ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Proyecto Fin de Carrera

Prestaciones de las técnicas Digital Fountain en redes de sensores

(Performance of Digital Fountain techniques over Wireless Sensor Networks)

Para acceder al Título de

INGENIERO DE TELECOMUNICACIÓN

Autor: Gonzalo Muñoz Corada

Octubre - 2013

INGENIERÍA DE TELECOMUNICACIÓN

CALIFICACIÓN DEL PROYECTO FIN DE CARRERA

Realizado por: Gonzalo Muñoz Corada
Director del PFC: Jorge Lanza
Título: "Prestaciones de las técnicas Digital Fountain en redes de sensores"
Title: "Performance of Digital Fountain techniques over Wireless Sensor Networks"

Presentado a examen el día: 30 de Octubre de 2013

para acceder al Título de

INGENIERO DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Luis Muñoz Gutiérrez
Secretario (Apellidos, Nombre): Jorge Lanza Calderón
Vocal (Apellidos, Nombre): Luis Sánchez González

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del PFC
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Proyecto Fin de Carrera Nº
(a asignar por Secretaría)

AGRADECIMIENTOS

En primer lugar me gustaría dar las gracias a Luis por ofrecerme la posibilidad de desarrollar este proyecto.

Dar también las gracias a Jorge por ayudarme durante el desarrollo del proyecto teniendo que encargarse de otro proyecto que no esperaba tener que dirigir.

A Juanra y Jose por ayudarme siempre que he tenido alguna duda o problema con las placas y guiarme con el desarrollo del proyecto. A los demás que trabajan en el laboratorio por ayudarme a que me integrase en el grupo.

Y por último dar la gracias a mi familia por apoyarme en todo momento durante la carrera.

RESUMEN

La aplicación de los códigos Digital Fountain son una de las múltiples técnicas que permiten la recuperación de errores en sistemas de transmisión sin tener que realizar retransmisiones. Así se logra optimizar el uso del canal sobre todo en casos donde el transmisor actúa como una fuente continua de datos.

Los procedimientos de actualizaciones de los firmwares de equipos distribuidos en una red de sensores son un claro ejemplo de entornos en los que se requiere una mejora y eficiencia de las comunicaciones entre un punto y múltiples destinatarios. De esto surge la posible aplicación de los códigos Digital Fountain a este tipo de procedimientos.

Durante el trabajo primeramente se planteará un estudio de las técnicas de codificación y decodificación más utilizadas. Seguidamente, se analizaran las diferentes redes de sensores que se encuentran en el mercado. Por último, se procederá a implementar los algoritmos correspondientes para su correspondiente validación empleando plataformas hardware comercial. Finalmente, se evaluarán las prestaciones derivadas de la utilización de estas técnicas, también conocidas como códigos de Luby, para flujos de datos entre un servidor y varios clientes.

ABSTRACT

The application of the Digital Fountain codes is one of the multiple techniques that enable error correction avoiding retransmissions in communication systems. This way channel use is optimized, especially when using continuous data sources.

Device software updates are examples of procedures where enhancements in point-multipoint environments are required. Then the application of Digital Fountain techniques could be profitable.

In order to achieve this, the first step is to study the most widespread coding and decoding techniques. Afterwards, the different sensor networks that are on the market will be analyzed. Then proceed to implement the corresponding algorithms for further validation using commercial hardware platforms.

Finally, a thorough evaluation of the performance gains that can be obtained with these techniques (which are usually referred to as Luby codes) will be carried out. This evaluation will embrace data flow between a server and two clients.

ÍNDICE

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

La digitalización de la información ha abierto un gran abanico de posibilidades para su intercambio entre las personas y sus múltiples dispositivos. El despliegue y desarrollo de Internet ha brindado paralelamente la posibilidad de definir nuevos esquemas de distribución más rápidos y económicos.

No obstante, para dar respuesta a todos estos nuevos servicios, el necesario y progresivo incremento de uso de la red, en cierta medida, se ha realizado de un modo descontrolado. Así, se han ido añadiendo nuevos servicios cada vez con mayores demandas sobre sistemas ya existentes, sin realizar en muchos casos un análisis previo sobre la idoneidad de estos sistemas, las redes que los interconectan y las metodologías de transmisión ligadas a estos nuevos modelos.

Los propios modelos de explotación y distribución de la información, por los cuales un conjunto de datos es demandado por múltiples usuarios, redunda en la transmisión de flujos de datos similares donde la única información que varía es la dirección de destino en la mayoría de los casos. Adicionalmente, esta alta demanda de información provoca la propia congestión de la red, lo que conlleva un gran número de retransmisiones debidas a paquetes perdidos antes de llegar a su destino o a causa de errores en estos provocados por el uso indebido del canal de comunicación.

Los servicios de mayor demanda en la actualidad comprenden la transmisión simultánea a múltiples usuarios de videos, música, etc. Retransmisiones de televisión, eventos deportivos a través de internet, descarga de películas, series y otros tipos de contenidos multimedia tienen normalmente un único emisor y múltiples destinatarios. Constituyen lo que se denominan transmisiones multicast. Otro modelo igualmente extendido son las redes P2P (Peer-to-Peer) o redes distribuidas en las que con el objetivo de crear un medio de transmisión eficiente, todos los nodos comportantes responsabilidades y operan de un modo similar. Aunque puedan ser empleadas con múltiples propósitos, su uso más común es para la transferencia de grandes ficheros digitales.

La característica común de estos escenarios es que se trata de fuentes continuas de datos que requieren de una baja latencia de transmisión, un elevado ancho de banda y cierta robustez para evitar la retransmisión de contenidos.

Por otro lado, centrándose en las redes subyacentes empleadas para la transmisión, las necesidades difieren según el modelo que se aplique. Las redes cableadas tradicionales se dimensionan considerando el tráfico previsto que circulará por el canal y el número de usuarios que hará uso de él. La misma filosofía se sigue con las redes inalámbricas. Sin embargo, el notable incremento de uso de éstas y la alta movilidad de los usuarios conectados hace más difícil determinar las necesidades instantáneas, lo que hace que las situaciones de congestión anteriormente señaladas se den con mayor frecuencia.

Por otro lado, el uso compartido del canal inalámbrico, unido a las numerosas imperfecciones de éste, obliga a plantear nuevos modelos de intercambio de datos más eficientes para así poder mantener la calidad de los servicios y la experiencia de usuario.

De todo lo anterior se extrae la necesidad de redefinir la filosofía de uso de las redes. Muchos protocolos orientados a conexión usualmente emplean mecanismos para aportar fiabilidad y controlar la congestión. La aplicación de estos protocolos nativos de redes cableadas no es directa a los entornos inalámbricos. Estudios como [1][2] demuestran, a través del análisis del canal punto a punto entre dos nodos, que los protocolos TCP/IP, ampliamente empleados en los sistemas cableados, no son adecuados en los inalámbricos y más concretamente la redes IEEE 802.11, popularmente conocidas como redes WiFi.

La solución para paliar estas deficiencias y adecuar las redes inalámbricas a la distribución masiva de contenidos puede residir en la aplicación de técnicas de codificación de canal.

Supóngase, por ejemplo, un servidor que distribuye una actualización de un software a miles de clientes. Considerando el servidor con una fuente continua de datos debe permitir que cualquier receptor se una a la sesión de forma dinámica, accediendo a la información de forma asíncrona. El uso de la técnica denominada *Data Carrousel* permite te comportamiento en esta aproximación, la fuente envía de forma cíclica todos los paquetes. Los receptores, por tanto, pueden unirse en cualquier momento y recibir paquetes hasta haber reconstruido completamente la información original. La clara desventaja está en el número de paquetes innecesarios que se recibirán hasta haber completado con éxito la reconstrucción. A medida que los receptores recién la información pueden detectar la perdida de paquetes (recepción fuera de secuencia, etc.), enviando solicitudes de retransmisión. Este procedimiento inundaría rápidamente al servidor, e incluso aunque éste tuviera capacidad suficiente para atenderlas, los paquetes retransmitidos serían de utilidad a tan sólo un pequeño conjunto de los clientes. Las técnicas de retransmisión adaptativas que ya resultan ineficientes en redes cableadas, lo son aún más en redes inalámbricas y satelitales, donde se da un incremento de la latencia y una mayor limitación de la capacidad del canal.

Los problemas observados en la aplicación de soluciones propuestas basadas en retransmisión adaptativa [REF], han llevado a muchos investigadores a considerar la aplicación de las técnicas Forward Error Correction (FEC), consistentes en el envío de información adicional adjunta a los paquetes transmitidos. El principal beneficio de esta aproximación es que los diferentes receptores se pueden recuperar parte de la información perdida mediante

la utilización de la información redundante. En principio esta idea reduce en gran medida el número de retransmisiones. Un ejemplo de este mecanismo es el empleo de códigos Reed Solomon (RS). Sin embargo, tanto los tiempos de codificación como decodificación tienen dependencia cuadrática con el volumen de datos. Por tanto, esta solución requiere limitar el tamaño del bloque de información a transmitir en cada momento. Además, si se supera la capacidad correctora del código sería necesaria la solicitud de la retransmisión de la información, con lo que no se solventa el problema anteriormente señalado.

Las aproximaciones descritas que limitan la necesidad de retransmisiones pueden verse como una aproximación imperfecta de lo que se conoce como *Digital Fountain*. Digital Fountain es conceptualmente más simple, más eficiente, y aplicable a un abanico más amplio de redes que las soluciones anteriores. En Digital Fountain, se inyecta en la red un flujo de paquetes codificados, a partir de los cuales cualquier receptor puede reconstruir los datos. La propiedad clave es que los datos se pueden reconstruir idealmente a partir de cualquier subconjunto K de estos paquetes (siendo K el número de paquetes que constituyen la información a transmitir). Actualmente existen varios códigos que siguen el paradigma Digital Fountain y cuyo abanico de aplicabilidad se extiende a multitud de aplicaciones. Esta filosofía de codificación de datos se puede aplicar a redes de distribución, redes de almacenaje y recuperación de datos distribuidas, redes con limitaciones energéticas, etc.

Una aplicación muy interesante de esta codificación es a las redes de sensores y a la internet de las cosas, puesto que en ambas se dispone de una gran cantidad de interconectados entre ellos y que en la mayoría de los casos comparten una misma información. Es ahí donde los códigos Digital Fountain pueden suponer una mejora en su gestión.

La Internet de las Cosas hace referencia a una idea surgida por la compañía Auto-Id Labs en el que todos los objetos que se encuentran en la actualidad, desde una lata, un libro o un zapato se equipasen con dispositivos de identificación minúsculos de manera que se pudiesen controlar desde cualquier parte del planeta. La idea es muy simple pero su aplicación al mundo actual es de una dificultad extrema.

Las redes de sensores se plantean como una aproximación a este paradigma Consisten en una red de nodos de tamaño reducido pero con ciertas capacidades de cómputo, también llamados nodos, equipados con sensores y que colaboran en una tarea común. Un ejemplo de este tipo de redes es el proyecto de investigación del 7º Programa Marco de la Unión Europea Smart Santander [94], surgido de la colaboración entre la Universidad de Cantabria y el ayuntamiento de Santander con el propósito de crear una infraestructura experimental a escala de toda una ciudad donde se despliegan las aplicaciones y servicios típicos de una ciudad inteligente. En este proyecto se plantea el despliegue de una red de 20.000 sensores, integrados bajo la denominada 'Internet de las cosas', donde cualquier dispositivo dispone de capacidad de comunicación para poder transmitir información útil para los usuarios, con el objetivo de construir una “Ciudad Inteligente”.

Al tratarse de un proyecto experimental y de investigación y al poseer un gran número de sensores, surge la problemática de una eficiente comunicación entre todos ellos, por ejemplo para las actualizaciones del firmware de los sensores. Las frecuentes actualizaciones que se incluyen en la operativa de los nodos deben ser distribuidas a todos los nodos de la red.

Esta actualización del firmware es un proceso peligroso para el dispositivo puesto que si se produce un error de comunicación durante la actualización se pueden provocar grandes daños al sensor. De ahí surge la idea de usar los códigos Digital Fountain en este tipo de red puesto que además de que proporciona una mayor eficiencia de canal haga que sea eficiente en términos de batería y le doten de una mayor seguridad a la comunicación.

1.1 OBJETIVOS Y MOTIVACIÓN

A raíz de esta introducción preliminar, se considera necesario el estudio de técnicas de codificación que puedan ser de aplicación a entornos de distribución de contenidos de forma masiva y permitan una transmisión eficiente y fiable. De entre las diferentes soluciones, el paradigma de codificación basado en técnicas Digital Fountain se presenta como aquél que mejor se ajusta a esta problemática.

El objetivo principal del presente proyecto es la evolución en el análisis y validación experimental de los códigos Digital Fountain realizado en el Proyecto de Fin de Carrera de Cristina Ysart "*Prestaciones de las técnicas Digital Fountain sobre Infraestructuras Inalámbricas*"[96] y su aplicación a una red de sensores real con las características propias de esta redes de bajo consumo y capacidad computacional. Para ello se plantean los siguientes objetivos específicos:

- Estudio de las técnicas de codificación Digital Fountain.
- Estudio de los diferentes tipos de redes de sensores.
- Análisis del comportamiento de los códigos Digital Fountain insertados en una red de sensores.

1.2 CONTENIDO DE LA MEMORIA

Esta memoria se ha estructurado en varios capítulos, que se describen brevemente a continuación:

- Capítulo 2. *Digital Fountain*

  En este capítulo se presentan, en primer lugar, los códigos de Paridad de Baja Densidad (*Low-Density Parity Check,* LDPC) para, a continuación, presentar los diferentes códigos Digital Fountain existentes, enfocando la explicación en la Transformada de Luby (*Luby Transform*, *LT*), centro de este proyecto.

- Capítulo 3. *Redes de sensores*

  En este capítulo, se exponen y se analizan los diferentes tipos de sensores que se pueden encontrar cuando uno se propone crear un sistema de comunicación a través de sensores.

- Capítulo 4. *Diseño y validación*

  Una vez introducidos los aspectos teóricos de LT y los conceptos básicos de las redes de sensores que se encuentran en el mercado, se dispone a la descripción de los diferentes protocolos definidos y su validación con su implementación en entornos de comunicaciones reales, resultado de lo cual se busca obtener el mecanismo más óptimo y adaptado a las redes empleadas, en este caso, redes de sensores.

- Capítulo 5. *Conclusiones y líneas futuras*

  A la vista de los resultados obtenidos en el capítulo anterior, se presentan unas conclusiones al respecto y se muestran posibles las líneas futuras de trabajo.

CÓDIGOS DIGITAL FOUNTAIN

2

Cuando se desea transmitir datos de forma robusta en la Internet, como es bien sabido, se emplea el protocolo TCP [7], que utiliza las retransmisiones para garantizar que todos los paquetes se reciben ordenadamente y libres de errores. Este esquema sin embargo, no es apropiado para utilizarse en canales con elevada latencia o con elevada pérdida de paquetes, ni en el caso de redes multicast, como se ha visto en el capítulo anterior.

Los esquemas Digital Fountain (DF) permiten recuperar el mensaje original sin necesidad de retransmisiones, de tal modo que el receptor puede recuperar los K símbolos, que constituyen un paquete, a partir de cualquier subconjunto de $K \cdot (1 + \varepsilon)$ símbolos codificados [16].

En este capítulo, como paso previo a los DF, se tratarán los códigos de baja densidad (*Low-Density Parity Check,* LDPC).

2.1 REQUERIMIENTOS DE UN PROTOCOLO IDEAL

Para determinar qué necesita un protocolo ideal [13], se supone un escenario en el que numerosos clientes acceden a una aplicación de un servidor que envía paquetes broadcast o multicast mientras haya al menos un cliente que desee tener acceso a los datos. Esta situación ejemplifica características importantes que ha de poseer un protocolo para transmitir grandes cantidades de información a gran cantidad de clientes. Para mantener el tráfico de la red lo más bajo posible, un protocolo escalable para distribuir software debería implementar las siguientes características:

1. Fiabilidad: que los datos se envíen correctamente a todos los clientes.
2. Eficiencia: tanto el número de paquetes a recibir por cada cliente para poder reconstruir los datos como el tiempo que necesite para ello cualquier receptor, debe ser mínimo. Idealmente, el tiempo de descarga para cada cliente no debería ser mayor que en el caso de una conexión punto a punto.

3. Bajo demanda: los clientes deberían poder iniciar la descarga del archivo cuando creyeran conveniente en diferentes instantes.
4. Tolerancia: el protocolo debería poder atender a una población heterogénea de receptores, con diferentes pérdidas de canal y diferentes tasas de código.

Partiendo de estas premisas en las siguientes secciones se identificarán y caracterizarán una serie de protocolos, métodos de codificación y algoritmos de transmisión que hagan realizable este paradigma de protocolo ideal.

2.2 Códigos LDPC

Los códigos de Baja Densidad (*Low-Density Parity Check,* LDPC), también conocidos como códigos Gallager en honor a Robert G. Gallager, se presentaron a principios de los años sesenta [10][11]; debido a las limitaciones existentes en la capacidad de cómputo y de almacenamiento de los equipos disponibles en la época, y a la aparición de los códigos RS, fueron abandonados hasta que en la década de los noventa se "redescubrieron" sus bondades.

Su denominación emana del hecho de que la matriz H contiene pocos unos en comparación con el número de ceros (matriz dispersa).

Este tipo de códigos se incluyen entre los códigos bloque lineales[1]; básicamente, pueden representarse de dos modos: mediante su matriz de paridad (como todos los códigos bloque lineales), o bien mediante una representación gráfica, conocida como Grafo de Tanner [12]; el paso de un modo a otro es directo (Figura 2.1).

El Grafo de Tanner es un grafo bipartito, es decir, los nodos se separan en dos conjuntos diferentes y se conectan mediante arcos. Los dos tipos de nodos se conocen como nodos variable (*v-nodes*) y nodos *check* (*check-nodes*). Se explican más detalladamente en 2.3.2.2

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

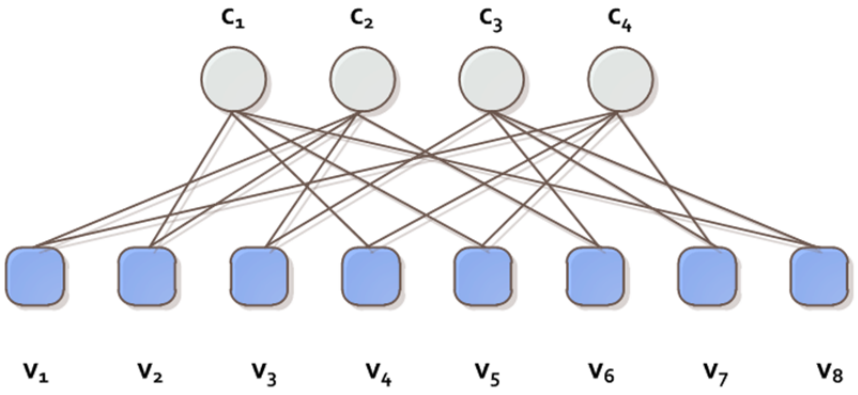

Figura 2.1 Gráfico de Tanner correspondiente a la matriz de paridad H

[1] . Un Código Bloque es Lineal cuando la combinación lineal de cualquier par de palabras código dadas, c_i y c_j, genera otra palabra código.

Si se definen w_r y w_c como el número de unos por fila y por columna respectivamente que posee la matriz H, ésta se considera de baja densidad si $w_c \ll k$ y $w_r \ll n$. De acuerdo a esto, la matriz de paridad ha de ser grande para considerarse de baja densidad.

Asimismo, a partir de estos parámetros se pueden clasificar los LDPC como:

1. LDPC regulares: si las filas y columnas de H tienen un peso uniforme, es decir, si w_c y $w_r = w_c \cdot (n/k)$ son constantes para cada fila y columna (en el ejemplo, $w_c = 2$ y $w_r = 4$); n y k son los valores que definen el código bloque (n, k).
2. LDPC irregulares: si las filas y columnas de G no tienen un peso uniforme, es decir, si w_r y w_c no son constantes.

Los códigos DF son un ejemplo de códigos LDPC irregulares. Fueron resultado de un largo proceso de estudio de los LDPC retomado por D. McKay, M. Luby, M. Mitzenmacher, A. Shokrollahi, J. Byers y D. Spielman. Desarrollaron diferentes tipos de LDPC [13] basándose en el esquema de Digital Fountain: los códigos Tornado, *Luby Transform (LT)* y los códigos Raptor. A continuación se van a presentar estos códigos, realizando un estudio más profundo en los Códigos LT, ya que son el tema principal del proyecto.

2.3 CÓDIGOS DIGITAL FOUNTAIN

En [5] se presentan algunos métodos para conseguir que los códigos LDPC trabajen de forma más eficiente. Con la intención de mejorar el rendimiento de estos códigos LDPC regulares, surge la idea de generar códigos LDPC irregulares [14]. La irregularidad de los grafos es la razón vital por la que los códigos DF son tan eficientes en la corrección de borrones.

Los códigos DF son aquellos que permiten implementar un método óptimo de distribución de contenidos, en el que una fuente genera una cantidad potencialmente infinita de paquetes y los envía por la red. Los receptores tan sólo necesitan recoger cierta cantidad de esos paquetes para reconstruir la información. El término viene por su analogía con una fuente: se puede imaginar una fuente de la que emanan continuamente gotas de agua; cualquier persona que tenga sed, no tiene más que llenar un vaso de agua de esa fuente sin importar gracias a qué gotas se ha llenado, ya que saciará su sed. La fuente es el servidor, las gotas los paquetes que se envían, y los vasos los receptores. En una situación ideal, si el mensaje original está formado por K símbolos, sólo se necesitarán K paquetes para recuperar la información.

Esta metáfora ayuda a comprender los códigos DF. Se trata de unos códigos *rateless,* en el sentido de que el número de paquetes que puede generar es potencialmente ilimitado (al igual que la fuente gotas de agua); además, estos paquetes codificados pueden ser generados dinámicamente. También son universales, porque son casi óptimos para cualquier tipo de canal, es decir, se pueden mandar tantos paquetes como sean necesarios para recuperar los datos independientemente de la estadística de borrones que tenga el canal. Los datos pueden reconstruirse a partir de un subconjunto cualquiera de K' paquetes, siendo K' algo mayor que K. Estos códigos asimismo, tienen una complejidad de codificación y decodificación muy pequeñas.

2.3.1 Códigos Tornado

Los trabajos de Luby *et ál.* desembocaron en 1997 en una nueva clase de códigos [15][16], denominados códigos Tornado, que fueron los primeros códigos en aproximarse eficientemente a DF. Fueron los primeros códigos publicados bajo la idea de DF. Estos códigos, mucho más eficientes que los RS en la corrección de borrones [16], se muestran como una alternativa mucho mejor como aproximación de DF.

En parte de la literatura [17] no se les considera DF, ya que al igual que con los RS, se debe fijar la tasa de código con anterioridad y por lo tanto, no son *rateless*. Se basan en grafos bipartitos en cascada (Figura 2.2) que deben ser elegidos correctamente para permitir una codificación y decodificación eficiente; los símbolos redundantes se crean mediante operaciones XOR [15].

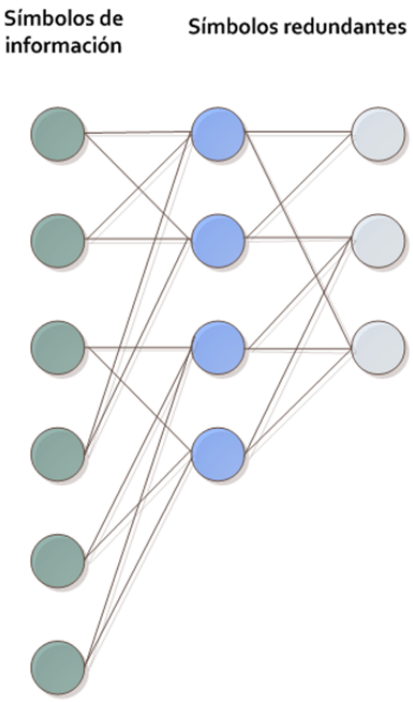

Figura 2.2 Grafo Códigos Tornado

Si el mensaje a transmitir de tamaño M se divide en k símbolos, empleando los códigos Tornado el receptor necesita conseguir algo más de k símbolos para recuperar el mensaje original (necesitaría un total de $f \cdot k$, donde f es el factor de overhead). La desventaja que tienen frente a los códigos RS es el número de paquetes extra que necesita recibir para reconstruir todos los símbolos. No obstante un buen diseño del código da como resultado un comportamiento global mejor. Según [15] el factor de *overhead* se estima alrededor de $f \approx 1.05$ para valores grandes de n y k. Los tiempos de codificación y decodificación son proporcionales a $k \cdot \log(1/(\mathrm{f}-1)\mathrm{M})$.

2.3.2 Transformada de Luby

Los códigos Transformada de Luby, o códigos LT fueron publicados por Luby en 2002 [18]. Los códigos LT, a diferencia de los Tornado, sí son rateless y no se ha de fijar tasa de código de antemano; asimismo los símbolos codificados se pueden crear dinámicamente.

Fueron los primeros códigos que cumplían completamente el concepto de Digital Fountain presentado en [16].

Los códigos LT permiten generar símbolos codificados dinámicamente de forma ilimitada. Es una de las características que hace que el rendimiento de los LT sea independiente de la probabilidad de borrón del canal.

Como es bien sabido, los protocolos habituales para la transferencia de ficheros se basan en dividir la información en K paquetes que son enviados a su destino. Debe garantizarse la correcta recepción de todos los paquetes para dar por concluida la transmisión de forma satisfactoria. No obstante, pueden producirse errores en la transmisión, por lo que se emplean diferentes técnicas para tratar de superarlos. Las más sencillas implican el uso de un canal de retorno a través del cual el cliente, en caso de recepción errónea, informa al servidor de qué paquetes deben ser retransmitidos.

Mediante la utilización de estos códigos se evita el empleo del canal de retorno. Por otra parte, los paquetes enviados se construyen operando de forma "aleatoria" sobre los símbolos de información (que conforman los paquetes), en lugar de enviar los símbolos originales (paquetes) en sí.

