Escola Politècnica Superior de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE CARRERA

TÍTULO DEL TFC: Seguridad para protocolos MAC cooperativos en redes de sensores inalámbricas

TITULACIÓN: Ingeniería Técnica de Telecomunicaciones, especialidad en Telemática

AUTORES: Daniel Casabona Gómez
Daniel Leiva Martín

DIRECTOR: Christos Verikoukis

CO-DIRECTORES: Luís Gonzaga Alonso Zárate
David Sánchez

FECHA: 9 de julio de 2009

Título: Seguridad para protocolos MAC cooperativos en redes de sensores inalámbricas

Autores: Daniel Casabona Gómez
Daniel Leiva Martín

Director: Christos Verikoukis

Co-Directores: Luís Gonzaga Alonso Zárate
David Sánchez

Fecha: 9 de julio de 2009

Resumen

La tecnología ZigBee está empezando a cobrar importancia dentro de las redes de corta-media distancia en el ámbito industrial. Su bajo coste de producción, su mínimo consumo y sus prestaciones han hecho que se piense en este sistema de telecomunicaciones inalámbrico antes que en otras tecnologías que han dominado el mercado en los últimos años.

Como toda red inalámbrica, el apartado de seguridad cobra una gran importancia, ya sea para proteger los datos o para garantizar que todos los nodos disponen de la misma probabilidad para acceder al canal de transmisión.

A lo largo de este TFC se ha realizado una pequeña introducción de la tecnología ZigBee, prestando especial atención al estándar 802.15.4 que marca las bases a nivel físico y MAC. El objetivo final ha consistido en desarrollar un ataque a nivel de acceso al medio para poder saltarse el algoritmo CSMA/CA y aumentar el volumen de transmisiones de los nodos atacantes.

Una vez demostradas las vulnerabilidades que presenta el entorno en el que se trabaja, se han buscado métodos de detección y contraataque para reajustar el funcionamiento de la red.

Se han utilizado las motas TelosB como dispositivos y trabajado con el entorno TinyOS. Todas las pruebas realizadas y decisiones tomadas se han documentado con un razonamiento lógico y, si ha sido necesario, mediante gráficas de resultados.

Title: Security for cooperative MAC protocols in wireless sensors networks

Authors: Daniel Casabona Gómez
Daniel Leiva Martín

Director: Christos Verikoukis

Co-Directors: Luís Gonzaga Alonso Zárate
David Sánchez

Date: July, 9th 2009

Overview

ZigBee technology is beginning to be important for short-medium distance networks in industry. This wireless telecommunications system has been considered before other technologies, that have previously dominated the market, because of it's low cost production, low current consumption and features.

Just as in a wireless network, security is of great importance, both to protect data and to ensure that all nodes have the same probability to access the transmission channel.

In this TFC has been made a short introduction of ZigBee technology, focusing on the 802.15.4 standard that sets the foundation for the physical level and MAC. The ultimate goal has tried to develop a attack media access to skip the algorithm CSMA/CA and increase the volume of transmissions of the attack nodes.

Once the vulnerabilities of the environment in which we work have been demonstrated, have been searched methods of detection and counterattack to reset the network.

TelosB motes have been used as devices and worked within the TinyOS environment. All tests and decisions have been documented with logical argument and if it's necessary, through graphical results.

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN 7

1.1. Motivación 7

1.2. Objetivo 7

1.3. Estructura 7

CAPÍTULO 2. COMUNICACIONES INALÁMBRICAS 9

2.1. Tipos de redes inalámbricas 9

2.2. Redes de sensores inalámbricas 10

2.3. Seguridad en redes inalámbricas 11

CAPÍTULO 3. ESPECIFICACIONES Y PROTOCOLOS 12

3.1. IEEE 802.15.4 12
3.1.1. Capa física 13
3.1.2. Capa MAC 13

3.2. ZigBee 19
3.2.1. Dispositivos y topología 20

CAPÍTULO 4. HERRAMIENTAS DE TRABAJO 22

4.1. Motas TelosB 22

4.2. TinyOS 23

4.3. Lenguaje de programación NesC 24
4.3.1. Ejemplo de wiring. 25

4.4. XubunTOS 25

4.5. Herramientas para debugar 25

CAPÍTULO 5. APLICACIÓN Y PRUEBAS 27

5.1. Idea inicial 27

5.2. Ocupar el medio 28
5.2.1. Análisis detallado 28
5.2.2. Análisis global aproximado 31
5.2.3. Prueba práctica realizada con las motas TelosB 32
5.2.4. Conclusiones 33

