inf. Theory, 2001, 47, pp. 569-584

[12] R. G. Gallager. *Low Density Parity-Check Codes*. In PhD thesis, Massachussets Institute of Technology, 1963.

[13] R. G. Gallager. *Low Density Parity-Check Codes*. IEEE Transactions on Information Theory, 8(1), Jan. 1962.

[14] D. J C MacKay. *Information Theory, Inference, and Learning Algorithms*. 2003, Cambridge University Press. ISBN 0-521-64298-1.

[15] Lin, S., and Costello, D.J. Jr. *Error control coding*, Pearson Prentice Hall, 2004, 2nd edn.

[16] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.

[17] Berrou C., Glavieux A., y Thitimajshima P., *Near Shannon limit error-correcting coding and decoding: Turbo Codes*, en Proceedings of ICC'93, Geneva, Switzerland, May 1993, pp. 1064-1070.

[18] Cover T. M. y Thomas J. A., *Elements of Information Theory, John Wiley & Sons*, ISBN 0-471-06259-6, 1991.

[19] Shannon C.E., *A Mathematical Theory of Communication*, The Bell System Technical Journal,Vol. 27, pp. 379-423, 1948.

[20] Jörg Nonnenmacher and Ernst Biersack. *Reliable Multicast: Where to use FEC*, Proceedings of IFIP 5th International Workshop, (1996).

[21] John E. McNamara. Technical aspects of data communication. Rockport, MA: Digital Press, 1982.

[22] Peterson, Wesley. Weldon, E.J. Error Correcting Codes. England, Cambridge Mass, London: The MIT Press, 1984.

[23] C. E. Shannon, *A mathematical theory of communication*, The Bell System Technical Journal, vol. 27, no. 1, pp. 101-111, 1948.

[24] R. W. Hamming, *Error detecting and error correcting codes*, The Bell system technical journal, vol. 26, no. 2, pp. 147-161, April, 1950.

[25] Berrou, C. Berrou and Glavieux, A., *Near Optimum Error Correcting Coding and Decoding: Turbo-Codes*, IEEE Trans. on Communications, vol. 44, no. 10, October 1996, pp. 1261-1271.

[26] B. TELECOM. (2012, August) International symposium on turbo codes and iterative information processing. [Online]. Available: http://conferences.telecom-bretagne.eu/turbocodes

[27] John M. Danskin and Geoffrey M. Davis and Xiyong Song, *Fast Lossy Internet Image Transmission*, 1996.

[28] Steven Ray Mccanne, *Scalable compression and transmission of Internet multicast video*,1996.

[29] A. Albanese, J. Blomer, J. Edmonds, M. Luby, M. Sudan, *Priority encoded transmission*, IEEE Trans. Inform. Theory 42 (6) (November 1996) 1737-1744.

[30] Rohit Puri and K. Ramchandran and K.W. Lee and V. Bharghavan, *Forward error correction (FEC) codes based multiple description coding for internet video streaming and multicast* , 2001.

[31] T. Mizuochi, T. Sugihara, Y. Miyata, K. Kubo, K. Onohara, S. Hirano, H. Yoshida, T. Yoshida, and T. Ichikawa, *Evolution and Status of Forward Error Correction*, Information Technology R&D Center, Mitsubishi Electric Corporation, Optical Society of America, 2012.

[32] Hocquenghem, A. (September 1959), *Codes correcteurs d'erreurs* (in French), Chiffres (Paris) 2: 147?156

[33] Bose, R. C.; Ray-Chaudhuri, D. K. (March 1960), *On A Class of Error Correcting Binary Group Codes*, Information and Control 3 (1): 68?79, ISSN 0890-5401

[34] Bose, R. C.; Ray-Chaudhuri, D. K., *Further Results on Error Correcting Binary Group Codes*, Sept. 1960.

[35] I. S. Reed and G. Solomon, *Polynomial Codes over Certain Finite Fields*, SIAM Journal of Applied Mathematics, Volume 8, pp. 300-304, 1960.

[36] Viterbi AJ (April 1967). *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm*. IEEE Transactions on Information Theory 13 (2): 260?269. doi:10.1109/TIT.1967.1054010.

[37] L. van de Meeberg, *A Tightened Upper Bound on the Error Probability of Binary Convolutional Codes with Viterbi Decoding*, IEEE Trans. Inform. Theory, vol. IT-20, pp. 389-391, May 1974.

[38] Massey, J. L. (1969), *Shift-register synthesis and BCH decoding*, IEEE Trans. Information Theory IT-15 (1): 122?127

[39] Ben Atti, Nadia; Diaz-Toca, Gema M.; Lombardi, Henri, *The Berlekamp-Massey Algorithm revisited*.