2.3.2.1 TERMINOLOGÍA

Antes de proceder a la descripción de los códigos LT, conviene fijar la notación y nomenclatura a utilizar. Tomando como base el diagrama de la Figura 2.3, los datos a enviar se consideran como una matriz de $K \cdot L$ bits que se dividen en K símbolos de información de L bits. Tras la codificación, a los símbolos creados se les denominará símbolos codificados (equivalentemente a paquetes codificados).

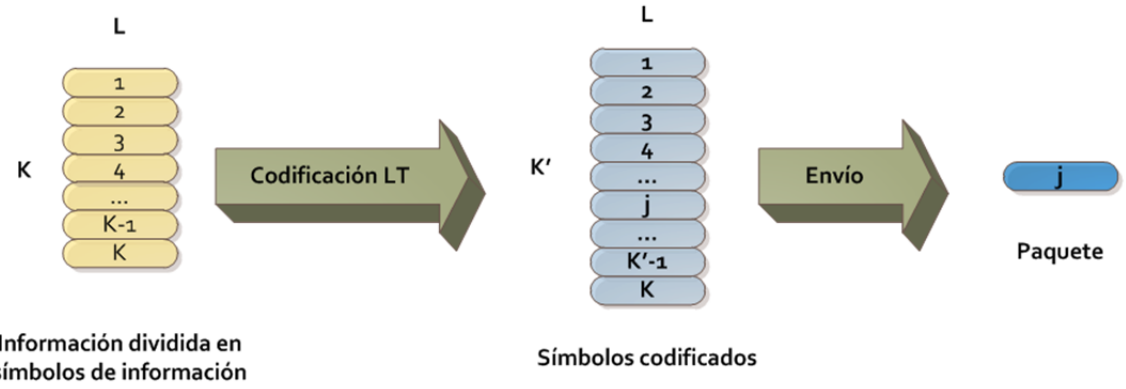

Figura 2.3 Terminología

2.3.2.2 PROCESO DE CODIFICACIÓN

El proceso de codificación de los códigos LT es simple. Al igual que los códigos LDPC, estos códigos también se pueden definir mediante un grafo bipartito o gráfico de Tanner.

Se considera un fichero de $K \cdot L$ bits, y se divide en K partes de la misma longitud, L bits; cada una de estas partes es un símbolo de información. A partir de esos K símbolos (de L bits cada uno), se crean los símbolos codificados sumando varios de ellos en base a una distribución de grado determinada (Figura 2.4). Se denomina grado al número de símbolos de información que se suman para obtener el símbolo codificado.

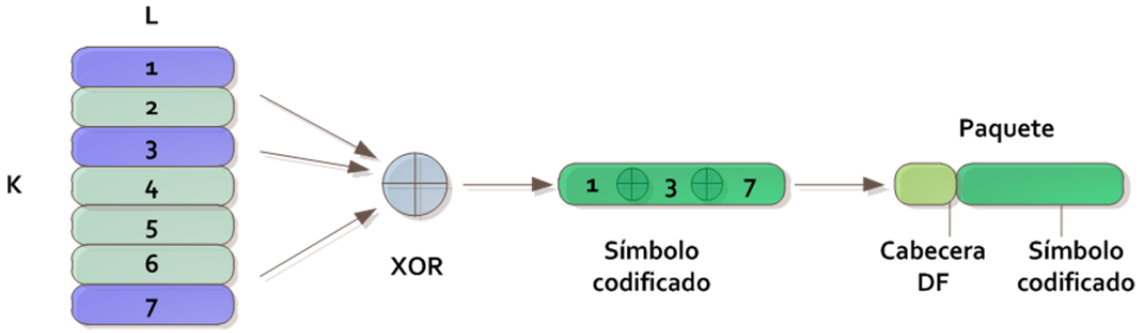

Figura 2.4 Ejemplo del proceso de Codificación

La codificación de los códigos LT se realiza de la siguiente manera:

1. Se elige el grado d del símbolo codificado a partir de la distribución de grado $\rho(d)$.
2. Se eligen d símbolos de información distintos del conjunto de los K disponibles.
3. Se realiza la operación XOR, bit a bit, módulo 2, de los símbolos seleccionados para obtener el símbolo codificado. A los símbolos seleccionados, se les conoce como *vecinos*.

La codificación puede representarse mediante un *grafo bipartito*, según se muestra en la Figura 2.5, en el que se pueden encontrar dos tipos de nodos: los símbolos de información o bloques, representados mediante círculos; y los símbolos codificados, representados mediante cuadrados. Los nodos de los diferentes grupos pueden estar conectados, pero no así los del mismo grupo. Las líneas que unen los nodos se conocen como *arcos*, y dos nodos que están conectados por un arco, *vecinos*. El número de vecinos por los que está formado un determinado símbolo codificado se conoce como *grado*.

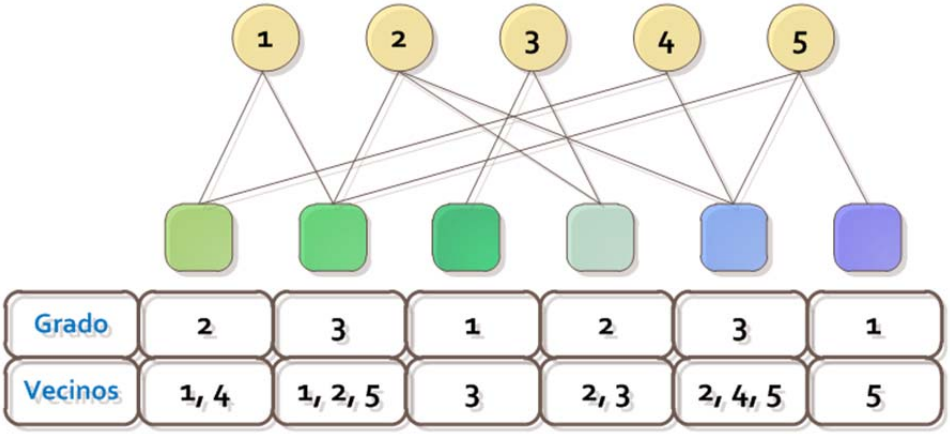

Figura 2.5 Grafo bipartito

Esta información se puede enviar al receptor explícitamente, formando tanto el grado como los vecinos de cada símbolo parte de la cabecera DF, o bien utilizando un generador de números pseudo-aleatorio de forma que el receptor, a partir de la misma semilla que en el emisor, sea capaz de replicar tanto el grado como los vecinos de cada símbolo de información.

Los códigos DF definen una fuente continua de datos. Sin embargo, previo al inicio de la transmisión se podrá negociar entre el emisor y el receptor un valor máximo de símbolos

codificados a recibir, superado el cual se interrumpirá la transmisión. Igualmente se podrán intercambiar información que describa el mensaje original, de tal forma que la transmisión se dará por concluida cuando el receptor haya reconstruido el mensaje original.

2.3.2.3 PROCESO DE DECODIFICACIÓN

Cuando se transmite un símbolo codificado a través de un PEC, se recibe correctamente o bien se pierde. El decodificador LT trata de recuperar los símbolos de información a partir de los símbolos codificados que recibe. Para decodificar, se asume que se conocen tanto el grado como los vecinos de cada símbolo codificado.

El proceso de decodificación se lleva a cabo de la siguiente manera:

1. Se recibe un paquete.
2. Se busca un símbolo codificado, e_m, que esté conectado a tan sólo un símbolo de información i_j. Si no es posible, volver al paso 1.
    a. Se asigna $i_j = e_m$
    b. Se calcula $e_l = e_l \oplus i_j, \forall\, l \neq m$, tal que el nodo e_l esté conectado al nodo i_j.
    c. Se quitan los arcos que unen i_j con e_l.
3. Volver al paso 1.

El proceso de decodificación se detiene si no se recuperan símbolos de información porque no hay nodos de grado 1 en el grafo bipartito. Si la decodificación finaliza sin recuperar todos los símbolos de información, se dice que el decodificador ha fallado. De otro modo, la decodificación ha tenido éxito.

En la Figura 2.6 se desarrolla un ejemplo de la decodificación suponiendo los símbolos de un bit. Las flechas rojas representan la situación $2a$ del algoritmo, es decir, se ha encontrado un paquete de grado uno y se copia su valor en el símbolo de información indicado. Las flechas azules representan la situación $2b$, cuando se hace la XOR entre el símbolo de información recién obtenido y los símbolos codificados que le tienen como vecino. Las flechas que desaparecen en cada iteración corresponden a la situación $2c$. Los símbolos de información se denotarán por $\{i_1, i_2, i_3, i_4, i_5\}$ y los codificados $\{e_1, e_2, e_3, e_4, e_5, e_6\}$.

En la iteración 1, se ve el grafo inicial. Seguidamente se encuentra un paquete de grado 1, por lo que se copia al símbolo de información al que está conectado, es decir, se realiza la operación $i_3 = e_3$. A continuación dado que i_3 es a su vez vecino de un símbolo codificado, e_4, y con arreglo del algoritmo de decodificación se tiene $e_4 = e_4 \oplus i_3 = 1$ y el símbolo codificado no altera su valor; este mismo símbolo codificado decrementa entonces su grado y queda conectado a tan sólo un símbolo de información, i_2, por lo que se copia su valor a éste, es decir $i_2 = e_4$ (ver iteración 3). En la cuarta iteración, habiendo conseguido el segundo símbolo de información, i_2 , y siendo éste vecino de dos símbolos codificados, e_2 y e_5, siguiendo el proceso de decodificación definido se realizan dos operaciones: $e_2 = e_2 \oplus i_2 = 0$ y $e_5 = e_5 \oplus i_2 = 0$; ambos símbolos decrementan su grado. El proceso de decodificación continúa hasta que se recupera el mensaje transmitido.

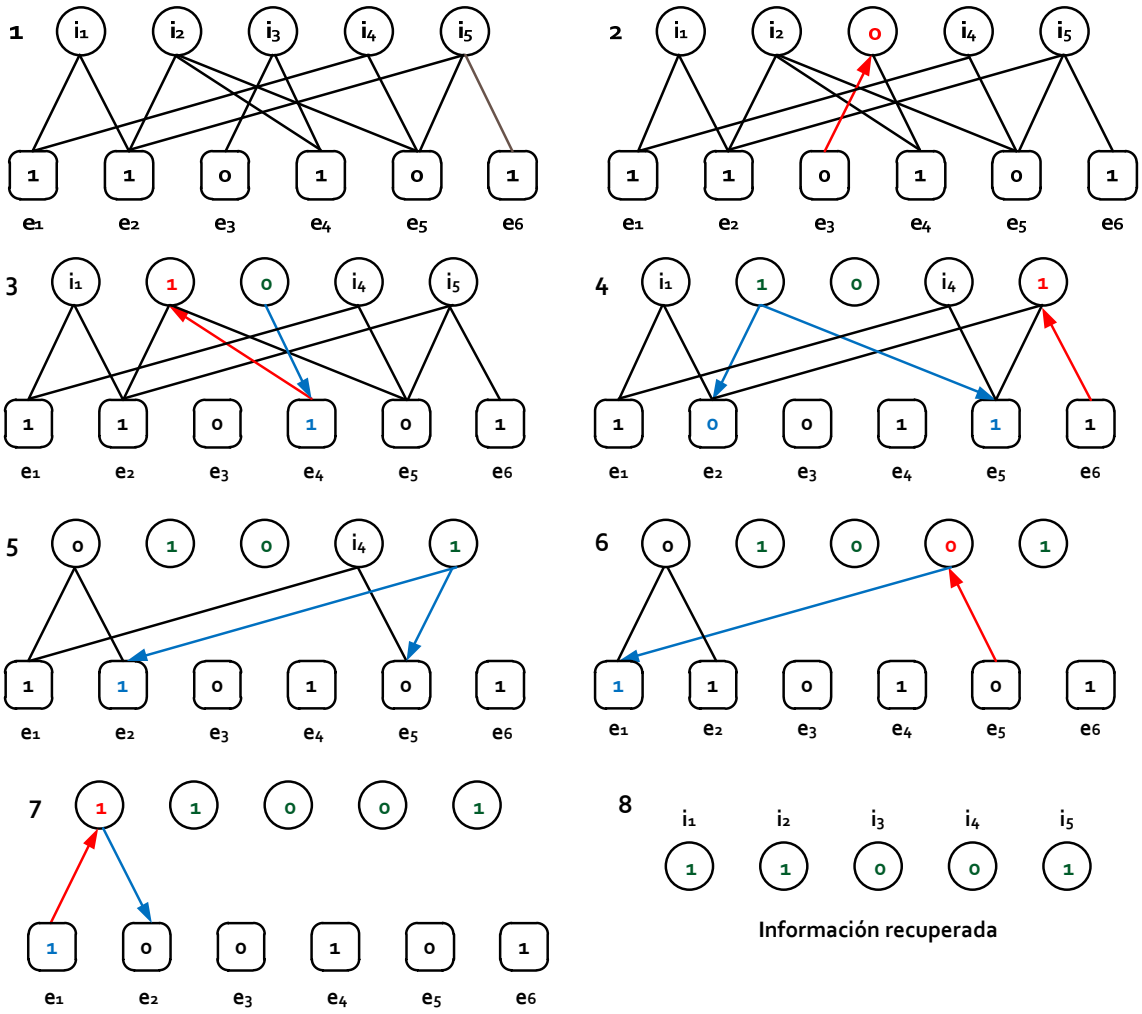

Figura 2.6 Ejemplo de Decodificación

Para que todo el proceso de decodificación resulte efectivo, es necesario elegir una distribución de grado adecuada ya que su elección resulta ***crítica*** en el diseño de los códigos LT. A continuación se analizan diversas distribuciones de grado y qué características deben incorporar para que su implementación real resulte eficiente.

2.3.2.4 DISTRIBUCIÓN DE GRADO

El comportamiento del proceso LT depende fuertemente de la distribución de grado, $\rho(d)$. Sin una distribución de grado adecuada, el concepto de los códigos LT sería inviable. Luby mostró en [18], que existen distribuciones eficientes para los LT.

Ya que la distribución de grado es el único factor que define la eficiencia de los LT, el objetivo es diseñar una que cumpla las siguientes condiciones:

1. Ha de precisar, en media, el mínimo número de símbolos para recuperar la información original y así mantener la redundancia baja.
2. El grado medio de los paquetes ha de ser lo más bajo posible, ya que se corresponde con el número de operaciones que son necesarias para crear un símbolo codificado.

Antes de describir la distribución de grado adecuada propuesta por Luby, se presenta un ejemplo que permite analizar el problema descrito de emplear todos los bloques o símbolos de información en los que se ha descompuesto el mensaje original.

El clásico problema de probabilidad consistente en colocar un conjunto de bolas en un conjunto de cajas puede verse como un caso especial del proceso LT en el que todos los símbolos tienen grado uno [19]. De este modo, cada símbolo codificado se crearía a partir de tan sólo un símbolo de información elegido aleatoriamente, copiando así su valor. A partir de esta premisa, se calcula el número de símbolos codificados necesarios para reconstruir los datos y el grado medio de cada uno, para responder a las preguntas anteriores.

Así, si se asimilan los símbolos de información a las *bolas*, y los símbolos codificados a las *cajas*, se plantea, dadas K cajas y fijada una caja concreta, calcular la probabilidad de que una vez arrojadas N bolas, no haya ninguna en la caja prefijada:

$$p(una\ bola\ no\ caiga\ en\ la\ caja\ prefijada) = \left(1 - \frac{1}{K}\right) \tag{2.1}$$

$$p(ninguna\ de\ las\ bolas\ caiga\ en\ la\ caja\ prefijada) = \left(1 - \frac{1}{K}\right)^N \approx e^{-\frac{N}{K}} \tag{2.2}$$

Puede verse entonces, que cuando $N = K$, la probabilidad de que una caja determinada esté vacía es aproximadamente $1/e$. Si se lanzan en total $3 \cdot K$ bolas, esta probabilidad cae a $1/e^3$, lo que equivale a que un 5% de las ocasiones dicha caja estará vacía [19]. Por tanto, hay que lanzar una gran cantidad de bolas para asegurar que todas las cajas estén cubiertas. Se calcula ahora el número medio de cajas vacías tras N tiradas. Partiendo de las siguientes variables aleatorias *(v.a.)*,

$$n_1 : primera\ caja\ vacía\ en\ N\ tiradas \begin{cases} 1, & vacía, probabilidad = e^{-\frac{N}{K}} \\ 0, & no\ vacía, probabilidad = 1 - e^{-\frac{N}{K}} \end{cases} \tag{2.3}$$

$$n : número\ de\ cajas\ vacías\ en\ N\ tiradas \tag{2.4}$$

y haciendo la media de n

$$E[n] = \delta = K \cdot e^{-\frac{N}{K}} \tag{2.5}$$

se obtiene el valor de δ, o el número medio de cajas vacías esperadas en N tiradas. Este número δ será pequeño (e indicará que todas las cajas tienen al menos una bola con una probabilidad de $(1 - \delta)$) sólo si

$$N > K \cdot \ln\left(\frac{K}{\delta}\right) \tag{2.6}$$

que resulta ser el número medio de paquetes necesarios (símbolos codificados), en media, para poder recuperar la información. Por tanto, el grado medio de estos ha de ser de $\ln(K/\delta)$.

Este análisis muestra que:

1. El número de símbolos codificados necesarios para recuperar los K símbolos de información con una probabilidad de al menos $1-\delta$ tiene un *valor inaceptable*, $\ln\left(\frac{K}{\delta}\right)$ veces más del mínimo posible.
2. Para cubrir todos los símbolos al menos una vez, la suma de los grados ha de ser al menos de $K \cdot \ln\left(\frac{K}{\delta}\right)$.

Otros análisis que suponen la variable aleatoria n como Binomial o Poissoniana, tampoco cumplen los requisitos esperados para la distribución de grado. Sin embargo, Luby ideó una distribución con un comportamiento ideal, denominada *Distribución Solitón Ideal (Ideal Soliton Distribution, ISD).*

2.3.2.4.1 DISTRIBUCIÓN SOLITÓN IDEAL

Antes de proceder con el desarrollo de la distribución Solitón Ideal, es necesario definir el concepto de ***rizo.*** Rizo es *el conjunto de símbolos de información obtenidos, pero que aún no han sido procesados*.

Para aclarar esta definición a continuación se presenta un ejemplo. Se suponen $K = 6$ símbolos de información y $K' = 10$ paquetes recibidos. En la Figura 2.7, los paquetes que se van a añadir al rizo en la iteración actual, se muestran en color azul; mientras que los que resultan redundantes (es decir, no ayudan a la decodificación), están marcados en verde. Asimismo, dentro de cada caja se encuentran los números de los paquetes vecinos; cuando aparezca un 0, quiere decir que ese paquete ya no tiene vecino en dicha posición. La evolución del rizo se encuentra en la columna de la derecha.

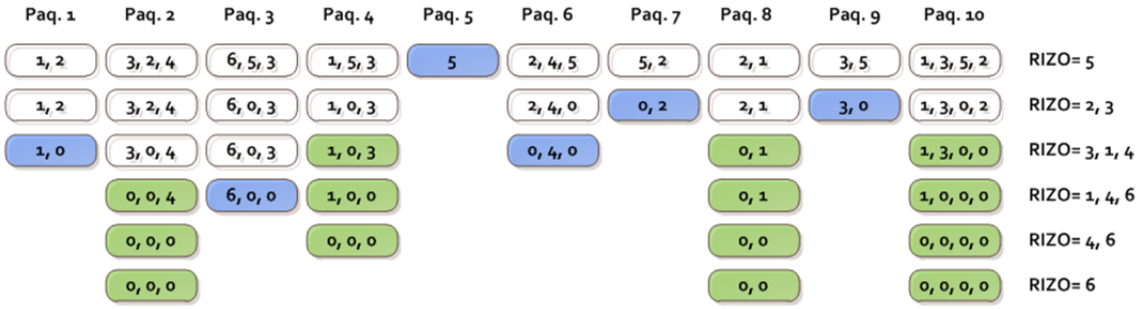

Figura 2.7 Ejemplo de evolución del Rizo

Del conjunto de paquetes recibidos, se observa que el paquete 5 es un paquete de grado uno. Según el algoritmo de decodificación descrito anteriormente, dicho paquete se añade al rizo y se guarda como símbolo de información; a continuación, se suma a los paquetes $\{3, 4, 6, 7, 9, 10\}$, que le tienen como vecino y se elimina del rizo.

En la siguiente iteración, tenemos dos nuevos paquetes de grado uno, el 2 y el 3. Procediendo del mismo modo se añaden ambos al rizo y se guardan como símbolos de información; en primer lugar se buscan los paquetes que tienen como vecino el símbolo de información 2, que resultan ser $\{1, 2, 6, 8, 10\}$ y se suman para posteriormente eliminarlo del rizo, y a continuación se repite el proceso con el símbolo de información 3. En la tercera iteración los paquetes 1 y 4 se añaden al rizo, por lo que los paquetes marcados en verde,

$\{4, 8, 10\}$, resultan redundantes (ya que contienen símbolos de información ya obtenidos). En la siguiente iteración la decodificación puede darse por terminada al recuperar el último símbolo de información, el 6.

Comprendido el concepto de rizo, seguidamente se desarrolla el proceso para la definición de la distribución Solitón ideal.

Una propiedad básica necesaria para cualquier distribución de grado, es que se añadan símbolos al rizo al mismo ritmo al que son procesados. Esta propiedad es la que inspiró el nombre de *Solitón*, ya que una ola solitón se propaga sin pérdida de forma ni velocidad durante grandes periodos de tiempo.

Otra propiedad esperada es que a medida que se decodifique, vayan quedando menos paquetes que contengan símbolos que ya están en el rizo, dado que resultan redundantes. Por otra parte, si el rizo se desvanece antes de haber procesado los K símbolos, el proceso de decodificación habrá fallado. Por tanto, el tamaño del rizo deberá ser lo suficientemente grande como para asegurar que no se desvanezca antes de tiempo.

Partiendo de estas premisas, seguidamente se obtiene de forma heurística la distribución ISD.

Sea $n_i^{(t)}$ el número de nodos de salida de grado i en el instante t. El instante $t = 0$ corresponde al inicio de la decodificación, momento de la recepción del primer paquete de grado uno pero que aún no ha sido procesado. En este instante, los nodos de grado i tienen $i \cdot n_i^{(0)}$ arcos en total que los conectan con los nodos de entrada, lo que significa que, en media, un paquete tiene $i \cdot n_i^{(0)}/N$ vecinos de grado i. Para mayor claridad se ha realizado un cambio de notación denominando N al número de paquetes o símbolos recibidos en lugar de n, como se ha venido haciendo hasta ahora. Por ejemplo, en la Figura 2.5, la media de nodos de salida de grado 3 que son vecinos de algún nodo de entrada es $((3 \cdot 2)/5) = 1.2$. Al procesar un nodo de salida de grado uno y eliminar los arcos que le conectan con sus vecinos, el número de paquetes de grado i cuyo grado se ha disminuido en uno se espera que sea de media $i \cdot n_i^{(0)}/N$. Si ya han sido decodificados t símbolos (en el instante de tiempo t), los arcos tienen sólo $n - t$ nodos de entrada a los que conectarse, por lo que la media resulta ser $i \cdot n_i^{(t)}/(N - t)$.

La condición deseada en términos de lo explicado anteriormente, es:

$$n_i^{(t)} = 1 \; \forall \; t \; \epsilon \; \{0, 1, \dots, N - 1\} \tag{2.7}$$

Teniendo en cuenta las condiciones presentadas, se está en disposición de construir la distribución ideal. De la ecuación (2.7) se tiene que $n_i^{(0)} = 1$. El valor de $n_2^{(0)}$ ha de ser tal que en el instante $t = 1$ el número de paquetes de grado uno sea nuevamente uno, lo que significa que uno de los paquetes de grado dos en el instante $t = 0$ ha disminuido su grado. Por tanto se tiene que:

$$\frac{2 \cdot n_2^{(0)}}{N} = 1 \; \leftrightarrow \; n_2^{(0)} = \frac{N}{2} \tag{2.8}$$

Y en general en cualquier instante t,

$$\frac{2 \cdot n_2^{(t)}}{N-t} = 1 \leftrightarrow n_2^{(0)} = \frac{N-t}{2} \tag{2.9}$$

Continuando este razonamiento recursivamente, se obtienen el resto de los valores $n_i^{(0)}$. El número de paquetes de grado dos en un instante t, es el mismo que el número de paquetes de grado dos en el instante $t-1$ menos los nodos que han disminuido su grado, más los nodos que previamente eran de grado tres; es decir,

$$n_2^{(t+1)} = n_2^{(t)} - \frac{2 \cdot n_2^{(t)}}{N-t} + \frac{3 \cdot n_3^{(t)}}{N-t} \tag{2.10}$$

$$\frac{N-t-1}{2} = \frac{N-t}{2} - 1 + \frac{3 \cdot n_3^{(t)}}{N-t} \tag{2.11}$$

$$\frac{1}{2} = \frac{3 \cdot n_3^{(t)}}{N-t} \rightarrow n_3^{(t)} = \frac{N-t}{2 \cdot 3} \tag{2.12}$$

Por lo tanto, el valor que se buscaba $n_3^{(0)} = N/(2 \cdot 3)$. La ecuación general de estado para cualquier nodo de grado i en cualquier instante t es

$$n_i^{(t+1)} = n_i^{(t)} - \frac{i}{N-t} \cdot n_i^{(t)} + \frac{(i+1)}{N-t} \cdot n_{i+1}^{(t)} \tag{2.13}$$

que para el caso de $n_{i+1}^{(t)}$ resulta:

$$n_{i+1}^{(t)} = \frac{N-t}{i+1} \cdot \left(n_i^{(t+1)} - n_i^{(t)} \right) + \frac{i}{i+1} \cdot n_i^{(t)} \tag{2.14}$$

ecuación con la que se pueden obtener recursivamente el resto de los valores. En realidad hay una forma más sencilla de obtener estos valores, que dice que el número de nodos de grado i en cualquier instante t para llegar a una distribución óptima es:

$$n_i^{(t)} = \frac{N-t}{i \cdot (i-1)} \tag{2.15}$$

Esta fórmula se infiere mediante inducción a partir de la ecuación (2.15).