5.3. Intercambio de identificadores 33

5.4. Deshabilitar CSMA/CA 35
5.4.1. CSMA/CA implementado en todos los nodos 35

5.4.2. CSMA/CA deshabilitado en un nodo 36

5.5. Deshabilitar CSMA/CA dentro de una red. 39

5.6. Adaptar el algoritmo de ataque 43

5.7. Desarrollo del ataque completo 43
5.7.1. Ataque final 43
5.7.2. Escenario con más de dos nodos 47

5.8. Detectar el ataque 49
5.8.1. Nivel local 49
5.8.2. Nodo dedicado dentro de la red 51

5.9. Contraataque 55

5.10. Pruebas finales. 56

CAPÍTULO 6. CONCLUSIONES 61

6.1. Conclusiones 61

6.2. Estudio de ambientalización 61

6.3. Líneas futuras 61

6.4. Valoración personal 62

BIBLIOGRAFÍA 63

ANEXO 1. TIPOS DE TRAMAS 65
Trama Beacon 65
Trama Datos 65
Trama Ack 66
Trama comandos MAC 66

ANEXO 2. REDES INALÁMBRICAS 67

ANEXO 3. ATAQUE POR FUERZA BRUTA Y SOLUCIÓN 68

ANEXO 4. VALORES PROCEDIMIENTO DESHABILITAR CSMA/CA 76

ANEXO 5. TIEMPO IFS 79

ANEXO 6. IDENTIFICADOR DE RED 80

ANEXO 7. OCUPAR EL MEDIO 82

CAPÍTULO 1. INTRODUCCIÓN

1.1. Motivación

Las redes inalámbricas están cobrando gran importancia en el mundo de las telecomunicaciones actuales, concretamente las redes de sensores empiezan a convertirse en algo habitual dentro de las nuevas tendencias de investigación.

La evolución de la tecnología ZigBee está provocando que se desarrollen multitud de aplicaciones en gran variedad de campos como pueden ser la medicina, la industria, la domótica o el medioambiente. Las propiedades de dichas redes caracterizadas por su bajo consumo y la posibilidad de trabajar con caminos múltiples, amplían el abanico de posibilidades de cara al futuro.

A nivel personal, las redes de sensores y la curiosidad por el funcionamiento de la tecnología ZigBee, han sido los principales motivos para elegir este proyecto.

1.2. Objetivo

El objetivo de este trabajo se ha centrado en conocer la tecnología ZigBee, y de forma más concreta investigar las vulnerabilidades que existen a nivel de acceso al medio. Tras conocer las posibles brechas en la seguridad de la capa MAC, se ha implementado una aplicación mediante la que se ha realizado un ataque en la red. Una vez demostrados los efectos del ataque sobre la red se han diseñado sistemas para detectar el mismo y contraatacar, con la finalidad de volver a disfrutar de un acceso al medio justo y equitativo.

Tras estos objetivos iniciales, existen una serie de metas secundarias como aprender a programar en NesC, ampliar conocimiento en redes de sensores, programar en java o conocer el entorno TinyOS. Son tareas complementarias sin las que no se podrían desarrollar los objetivos principales.

1.3. Estructura

Este trabajo de final de carrera, se divide en 6 capítulos junto con un anexo en el que se proporciona información complementaria para entender claramente algunos puntos tratados en el TFC.

Los primeros puntos del proyecto se centran en describir tanto la tecnología sobre la que se trabaja, como las herramientas que se han utilizado. El primer punto habla de forma general sobre las redes inalámbricas y más en concreto de las redes de sensores. Posteriormente se desarrollan las características de la tecnología ZigBee y concretamente del estándar 802.15.4 que son dos pilares básicos en el TFC. Para finalizar esta parte del trabajo, se detallan las

características de las herramientas utilizadas, prestando especial atención a las motas TelosB.

El último capítulo del proyecto se centra en desarrollar por completo el ataque en una red de sensores a nivel de acceso al medio y un método para contrarrestarlo, enumerando los pasos que se han seguido y las dificultades que se han encontrado antes de adaptar la idea original y completarla con éxito.

CAPÍTULO 2. COMUNICACIONES INALÁMBRICAS

La tecnología inalámbrica está ganando adeptos a pasos agigantados. Actualmente multitud de empresas y usuarios prefieren implementar redes sin hilos por las ventajas que esto conlleva.

La principal ventaja es la eliminación del medio físico en la transmisión. Los cables han dejado paso a la comunicación por ondas electromagnéticas, facilitando la instalación de dispositivos y ampliando las posibilidades de ubicación. Eliminar el medio físico también tiene sus puntos negativos que se centran en la bajada de velocidades de transmisión y el riesgo a nivel de seguridad que existe en cada comunicación.