[40] G. Ungerboeck, *Channel coding with multilevel/phase signals*, IEEE Trans. Inform. Theory, vol. IT-28, pp. 55?67, 1982.

[41] Tanner, R.M., *A Recursive Approach to Low Complexity Codes*, IEEE Trans. Inform. Theory, 1981, vol. 27, no. 9, pp. 533-547.

[42] Ashish Khisti, *Tornado Codes and Luby Transform Codes*, October 22, 2003.

[43] Vahid Tarokh, Nambi Seshadri, and A. R. Calderbank (March 1998). *Space-time codes for high data rate wireless communication: Performance analysis and code construction*. IEEE Transactions on Information Theory 44 (2): 744-765.

[44] P. Maymounkov, *Online Codes*. (Technical Report), November 2002.

[45] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer (October 2007), *Raptor Forward Error Correction Scheme for Object Delivery*, RFC 5053 (IETF).

[46] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, L. Minder (May 2011), *RaptorQ Forward Error Correction Scheme for Object Delivery*, IETF.

[47] M.Luby and M. Mitzenmacher, A. Shokrollahi, D. Spielman, V. Stemann, *Prectical loss-resilient codes*, 1997.

[48] M.Luby and M. Mitzenmacher, A. Shokrollahi, D. Spielman, *Eficient Erasure Correcting Codes*, IEEE Transactions on Information Theory, vol. 47, issue 2, pp. 569-584, February 2001.

[49] Biography of Claude Elwood Shannon: http://www2.research.att.com/~njas/doc/shannonbio.html

[50] Sklar, B., *Digital Communications: Fundamentals and Applications*, Second Edition (Upper Saddle River, NJ: Prentice-Hall, 2001).

[51] Robert G. Gallager, Massachusetts Institute of Technology. http://www.rle.mit.edu/rgallager/biography.htm

[52] Vucetic, B. & Yuan, J., *Space-Time Coding*, John Wiley & Sons, UK, 2003.

[53] H. Jafarkhani, *Space-Time Coding: Theory and Practice*, Cambridge University Press, 2005.

[54] Bernard M. T. Leiner, *LDPC Codes - A brief tutorial*, Available at http://bernh.net/media/download/papers/ldpc.pdf, Apr. 2008.

[55] V. Roca and C. Neumann, *Design, Evaluation and Comparision of Four Large Block FEC Codes, LDPC, LDGM, LDGM Staircase and LDGM Triangle, plus a Reed-Solomon Small Block FEC Codec*, INRIA, Tech. Rep., 2004.

[56] Vincent Roca and Christoph Neumann and Block Fec Codec, *Design, Evaluation and Comparision of Four Large Block FEC Codes, LDPC, LDGM, LDGM Staircase and LDGM Triangle, plus a Reed-Solomon Small Block FEC Codec*, 2004.

[57] Christoph Neumann and Aurélien Francillon, *Impacts of packet scheduling and packetloss distribution on fec performances: Observations and recommendations*, In ACM CoNEXT'05 Conference, 2005.

[58] James S. Plank and Michael G. Thomason, *On the practical use of LDPC erasure codes for distributed storage applications*, IEEE, Florence, Italy, June, 2004.

[59] Mathieu Cunche, Vincent Roca, *Improving the Decoding of LDPC Codes for the Packet Erasure Channel with a Hybrid Zyablov Iterative Decoding/Gaussian Elimination Scheme*, INRIA, No 6473, 2008.

[60] Montalban Gutierrez, R.; Seco-Granados, G.; , *Efficiency comparison of LDPC-LDGM and Raptor codes for PL-FEC with very large block sizes*, Wireless Telecommunications Symposium, 2009. WTS 2009 , vol., no., pp.1-6, 22-24 April 2009
doi: 10.1109/WTS.2009.5068986
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5068986&isnumber=5068929

[61] Gilbert E. N., *Capacity of a burst-noise channel*, Bell System Technical Journal 39: 1253?1265, 1960.

[62] Elliott E. O., *Estimates of error rates for codes on burst-noise channels*, Bell System Technical Journal 42: 1977?1997, 1963.

[63] Michael Luby; Tiago Gasiba; Thomas Stockhammer; Mark Watson, *Reliable Multimedia Download Delivery in Cellular Broadcast Networks*, Broadcasting, IEEE Transactions on , vol.53, no.1, pp.235-246, March 2007

[64] Matsuzono K., Detchart J., Cunche M., Roca V., Asaeda H., *Performance analysis of a high-performance real-time application with several AL-FEC schemes*, Local Computer Networks (LCN), 2010 IEEE 35th Conference on , vol., no., pp.1-7, 10-14 Oct. 2010