Por último, los valores que se buscaban son $n_i^{(0)}$, que por la ecuación (2.15) son:

$$n_i^{(0)} = \frac{N}{i \cdot (i-1)} \; \forall \, i \in \{2, 3, \dots, N-1\} \tag{2.16}$$

Y $n_1^{(0)} = 1$. Para obtener finalmente la distribución de probabilidad normalizada, se dividen estos valores por el número de paquetes obtenidos, llegando a la distribución solitón, que resulta ser:

$$\rho(i) = \begin{cases} 1/K & i = 1 \\ 1/i(i-1) & i = 2, \dots, K \end{cases} \tag{2.17}$$

En la Figura 2.8 se muestra esta distribución para el caso de un grado máximo de paquete de 19.

En términos de los paquetes necesarios para recuperar la información, esta distribución es la adecuada ya que tan sólo se necesitarían K paquetes para recuperar los K símbolos originales; pero desafortunadamente es bastante *frágil* en la práctica, ya que pequeñas fluctuaciones en su comportamiento hacen que el proceso de decodificación falle. Esto es debido a que está diseñada de tal modo que cada vez que se procese un símbolo del rizo y se elimine, se incluya otro, haciendo que el tamaño del rizo sea siempre uno.

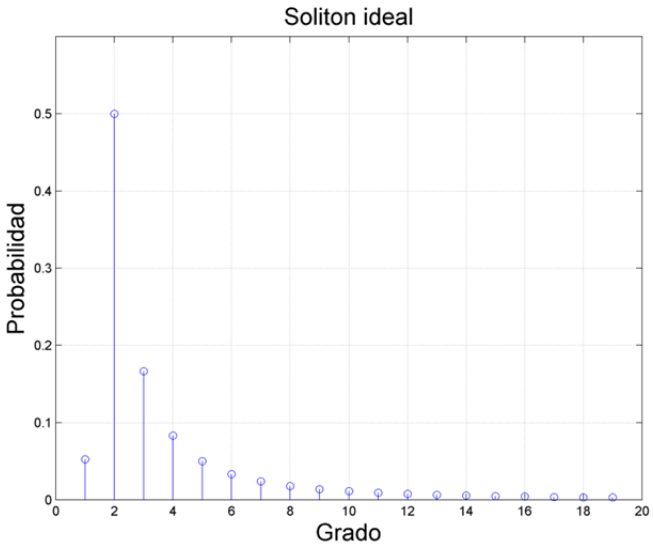

Figura 2.8 Función Solitón Ideal (ISD) para $K = 19$

En la Figura 2.9 puede verse un ejemplo del comportamiento ideal, con un tamaño de rizo constante e igual a uno durante el proceso de decodificación.

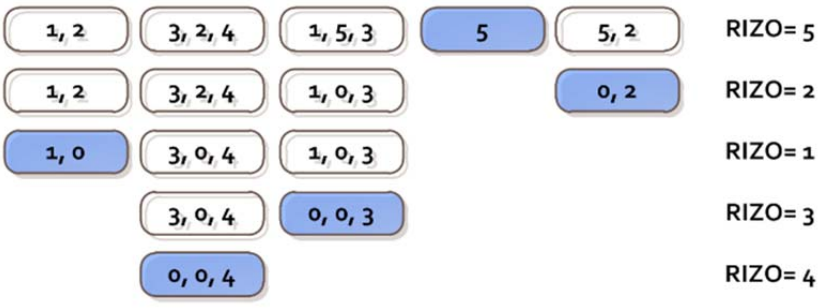

Figura 2.9 Comportamiento esperado del Rizo en ISD

A pesar de su fragilidad en la práctica, su descripción y análisis resultan necesarios para comprender mejor la d*istribución de solitón reforzado* (*Robust Soliton Distribution*, RSD). El trabajo de Luby [18] se ocupa principalmente de esta distribución, que es una versión avanzada de ISD. El objetivo en este caso es mantener un tamaño de rizo lo suficientemente

grande como para que no se desvanezca antes de que se consiga decodificar el mensaje original, y a la vez lo suficientemente pequeño como para que no haya mucha redundancia.

2.3.2.4.2 DISTRIBUCIÓN SOLITÓN REFORZADO

La distribución Solitón Reforzado (*Robust Soliton Distribution*, RSD) presenta mejores prestaciones que la ISD. Esta distribución mejorada, incluye dos nuevos parámetros: c y δ, que aseguran un tamaño de rizo de aproximadamente

$$\mathrm{R} \equiv c \cdot \ln\left(\frac{R}{\delta}\right) \cdot \sqrt{K} \tag{2.18}$$

durante todo el proceso de decodificación, lo que garantiza que haya una alta probabilidad de que el rizo no se desvanezca antes de completar la decodificación.

El parámetro δ es un límite de la probabilidad de fallo en la decodificación después de haber recibido K' paquetes (como se vio en el ejemplo de las bolas y las cajas).

El parámetro c es una constante de orden 1, si nuestro objetivo es probar el Teorema de Luby sobre los códigos LT; en la práctica, sin embargo, puede tomarse como un parámetro libre, que da buenos resultados con un valor algo menor que uno.

Defínase una función positiva $\tau(i)$ de la forma:

$$\tau(i) = \begin{cases} R/iK & i < (K/R) - 1 \\ R \cdot \ln(R/\delta)/K & i = (K/R) \\ 0 & i > (K/R) \end{cases} \tag{2.19}$$

Al sumarla a la función solitón $\rho(i)$, y normalizándola, se obtiene, la *distribución de solitón reforzado*:

$$\mu(i) = \frac{\rho(i) + \tau(i)}{Z} \quad \text{siendo} \quad Z = \sum_{i} \big(\rho(i) + \tau(i)\big) \tag{2.20}$$

Aplicando esta distribución, el número de paquetes necesarios para poder decodificar con éxito con una probabilidad de al menos $1 - \delta$ es de $K' = K \cdot Z$. Esto implica que el número de paquetes esperados de grado i es $K \cdot \big(\rho(i) + \tau(i)\big)$.

Para minimizar el número de paquetes usados para reconstruir los datos, es importante minimizar el tamaño del rizo, evitando así la aparición de muchos paquetes redundantes que cubran símbolos de información que se encuentran previamente en el rizo.

El hecho de añadir la función $\tau(i)$ debería asegurar que:

1. El proceso comienza con un tamaño de rizo suficiente.
2. La disminución del tamaño del rizo por el tratamiento de un símbolo debe contrarrestarse con la inclusión de otro símbolo.
3. Al final del proceso se consigue que todos los símbolos se hayan recuperado ubicando $\tau(K/R)$ en un valor de grado alto.

Un ejemplo de RSD para el mismo caso que en la Figura 2.8 se muestra en la Figura 2.10. Al comparar ambas figuras, se puede ver el pico en el caso de los paquetes de grado 5, obtenido gracias a la función $\tau(i)$.

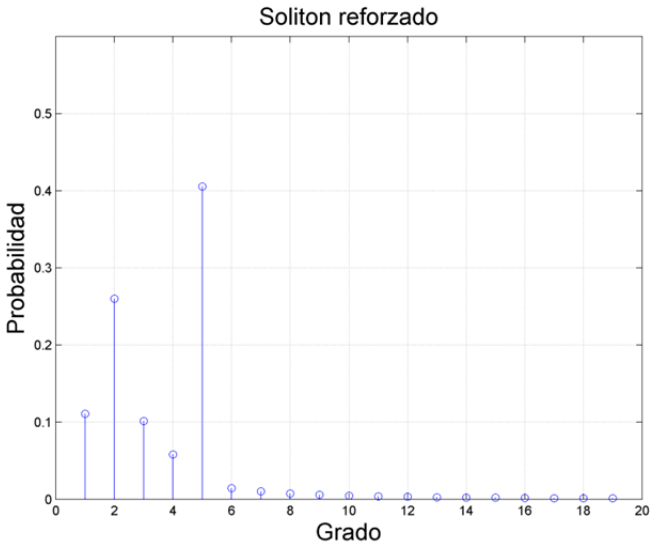

Figura 2.10 Función Solitón Reforzado (RSD) para $K = 19$

Esta distribución se usó en [18] para probar que el mensaje original se puede recuperar a partir de $N + O\left(\sqrt{N} \cdot ln^2(K/\delta)\right)$ símbolos con una probabilidad de $1 - \delta$. La complejidad de codificación y decodificación son entonces $O(\log(K/\delta))$ en términos de operaciones aritméticas.

2.3.3 Códigos Raptor

Los códigos Raptor fueron desarrollados por Amin Shokrollahi. La complejidad de codificación y decodificación es lineal, y su eficiencia es mejor que la de los LT: el coste, en media, de decodificación por símbolo es de $O(\log(N))$, resultando un coste total de $O(N \cdot \log(N))$ [3]. Esto se consigue relajando el requerimiento de que todos los símbolos estén cubiertos.

Estos códigos se consideran una extensión de los LT: el mensaje original se precodifica, dando lugar a unos nodos intermedios en el gráfico de Tanner; estos nodos intermedios son a los que se les aplica entonces la codificación LT. El código con el que se codifica se denota C. El proceso se puede observar en Figura 2.11.

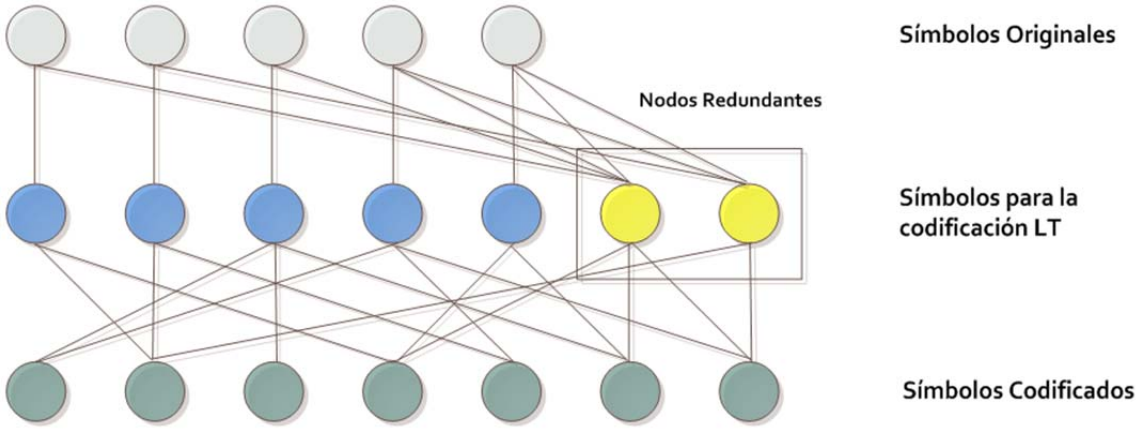

Figura 2.11 Grafo Código Raptor

Así definidos, se pueden considerar los LT como un tipo de código Raptor sin precodificación. La precodificación se puede realizar también en varios pasos, por ejemplo aplicando primero un código Hamming, a continuación un LDPC y entonces codificar los símbolos resultantes mediante LT.

También existen una subclase de códigos Raptor sin codificación LT, denominados *Pre-Coding-Only*, o códigos Raptor PCO [20].

Usando un código corrector de borrones como precódigo, se evita la necesidad de cubrir todos los símbolos de LT: sólo se necesita recuperar una fracción constante, ya que el mensaje original se puede recuperar gracias a la capacidad de corrección de borrones del precódigo.

La decodificación se realiza usando primero el decodificador LT para obtener los nodos intermedios, y a continuación se obtienen los símbolos originales aplicando el algoritmo de decodificación de C.

En definitiva, si se realiza una pequeña comparativa entre los códigos, cabe resaltar que los RS pueden decodificar con tan sólo K símbolos recibidos, pero sin embargo, los símbolos que se pueden crear son limitados y además su algoritmo de decodificación es cuadrático en el tiempo.

Los Tornado necesitan más de K símbolos para decodificar y los símbolos que puede crear también son limitados, pero su tiempo de decodificación es proporcional a K.

Los LT, como los Tornado, necesitan más de K paquetes para decodificar, pero su tiempo de decodificación es proporcional a $K \cdot \ln K$ y es capaz de generar un número ilimitado de símbolos codificados diferentes.

Por último, los Raptor también necesitan más de K paquetes, pero su tiempo de codificación, dependiendo del precódigo, puede llegar a ser proporcional a K y, como los LT, puede generar un número ilimitado de símbolos codificados.

2.4 INVESTIGACIÓN Y APLICACIONES

A continuación se presentan las principales investigaciones realizadas sobre los códigos LT en diferentes campos. Algunas aplicaciones requieren minimizar el overhead provocado por los paquetes redundantes, otras requieren reducir el coste computacional de los procesos de codificación y decodificación. Por ello, las investigaciones se centran en la mejora de la distribución de grado propuesta por Luby con diferentes objetivos.

2.4.1 MEJORAS DE LA DISTRIBUCIÓN DE GRADO

En las implementaciones reales de estos códigos, se utilizan generadores de números pseudo-aleatorios (*Pseudo Random Number Generator, PRNG*) para determinar el grado y los vecinos de un determinado símbolo, lo que reduce la aleatoriedad. En [21] se prueba que el uso de generadores de congruencia lineal (*Linear Congruential Generator, LCG*), que es un tipo de PRNG, ofrecen los mismos resultados que un generador de números aleatorios real. En [22] se propone una distribución RSD modificada, que consta tan sólo de un parámetro, implementada utilizando un *Kent Chaotic Map* para obtener los números. Los resultados muestran que esta distribución modificada tiene una eficiencia de codificación comparable con RSD y además reduce las operaciones a realizar para la decodificación, ya que se aumenta la probabilidad de los símbolos de menor grado. En [23] utilizan tres esquemas de creación de números aleatorios diferentes, pero ninguno de ellos destaca frente a los otros en todos los escenarios.

En [24] se demuestra que el hecho de mantener el tamaño del rizo constante durante la decodificación es una característica no deseable para los códigos LT; por ello, se propone un diseño en el que la distribución de grado lleva a un tamaño de rizo decreciente. Los resultados muestran que este diseño ofrece mayor robustez y menor overhead.

A fin de adaptar la función de grado para diferentes propósitos, en [25] utilizan un algoritmo multiobjetivo (*MultiObjective Evolutionary Algorithm, MOEA*). En el estudio se demuestra que las funciones optimizadas superan a la RSD tanto en overhead como en coste computacional. Por ello, se aconseja adaptar la función de grado de los códigos LT de acuerdo a las características de la aplicación en la que vaya a ser utilizada, para así mejorar su comportamiento.

En [26] proponen una distribución de grado sub-óptima que ayuda a mejorar la eficiencia del canal en un sistema de comunicación por satélite.

2.4.2 MEJORAS PARA LA TRANSMISIÓN DE CONTENIDO MULTIMEDIA

Otras investigaciones se centran en mejorar las características de los códigos para la distribución de contenido multimedia. En [27] proponen la utilización de un esquema de ventana deslizante para este propósito. La ventana se va deslizando a medida que los datos se decodifican, dejando una solapamiento entre sucesivos pasos, lo que permite incrementar virtualmente el tamaño del bloque. El esquema propuesto ofrece una mayor fiabilidad;

además, el proceso de decodificación es menos complejo y requiere menor cantidad de memoria respecto a los códigos tradicionales.

En [28] se utiliza el esquema de ventana deslizante, junto con un método de protección desigual frente a errores (*Unequal Error Protection, UEP*); y en [29] los mismos autores aplican este esquema para transmisiones de vídeo multicast en canales con pérdidas obteniendo buenos resultados.

Hacia el año 2008, el 3GPP (*Third Generation Partnership Project*) [30] y DVB (*Digital Video Broadcasting*) [31] estandarizaron una versión mejorada de los códigos LT. Asimismo, JVT (*Joint Video Team*) estandarizó SVC (*Scalable Video Coding*) como una extensión de H.264/AVC [32], lo que hizo que se produjeran investigaciones en ese campo. En un sistema de codificación de vídeo multicapa (*layered video coding system*), un vídeo se codifica en capas con diferente importancia. Las capas inferiores son más importantes que las superiores, es decir, una capa resulta inservible si no se tienen todas las capas inferiores a ella. Por ello, el esquema de protección desigual resulta beneficioso y necesario en la transmisión de este tipo de vídeos.

En [89] y [90] se exponen una serie de mejoras a estos códigos que dotarían de una mayor seguridad y robustez a este tipo de sistemas. Los principales cambios que proponen y que pueden influir en el desarrollo del presente proyecto son la aplicación de mecanismos de transmisión multiventana y la gestión del proceso de decodificación de forma que únicamente se escriba la información al finalizar el proceso de decodificación.

En [33] proponen una modificación de los códigos LT, a la que llaman códigos LT de protección desigual (*UnEqual Protected LT Code, UEPLT code*) para su aplicación en vídeos multicapa. Mediante esta modificación, cada una de las capas se protege de acuerdo a su importancia; así, las capas más importantes se recuperan más rápida y fácilmente. De este modo, un cliente puede reproducir vídeos dependiendo de la disponibilidad de ancho de banda que tenga.

2.4.3 APLICACIONES

La codificación basada en Digital Fountain se puede emplear en sistemas de almacenamiento distribuido de información. DIBS (*Distributed Internet Backup System*), es un sistema de almacenamiento y recuperación de datos peer-to-peer (P2P). En este tipo de sistemas, los datos se dividen en bloques y se distribuyen entre los diferentes nodos de almacenamiento. Así, si un usuario quiere recuperar sus datos, debe solicitar todos los bloques a los nodos de almacenamiento. En [34] mejoran el sistema DIBS, entre otros aspectos, mediante el uso de códigos LT en lugar de RS. En [35] se propone una arquitectura de almacenamiento denominada RobuSTore, que combina el uso de los códigos LT y los mecanismos de acceso especulativo para lectura y escritura en paralelo en sistemas distribuidos.

Otro posible uso de los códigos Digital Fountain es su aplicación a las comunicaciones a través de redes de baja tensión (PLC) [91]. Estas redes están despertando un creciente interés en la industria y en la comunidad científica puesto que la utilización de los cables de baja tensión ya existentes en los edificios, redunda en su sencillo y poco costoso despliegue. En los

estudios realizados reportan que los códigos Digital Fountain proporcionan mejores resultados que el protocolo de comunicación TCP, el cual presenta serios problemas en este tipo de canales.

Por último, una de las aplicaciones más interesantes de los códigos, se encuentra en su uso en las redes de sensores inalámbricas (*Wireless Sensor Netwotks, WSN*). Estas redes están tomando cada vez más relevancia para un amplio rango de aplicaciones como la vigilancia, el ámbito médico, o el seguimiento de los niveles de contaminación ambiental. Uno de los retos a la hora de diseñar estas redes, es el consumo de energía; para reducirlo, se necesitan tanto una modulación como un esquema de codificación energéticamente eficientes y así prolongar la vida del sensor. Una investigación publicada recientemente [36], muestra una forma de diseñar redes de sensores inalámbricas fiables, de baja complejidad y de bajo consumo (*Green Modulation/Coding, GMC*). Se muestra que los códigos LT ofrecen beneficios para este tipo de redes gracias a no tener una tasa de decodificación fija. Así, su simplicidad y flexibilidad junto con el uso de la modulación MFSK (*Multiple Frequency-Shift Keying*) lo hacen ser considerado para su utilización en redes de sensores inalámbricas dinámicas.

REDES DE SENSORES 3

Hoy en día existen numerosos tipos de sensores formando parte de un gran número de sistemas y dispositivos electrónicos (sensores de temperatura en estaciones meteorológicas, acelerómetros en móviles y tabletas, sensores infrarrojos en sistemas de alarma o detección de movimiento, etc.). La mayor parte de estos sensores se limitan a funcionar como un transductor que realiza un tipo concreto de medición (una o más variables del entorno) y envía dicha información a un procesador central, dentro del mismo sistema electrónico.

Sin embargo, durante los últimos años, ha aparecido una nueva generación de sensores, independientes de un sistema electrónico concreto, que incorporan en un mismo dispositivo el transductor (de la o las variables que interesan medir), la alimentación del propio dispositivo y un módulo de comunicación dotado de cierta inteligencia propia, capaz de organizarse a sí mismo y de interconectarse de forma inalámbrica con otros nodos semejantes. Como resultado de su despliegue surgen las llamadas redes de sensores inalámbricos (Wireless Sensor Networks, WSN), consistentes en redes formadas por multitud de sensores individuales que intercambian información entre sí y/o con un nodo central sin necesidad de cables y mediante un protocolo de comunicación pre-establecido.

Los diferentes nodos pueden operar no solo como sensores, sino que sus capacidades de procesamiento y comunicación permiten que funcionen como actuadores, es decir, mecanismos capaces de activar procesos automatizados sobre sistemas externos bien siguiendo instrucciones remotas o a partir de decisiones tomadas de manera autónoma a partir de la información sensada (i.e. encender luces ante una disminución de la intensidad lumínica). Junto a los sensores, constituyen las redes inalambricas de sensores y actuadores (Wireless Sensor & Actuator Networks, WSAN).

Las WSN comenzaron siendo utilizadas en aplicaciones militares, más en la actualidad proliferean multitud de aplicativos donde un sistema de control inalámbrico facilita la instalación. Algunos ejemplos son la domótica, medicina o en fábricas industriales o las ambiciosas ciudades inteligentes. El diseño eficiente e implementación de las redes inalámbricas de sensores se ha convertido en un área de investigación emergente en los

últimos años, debido al gran potencial de estas redes para cubrir grandes áreas con un coste relativamente bajo.

A continuación se introducirán las caracteristicas y posiblidades de la redes de sensores inalambricas, ahondando en las arquitecturas y topologías más usuales, así como en las soluciones tecnologícas empleadas en su despliegue.

3.1 CONCEPTOS GENERALES

Una red de sensores inalámbrica o su extensión, una red de sensores y actuadores, es una red inalámbrica ad-hoc formada por un conjunto de dispositivos autónomos e inteligentes y de carácter multifuncional, con capacidades de sensado y recogida de datos, que se comunican entre sí o con una una base central con el objetivo de intercambiar la información capturada por ellos mismos y tomar las acciones pertienentes sobre los sistemas bajo su inspección y control. Estas redes deben adaptarse a los cambios (posición, alimentación, etc.) puntuales en los diferentes nodos que las conforman.

En general, en una WSN se pueden encontrar los siguientes elementos:

- Sensores: capaces de medir las condiciones y la información deseada del lugar o del medio en el que se encuentran.
- Nodos: dispositivos que recogen la información recibida por los sensores y la transmiten a otros nodos o a la estación base.
- Pasarelas o Gateways: elementos que conectan los nodos de la red de sensores a una red externa o red troncal.
- Estación base: consistente en un elemento que recoge todos los datos provenientes de la red de sensores. Suele tratarse de un ordenador común o un elemento que lo sustituya.

Usualmente se emplea el termino mota para referirse a los nodos en una red de sensores. Este dispositivo integra, al menos, sensores, un micro controlador, un conversor A/D y un chip de comunicación radio.

3.1.1 NODOS SENSORES Y ACTUADORES

La arquitectura de un elemento sensor o actuador se puede dividir en tres bloques funcionales, tal como se refleja en Figura 3.1:

- Bloque sensor/Bloque actuador: define la funcionalidad del dispositivo, como sensor (medición de un parámetro) o como actuador (mecanismo activador/interventor).
- Bloque de procesado: conjunto de elementos que dotan al dispositivo de inteligencia de computación y toma de decisiones.
- Bloque de comunicación: envía y recibe mensajes hacia y desde los nodos y repetidores vecinos.

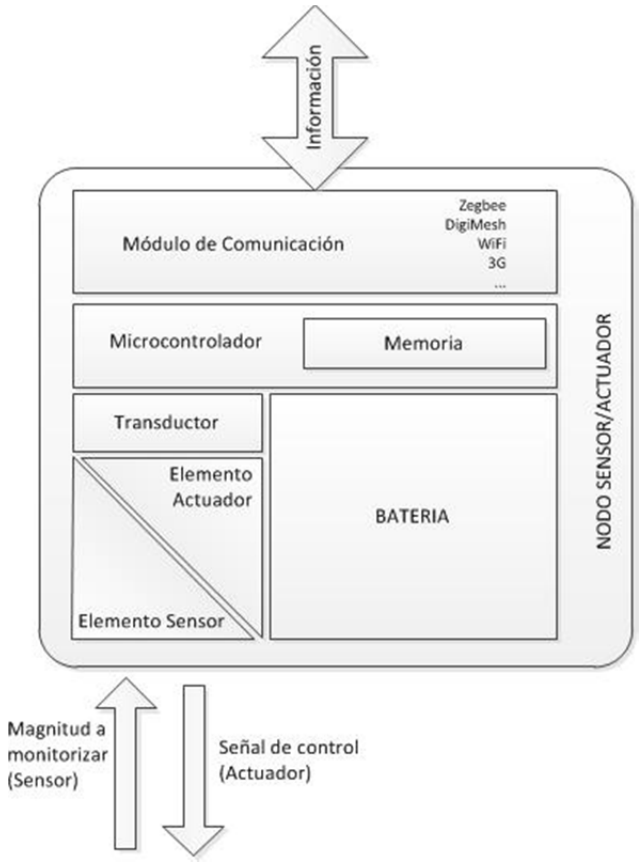

Figura 3.1 Esquema genérico de un nodo sensor o actuador

Se pueden establecer una serie de características que se dan en los nodos:

- Suelen estar limitados por el consumo energético, al estar usualmente alimentados por baterías. Y limitados en memoria y capacidad de proceso para ahorrar costes.
- Son autónomos y operan de forma independiente.
- Tienen una alta probabilidad de fallo, debido a las condiciones a las que se exponen y a su bajo coste.
- Deben incorporar medidas que permitan su adaptación al entorno.

El tamaño de un nodo puede variar, al igual que el coste, desde cientos de euros a unos pocos céntimos, dependiendo de la complejidad requerida. El tamaño y la limitación del consumo en los nodos sensores son los causantes de las limitaciones de recursos como la energía (<50mw), la memoria (<1Mbyte), la velocidad de cálculo (<500 Mhz) y el ancho de banda (250kbps).