2.1. Tipos de redes inalámbricas

Las tecnologías inalámbricas que existen actualmente, tienen sus respectivas limitaciones de velocidad y rango de alcance. En la **Fig.2.1** se pueden apreciar los estándares inalámbricos relacionados con estos dos parámetros.

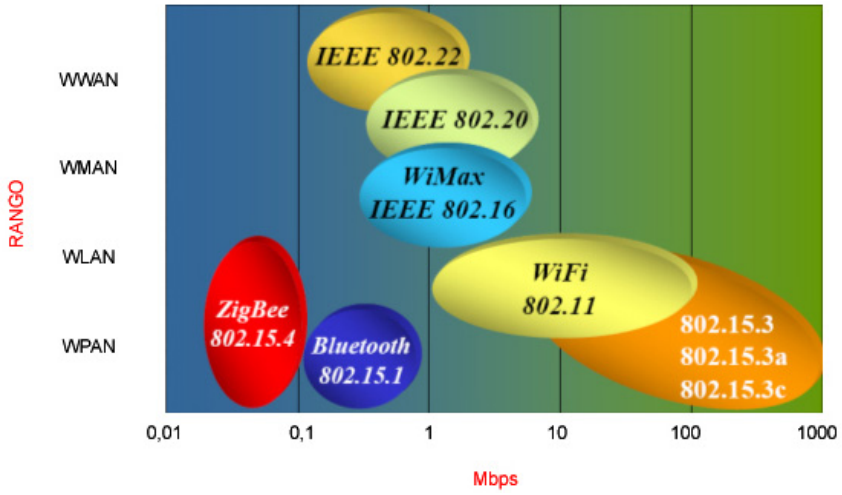

Fig.2.1: Clasificación redes.

Si se realiza una clasificación por rango de cobertura, se puede hablar de:

- WPAN (*Wireless Personal Area Networks):* Redes personales caracterizadas por tener un área de cobertura limitada. Entre las tecnologías que pueden formar este tipo de redes destacan la novedosa ZigBee o la ya consolidada Bluetooth.

- WLAN (*Wireless Local Area Network*): Redes que aparecen como alternativa a las redes LAN cableadas, creando un sistema de comunicación de datos más flexible. Las tecnologías relacionadas con

este tipo de redes, son WI-FI y, si se habla de distancias relativamente cortas, puede incluirse ZigBee.

- WMAN (*Wireless Metropolitan Area Networks*): Redes de área metropolitana que pueden abarcar hasta 4 Km. de distancia. Entre las tecnologías que pueden encontrarse en este tipo de redes destacan LMDS y WiMax, con ciertas similitudes a WI-FI pero basadas en el estándar IEEE 802.16.

- WWAN (*Wireless Wide Area Network):* Red a nivel mundial con una gran cobertura, en la que destacan tecnologías como GPRS y UMTS, ambas utilizadas para la telefonía móvil principalmente.

Existen multitud de clasificaciones de redes inalámbricas, en función de los parámetros estudiados, otro ejemplo se puede encontrar en el Anexo 2.

2.2. Redes de sensores inalámbricas

Las redes de sensores, se caracterizan por el gran número de nodos distribuidos que mantienen una comunicación rápida y flexible de forma constante.

Desplegar este tipo de redes ofrece varias ventajas entre las que se pueden destacar robustez, diversidad, redes auto-organizativas o escalabilidad. La funcionalidad es bastante variada y son diversos los sectores que están empezando a implementar este tipo de redes, como la domótica, la agricultura o el medio ambiente.

En función del propósito final de la red, los dispositivos pueden disponer de unas características u otras. Por regla general este tipo de redes se utilizan para capturar datos, por lo que es habitual que dispongan de sensores de humedad, temperatura, luz o vibración entre otros.

En este trabajo, se han utilizado los dispositivos TelosB, comúnmente llamados motas, pero simplemente se han estudiado a nivel de comunicaciones. En el punto dedicado a las herramientas del proyecto se han definido sus características y sus funciones principales.

Si las redes de sensores presentan una topología mallada, en la que existe una comunicación multidireccional, los nodos pueden realizar una comunicación auto-organizativa donde se transmite la información recogida y enviada por otras motas hasta que llega a la mota destino. De esta forma existen múltiples caminos que proporcionan gran estabilidad ante posibles caídas de dispositivos intermedios. Las motas destino serán las encargadas de hacer llegar en última instancia la información recogida por los sensores al servidor de la aplicación con la que se esté trabajando.