Al tener un micro controlador interno, dentro de estos nodos se pueden realizar pequeños procesamientos de los datos con el objetivo de reducir el tráfico circulante por la red; puesto que sólo se enviarán los datos o las informaciones estrictamente necesarias, requieriéndose así una programación eficiente.

3.1.2 ARQUITECTURAS

Tomando como elementos de una red los nombrados anteriormente, podemos distinguir:

3.1.2.1 ARQUITECTURA CENTRALIZADA

En este tipo de red (Figura 3.2) los nodos se encargan de recoger información, tras lo que enviarán sus datos a la pasarela o gateway más cercano (si es que la hubiera, pues podría hacerlo directamente a la estación base), que dirigirá el tráfico de la red. Esto provocará dos problemas. Uno, el cuello de botella, provocando la lentitud de los envíos de los datos, y dos, como consecuencia un mayor consumo de energía, disminuyendo el tiempo de vida de los nodos.

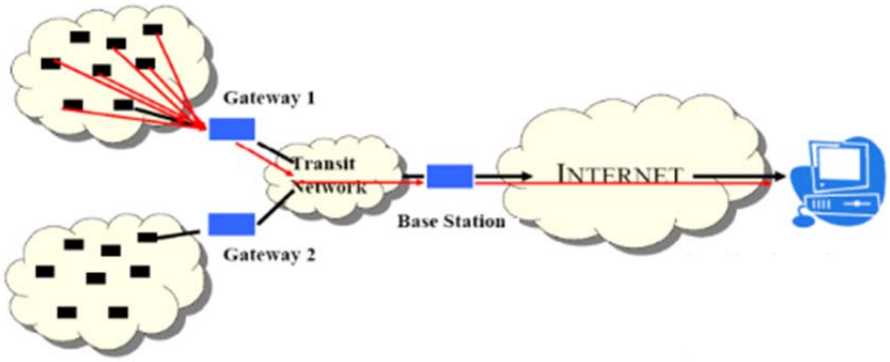

Figura 3.2 Esquema de arquitectura centralizada

3.1.2.2 ARQUITECTURA DISTRIBUIDA

Los nodos se comunicarán con sus nodos vecinos, cooperando y obteniendo una respuesta única que será enviada al coordinador del *clúster* que se encargará de comunicar a la estación base (Figura 3.3).

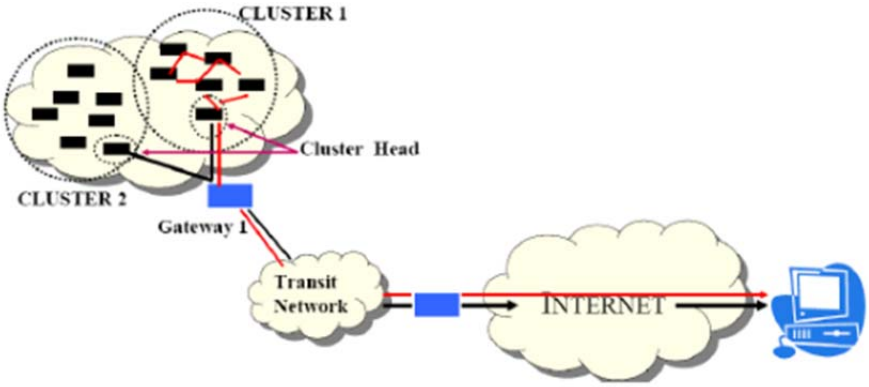

Figura 3.3 Esquema de arquitectura distribuida

3.1.3 TOPOLOGÍAS DE RED

Estas redes deben de auto-organizarse y auto-mantenerse para que de esta forma se reduzcan los costes totales y se facilite su uso. Dependiendo de los requerimientos de la aplicación, existen múltipes formas desde el punto de vista lógico en las que disponer los enlaces entre los nodos. A continuación se muestran algunas de las más empleadas (ver Figura 3.4).

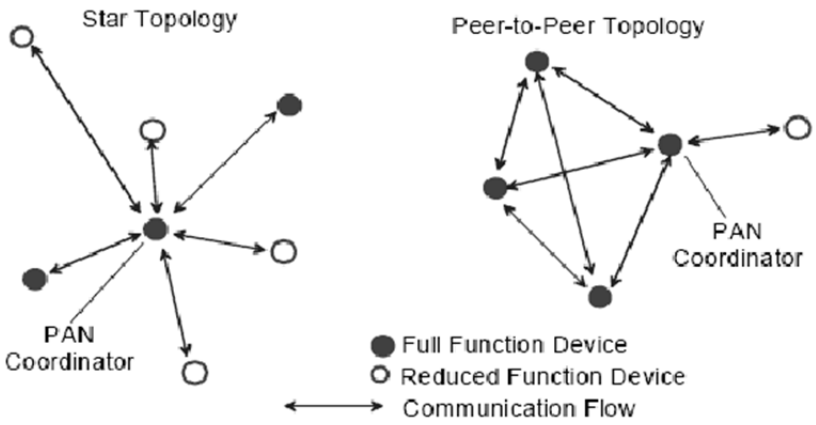

Figura 3.4 Topología en estrella y red peer-to-peer

3.1.3.1 EN ESTRELLA

En este tipo de configuración es necesario un coordinador por cada grupo de nodos o PAN (Personal Area Network), hacia el cual el resto de nodos dirigen su comunicación. Cada dispositivo establece una comunicación bireccional con el nodo maestro. Dicho coordinador es el encargado de iniciar, terminar o redirigir la comunicación a través de la red. Aparte de esto, realiza la propia función que tenga asignada, por ejemplo, actuar como nodo central de recogida de datos.

3.1.3.2 PEER-TO-PEER

Según esta topología cada dispositivo puede comunicarse con otros de la misma red que estén a su alcance. Conforman redes ad-hoc, es decir, redes sin infraestructura física preestablecida, ni necesidad de un control central que coordine su actividad.

Además pueden ser auto-organizativas, con restauración de caminos, con nodos que puedan trabajar como emisores o receptores y tener la capacidad de establecer caminos de comunicación entre nodos sin visibilidad directa y, modificar estos caminos si alguno de los nodos del encaminamiento falla, etc. Pero estas funciones han de ser establecidas por el protocolo de comunicaciones que se emplee entre los nodos.

Con esta topología de red es posible crear estructuras más complejas, como redes malladas, en las que no existe una jerarquía entre los nodos.

3.2 TECNOLOGÍAS DE COMUNICACIÓN INALÁMBRICA

Seguidamente se estudian las tecnologías comunicación inalámbrica de nivel físico y de enlace que potencialmente pueden ser empleadas en redes de sensores.

3.2.1 BLUETOOTH

El estándar IEEE 802.15.1[39] o Bluetooth trabaja en la banda ISM (Industrial Scientific & Medical) a una frecuencia de 2,4 GHz con 79 canales de 1 MHz entre 2402 y 2480 MHz. Esto hace muy alta la probabilidad de interferencia debido la proximidad de redes 802.11, hornos microondas, mandos de garaje, etc. Para evitar en lo posible las interferencias, Bluetooth usa un sistema llamado Frequency Hop Spread Spectrum (FHSS). Dicho sistema realiza 1600 saltos de frecuencia por segundo entre los 79 canales radio. Es decir, divide la información en pequeños paquetes, los cuales se transmiten siguiendo una secuencia determinada a través de los 79 canales disponibles.

El rango de alcance de Bluetooth es de 10 metros con un consumo de 1 mW, aunque dicho rango puede alcanzar los 100 metros si aumentamos la potencia de transmisión hasta los 100 mW. La tasa de transmisión es de 1 Mbps cuando se utiliza modulación GFSK (Gaussian Frequency Shift Key). Aunque para versiones más avanzadas del estándar (Bluetooth 2.0) se pueden alcanzar velocidades de hasta 2 Mbps gracias a la combinación de GFSK con PSK (Phase-Shift Keying).

Bluetooth está basado en una arquitectura maestro-esclavo. En la que los dispositivos se organizan en celdas denominadas “piconets”, que pueden estar formadas de hasta un máximo de 8 nodos, de los cuales uno es el maestro, encargado de organizar la comunicación entre los demás integrantes de la red o esclavos. Además, cuenta con cinco estados distintos de funcionamiento para reducir el consumo.

3.2.2 WI-FI

Esta tecnología se basa en el estándar IEEE 802.11[40] y pretende ser la alternativa inalámbrica a las redes de área local (LAN).

Trabaja en la banda ISM y existen varios estándares Wi-Fi, los principales son:

- 802.11b: trabaja a 2,4 GHz, tiene un alcance de aproximadamente 100 metros con una tasa de transferencia de 11 Mbps. Su principal ventaja es el bajo coste de los dispositivos que lo implementan.
- 802.11a: trabaja a una frecuencia de 5 GHz y su tasa de transferencia puede alcanzar los 54 Mbps. Al ser una frecuencia menos usada, no se dan tantas interferencias. No obstante tiene un rango de unos 30 metros y el coste de los dispositivos es mayor que los de 802.11.

- 802.11g: trabaja a 2,4 GHz, compatible con 802.11b, puede alcanzar velocidades de hasta 54 Mbps en un rango de alcance de unos 100 metros. Por contra, su coste y consumo de potencia son mayores.
- 802.11n: incrementa significativamente la velocidad de transmisión hasta un máximo de 600Mbps.

Existen dos tipos de elementos de red: AP (Access Point) y las estaciones. El primero está conectado a una red Ethernet y, el segundo se trata de estaciones inalámbricas, en general ordenadores personales, teléfonos móviles, entre otros, equipados con un interfaz de red inalámbrico (NIC: Network Interface Card).

Dependiendo de la forma de conectar los elementos de red anteriores se pueden definir diferentes configuraciones para las redes WLAN:

- Infraestructura: cada celda, o BSS (Basic Service Set), es el área de cobertura de un AP que forma parte de una red cableada. Los dispositivos inalámbricos que se encuentran dentro de la celda se conectan a la red o entre ellos mismos a través del AP.
- Ad hoc: en este caso dos o más dispositivos 802.11 se conectan directamente entre ellos sin tener que canalizar previamente la información a través de un AP. A esta configuración también se la conoce como “peer to peer”.

3.2.3 Ultra-Wide Band

Basado en los trabajos del IEEE Task Group 802.15.3a[41], Ultra-Wide Band (UWB)[42] es una tecnología en el rango de las PAN (Personal Area Network) que pretende saciar la necesidad del intercambio de datos con una alta tasa de transferencia en distancias cortas, como puede el ejemplo de videocámaras digitales, sistemas home cinema, etc.

UWB hace uso de un espectro muy ancho de frecuencias: desde los 3,1 hasta los 10,6 GHz con un ancho de canal de 500MHz. Pero, al contrario que Bluetooth y Wi-Fi, no realiza el envío de información de forma continua sino en determinados instantes temporales, utilizando gran parte de ese ancho de banda y emitiendo con menos potencia. Así, el consumo de los dispositivos basados en este estándar es relativamente bajo. Además esta tecnología es capaz de hacer uso de un mismo canal sin interferencias.

Con UWB se pueden alcanzar tasas de transmisión de 110 Mbps con un rango de alcance de 10 metros aproximadamente. Sin embargo, en su contra para que esta tecnología se consolide queda por resolver la poca distancia de transmisión, la contaminación electromagnética, la interoperabilidad, entre otros.

3.2.4 Z-WAVE

Z-Wave[43] es un protocolo para comunicación inalámbrica diseñado para aplicaciones remotas en el hogar como iluminación, accesos, detectores de humo,… Opera en el rango de los 900MHz perteneciente a la banda ISM, y tiene una cobertura de unos 30 metros.

Permite una topología mallada o mesh, que admite en torno a los 232 dispositivos, en la que puede haber uno o varios coordinadores y donde la tasa de transmisión alcanza los 50Kbps. Cada nodo permite enrutar información hacia otro lo que hace que el radio de cobertura crezca en relación con el de una sola unidad, lo que significa la existencia de ciertos nodos, llamados repetidores, no puedan dormir.

3.2.5 WIBREE - BLUETOOH ULTRA LOW ENERGY

Wibree[44] es una nueva tecnología digital de radio interoperable para pequeños dispositivos. Permite la comunicación entre dispositivos móviles, computadores y otros dispositivos más pequeños (de pila de botón). Está diseñada para que funcione con poca energía y a poca distancia, opera a una frecuencia de 2.4 GHz (ISM) y cuenta con una tasa de transferencia de 1 Mbps en la capa física. Además da soporte de seguridad utilizando el sistema de cifrado AES (Advanced Encryption Standard).

Las principales características de WiBree son que tiene un consumo de energía durante la conexión ordinaria 10 veces menor que Bluetooth 2.1, y es 50 veces más rápido en las transferencias de datos. Wibree tal y como está definido no es compatible con los actuales estándares Bluetooth. Desde junio de 2007, se le conoce como "Bluetooth Low Energy Technology" o “Bluetooth ULP” (Ultra Low Power).

3.2.6 IEEE 802.15.4

La norma IEEE 802.15.4[45], implementa las capas física (PHY) y de control de acceso al medio (MAC: Medium Access Control). Esta tecnología está enmarcada dentro de un subgrupo de las redes de área personal, las LR-WPAN (Low Rate WPAN, baja velocidad de transmisión), y está orientada a la implementación de redes de sensores inalámbricos y actuadores.

Dentro de las características más importantes se puede destacar que alcanza tasas de transmisión de hasta 250 Kbps, en un rango de cobertura de entre 10 y 100 metros y que opera en las bandas ISM a las frecuencias de 2.4 GHz, 868 MHz (Europa) y, 915 MHz (Estados Unidos).

Otra característica esencial es la optimización del consumo de los elementos que componen la red. Son los nodos finales (típicamente sensores o actuadores) los elementos con menor consumo, ya que permiten ser configurados de forma que permanezcan inactivos hasta que ocurra un determinado evento o tengan que enviar información. Este protocolo permite dos tipos de topologías en estrella y peer to peer en visión directa.

3.2.7 ZIGBEE

ZigBee[46] es un estándar hardware y software que parte del estándar IEEE 802.15.4. Incluye especificaciones en múltiples niveles del modelo de referencia OSI, desde el modo de uso de la capa física y MAC, hasta perfiles de usuario, descubrimiento de topologías, seguridad, y mensajería a niveles de red y aplicación. Al igual que IEEE 802.15.4 trabaja en la banda ISM a 2,4 Ghz, pero tiene una velocidad de transmisión menor.

3.2.8 OTRAS ESPECIFICACIONES

Se incluyen a continuación un conjunto de protocolos o soluciones tecnológicas apoyados en IEEE 802.15.4 para dar soporte a la gestión de redes de sensores:

- Wireless HART [47][48]: versión para comunicación inalámbrica del protocolo HART (Highway Addressable Remote Transducer Protocol) usado en aplicaciones industriales y automoción que requieren tiempo real. Puede operar según una topología mallada o estrella y, opera en la banda ISM de 2,4 Ghz.
- ISA - SP100 [49]: también es usado en el ámbito de la automoción. Se caracteriza por su bajo consumo, con una cobertura de hasta 100 metros, estando pensado para un monitoreo periódico y de control de procesos. Desarrollado por la ISA(Industrial Society of Automation), está en proceso de convertirse en estándar
- IETF IPv6 – LoWPAN [50]: (IPv6 over Low power Wireless Personal Area Networks) mplementa una adaptación de IPv6 sobre IEEE 802.15.4 para permitir comunicaciones eficientes a un dispositivo acceder desde y hacia internet. (IETF, Internet Engineering Task Force).

3.2.9 COMPARATIVA

Antes de proceder con la comparativa de las diferentes tecnologías enumeradas anteriormente, se muestran los requerimientos del sistema objetivo. Una red de sensores se caracteriza por:

- Tráfico no elevado, y no continuo
- Nodo operativo cuando requiera realizar alguna tarea, bien transmisión, recepción o cómputo.
- Baja tasa binaria.
- Complejidad de programación baja.
- Bajo consumo y larga vida de las baterías.

Partiendo de estas premisas se procede el análisis sobre qué tecnología es la más conveniente. Las alternativas inalámbricas anteriormente mencionadas trabajan en la misma banda ISM a excepción de UWB que trabaja desde los 3,1 a los 10,6 Ghz, banda recientemente legalizada.

Wi-Fi requiere una actividad casi ininterrumpida de los dispositivos en la red. Este estándar presenta una elevada tasa binaria, capaz de transmitir gran cantidad de datos entre

dispositivos, lo que supone un gran consumo de energía. Por lo que Wi-Fi queda descartada por su alto consumo.

UWB, por su parte, aunque energéticamente podría ajustarse y proporciona una elevada tasa de transferencia, el radio de cobertura de en torno a 10 metros es pequeño. Esto unido a la escasa disponibilidad de dispositivos en el mercado hace que se descarte.

Otra tecnología que se podría considerar, aunque no haya sido analizada en profundidad, es RFID pasiva, pero su reducida cobertura, de unos pocos centímetros se descarta. RFID activa aunque con mayor cobertura no permite el empleo de dispositivos inteligentes ni la implementación de las características de comunicaciones previstas para el sistema. Igualmente, aunque no se haya mencionado en apartados anteriores, la comunicación infrarroja a pesar de tener una cobertura de hasta 200 metros, presenta como mayor problema la necesidad de tener que evitar la presencia de obstáculos entre los nodos, lo que implicaría que toda la red debería tener en línea de vista directa.

Bluetooth está diseñado para la transferencia de datos a distancias cortas. IEEE 802.15.4 tiene un consumo similar, pero menor consumo en el estado de espera. Esto es debido a que los dispositivos en redes Bluetooth deben dar información a la red frecuentemente para mantener la sincronización, así que no pueden ir fácilmente a modo de sueño. Además el tamaño de las piconets tiene un valor máximo de 8 nodos. No obstante su necesidades de establecer el pareado entre los nodos, dificulta su despliegue en redes de sensores.

IEEE 802.15.4 en cambio, se trata de un protocolo cuya tasa de transmisión no supera los 250Kps, y con una cantidad de datos baja es más que suficiente. No precisa de gran capacidad de cómputo, ni de grandes potencias de transmisión. Precisamente el objetivo buscado. Zigbee por su parte, añade overhead por lo que la velocidad de transmisión bajará. Al contrario que 802.15.4 que permite la comunicación por visión directa, Zigbee enruta paquetes a través de nodos intermedios, es decir permite una topología mucho más compleja, que en el presente proyecto no es necesaria.

En cuanto a consumo, 802.15.4 ofrece mayores prestaciones como se puede observar en la Tabla 3-1.

Existen también soluciones propietarias como Z-Wave, Wibree y, otras no descritas como EnOcean[51], ANT[52], MiWi[53], etc., que a pesar de ser robustas y sencillas tienen el inconveniente de ser difícilmente escalables, no siendo interoperables y forzando al cliente a permanecer con un único fabricante, con las limitaciones que eso supone.

Es por ello que si se desea disponer de una solución interoperable y escalable, es imprescindible elegir una tecnología abierta. De entre todas mencionadas anteriormente, resulta especialmente interesante IEEE 802.15.4, que se ajusta más al despliegue de redes o aplicaciones en las que no es necesaria una gran velocidad de transmisión, pero sí es importante minimizar la complejidad, el consumo y el coste, al tiempo que permite la interconexión de una gran cantidad de nodos. Adicionalmente, para los requerimientos establecidos IEEE 802.15.4 es mucho más eficiente energéticamente que todas las tecnologías anteriormente mencionadas y, por lo tanto, mucho más adaptado al uso en redes de sensores.

Protocolo	802.15.4	Bluetooth	Wi-fi	UWB	Z-Wave	Wibree
Alcance	100m	10m	10-100m	10m	1-30m	1-10m
Banda de frecuencias	2,4Ghz	2,4Ghz	2,4-5 Ghz	3,1-10,6Ghz	900Mhz	2,4Ghz
Régimen binario	250Kbps	1Mbps	11,54,300Mbps	110Mbps	40Kbps	1Mbps
Consumo	1,8 mA en Tx 5.1 µA en reposo	40 mA en Tx 0.2 mA en reposo	400 mA en Tx 20 mA en reposo	--	23 mA en Tx 3 uA en reposo	--

Tabla 3-1 Comparativa de tecnologías radio de transmisión

3.3 IEEE 802.15.4

El estándar IEEE 802.15.4, tal como se ha anticipado anteriormente, va a ser la elección como norma de comunicación entre los nodos de la red implementada en este proyecto. Por esta razón a continuación se realiza una descripción más detallada.

3.3.1 ESTÁNDAR

IEEE 802.15.4 es el estándar que define la capa de nivel físico y parte de enlace (MAC) para redes inalámbricas de área personal, PAN, y de baja tasa de transmisión de datos (conocidas por su acrónimo en inglés LR-WPAN {Low Rate}).

Nace en 2003 de la necesidad de tener una norma que defina la comunicación en una red de múltiples nodos, los cuales deben tener una baja tasa de datos, baja complejidad, bajo consumo de energía, corto alcance y bajo coste, además de ser sencillos de instalar y hacer uso de bandas no licenciadas.

Entre sus principales características están:

- Tasas de transferencia de datos de 20 Kbps, 40 Kbps, 110 Kbps, 250 Kbps o 851 Kbps dependiendo del rango de frecuencias que se emplee.
- Configuración de red en estrella o mallado total (peer-to-peer).
- Direccionamiento corto de 16 bits o extendido de 64 bits.
- Asignación de intervalos de tiempo garantizados (GTS, Guaranteed Time Slot).
- Uso de CSMA-CA como técnica de acceso al medio.
- Mecanismo de asentimientos para garantizar fiabilidad en las transmisiones.
- Consumo de energía muy pequeño.
- Control de potencia, e indicación de calidad del enlace (LQI12, Link Quality Indicator).

3.3.2 ARQUITECTURA

En la Figura 3.5, se puede observar el nivel de ejecución de 802.15.4 comparándolo con el modelo de capas OSI.

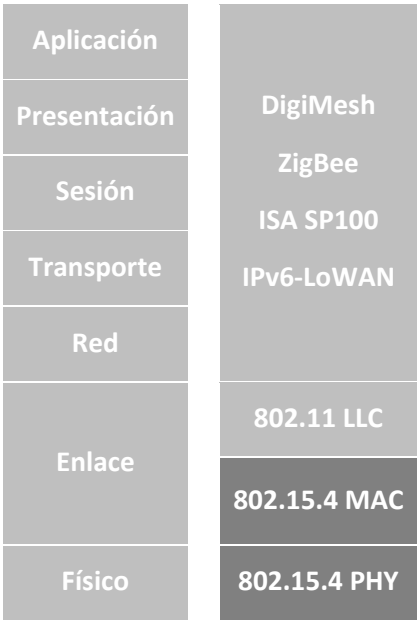

Figura 3.5 Modelo OSI

Similar a todos los estándares inalámbricos IEEE 802, como se muestra en la figura, el estándar IEEE 802.15.4 normaliza sólo las capas física o PHY y de control de acceso al medio o MAC. Por encima pueden ir otros protocolos como Zigbee.

La capa física se encarga de interactuar con el medio físico de transmisión así como con la capa MAC. Esta capa provee dos servicios a la capa MAC: Servicio de Datos PHY, que habilita la transmisión y recepción de Unidades de Datos de Protocolo PHY (PPDU) a través del canal de radio; y el Servicio de Administración, el cual brinda mecanismos para el control y configuración de la interfaz de radio desde la capa MAC.

La capa MAC, una sub-capa de la capa de enlace, se encarga de brindar acceso a los canales físicos para todo tipo de comunicación. En la se ve que la capa de enlace está formada por la subcapa 802.2 LLC (Logical Link Control) y, la sub-capa MAC. 802.2 LLC es común en todos los estándares IEEE 802 y accede a la sub-capa MAC, sin embargo su explicación está fuera del alcance del estándar IEEE 802.15.4.

La sub-capa MAC ofrece los siguientes servicios a las capas superiores: Servicio de Datos MAC (MCPS, MAC Common Part Layer), el cual permite enviar y recibir datos a la siguiente capa superior; y el Servicio de Administración MAC (MLME, MAC Layer Management Entity), que brinda mecanismos para el control y configuración de comunicaciones, interfaz de radio y creación de redes desde la siguiente capa superior.

3.3.3 CAPA FÍSICA

3.3.3.1 CARACTERISTICAS

Utiliza la técnica DSSS, Direct Sequence Spread Spectrum, para modular la información antes de ser enviada a la capa física. Básicamente cada bit de información se modula en 4 señales distintas, lo que hace que ocupe un gran ancho de banda. En cambio usa una densidad espectral de potencia baja para cada señal. Esto hace que sea inmune a interferencias en la banda de frecuencias de uso, y mejora la relación a ruido, SNR, en el receptor. De manera que la detección y decodificación son más fáciles.

Una característica de las redes sobre IEEE 802.15.4 es que trabajan en bandas no licenciadas. Según la región geográfica las bandas frecuenciales de operación son:

- 868 – 868.6 Mhz (Europa)
- 902 – 928 Mhz (EEUU)
- 2400 – 2483.5 Mhz (Mundial)
- 3100 – 10600 Mhz (variable dependiendo del país)

Se tienen 27 canales, 16 en la banda 2400 MHz, 10 en la banda 915 MHz y 1 en la banda 868 MHz. El coordinador de una red se encarga de escoger el canal adecuado, para esto realiza un escaneo de todos los canales y elige el mejor, que por lo general es el canal con menos interferencias.

3.3.3.2 FUNCIONES

Las tareas que realiza la capa física IEEE 802.15.4 son:

- Activación y Desactivación del transceptor de radio.
- Detección del nivel de energía en el canal de trabajo.
- Brindar un Indicador de Calidad de Enlace (LQI, Link Quality Indicator) de los paquetes recibos, basados en su nivel de potencia.
- Realizar la valoración de canal libre (CCA, Clear Channel Assessment), utilizado por la Sub-Capa MAC para el algoritmo CSMA/CA (Acceso Múltiple por Detección de Portadora con Evasión de Colisiones). Elegir el canal de frecuencia de trabajo.
- Transmisión a través del canal físico de los mensajes.

3.3.4 SUB-CAPA MAC

Las tareas que realiza la sub-capa MAC IEEE 802.15.4 son:

- En el caso de nodo coordinador, debe generar señales de sincronización, asociación/desasociación.
- Brindar diferentes servicios de seguridad.
- Utilizar el mecanismo CSMA/CA para el acceso al canal.
- Enviar tramas de acuse de recibo.
- Administrar el mecanismo de las ranuras de tiempo dedicadas.
- Administrar la asociación y desasociación de nodos en la red PAN.

Una de las ventajas de la capa MAC en el IEEE 802.15.4 es que sólo dispone de 21 primitivas de servicio o comandos, frente a las 131 primitivas de Bluetooth. Esto hace que el hardware que se utiliza pueda ser más sencillo y económico. El estándar de la capa MAC implementa dos modos de funcionamiento: el modo Beacon y el modo Beaconless.

3.3.5 Mecanismos de robustez

3.3.5.1 CSMA/CA

Es necesario resaltar la implementación del mecanismo de CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) en la transmisión de mensajes para evitar colisiones con otros mensajes que están siendo transmitidos en el mismo canal por otro nodo.

Cuando se transmite un mensaje, la sub-capa MAC solicita a la Capa Física que realice una Valoración de Canal Libre (CCA, Clear Channel Assessment), si se detecta que el medio está libre entonces se procede a transmitir, de lo contrario, si el medio está siendo utilizado, la sub-capa MAC espera un periodo de tiempo aleatorio (Random Back-Off Period) y después reinicia el proceso.

3.3.5.2 ACUSE DE RECIBO

Al enviar un mensaje, si se ha recibido satisfactoriamente, el nodo receptor se encarga de notificar al origen que se recibió con éxito. Esto se hace mediante un mensaje de acuse de recibo (acknowledgment frame o simplemente ACK). Estos mensajes son enviados sin utilizar CSMA/CA.

3.3.5.3 OTROS

IEEE 802.15.14 incorpora un control de secuencia (FCS) para validar la integridad de los datos.

3.3.6 Elementos de una red 802.15.4

Existen dos tipos de dispositivos según su hardware y software: los Dispositivos de Función Completa (FFD), que son capaces de hacer uso de todos los servicios de la MAC IEEE 802.15.4; y los Dispositivos de Función Reducida (RFD), que no pueden hacer uso de todos los servicios de la MAC porque tienen hardware limitado.

Además, los nodos de la red pueden ser clasificados según el rol que cumplen:

- Coordinador PAN: Se encarga de asignar un ID (identificador) a la red, encontrar el canal de trabajo adecuado, asignarse una dirección corta, administrar la asociación de nuevos nodos y según la topología retransmitir mensajes entre nodos. Por lo que tiene que ser necesariamente un FFD.
- Router: En una red con topología de árbol, aparte del coordinador PAN, también puede existir un Coordinador o Router, que se encarga de administrar las solicitudes de asociación a la red y retransmitir mensajes entre nodos. También debe ser un FFD.
- Dispositivo: Es un nodo que no cumple funciones de coordinador, siendo FFD o RFD.

3.3.6.1 TOPOLOGÍA DE LA RED

Cómo se comentó anteriormente en el apartado 3.1.3, los nodos se pueden agrupar lógicamente en estrella o formar una red peer-to-peer.

3.3.7 HARDWARE IEEE 802.15.4

3.3.7.1 MODULOS XBEE DIGI

XBee[53] describe una familia de módulos radio, compuestos básicamente por un transmisor/receptor RF y un microprocesador. Los módulos están diseñados para dar interoperabilidad entre múltiples plataformas, incluyendo topologías punto-multipunto, Zigbee/Mesh, basadas en el estándar 802.15.4. Trabajan en 2.4 GHz y 900 MHz con una tasa binaria de 250 Kbps.

Actualmente, y en función del módulo RF implementado, XBee puede trabajar con:

- **ZigBee**: Módulos XBee, XBee-PRO ZigBee y XBee-PRO ZB ofrecen solución a redes ZigBee mesh. La familia ZigBee PRO son compatibles con otros dispositivos ZigBee.
- **DigiMesh**: es un protocolo propietario de Digi, basado en el estándar 802.15.4, para redes malladas. Este protocolo incluye funciones de descubrimiento de nodos, detección de nodos caídos, y soporte para nodos routers que pueden entrar en estados de bajo consumo, entre otras características. Por todo ello es una solución adecuada en donde el uso de baterías es crítico.
- **IEEE 802.15.4**: Módulos XBee y XBee-PRO 802.15.4 OEM RF usan el protocolo 802.15.4 para redes "peer to peer" y punto multipunto. Están diseñados para aplicaciones de alto throughtput que requieren baja latencia y tiempos predecibles de comunicación.

Los módulos XBee permiten comunicaciones robustas en configuraciones punto a punto, peer to peer y multipunto/estrella y una fácil configuración maximizando el rendimiento en la comunicación. Además, pueden ser fácilmente configurados desde un ordenador utilizando el programa X-CTU[54], desarrollado por Digi, o bien desde un microcontrolador. Estos módulos incluyen dos modos de funcionamiento, API (Application Programming Interface) y AT (o modo comandos). Por todo ello, hoy en día tienen una alta aceptación entre los desarrolladores y proveedores de servicios.

3.3.7.2 OTROS FABRICANTES

Además del mencionado anteriormente, existen una serie de fabricantes que desarrollan módulos con características similares por ejemplo:

- Módulo RP-M100 de FirmTech[56].
- ProBee-ZE10 y ProBee-ZE20S de Sena Technologies [57].
- ProFLEX01 y SiFLEX02 de LS Research[58].

3.4 PLATAFORMAS DE NODOS SENSORES

Una vez estudiado su hardware de comunicación, se plantean diferentes plataformas que permitan implementar la inteligencia de las motas. Tanto el hardware como el software de las motas debe ser configurable con cierta flexibilidad, lo que deriva en que existan en el mercado multitud de sistemas.

3.4.1 CROSSBOW

Crossbow[59][60] es un fabricante de motas especializado que desarrolla plataformas hardware y software que da soluciones para las redes de sensores inalámbricas. Entre sus productos se encuentran las plataformas Mica, Mica2, MicaZ, Mica2dot, Telos y TelosB.

Las motas formas redes tipo Adhoc, alguna de sus características son que permite el descubrimiento de rutas, y que son auto-organizativas. Además, utiliza el sistema operativo TinyOS encargado de las comunicaciones radio y, de minimizar el consumo manteniendo la mayor parte del tiempo al sistema durmiendo, y sólo cuando ocurre un evento se despierta para atenderlo. El lenguaje utilizado para programar las motas es el "NesC"[61], parecido a C, y es complilado por Cygwin[62] (un emulador de Linux).

Además de Crossbow existen otras soluciones como MOTEIV, que desarrolla las plataformas Tmote Sky y Tmote Invent. Y Shockfish que desarrolla la plataforma TinyNode[63].

A continuación se ahondará en algunas de las plataformas mencionadas.

3.4.1.1 TELOSB

TelosB[64] es una plataforma para aplicaciones en redes de sensores de muy bajo consumo y alta recolección de datos. Lleva integrada sensores de temperatura, humedad y radiación, una antena, un microcontrolador y, pueden ser fácilmente programados.

El microcontrolador es el MSP430 F1611 (procesador RISC de 16 bits) que permanece en estado de sueño la mayor parte del tiempo, se activa tan rápido como puede para procesar, enviar y, una vez finalizado vuelve al estado de sueño. De forma que su consumo en modo activo es de 1,8 mA y, de 5,1 µA en estado de sueño.

Utiliza un controlador USB del fabricante FTDI para comunicarse con un ordenador y lleva el módulo de radio CC2420 (de Chipcon), el cual envía y recibe bajo el estándar IEEE 802.15.4, a una frecuencia de 2,4Ghz y con una velocidad de transmisión de 250Kbps. La antena puede ser apagada para ahorrar energía y, tiene una cobertura de 50 metros en interiores y, 125 metros en exteriores. En la Figura 3.6 se puede observar una mota TelosB:

Figura 3.6 TelosB

Utiliza el sistema operativo TinyOS[2] en el enrutamiento e implementación de las comunicaciones. Y se alimenta con 2 baterías de tipo AA, aunque igualmente lo puede hacer a través de su puerto USB (que además sirve para programación). También proporciona la capacidad de añadir dispositivos adicionales mediante los dos conectores de expansión de los que dispone, estos pueden ser configurados para controlar sensores analógicos, periféricos digitales y displays LCD. Entre las distintas casas que lo fabrican se encuentran MoteIV y, Crossbow.

3.4.1.2 MICAZ

MicaZ [65] es un tipo de mote fabricado por la empresa Crossbow Technology, que puede ser utilizado para formar redes inalámbricas de sensores de bajo consumo. Entre algunas de sus características está que trabaja a la frecuencia de 2,4Ghz, cumple la especificación del IEEE 802.15.4, alcanza una tasa binaria de 250Kbps, cuenta con entradas analógicas, digitales, soporte de I2C, SPI (Serial Peripheral Interface) y UART (Universal Asynchronous Receiver-Transmitter), y utiliza el sistema operativo TinyOS.

En la Figura 3.7 podemos ver el aspecto exterior de este mote, donde al igual que en modelos de otros fabricantes el mayor espacio es el ocupado por las 2 baterías del tipo AA.

Figura 3.7 MicaZ

[2] TinyOS es un sistema operativo de código libre, pensado para crear códigos que optimicen los escasos recursos de un microcontrolador que controla una red de sensores y un módem inalámbrico.

3.4.2 WASPMOTE

En el año 2009 la plataforma Waspmote[66] fue lanzada por la empresa Libelium (Spin-off de la Universidad de Zaragoza). El proyecto se centró en conseguir una plataforma de muy bajo consumo, alcanzando así los 0,7uA en modo hibernación. Hace uso de los módulos radio Xbee, lo que le permite trabajar a las frecuencias de 2,4GHz, 900Mhz y 868MHz (bandas ISM), y con los protocolos IEEE 802.15.4 y Zigbee. En la Figura 3.8 se muestra el mote con un módulo de radio, entrada USB, tarjeta SIM, lector de tarjetas MMC/SD y el conector de expansión.

Estos módulos están basados en una arquitectura modular por lo que es sencillo ampliar sus capacidades. Para ello se han desarrollado diferentes escudos o shields siendo algunos de los más utilizados los de GPS, GPRS, lectores de tarjetas SD, sensores para monitorización ambiental CO_2, O_2, presión atmosférica, de luminosidad, vibración, nivel de líquidos y módulos de entrada salida genéricos como ADC, entradas digitales, etc.

Figura 3.8 Waspmote

3.4.3 ISENSE

iSENSE[67] es una plataforma hardware y software para redes inalámbricas. Están constituidos principalmente con microcontroladores RISC de 32 bits y trabajan con la norma IEEE 802.15.4. Pero, además, está disponible un protocolo propietario mesh o para redes en malla, que también incluye en capas superiores soporte para conexión a internet tanto en IPv4 como IPv6. Son programables a través del lenguaje C y, están fabricados por la empresa Coalsenses. Existe una variedad de módulos según su función como “Core Module 3”, o “Gateway Module” y distintos escudos o shields que los complementan. En la Figura 3.9 se pueden observar algunos de ellos. Para más información se puede acudir a la página web del fabricante[68].

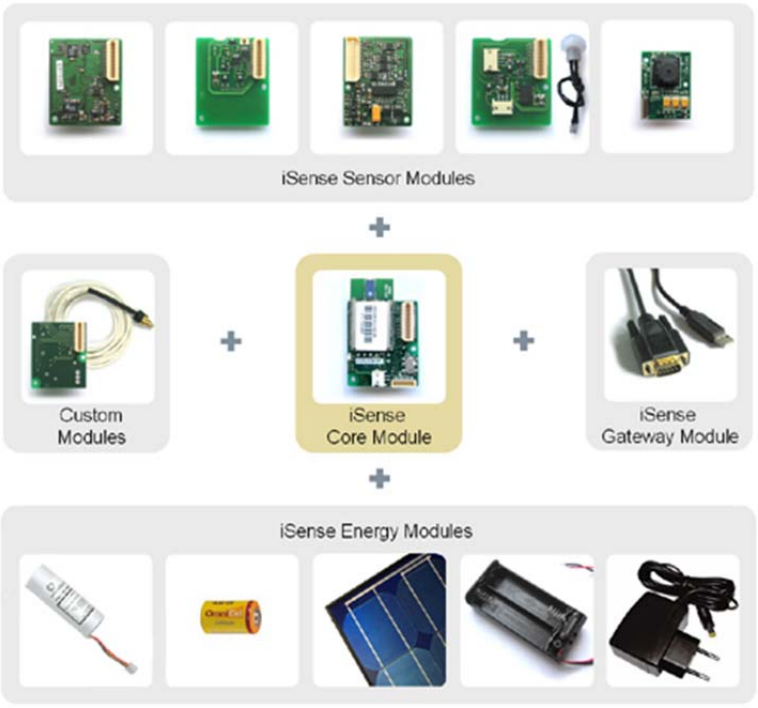

Figura 3.9 Módulos iSense

3.4.4 ARDUINO

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. No es más que una placa con un circuito que comunica un procesador con diferentes puertos de entrada o salida, y con una memoria E^2PROM. Esta memoria actúa a modo de pequeño disco duro, donde se almacenan los programas diseñados a ejecutar. Su unidad de procesamiento está basada en los microcontroladores Atmega168 o Atmega8.

Figura 3.10 Arduino UNO

Arduino puede tomar información del entorno a través de transductores conectados a los múltiples pines de entrada y, operar como actuador mediante los interfaces de salida a los que se pueden conectar luces, motores, etc. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino[70], basado en Wiring[71], empleando el entorno de desarrollo Arduino, basado en Processing[72]. Tanto el diseño de la placa como el software de programación están disponibles bajo licencia abierta y cualquiera es libre de adaptarlo a sus necesidades. Un ejemplo de ello son los sistemas de Libelium, como el WaspMote que se pueden considerar un fork del sistema Arduino.

Lo que hace tan atractivo Arduino es su facilidad para aprender a utilizarlo y, su sencillez al programarlo, además del gran número de librerías disponibles, fruto de la gran comunidad de usuarios de que dispone.

Existen una gran variedad de diseños de placas Arduino, creadas según requerimientos específicos como la Arduino FIO, la EthernetDUINO, entre otras [69] y de proyectos basados en Arduino, como Mosquino, ZArdino, Imaguino... creadas para una función específica y, totalmente compatibles con el IDE de Arduino[73].

3.4.5 PINGÜINO

Pingüino[74][76] dio sus primeros pasos en el año 2008, se trata de una plataforma de hardware y software libre para la experimentación con microcontroladores. Es similar a Arduino pero basada en un microcontrolador PIC18F2550, de estructura RISC (Reduced Instruction Set Computer), que incluye un módulo USB nativo y una UART para la comunicación serie. Cuenta con su propio Entorno de Desarrollo Integrado (IDE) de uso y apariencia similar al de Arduino, válido para Windows, Linux y MAC OS X.

El módulo USB integrado permite comunicarse directamente con el PC y reduce el costo del hardware, dejando además libre el puerto UART del microcontrolador para otras aplicaciones.

El circuito básico Pingüino es muy simple y sólo precisa de unos pocos componentes; todo el software necesario está disponible para bajarse de internet gratuitamente. Su IDE está basado en Python[76]. Cuenta con un preprocesador que traduce instrucciones Arduino directamente a C, lo que reduce la longitud del código y velocidad ejecución.

PowerJaguar es otro proyecto parecido que utiliza el mismo controlador PIC. Pero aún está en desarrollo. No cuenta aún con IDE propio.

3.4.6 RASPBERRY

Raspberry Pi[77] es una placa computadora (SBC) de bajo coste desarrollada en Reino Unido por la Fundación Raspberry Pi.

El diseño incluye un procesador ARM1176JZF-S a 700 MHz y 256 MB de memoria RAM con el objetivo de ejecutar un sistema operativo Linux o RISC OS.2 3. Utiliza una tarjeta SD para el almacenamiento permanente. (ARM: Advanced RISC Machine).

Se trata de un sistema embebido que se puede conectar a un monitor o a un televisor, cuenta con puertos USB, slot SD (donde se carga el sistema operativo), puerto HDMI, micro USB para alimentación y un puerto Ethernet. Dada su juventud aún no se encuentra debidamente probada para su uso como nodos sensores.

3.4.7 PICAXE

Este tipo de soluciones consiste en adquirir un controlador y montar el resto de la placa según tus necesidades. Picaxe[78] consiste en un microcontrolador basado en un chip PIC, que tiene cargado un bootloader que permite ser programado fácilmente con un simple programa en BASIC o un diagrama de flujo.

Existen múltiples versiones de este chip con diferente número de pins (8, 14, 18, 20, 28 y 40), dependiendo del requerimiento, configurables como salida/entrada digital y, entrada analógica... Además cuenta con interfaces PWM, I^2C, SPI y, RS232.

3.4.8 TST SISTEMAS

TST Sistemas [89] es una PYME creada en 2007 por investigadores de la Universidad de Cantabria cuyo objetivo es el de proyectar al mundo empresarial el know-how acumulado tras una intensa labor de investigación en el ámbito de tecnologías inalámbricas. En el desarrollo de nuevos dispositivos algunas características básicas en ellos son el desarrollo rápido de aplicaciones gracias a librerías API de TST, la programación sencilla en ANSI C y el uso de un potente micro controlador ARM de 32 bit de bajo consumo. Las placas más interesantes bajo el punto de vista de crear una red de sensores son el TSgaTe y el TSmoTe.

El TSmoTe es un sistema embebido diseñado para permitir desarrollos rápidos y sencillos de aplicaciones inalámbricas de monitorización, control remoto y soluciones M2M. Contiene múltiples interfaces de interconexión para conectarle sensores y actuadores

Figura 3.11 Modulo TSmoTe

El concentrador TSgaTe es una plataforma más potente que el TsMote y que posee otro tipo de requerimientos. Además posee un puerto Ethernet a través del cual o de alguno

de sus módulos de expansión el TSgaTe ejerce de Gateway entre dispositivos TSmoTe y aplicaciones software en servidores remotos.

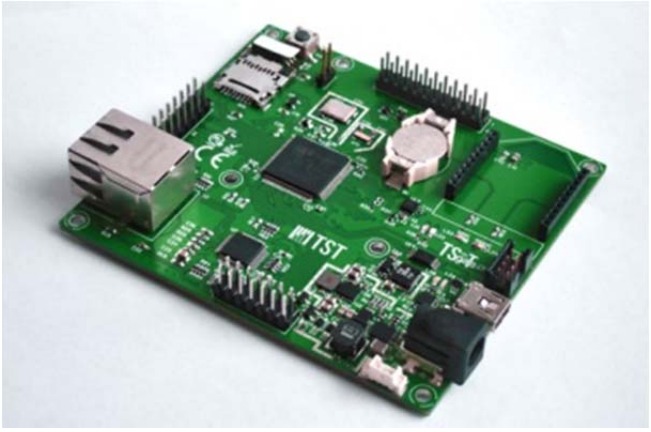

Figura 3.12 Modulo TSgaTe

DISEÑO Y VALIDACIÓN 4

A raíz del estudio realizado se ha identificado la solución basada en IEEE 802.15.4 como la más adecuada para su utilización en redes de sensores. Esto es así dado que su modo de operación supone un consumo de energía muy bajo manteniendo un buen balance con el rendimiento de las comunicaciones, tanto en alcance como en ancho de banda, para los requerimientos de este tipo de redes.

Si bien existen capas de control y gestión específica que operan sobre IEEE 802.15.4, en este proyecto se ha preferido trabajar de forma nativa para así tener un mayor control sobre la operativa del protocolo y de esa manera contar con la oportunidad de tener mayores posibilidades de experimentación.

El presente trabajo busca realizar una prueba de concepto de las posibilidades de una red de sensores desatendida y aplicarla a un caso real como puede ser el proyecto Smart Santander usando sus sensores como modelo de uso.

4.1 ARQUITECTURA SMART SANTANDER

El proyecto Smart Santander, como se explicó en la introducción, consiste en una gran red de sensores, en concreto 20.000 sensores para 4 diferentes ciudades europeas, de los cuales 12.000 se instalaran en Santander.

La red que se pretende instalar en Santander está compuesta de alrededor de 2000 IEEE 802.15.4 terminales desplegados en una arquitectura de 3 niveles:

- Nodo IoT. Es el responsable de detectar el parámetro que se desea medir. Están distribuidos conformando 23 clusters, cada uno con una topología de estrella. Entre el nodo y el clúster se forma una red inalámbrica en la que toda la información que se captura se envía a los Gateway.

- Repetidores. Estos nodos están colocados en lugares altos de la ciudad como farolas o semáforos para funcionar como nodos de reenvió de toda la información asociada a lo las diferentes medidas.

- Gateway. Es el terminal encargado de reunir toda la información que ha llegado a través de los nodos IoT y los repetidores y pasársela a la base de datos de Smart Santander

La comunicación entre los nodos de la IoT y los repetidores con los nodos que funcionan como Gateway se lleva a cabo a través del Testbed Runtime (TR). El TR crea una red superpuesta para facilitar el direccionamiento de los nodos y el intercambio de mensajes con nodos conectados localmente independientes de las conexiones actuales de red subyacentes. Se lleva a cabo el reenvío de mensajes y ofrece primitivas de comunicación que se utilizan para el control y la gestión de los experimentos y la propia WSN. El diseño del TR define los servicios de conexión que manejan los mensajes intercambiados con los nodos de la IoT. La arquitectura de la TR implica que existe una conexión por cada nodo de la IoT en el banco de pruebas al que se accede a través de un conector exclusivo.

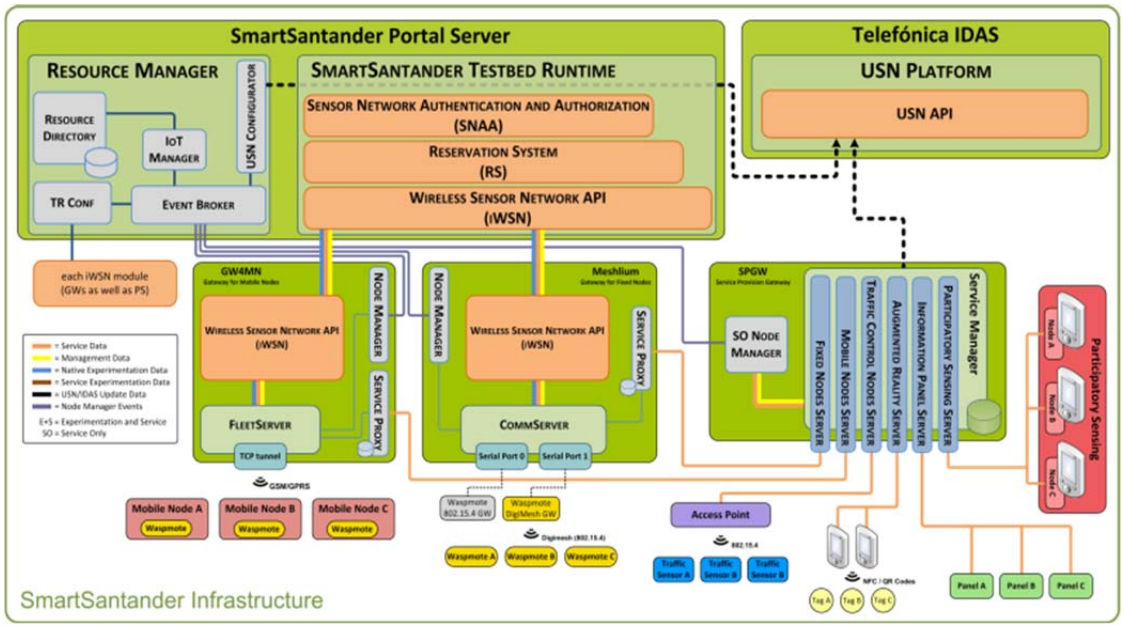

Figura 4.1 Infraestructura de la red Smart Santander[94]

En la Figura 4.1 se puede observar un resumen de la infraestructura usada en la red Smart Santander.

Actualmente en Smart Santander la red de sensores desplegada usa principalmente dispositivos de la empresa Libelium, entre los que se encuentran los Meshlium, Waspmote y Waspmote Gateway debido a que están orientados a desarrolladores de software y permiten una total interacción en la programación de los sensores.

4.2 PLATAFORMA HARDWARE

Como se ha comentado el presente proyecto se centra en la evaluación de los códigos Digital Fountain para su posterior despliegue en la plataforma del proyecto Smart Santander, descrita anteriormente. No obstante para no interferir sobre los trabajos que se estaban realizando durante el desarrollo del este proyecto se ha preferido emular el despliegue de Smart Santander a escala entorno. La red empleada se describe en la Figura 4.2

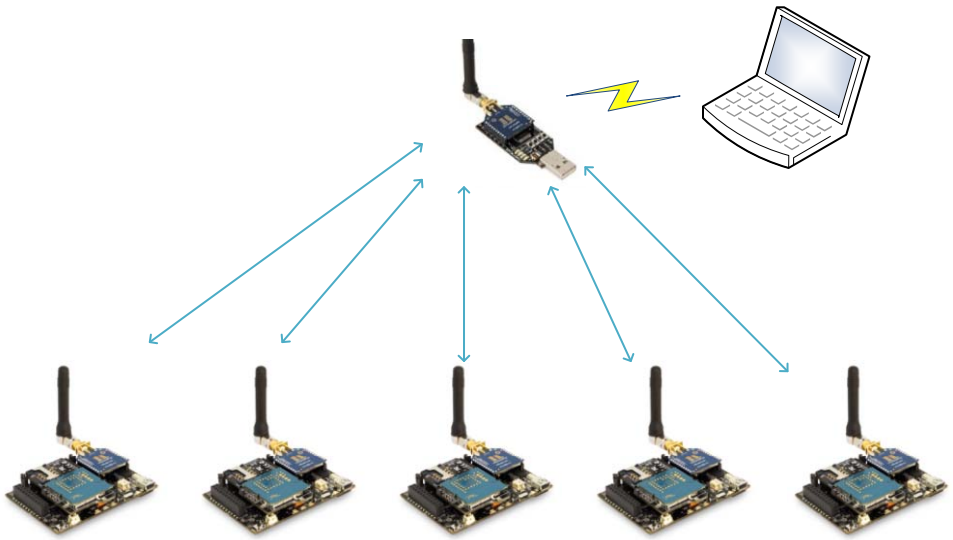

Figura 4.2 Arquitectura del proyecto

El diagrama presenta un Waspmote Gateway, que conectado a un servidor que ejecuta la inteligencia de la codificación Digital Mountain, actúa como emisor del conjunto de datos codificados emulando la transmisión de actualizaciones de firmware Over-The-Air de los Waspmotes. Estos recibirán los paquetes y los decodificarán para obtener el fichero original. Los paquetes llegan encapsulados en un formato de trama diseñado dentro del presente proyecto, que se describe más adelante.