2.3. Seguridad en redes inalámbricas

El tema de la seguridad adquiere una especial importancia en las redes inalámbricas (ver [11] y [15]). El principal inconveniente de este tipo de redes se centra en que las comunicaciones ya no se realizan sobre un medio físico cerrado. Al utilizar la transmisión por ondas electromagnéticas, se utiliza un canal común, abierto y accesible.

Todo agente externo situado dentro de la cobertura de la red inalámbrica puede acceder a la misma, ya sea de un modo pasivo donde sólo escucha la información que se transmite, o puede actuar activamente sobre el funcionamiento de la red, alterando la información transmitida o inyectando paquetes sin autorización.

Cuando se habla de seguridad, también hay que tener en cuenta todos aquellos mecanismos no autorizados que se pueden utilizar para saltarse las reglas básicas dentro de la red. Deshabilitar o modificar el acceso al medio son ataques que pueden realizar nodos autorizados dentro de la red. De esta forma se sitúan en una posición de ventaja respecto al resto de estaciones a la hora de transmitir.

La privacidad de los datos en estas redes provoca cierta inseguridad que ha ido desapareciendo gracias a los diversos sistemas de protección de datos mediante cifrado. Actualmente las redes más utilizadas que trabajan bajo el estándar 802.11 incorporan variadas técnicas de cifrado que están recogidas en la norma 802.11i. En cuanto a la seguridad en el acceso al medio, se tiende a utilizar un nodo que gestione este acceso asignando un tiempo determinado a cada estación para transmitir.

En las redes de sensores, las medidas a nivel de seguridad son bastante similares a las que se pueden encontrar en redes mayores. La evolución de dichos sistemas es algo más lenta por su pequeño mercado y por el hecho de que muchas veces se trabaja sobre código abierto sin un respaldo económico.

Este TFC se centra en la seguridad que presenta la capa MAC y las posibilidades que presenta para provocar desigualdades en las transmisiones dentro de la red.

CAPÍTULO 3. ESPECIFICACIONES Y PROTOCOLOS

El sistema de comunicaciones con el que se ha trabajado está compuesto por diferentes especificaciones y protocolos como se puede observar en la **Fig.3.1**. ZigBee trabaja sobre el estándar IEEE 802.15.4 que a su vez define el nivel físico y el control de acceso al medio.

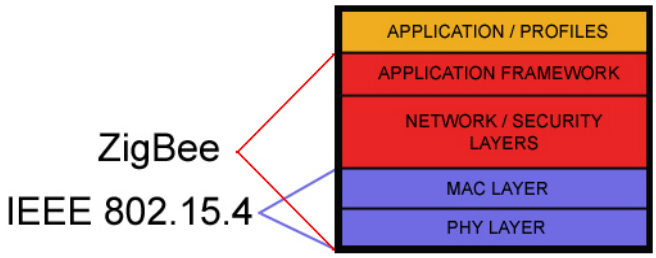

Fig.3.1: Pila ZigBee.

A continuación se explicarán con mayor detalle el estándar IEEE 802.15.4 y ZigBee.

3.1. IEEE 802.15.4

El IEEE 802.15.4 es un estándar de comunicaciones inalámbricas con bajas tasas de transmisión de datos, define el nivel físico y el control de acceso al medio (ver [5]). Por ello es la base sobre la que se define ZigBee (ver **Fig.3.1**).

El propósito de este estándar es definir los niveles de red básicos para que una red pueda dar servicio, priorizando el bajo coste y la baja tasa de transmisión de datos.

En la forma básica del estándar IEEE 802.15.4 el área de comunicación es de 10 a 75 metros y la tasa de transferencia es de 250 Kbps. Se han definido varios niveles físicos para tener alternativas para sistemas empotrados con requerimientos de consumo menores con tasas de 100 Kbps, 40 Kbps y 20 Kbps.

Estos dispositivos están adaptados para su uso en tiempo real por medio de slots de tiempo garantizado, son capaces de evitar colisiones gracias al CSMA/CA y tienen un soporte integrado para comunicaciones seguras. Además de todo esto el 802.15.4 ha sido concebido para que estos dispositivos tengan un bajo coste de fabricación.

3.1.1. Capa física

El nivel físico (PHY) controla el transceptor de radiofrecuencia y realiza la selección de canales junto con el control de consumo y de la señal. En la **Fig.3.2** se puede observar una representación del espectro del canal físico. El estándar IEEE 802.15.4 opera en las siguientes bandas de frecuencia:

- 868-868,8 MHz: Europa.
- 902-928 MHz: Norte América.
- 2400-2483,5 MHz: uso libre en todo el mundo.