4.2.1 MESHLIUM

Meshlium es un router Linux que actúa como un Gateway en la red de sensores de Waspmote. Puede contener hasta cinco diferentes interfaces radio: WiFi 2.4GHz, WiFi 5GHz, 3G/GPRS, Bluetooth y ZigBee. De la misma manera, también puede integrar un módulo GPS para aplicaciones móviles y opera mediante conexión a la red eléctrica, cargado con energía solar o por batería. En la Figura 4.3 se muestra un modelo de Meshlium. Dependiendo del tipo de interfaz radio que se desee usar, el radio de transmisión será mayor o menor. Por ejemplo, usando la interfaz WiFi AP en la banda de 2.4 GHz, se puede alcanzar una distancia máxima de

500 metros dependiendo también de la antena que se use y de la línea de vista del módulo, usando una potencia de transmisión de 20 dBm.

Los nuevos módulos de Meshlium, los Meshlium Xtreme, permiten detectar aparatos que empleen interfaces WiFi o Bluetooth. Este avance se enfoca a nuevas utilidades como por ejemplo conocer la cantidad de coches y gente que se encuentra en un lugar en un momento determinado, así como la evolución de la congestión del tráfico en una determinada zona.

Figura 4.3 Modulo Meshlium

4.2.2 WASPMOTE

Waspmote, sobre el que se han realizado los desarrollos de este proyecto, es un dispositivo que funciona como sensor orientado a desarrolladores de software. Funciona con diferentes protocolos (ZigBee, Bluetooth, GPRS) y frecuencias (2.4GHz, 868MHz, 900MHz) siendo capaz de alcanzar enlaces teóricos de hasta 12 kilómetros. Cuenta con un modo de hibernación que le permite ahorrar batería cuando no está transmitiendo o ejecutando algún cálculo. Dispone de un conjunto de librerías API y un compilador propio que permiten que el programador comience a trabajar con la plataforma de una manera rápida. Adicionalmente incluye la opción de emplear una tarjeta micro SD de capacidad hasta 32 GB que puede ser usada como almacenamiento extra puesto que la memoria interna de modulo es de 8 KB la SRAM, 4 KB la EEPROM y 128 KB la memoria FLASH. El modulo que se usó en el proyecto es el que se puede ver en la Figura 4.4.

Figura 4.4 Modulo Waspmote

Ahora se encuentra en el mercado una nueva versión del Waspmote, el Waspmote Plug & Sense!, que permite a los desarrolladores olvidarse de la parte electrónica y poder centrarse en la parte de servicios y aplicaciones. Con este nuevo módulo (Figura 4.5) se pueden desplegar redes de conexión inalámbrica de manera rápida y con el mínimo coste de mantenimiento. Además, esta encapsulado en una carcasa resistente al agua.

Figura 4.5 Modulo Waspmote Plug & Sense!

4.2.3 WASPMOTE GATEWAY

Este dispositivo sirve de enlace entre los equipos de desarrollo y los Waspmote. Posee las mismas características que el Waspmote salvo la de almacenamiento puesto que es un punto de acceso únicamente con el interfaz radio que redirige la información recibida hacía el equipo de control. En la Figura 4.6 se puede observar una imagen del Gateway.

Figura 4.6 Waspmote Gateway

4.3 FORMATO DE LAS TRAMAS

Los paquetes usan un formato especial puesto que los datos además de encapsularse sobre una trama de transmisión 802.15.4, que a su vez se encapsula dentro del protocolo digimesh, propietario de la empresa Digi, proveedora de los módulos IEEE 802.15.4 empleados. Además, la arquitectura de comunicaciones de los waspmote requiere el uso de cabeceras adicionales. Todo ello se puede observar en la Figura 4.7.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x16	
	Frame-specific Data	Frame Type	3	0x10	
		Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following address is also supported: 0x000000000000FFFF - Broadcast address
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x0A	
			11	0x01	
			LSB 12	0x27	
		Reserved	13	0xFF	Set to 0xFFFE.
			14	0xFE	
		Broadcast Radius	15	0x00	Sets maximum number of hops a broadcast transmission can occur. If set to 0, the broadcast radius will be set to the maximum hops value.
		Transmit Options	16	0x00	Bitfield: bit 0: Disable ACK bit 1: Don't attempt route Discovery. All other bits must be set to 0.
		RF Data	17	0x54	Data that is sent to the destination device
			18	0x78	
			19	0x44	
			20	0x61	
			21	0x74	
			22	0x61	
			23	0x30	
			24	0x41	
	Checksum		25	0x13	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Figura 4.7 Encapsulación de la trama de datos[98]

En la imagen se puede ver como los datos útiles (RF data) se encapsula dentro de una trama digimesh dejando un tamaño máximo de 73 bytes para el protocolo que se desarrolla a continuación. A su vez, los datos que nosotros queremos enviar se encapsulan usando unas cabeceras por lo que el tamaño máximo de la información útil se vuelve a ver disminuido.

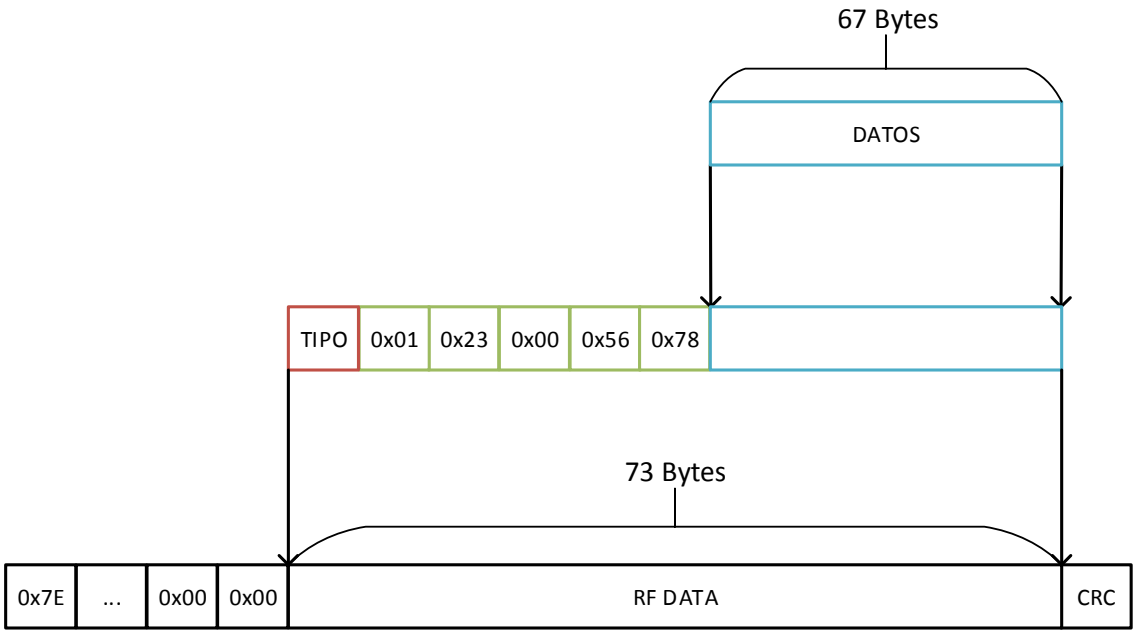

Figura 4.8 Encapsulación de trama de datos por ambos protocolos

Como se puede observar en la Figura 4.8, los 73 bytes comentados anteriormente se ven reducidos a 67 bytes debido a que los cinco bytes propios del protocolo Waspmote necesarios para su correcto funcionamiento, que son los que el recuadro es verde, y un byte para definir el tipo de trama, el recuadro que pone TIPO. El formato de este payload variará dependiendo del formato que apliquemos, siendo crucial el diseño para tratar de aprovecharlo al máximo y optimizar el envío de los paquetes.

4.4 Protocolos de Comunicación

Tal como se ha venido comentando a lo largo de esta memoria, los códigos Digital Fountain tienen un especial interés en la transmisión de datos de forma continua en entornos punto-multipunto. En esta situación, el nodo transmisor operando como fuente de contenido, codificará la información siguiendo los algoritmos descritos en el Capítulo 2. Por su parte, los nodos receptores decodificarán los datos recibidos y serán capaces de obtener libre de errores la información proporcionada por la fuente.

Buscando ajustar la implementación desde un inicio a este escenario, se han desarrollado un conjunto cliente-servidor, donde el servidor hará las funciones de transmisor y codificador de la información y el cliente incluirá las funciones de decodificación. Aunque en un sistema real, la fuente será continua y no será necesaria una petición por parte de los clientes para iniciar la transmisión, para mantener el sistema de experimentación más controlado y poder analizar más fehacientemente las diferentes métricas se ha decidido añadir mensaje de solicitud y finalización de la descarga con el que inicializar el sistema. Esta opción se empleará únicamente en la evaluación del sistema de codificación.

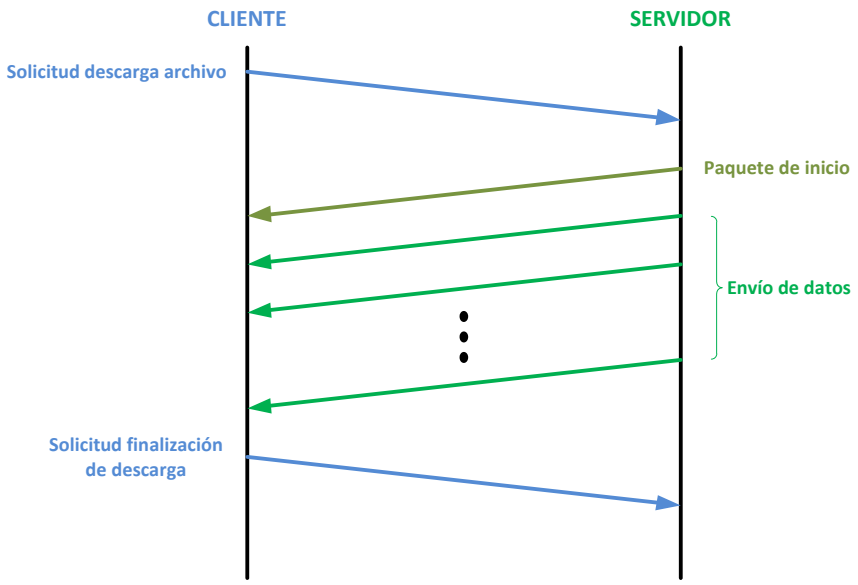

Figura 4.9 Esquema de comunicación cliente-servidor

El mecanismo de comunicación entre los procesos es sencillo, como se ilustra en la Figura 4.9. El cliente solicita la descarga de un fichero o bloque de información, el servidor le manda un paquete de inicio con los datos necesarios para, a continuación, iniciar el envío de los paquetes de datos. Los nodos clientes ajustan su configuración de manera automática según la información emitida. Una vez el cliente ha reconstruido el fichero, el cliente envía una última trama al servidor para que detenga el envío de paquetes. De esta forma se evita el inundar innecesariamente la red de sensores una vez que los nodos interesados han recibido toda la información.

Un aspecto importante en la implementación de la codificación es la elección de los parámetros delta y c. En este caso, se eligió como valor de delta 0.9 y como valor de c 0.1. Otro aspecto importante es que debido a la configuración básica de la comunicación entre Waspmote y el Gateway, el cliente envía una confirmación por cada paquete recibido, que es un factor que puede incluir cierto retraso en la comunicación ya que para la implementación de las técnicas Digital Fountain no se necesita confirmaciones de los paquetes.

Tomando como base este protocolo, a continuación se describen y evalúan las alternativas exploradas en el presente proyecto para el intercambio de datos haciendo uso de la codificación Digital Fountain. Las diferentes versiones diseñadas han buscado optimizar no solo el rendimiento en términos de uso de red, sino también en cuanto a carga computacional en los nodos.

4.4.1 Primera Versión

En el primer protocolo, se realizó el esquema detallado anteriormente, y se iban almacenando los paquetes descodificados en la memoria interna de propio Waspmote. Una vez finalizado el protocolo y realizando las primeras pruebas se pudo observar que debido al uso de las distintas librerías que se necesitan para poder ejecutar un programa en el Waspmote, la memoria del dispositivo se veía reducida en gran cantidad, dejando muy poco

espacio para poder almacenar los paquetes descodificados y provocando fallos en la recepción de los paquetes y que el programa se quedase sin memoria cuando se ejecutaba. Se partió de protocolo creado por Cristina Ysart en el proyecto "*Prestaciones de las técnicas Digital Fountain sobre Infraestructuras Inalámbricas*"[96]. En este protocolo lo que se hace es enviar el grado y el contador de paquetes. El formato de la trama es el que se muestra a continuación.

Figura 4.10 Formato trama Diseño Cristina Ysart

La codificación basada en Digital Fountain se sustenta en la generación aleatoria tanto del grado del paquete como de los identificadores de los bloques. La estructura planteada implica que todos los nodos emplean la misma semilla como referencia para todos los cálculos. De esta forma y con el contador de paquetes son capaces de sincronizar el generador de números aleatorios

Para el caso que nos ataña hemos detectado una problemática en esta operativa. Con un servidor de fuente continua que está distribuyendo durante días el mismo contenido, los clientes que se conectan relativamente poco después del inicio de la transmisión, no tienen problemas en empezar a calcular los datos; pero si se supone un cliente que se conecta al cabo de varios días, el retardo que supondría el calcular los datos referentes a los paquetes anteriores que no ha recibido hasta llegar al suyo, sería prohibitivo.

Es por ello que se decide mejorar la implementación, y partiendo de una semilla base se crea un número aleatorio que será la semilla para una función random nueva que en vez de usar una semilla normal, usa un contexto de semillas. Esta solución no implica correlación entre los valores generados a partir de las diferentes semillas, por lo que se puede considerar que la aleatoriedad sigue existiendo. Para desplegar esta solución es necesario que tanto el servidor como los clientes compartan el mismo generador de códigos aleatorios. Las pruebas preliminares y el análisis del código de las funciones del servidor y los clientes (waspmote) revelaron que no compartían la misma metodología. Por lo tanto se decidió mantener el generador de los waspmote y trasladarlo al servidor. A continuación se muestra el código de estas funciones.

```
/*GENERADOR ALEATORIO*/
int do_rand1(unsigned long *ctx)
{
#ifdef  USE_WEAK_SEEDING

        return ((*ctx = *ctx * 1103515245L + 12345L) %
                ((unsigned long)32767 + 1));
#else   /*
         * Compute x = (7^5 * x) mod (2^31 - 1)
         * wihout overflowing 31 bits:
         *      (2^31 - 1) = 127773 * (7^5) + 2836
         */
        long hi, lo, x;

        x = *ctx;
        /* Can't be initialized with 0, so use another value.
*/
        if (x == 0)
                x = 123459876L;
        hi = x / 127773L;
        lo = x % 127773L;
        x = 16807L * lo - 2836L * hi;
        if (x < 0)
                x += 0x7fffffffL;
        return ((*ctx = x) % ((unsigned long)32767 + 1));
#endif  /* !USE_WEAK_SEEDING */
}


int rand_r1(unsigned long *ctx)
{
        return do_rand1(ctx);
}


static unsigned long next = 1;

int rand1(void)
{
        return do_rand1(&next);
}


void srand1 (unsigned int seed)
{
        next = seed;
}
```

Pero para poder usar este código, hay que comprobar que cumpla los requisitos de los códigos Digital Fountain y el generador pseudoaleatorio tiene que tener una distribución uniforme. Las ejecuciones realizadas empleando este generador revelan la uniformidad, tal como se puede ver en la Figura 4.11. Aunque no se trata de una distribución uniforme

perfecta, sí se puede considerar válida para el sistema que se está desarrollando en este proyecto.

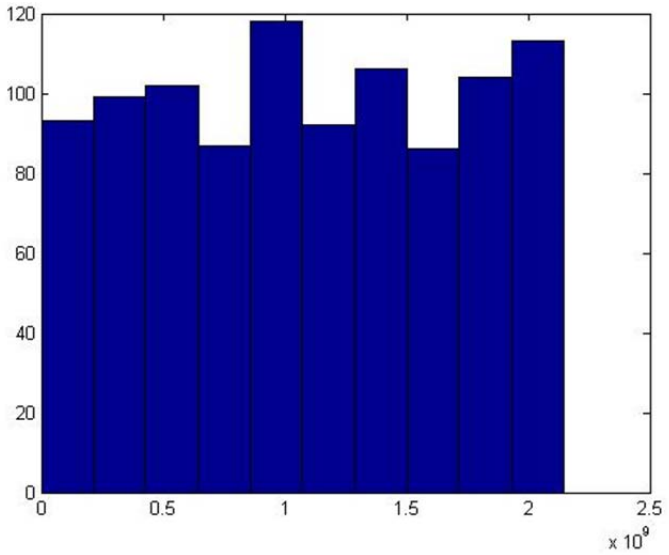

Figura 4.11. Histograma generador pseudoaleatorio

Aplicando estas modificaciones, en la trama inicial que envía el servidor al cliente se envía el tamaño de fichero a enviar, a partir del cual se puede obtener tanto el tamaño de la ventana como el tamaño del paquete de datos que serán esenciales para el progreso de la decodificación.

El payload de los paquetes que le llegaban al cliente del servidor tenía la estructura mostrada en la Figura 4.12.

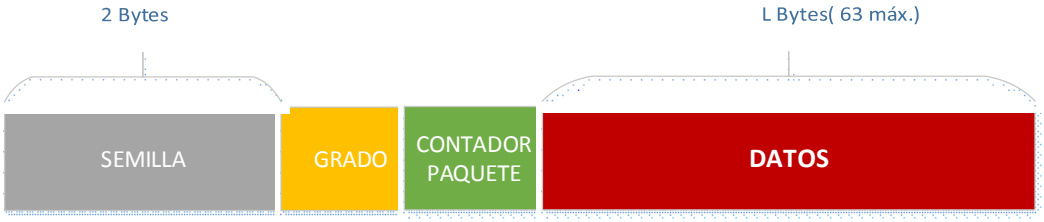

Figura 4.12 Payload de la trama de datos para la primera versión del protocolo

Se envía la semilla y el grado a partir de las cuales se pueden calcular los vecinos de los que depende el paquete. De esta manera se reduce mucho el gasto computacional puesto que solo que habrá que ejecutar la función random tantas veces como vecinos tenga el paquete, mientras que de la otra manera había que ejecutarlo tantas veces como paquetes se han recibido y luego calcular los vecinos. El contador de paquete se envía solo para realizar pruebas del rendimiento del protocolo. Esta modificación supone una mejora bastante significativa. Al enviar la semilla y el grado y el contador del paquete en el paquete se pueden enviar hasta 63 bytes de información.

Un ejemplo práctico de la trama de un paquete de datos se puede ver en la Figura 4.13 en el que se pueden ver los diferentes encapsulamientos que se realizan de la trama de datos.

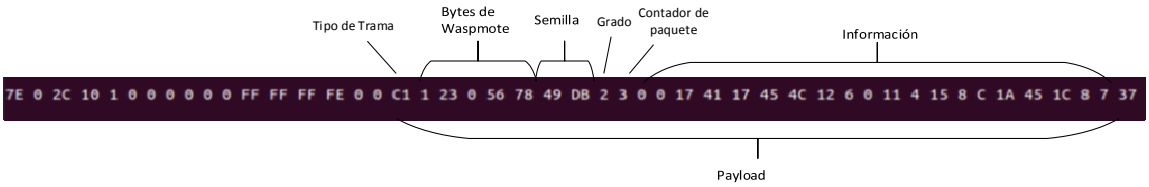

Figura 4.13 Trama real de datos para la primera versión del protocolo

El esquema de decodificación que sigue el cliente se puede ver en la Figura 4.14 en la que como se ve para que pueda empezar la decodificación se necesita que llegue un paquete de grado 1. Mientras que no llegue algún paquete de grado 1 se almacenan los paquetes que van llegando. Al recibir un paquete de grado 1, se guarda en la lista de paquetes decodificados y se busca en la lista de los paquetes sin decodificar si alguno se puede decodificar, en caso positivo se guarda y se vuelve a buscar, en caso de que no se pueda se recibe otro paquete y se vuelve a comprobar su posibilidad de decodificación y se repite el proceso de guardado y búsqueda hasta que se complete el fichero.

Tras realizar las pruebas se observa que este modo de funcionamiento requiere que el Waspmote tenga mucha memoria y potencia de cálculo. La memoria se necesita porque en el caso que se quiera enviar un firmware de gran tamaño, éste hay que almacenarlo en la memoria interna y es por ello que se llega a la conclusión de que hay que usar parte del almacenamiento en la SD del Waspmote. Otro problema que surge es que si el firmware es grande, esto implica que serán muchos trozos los que hay que decodificar, trabajo que con esta operativa y la potencia de cálculo del Waspmote se hace inviable.

Es por todo es por lo que hay que pensar una segunda versión que nos dé una solución a estos problemas y que se adecue de una mejor manera a las características de la plataforma que estamos empleando.

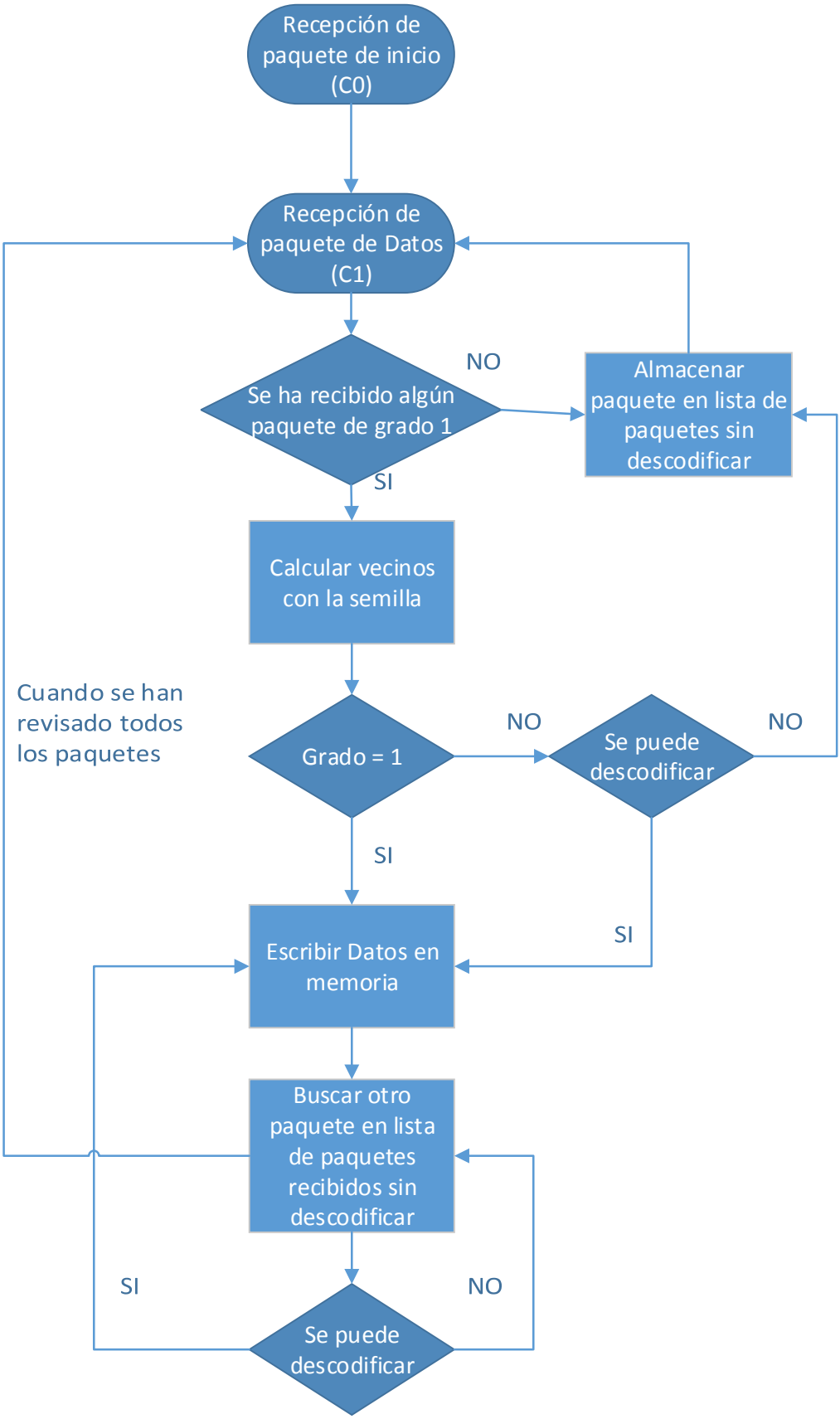

Figura 4.14 Diagrama de flujo de la primera versión del protocolo

4.4.2 Segunda Versión

El segundo protocolo se puede considera una extensión natural del primero. Si dado que no podemos realizar la transmisión porque no tenemos memoria suficiente debido al gran tamaño de firmware, la idea que se plantea es particionar el firmware que se manda en distintos subbloques más pequeños, a los que hemos denominado ventanas.

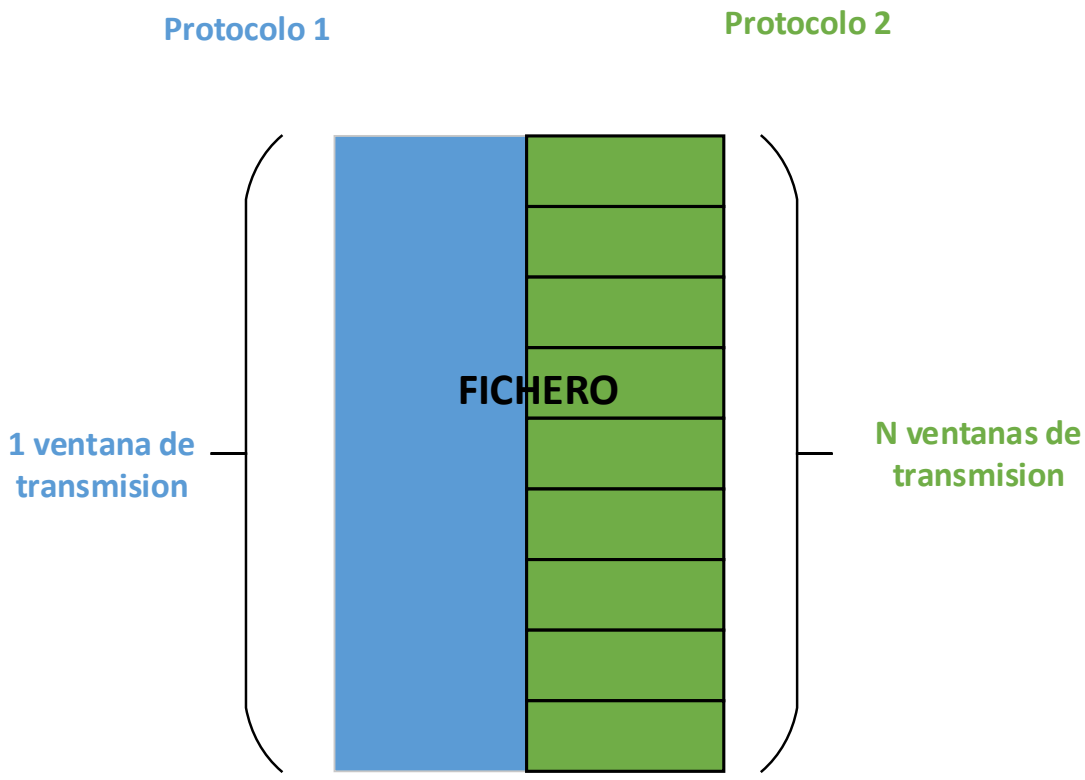

Figura 4.15 Ejemplo multi-ventana

El modelo de enventanado consiste en que un cliente no admitirá bloques de ventanas que no sean en la que se encuentra él en su proceso de decodificación. Si el servidor envía paquetes confeccionados por bloques de otras ventanas, el cliente los descartará y no aplicará los mecanismos de decodificación. Cuando tenga la ventana terminada, le pedirá al servidor la siguiente como se puede observar en la Figura 4.16. En caso que existan varios clientes, será el primero que finalice cada ventana el que regule el envío de paquetes de nuevas ventanas. Del mismo modo el último cliente en decodifica una ventana específica será el que detenga la emisión de paquetes vinculados a esa ventana. En el uso del modelo de enventanado es importante definir el tamaño de la ventana para alinearlo al tamaño de la memoria y a las capacidades de cómputo del Wasmpote.

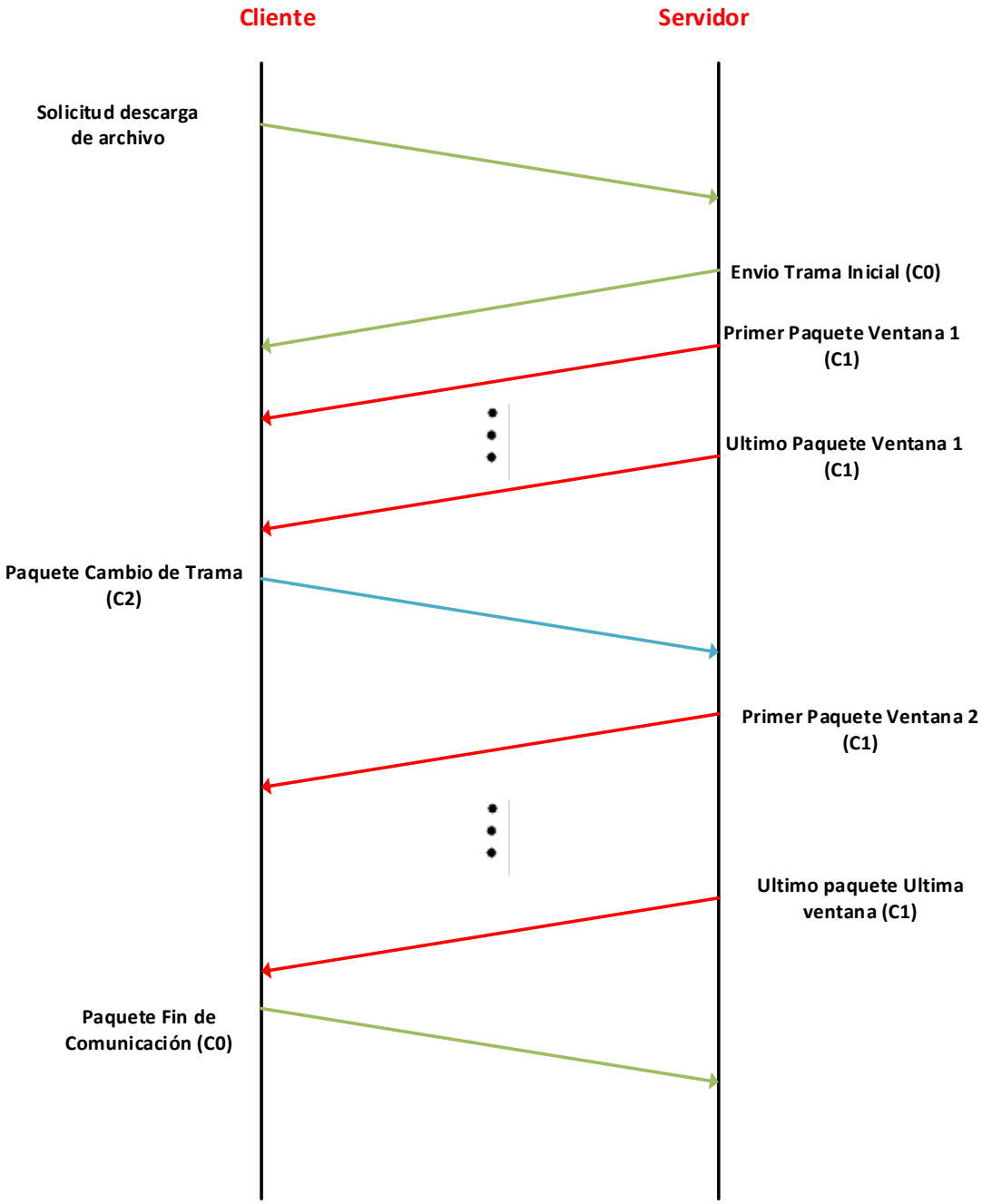

Figura 4.16 Comunicación Multi-Ventana

Debido a las limitaciones de memoria, se debería usar la SD para almacenar los paquetes que se iban recibiendo y decodificando. Puesto que las instrucciones de lectura y escritura en la SD son instrucciones bastante lentas, se decidió que el segundo protocolo utilizaría tanto la memoria interna, para almacenar los vecinos y poder realizar operaciones de manera más rápida, como la SD, donde se almacenarían los paquetes que llegan y los paquetes que ya han sido descodificados. El desarrollo de este protocolo es el mismo que se puede observar en la Figura 4.14 solo que en este caso en vez almacenar el paquete en memoria interna se almacena en la SD.

En esta versión, se tuvo que cambiar el formato de trama puesto que ahora se tiene que llevar la cuenta de que ventana se estaba transmitiendo para que el cliente no recibiera

tramas de una ventana diferente a la que se estaba decodificando. El formato de trama se puede observar en la Figura 4.17. Además se cambió el sistema de almacenamiento de paquetes del cliente como se ha explicado anteriormente.

Figura 4.17 Payload de la trama de datos para la segunda versión del protocolo

A la hora de probar el sistema de comunicaciones con ficheros de datos, se comprobó que para enviar un archivo de 10 Kbyte, usando un tamaño de ventana de 10 paquetes y un tamaño de datos de 40 bytes, se tarda alrededor de 2,28 minutos usando el método de codificación robusta mientras que si se usa la codificación ideal el dispositivo se bloquea debido a la falta de memoria. Es un tiempo alto puesto que empleando la metodología básica consistente en enviar los datos de fichero de forma secuencial se han obtenido tiempos aproximados de 60 segundos para ficheros de entre 75 y 85 Kbytes. Además, el sistema solo aceptaba un tamaño de ventana máximo de 11 paquetes y cada paquete solo podía transmitir 40 bytes, puesto que si se subía el tamaño de ventana o el del paquete, el sistema dejaba de funcionar debido a que no poseía memoria suficiente para ejecutar el programa.

Debido a estos contratiempos se decidió cambiar el proceso de decodificación puesto que era donde había problemas por la poca memoria de los Waspmote e intentar reducir al mínimo el número de ciclos ejecutados en la descodificación de cada paquete para reducir el tiempo total así como el sistema de almacenamiento de los paquetes recibidos por el cliente.

4.4.3 Tercera Versión

Como se ha dicho anteriormente, la idea principal en el desarrollo de la tercera versión del protocolo de comunicación se basó en la reducción de ciclos de ejecución en la decodificación y de uso de memoria, tanto de la memoria interna del Waspmote, ya que limita la ejecución de la comunicación a tamaños de ventana pequeños, como de la memoria SD, puesto que los accesos a esta memoria son lentos y ralentizan bastante la comunicación. Además, para que los procesos fueran más eficientes se modificaron las librerías de control de Waspmote logrando con esta nueva versión más memoria libre en los nodos.

El esquema empleado en eta versión del protocolo cambio totalmente, partiendo del formato de trama y finalizando en el proceso de decodificación. Ahora, en vez de enviar la semilla y el grado, se usa una máscara de 4 bytes en el que mediante bits activos se saben cuáles son los paquetes por los que está compuesto ese paquete. Si el bit está activo, el payload de la trama incluye información de ese vecino. De esta manera se puede tener un

tamaño de ventana máximo de 32 vecinos. De las pruebas con los protocolos anteriores y del análisis del uso de memoria se obtiene que ventanas de más de 32 bloques requieren equipos más potentes que el Waspmote y además no ofrecen mejoras. El uso de esta mascara se pude comprobar en la Figura 4.18 en el que se ven dos ejemplos de cómo se usa la máscara. En ambos casos los paquetes que ya se han decodificado son los vecinos 1, 2 y 4.

En el primer caso llega un paquete que depende de los vecinos 1,2 y 5. Al realizar la operación AND entre el paquete recibido y los paquetes ya decodificados, nos queda una trama de bits en la que están activos los vecinos de los que depende el paquete que ya tenemos decodificados. Realizando la operación XOR entre esta trama y el paquete recibido nos da como resultado los vecinos de los que depende el paquete que no tenemos decodificados. Se cuenta el número de bits activos, gracias al código obtenido en [97], y si es 1 significa que este paquete se puede decodificar como sucede en este caso y sabemos cuál es el vecino que se puede, en este caso el 5 debido a la posición del bit activo. En el segundo caso se realiza el mismo proceso de decodificación pero en ese caso obtenemos dos bits al final, con lo que ese paquete no se puede decodificar de momento. Este nuevo procedimiento de decodificación se sustenta únicamente en operaciones lógicas de rápida ejecución y su latencia es prácticamente independiente del número de trozos incluidos en el paquete codificado.

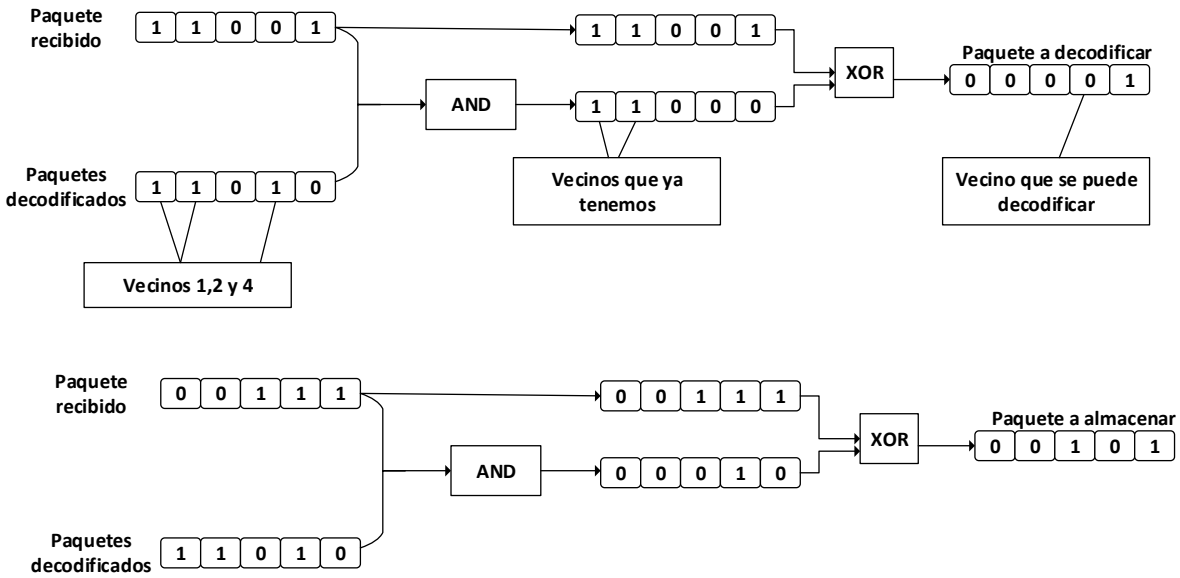

Figura 4.18 Ejemplos operaciones lógicas con mascara

En la trama también se envían el número de ventana que se está transmitiendo y el contador de paquete. Además se cambió la API del cliente a unas nuevas desarrolladas por investigadores del proyecto Smart Santander. De esta manera se aumentó en gran medida la memoria libre de dispositivo pasando de tener 700 bloques de memoria libre con la API antigua y 2300 bloques con la nueva. En cuanto al formato de trama, se redujo el número de bytes de la cabecera del protocolo Waspmote pasando de usar 6 a usar 3, como se puede ver

en la Figura 4.19. En este caso se reduce la información que se puede enviar pasando de 67 bytes a 64 en este caso.

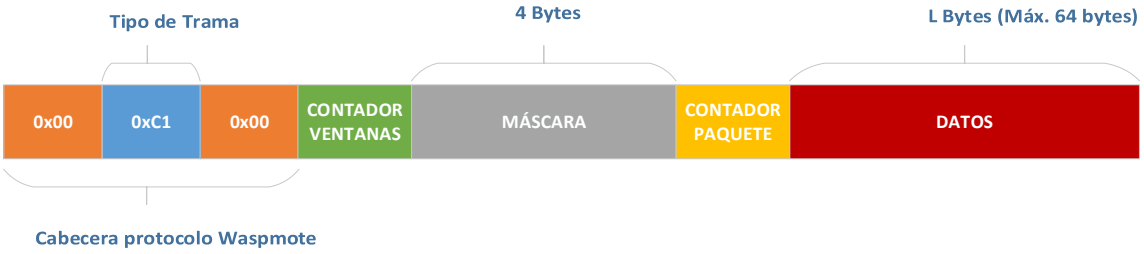

Figura 4.19 Payload de la nueva trama de datos

En el nuevo protocolo, la información de cada paquete se almacena en una lista de tamaño predefinido antes de la ejecución de programa, que esta predefinida como 1.5 veces el tamaño de la ventana, porque teóricamente recibiendo 1.05 veces el tamaño de ventana paquetes se puede descodificar una ventana completa[15][19]. Se establece 1.5 para tener un margen sobre el caso teórico.

El funcionamiento del nuevo protocolo de descodificación consiste en trabajar con la máscara como se ha explicado con anterioridad. Cuando se llega a la situación de que un paquete se puede descodificar, se almacena el puntero de memoria de donde se encuentran los datos de ese paquete en la lista explicada en el párrafo anterior y se sigue con el proceso de decodificación. Cuando se sabe que se puede decodificar la ventana entera, se procede a realizar la decodificación de la información de los paquetes realizando las operaciones XOR entre paquetes.

Una vez detallado el proceso, se procede a realizar pruebas para comprobar el rendimiento del protocolo. En la primera de ellas se comprueba cómo evoluciona el uso de la memoria interna del dispositivo a medida que se aumenta tanto el tamaño de la ventana como el tamaño de datos del paquete, dos factores que se consideran clave en el desarrollo de proyecto.

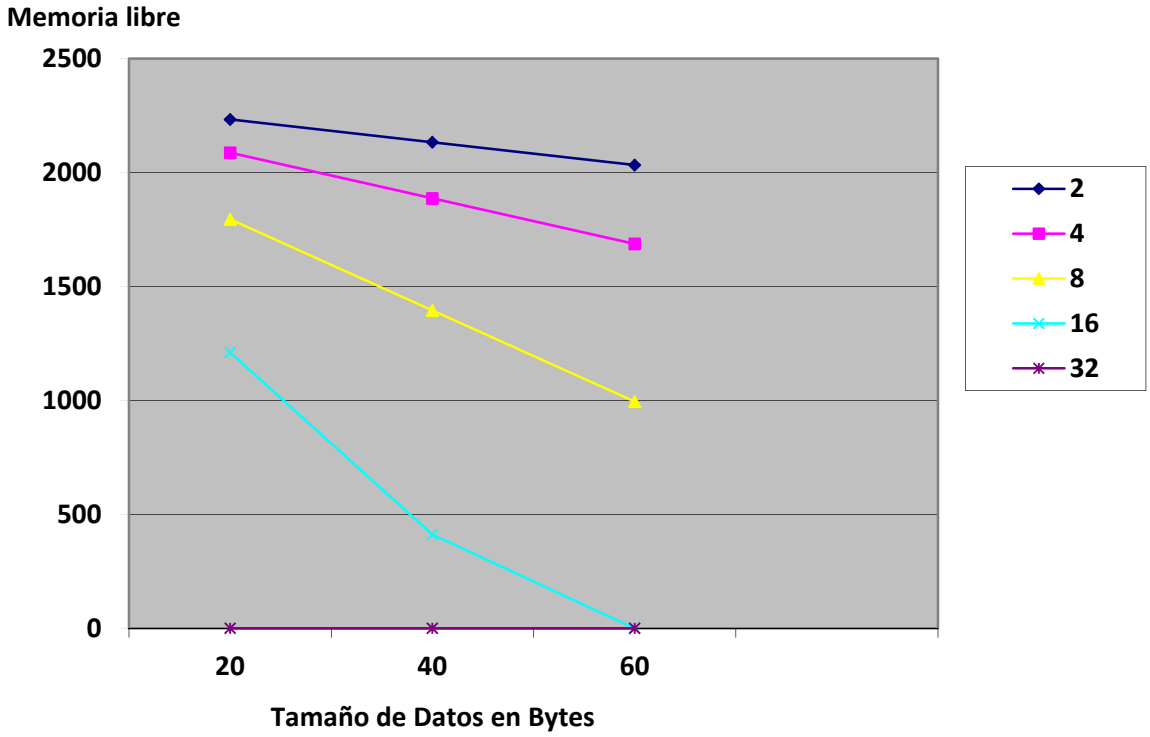

Figura 4.20 Prueba Memoria libre en dispositivo

Como se puede observar en la Figura 4.20, a medida que se aumenta el tamaño de los datos o el número de vecinos del paquete, la memoria libre del Waspmote se reduce de una manera drástica, llegando a un punto en el que no queda nada de memoria libre, punto crítico cuando el tamaño de los datos de la trama es de 60 bytes y se está usando un tamaño de ventana de 16 bytes.

Además se implementó este protocolo pero utilizando como almacenamiento solo la SD a modo de comparativa. En la siguiente grafica se siguió el mismo proceso que en la anterior prueba y como se puede observar obtenemos unos resultados de memoria libre bastante más positivos puesto que en ningún momento se llegó a un punto crítico de memoria como es el caso de usar solo la RAM de dispositivo.

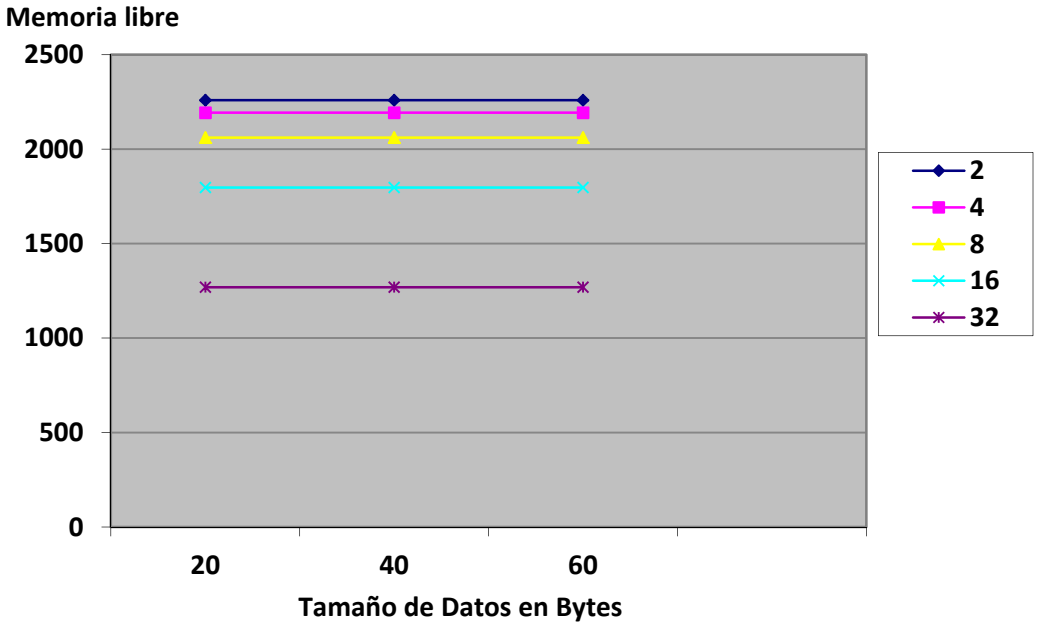

Figura 4.21 Prueba Memoria libre en dispositivo usando SD

En este caso, al almacenar los paquetes que llegan en la SD la descodificación no ocupa mucha memoria. Hemos ganado en memoria pero a la vez hemos perdido en velocidad como se puede observar en las siguientes graficas en las que también se diferencia entre el uso de la codificación ideal y la robusta.

En la Tabla 4-1 se muestra el rendimiento de la tercera versión del protocolo usando tanto almacenamiento solo en la memoria interna como solo en la SD usando el método de codificación ideal, mientras que en la Tabla 4-2 se usó el método codificación robusta. Para el cálculo de los datos de las tabla expuestas a continuación, se usaron paquetes con un tamaño de datos de 40 bytes, un tamaño de ventana de 12 paquetes. Además, el fichero que se envió era un fichero de 10 kilobytes.

	Almacenamiento en la memoria interna	**Almacenamiento en la SD**
Paquetes enviados por el servidor	539	846
Paquetes usados por el cliente	421	385
Tiempo transmisión (milisegundos)	72891	105026
Memoria libre (Bytes)	656	1898

Tabla 4-1 . Rendimiento de la tercera versión de protocolo con codificación ideal

	Almacenamiento en la memoria interna	Almacenamiento en la SD
Paquetes enviados por el servidor	476	820
Paquetes usados por el cliente	363	354
Tiempo transmisión (milisegundos)	67261	107676
Memoria libre (Bytes)	656	1898

Tabla 4-2 . Rendimiento de la tercera versión de protocolo con codificación robusta

Por último y para hacer la comunicación más real se ha adecuado el comportamiento del servidor y los clientes para que soportar que el servidor soporte la gestión de varios clientes con ventanas diferentes, etc. Para esta prueba en concreto, se implementó la situación de un servidor y dos clientes. El funcionamiento de este avance se puede observar en la Figura 4.22. El servidor divide sus esfuerzos enviando un paquete de la ventana en la que esta uno de los clientes y el siguiente paquete de la ventana en la que está el otro cliente y así sucesivamente.

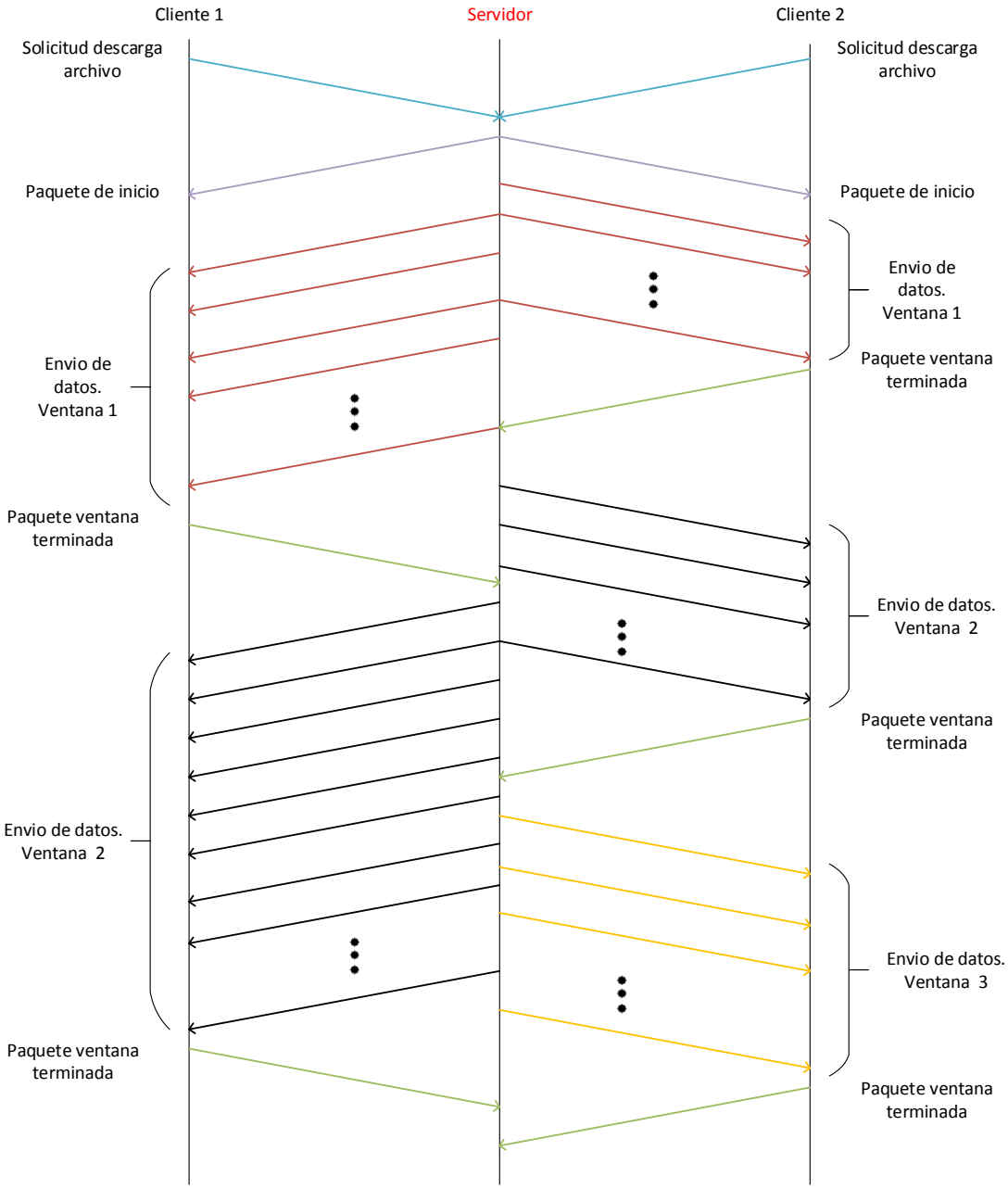

Figura 4.22. Tercera versión de protocolo con dos clientes

Los resultados que se obtuvieron se pueden ver en las tablas que se muestran a continuación. Se usaron las mismas características de la comunicación (tamaño de ventana, tamaño de datos, tamaño de fichero…) que en la parte anterior y también se diferenciara entre el uso del tipo de codificación ideal y robusta.

	CLIENTE 1		CLIENTE 2	
	Almacenamiento en la memoria interna	**Almacenamiento en la SD**	**Almacenamiento en la memoria interna**	**Almacenamiento en la SD**
Paquetes enviados por el servidor	471	935	471	935
Paquetes usados por el cliente	403	404	410	429
Tiempo transmisión (milisegundos)	96538	177610	95159	155699
Memoria libre (Bytes)	656	1898	656	1898

Tabla 4-3. Rendimiento de la tercera versión de protocolo con dos clientes y codificación ideal

	CLIENTE 1		CLIENTE 2	
	Almacenamiento en la memoria interna	**Almacenamiento en la SD**	**Almacenamiento en la memoria interna**	**Almacenamiento en la SD**
Paquetes enviados por el servidor	468	722	468	722
Paquetes usados por el cliente	347	374	350	363
Tiempo transmisión (milisegundos)	86400	140075	87432	137447
Memoria libre (Bytes)	656	1898	656	1898

Tabla 4-4. Rendimiento de la tercera versión de protocolo con dos clientes y codificación robusta

Como se puede observar en las tablas la codificación robusta mejora considerablemente el rendimiento de la transmisión, obteniendo mejores resultados tanto de tiempo de transmisión como de número de paquetes enviados y usados. También se observa

que el hecho de introducir nuevos nodos en la red no afecta al número de paquetes ni al tiempo de transmisión. La principal diferencia con respecto al funcionamiento de mandar los bloques de manera secuencial como se hace en el proyecto Smart Santander es que en este caso el sistema se escala. Como se puede en la Tabla 4-4, el servidor necesita un tiempo de transmisión muy similar para enviar el fichero a ambos clientes.

Por último, la elección entre uno de los dos mecanismos de almacenamiento, el de la RAM o el de la SD, cabe decir que todo dependerá de las características de la red de sensores en la que se esté trabajando y lo que se busque en la transmisión, eficiencia de la transmisión y rapidez. En este caso, se considera más adecuado el uso del almacenamiento en la memoria interna del dispositivo puesto que en el proyecto Smart Santander lo que se busca es que los datos de los sensores estén cuanto antes en la base de datos para poder analizarlos y usarlos.

5 CONCLUSIONES Y LÍNEAS FUTURAS

En el presente proyecto se han abordado aspectos relativos al estudio y diseño de los códigos Digital Fountain. En primer lugar se ha realizado un estudio de los códigos de Luby para, a continuación, implementar el codificador y el decodificador. Seguidamente, se modificó el algoritmo para que los datos a reconstruir se fueran decodificando en tiempo real. Por último, se aplicó este algoritmo a la red de sensores de Smart Santander y se mejoró progresivamente el algoritmo para alcanzar un óptimo funcionamiento.

A continuación se recogen las principales conclusiones que derivan del trabajo desarrollado. En la evolución del proyecto se han probado diferentes protocolos de decodificación y como se ha comprobado a medida que se avanzaba se mejoraba en términos de uso de memoria en el dispositivo y latencia del proceso. Todo ello se ha realizado sobre un entorno bastante restrictivo en cuanto a memoria y capacidad de cómputo.

En cuanto al protocolo tomado como definitivo, cabe la posibilidad de incluir alguna mejora en su implementación, pero su funcionamiento se considera suficientemente óptimo. Por ejemplo, aunque la máscara de los vecinos se ha restringido a un tamaño máximo de ventana de 32 vecinos, esta podría extenderse fácilmente pero no se considera rentable debido al overhead que introducen y que como se ha comprobado el uso de memoria de dispositivo se vería notablemente afectada, pudiendo empeorar el comportamiento global.

Se ha comprobado cómo afecta introducir más nodos en la comunicación. En este sentido la operativa del servidor se mantiene y el número de paquetes que se lanzan a la red se mantiene estable, ya que la misma información se puede reutilizar para cualquiera de los clientes. No obstante, dado que los clientes no se encuentran en el mismo punto del proceso de decodificación, se ha detectado un incremento de la latencia ya que cuando se solapan varias ventanas la transmisión de paquetes descartados por algún nodo reduce el ancho de banda efectivo. Otro aspecto que se ha comprobado es la mejora en el uso de la codificación

ideal y de la robusta, suponiendo mucho más rentable el uso de la codificación robusta tanto en tiempo de transmisión como en número de paquetes enviados.

La codificación de canal tiene como objetivo ofrecer una transmisión fiable al usuario. Esto lo consigue añadiendo redundancia, y permitiendo así al cliente detectar y/o corregir los posibles errores que se hayan podido producir en la información.

Los códigos Digital Fountain siguen esta filosofía, pero de un modo algo diferente. El cliente no debe recoger determinados paquetes para reconstruir la información; sólo debe recoger un número de paquetes independientemente de cuáles sean estos paquetes. Esta filosofía supone un cambio total en comparación con las diversas técnicas existían hasta ahora; además, ofrece numerosos beneficios.

Sin embargo, el uso de códigos Digital Fountain para este tipo de redes y ésta en particular no es la solución más adecuada puesto que si la red está bien planteada y no hay muchos errores, un esquema de transmisión y retransmisión básica funciona mejor. Sin embargo en el caso en el que la red tenga muchos errores de transmisión se deberían apreciar notables mejoras. Este aspecto se plantea como línea futura de investigación extendiendo los trabajos realizados en este proyecto a redes más amplias y con canales degradados como son los que se pueden encontrar en una red de sensores como la desplegada en la ciudad de Santander.

Con vistas a mejorar y continuar los trabajos presentados en este proyecto sería posible realizar más optimizaciones de la implementación tanto del codificador como del decodificador, lo que llevaría a mejoras en el rendimiento. Además se implementaran otros tipos de códigos Digital Fountain que puede que mejoren el rendimiento de los actuales.

Si se centran las mejoras en los propios algoritmos de codificación y decodificación, a lo largo de la elaboración de este proyecto han surgido ciertas ideas que podrían mejorar el rendimiento de los procesos. Así, actualmente siempre se espera a tener un paquete o bloque de grado 1 para iniciar el proceso de decodificación. No obstante, sería interesante analizar la viabilidad iniciar la codificación a partir de bloques de grado 2 que se encuentren dentro de bloques de grado 3. Es decir, dado un símbolo codificado con vecinos $\{3, 5\}$, realizar la búsqueda de algún símbolo de grado 3 que lo contenga, como podría ser el caso de uno cuyos vecinos fueran $\{3, 5, 7\}$; así, sumando ambos, se obtendría el símbolo de información 7 y se podría iniciar la codificación. Sería interesante investigar este caso analizando el uso extra que se realizaría de memoria, así como el posible aumento o disminución del tiempo de decodificación que supondría.

Otra posible mejora en la codificación en cuanto al grado seria la limitación del grado. Como se puede observar en la Figura 2.10 en torno al 90 por ciento de los paquetes que se crean con el soliton reforzado o robusto tienen un grado menor o igual que 5. Además, se ha comprobado durante el desarrollo de proyecto que es muy poco probable que cuando se sabe que un paquete se puede decodificar debido a la evolución del proceso de decodificación, este tenga un grado mayor de 3 o 4. Es por ello podría suponer una mejora tanto en tiempo de transmisión como en número de paquetes enviados desechar los paquetes que tuviesen un grado mayor que 5 y no enviarlos para que no supongan un cargo computacional adicional en el cliente.

Un nuevo horizonte en la implementación de los códigos Digital Fountain es el de la aplicación de Network Coding [92][93] en la implementación en diferentes tipos de redes. El Network Coding fue introducido por primera vez por Ahlswede como una alternativa al principio clásico de almacenar y reenviar de las redes IP. En este caso, el servidor no se encarga de codificar el paquete que va a enviar, sino que son los nodos intermedios entre el cliente y el servidor los que se encargan de codificar paquetes pertenecientes a distintos flujos. El cliente al trabajar en modo promiscuo puede almacenar en su buffer los paquetes que 'escucha' dentro de su área de cobertura, aun cuando no van dirigidos hacer el. Estos paquetes son un componente esencial, ya que serán necesarios para realizar las operaciones de decodificación del paquete codificado. Usando Network Coding se han conseguido mejores tiempos de transmisión ya que aumenta el throughput, así como una mayor eficiencia del canal ya que se realizan menos envíos.

Otro punto de estudio es la aplicabilidad de los códigos DF en aplicaciones reales. Se han introducido algunas líneas de investigación y aplicación más actuales. Así, la aplicación de los códigos DF para la difusión de video merece ser analizada en detalle. En este caso el estudio podría realizarse desde la aplicación de la codificación de canal a entornos con requerimientos de tiempo real, donde en la mayoría de los casos se almacenan conjuntos de datos (buffer) para mejorar la experiencia de usuario. Emplear los códigos DF para la transmisión de los datos de dicho buffer, determinar los requerimientos temporales y de tamaño de ese buffer o ventana de transmisión, etc. podrían ser algunos de los puntos a analizar.

Por último, una posible mejora en la valoración de los códigos Digital Fountain aplicados a redes de sensores seria utilizar otras plataformas de sensores de las detalladas con anterioridad en el capítulo 3 que tienen una mayor capacidad pero que siguen teniendo las características que se consideran necesarias para una red de sensores como el bajo consumo.

La solución de redes de sensores empleada en este proyecto ha revelado diversas deficiencias en procesado y lo más importante en cuanto a capacidades de almacenamiento, lo que dificulta los procesos de decodificación.

ACRÓNIMOS

3GPP	Third Generation Partnership Project
ACK	Acknowledgement
AES	Advanced Encryption Standard
AJAX	Asynchronous JavaScript And XML
AP	Access Point
API	Application Program Interface
ARM	Advanced RISC Machine
ARQ	Automatic Repeat request
AWGN	Additive White Gaussian Noise
BEC	Binary erasure channel
BSC	Binary Symmetric Channel
BSS	Basic Service Set
CCA	Clear Channel Assessment
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CSS	Cascading Style Sheets
DF	Digital Fountain
DIBS	Distributed Internet Backup System
DOM	Document Object Model
DSSS	Direct Sequence Spread Spectrum

DVB	Digital Video Broadcasting
E^2PROM	Electrically Erasable Programmable Read-Only Memory
EEC	Error-Correcting Coding
FCS	Frame Check Sequence
FEC	Forward Error Correction
FFD	Full Function Device
FHSS	Frequency Hop Spread Spectrum
FIO	Funnel I/O
FTDI	Future Technology Devices International
GFSK	Gaussian Frequency Shift Key
GMC	Green Modulation/Coding
GPRS	General Packet Radio Service
GPS	Global Positioning System
HART	Highway Addressable Remote Transducer Protocol
HTTP	Hypertext Transfer Protocol
I^2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv6-LoWPAN	IPv6 over Low power Wireless Personal Area Networks
ISA	Industrial Society of Automation
ISD	Ideal Soliton Distribution
ISM	Industrial Scientific & Medical
JSON	JavaScript Object Notation
JVT	Joint Video Team

LAN	Local Area Network
LCG	Linear Congruential Generator
LDPC	Low-Density Parity Check
LQI	Link Quality Indicator
LR-WPAN	Low Rate Wireless Personal Area Network
LT	Luby Transform
LLC	Logical Link Control
MAC	Media Access Control
MCPS	MAC Common Part Layer
MFSK	Multiple Frequency-Shift Keying
MIME	Multipurpose Internet Mail Extensions
MLME	MAC Layer Management Entity
MOEA	MultiObjective Evolutionary Algorithm
MTU	Maximum Transfer Unit
NFC	Near Field Communication
NIC	Network Interface Card
OSI	Open System Interconnection
PAN	Personal Area Network
PCO	Pre-Coding-Only
PEC	Packet Erasure Channel
PIC	Peripheral Interface Controller
PPDU	PHY Protocol Data Unit
PRNG	Pseudo Random Number Generator
PSK	Phase-Shift Keying
PWM	Pulse-Width Modulation

REST	Representation State Transfer
RFD	Reduced Function Device
RFU	Reserved for Future Use
RIA	Rich Internet Applications
RISC	Reduced Instruction Set Computing
RPC	Remote Procedure Calls
RS	Reed-Solomon
RSD	Robust Soliton Distribution
SBC	Single-Board Computer
SOA	Service-Oriented Architecture
SPI	Serial Peripheral Interface
SVC	Scalable Video Coding
TCP	Transport Control Protocol
TR	Testbed Runtime
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
UEP	Unequal Error Protection
ULP	Ultra Low Power
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UWB	Ultra-Wide Band
WLAN	Wireless Local Area Network
WSAN	Wireless Sensor & Actuator Networks
WSN	Wireless Sensor Network
XML	eXtensible Markup Language

BIBLIOGRAFÍA

[1] Ramón Agüero. Contribución a la mejora de las prestaciones en redes de acceso inalámbricas no convencionales. Tesis Doctoral, Noviembre 2007.

[2] Luis Sánchez. Contribution to the Cross-Layer Optimization of Intra-Cluster Communication Mechanisms in Personal Networks. Tesis Doctoral, Diciembre 2008

[3] E. Shannon. A mathematical theory of communication. Bell System Technical Journal, 27:379–423 and 623–656, July and October 1948.

[4] Bruce A. Carlson, Paul B. Crilly, and Janet C. Rutledge. Communication Systems: An Introduction to Signals and Noise in Electrical Communication. McGraw-Hill, 4th edition, 2002.

[5] David J. C. Mackay. Information Theory, Inference, and Learning Algorithms, Cambridge University Press, 2003.

[6] Tanenbaum, Andrew S. (2003). Computer networks. Upper Saddle River, New Jersey: Prentice Hall.

[7] RFC 793, Transmission Control Protocol, Darpa Internet Program, Protocol Specification, 1981

[8] Moon, Todd K. (2005). Error Correction Coding. New Jersey: John Wiley & Sons. Wiley

[9] Forney GD (March 1973). The Viterbi Algorithm. Proceedings of the IEEE 61 (3): 268-278.

[10] Robert G. Gallager. Low-Density Parity-Check Codes. PhD thesis, Massachusetts Institute of Technology, 1960.

[11] Robert G. Gallager. Low-density parity-check codes. IEEE Transactions on Information Theory, 8(1): 21-28, January 1962.

[12] Michael Tanner. A recursive approach to low complexity codes. IEEE Transactions on information theory. Vol IT-27, No. 5, September 1981.

[13] M. Mitzenmacher: Digital Fountains: A Survey and Look Forward, Harvard University, Division of Engineering and Applied Sciences.

[14] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, and Daniel A.Spielman. Improved low-density parity-check codes using irregular graphs. IEEE Transactions on Information Theory, 47(2):585–598, February 2001

[15] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, Daniel A. Spielman, and Volker Stemann. Practical loss-resilient codes. In Proceedings of 29th Symposium on Theory of Computing, pages 150–159, 1997.

[16] John W. Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. A digital fountain approach to reliable distribution of bulk data. In SIGCOMM, pages 56–67, 1998.

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

[18] Michael Luby. LT Codes. In Proceedings of The 43rd Annual IEEE Symposium on Foundations of Computer Science, pages 271–282, 2002.

[19] David J.C. McKay. Fountain Codes. Cavendish Laboratory, University of Cambridge.

[20] Amin Shokrollahi. Raptor Codes. IEEE Transactions on Information Theory, Vol. 52, No.6, June 2006. 2561-2567.22U, Application Programming Interface, Advanced Card Systems Limited

[21] Chris Harrelson, Lawrence Ip, Wei Wang. Limited randomness LT Codes. UC Berkeley.

[22] Zengqiang Chen, Qian Zhou. Implementation of LT Codes with a revised Robust Soliton Distributionby using Kent Chaotic Map. 2010 International Workshop on Chaos-Fractal Theory and its Applications.

[23] Siotai Cheong, Pingyi Fan. The effect of random encoding generators on the performance of LT codes. Tsinghua University.

[24] Jesper H. Sørensen, Petar Popovski, Jan Østergaard. Design and analysis of LT Codes with decreasing ripple size. Aalborg University.

[25] Chih-Ming Chen, Ying-ping Chen, Tzu-Ching Shen and John K. Zao. Optimizing degree distributions in LT codes by using the multiobjective evolutionary algorithm based on decomposition.

[26] Hongpeng Zhu, Guangxia Li, Zhidong Xie. Advanced LT codes in satellite data broadcasting system. PLA University of Sciencie and Technology.

[27] Mattia C.O. Boginom Pasquale Cataldi, Marco Grangetto, Enrico Magli, Gabriella Olmo. Sliding Window Digital Fountain codes for streaming of multimedia contents. Politecnico di Torino.

[28] Shakeel Ahmad, Raouf Hamzaoui, Marwan Al-Akaidi. Unequal error protection using LT codes and block duplication. University of Konstanz, Germany. The Monfort University, Leicester.

[29] Shakeel Ahmad, Raouf Hamzaoui, Marwan Al-Akaidi. Video multicast using unequal error protection with Luby Transform Codes. The Monfort University, Leicester.

[30] 3GPP TS 26.346 V6.1.0, Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service; Protocols and Codecs, June 2005.

[31] ETSI DVB TM-CBMS1167, IP Datacast over DVB-H: Content Delivery Protocols, Sept. 2005, draft Technical Specification, http://www.dvb.org.

[32] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," IEEE Transactions on Circuits and Systems for Video Technology, scheduled September 2007.

[33] Sheng-Kai Chang, Kai-Chao Yang, Jia-Shung Wang. Unequal protected LT code for layered video streaming. National Tsing-Hua University, Taiwan.

[34] Faruck Morcos, Thidapat Chantem, Philip Little, Tiago Gasiba, Douglas Thain.iDIBS: An improved distributed backup system. University of Notre Dame, United States, Technische Universität München, Munich.

[35] Huaxia Xia, Andrew A. Chien. RobuSTore: a distributed storage architecture with robust and high performance. UC San Diego.

[36] Jamshid Abouei, J. David Brown, Konstantinos N. Plataniotis, Subbarayan Pasupathy. On the energy efficiency of LT codes in Proactive Wireless Sensor Networks.

[37] David Gómez, Caracterización del comportamiento de las técnicas de QoS sobre WLAN: Estándar IEEE 802.11e, Proyecto Fin de Carrera, Universidad de Cantabria, Julio 2009.

[38] Sistema SOSUS, http://es.wikipedia.org/wiki/SOSUS

[39] Estándar 802.15.1, Bluetooth, http://www.ieee802.org/15/pub/TG1.html

[40] Estándar 802.11, Wi-Fi, http://ieee802.org/11/

[41] Estándar 802.15.3, http://www.ieee802.org/15/pub/TG3a.html

[42] Ultra-Band Wide: ECMA Std. 368: "High Rate Ultra Wideband PHY and Standard", 3rd Edition, December 2008

[43] Página oficial de Z-Wave Alliance, http://www.z-wavealliance.org

[44] Ultra Low Power Bluetooth o Wibree, http://www.eetimes.com/design/microwave-rf-design/4012680/Introduction-to-Bluetooth-ULP, http://vic-en-6to.blogspot.com.es/2012/05/dispositivos-moviles-wibree-ulp.html

[45] Estándar 802.15.4, http://www.ieee802.org/15/pub/TG4.html

[46] Página oficial de Zigbee Alliance, http://www.zigbee.org/

[47] Página oficial protocolo HART, http://www.hartcomm2.org/index.html

[48] Articulo interés sobre redes wireless HART, http://rii.galeon.com/aficiones2095541.html

[49] Página oficial protocolo ISA, http://www.isa.org

[50] Página oficial protocolo IETF IPv6, http://ietfreport.isoc.org/ids-wg-6lowpan.html

[51] Página oficial de EnOcean, http://www.enocean.com/home/

[52] Página oficial ANT, http://www.thisisant.com/

[53] MiWi Wireless Networking Protocol Stack, David Flowers y Yifeng Yang Octubre 2011: http://www.microchip.com/stellent/idcplgIdcService=SS_GET_PAGE&nodeId=1824&appnote=en520606

[54] Digi International, http://www.digi.com/

[55] X-CTU, http://www.digi.com/support/kbase/kbaseresultdetl?id=2125

[56] Firmtech, http://www.firmtech.co.kr/01pro/main_eng.php?index=100&proinfo=103

[57] Sena Technologies, http://www.sena.com/products/industrial_zigbee/index.php?tab_menu=OEM

[58] LS Research: http://es.mouser.com/lsresearch-prosiFLEX/

[59] Crossbow, http://www.xbow.com/

[60] Tutorial uso de motas crossbow, http://www.pages.drexel.edu/~kws23/tutorials/motes/motes.html

[61] Lenguaje de programación NESC, http://standards.ieee.org/about/nesc/index.html

[62] Compilador CYGWIN, http://www.cygwin.com/

[63] Shockfish, plataforma Tinynode, http://www.tinynode.com/

[64] TelosB, http://bullseye.xbow.com:81/Products/productdetails.aspx?sid=252

[65] MicaZ, http://bullseye.xbow.com:81/Products/productdetails.aspx?sid=164

[66] Waspmote, http://www.libelium.com/products/waspmote

[67] Módulos iSense, http://www.coalesenses.com/index.php?page=isense-hardware

[68] Coalsenses, http://www.coalesenses.com/

[69] Placas Arduino, http://arduino.cc/es/Main/Hardware

[70] Lenguaje Arduino, http://arduino.cc/es/Reference/HomePage

[71] Lenguaje Wiring, http://wiring.org.co/

[72] Processing, http://www.processing.org/

[73] Placas similares a Arduino, http://arduino.cc/playground/Main/SimilarBoards

[74] Página oficial proyecto pingüino, http://www.hackinglab.org/pinguino/index_pinguino.html

[75] Tutorial de uso, https://sites.google.com/site/pinguinotutorial/home

[76] Lenguaje Python, http://www.python.org/

[77] Página official proyecto Raspberry, http://www.raspberrypi.org/

[78] Picaxe, http://www.picaxe.com

[79] Página web de la empresa Libelium, http://www.libelium.com/

[80] Listado de placas oficiales de Arduino, http://arduino.cc/en/Main/Hardware

[81] Chip Wiznet w5100, http://www.wiznet.co.kr/Sub_Modules/en/product/Product_Detail.asp?cate1=&cate2=&cate3=&pid=1011

[82] Libería Ethernet, http://arduino.cc/es/Reference/Ethernet

[83] Librería SD, http://arduino.cc/en/Reference/SD

[84] SquidBee, http://www.libelium.com/squidbee/index.php?title=Main_Page

[85] Módulos XBee Serie 1,http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/XBee-series1-module#overview

[86] Módulos XBee Serie 2, http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/XBee-series1-module#docs

[87] O'Reilly, Making Things Talk 2nd Edition Sep. 2011 by Tom Igoe

[88] O'Reilly, Getting Started with the Internet of Things 1st Edition May 2011 by Cuno Pfister

[89] Yee Wei Law, Yu Zhang, Jiong Jin, Marimuthu Palaniswami, and Paul Havinga. Secure Rateless Deluge: Pollution-Resistant Reprogramming and Data Dissemination for Wireless Sensor Networks. 19 July 2010.

[90] Yee Wei Law, Yu Zhang, Marimuthu Palaniswami and Xing She Zhou. A Secure Method for Network Coding-based Reprogramming Protocols in Wireless Sensor Networks. 2011

[91] Pedro J. Piñero Escuer y Pilar Manzanares Lopez. Códigos Fountain y su aplicación a las comunicaciones a través de redes de baja tensión. 2011

[92] Sachin Katt, Hariharan Rahul, Wenjun Hu Dina, Katabi, Muriel M´edard, Jon Crowcroft. XORs in the Air: Practical Wireless Network Coding

[93] David Gomez, Soffiane Hassayoun, Arnaldo Herrero Ramon Agüero and David Ros. Impact of Network Coding on TCP Performance in Wireless Mesh Networks.

[94] Página web del Proyecto Smart Santander http://www.smartsantander.eu/

[95] Página web de TST Sistemas http://www.tst-sistemas.es/

[96] "Prestaciones de las técnicas Digital Fountain sobre Infraestructuras Inalámbricas" Cristina Ysart 2008

[97] Codigo contador de bits http://stackoverflow.com/questions/109023/how-to-count-the-number-of-set-bits-in-a-32-bit-integer

[98] Pagina web de Digi www.digi.com