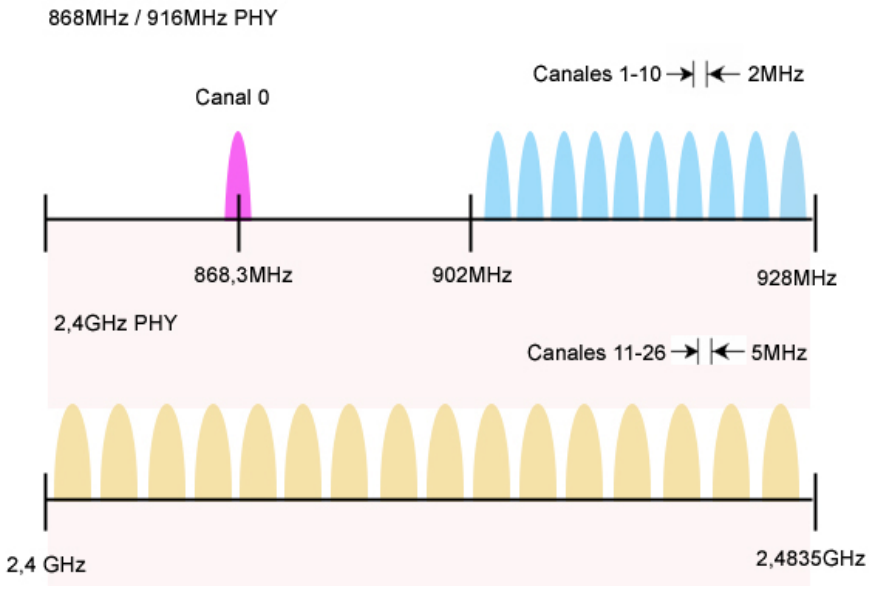

Fig.3.2: Espectro del canal físico que ocupa estándar IEEE 802.15.4

3.1.2. Capa MAC

La capa MAC se encarga de mantener y gestionar las comunicaciones entre los diversos dispositivos de la red. En función del tipo de red en la que se trabaje, proporciona herramientas para la gestión de beacons, la gestión de GTS (tiempo de slots garantizados) y envío de tramas de reconocimiento.

Esta capa proporciona 5 tipos de trama diferentes:

- Trama de datos (ver Anexo 1)
- Trama de beacon (ver Anexo 1)
- Trama ACK (ver Anexo 1)
- Trama de comandos (ver Anexo 1)
- Supertrama, sólo para redes con beacon.

A continuación se detallan los campos de la trama general MAC (ver **Fig.3.3** y **Fig.3.4)**, mientras que los demás tipos de tramas comentadas se han tratado en el Anexo 1 por falta de espacio.

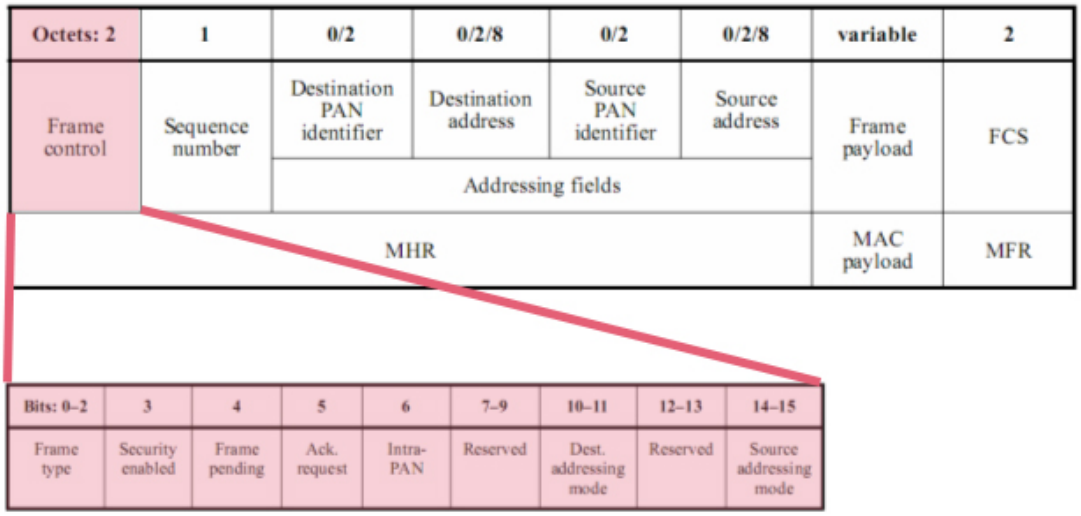

Fig.3.3: Campos de la trama general MAC y desglose campo frame control.

- Campo **Frame control:**
  - Frame type: Indica el tipo de paquete enviado (Ack, datos, comandos MAC o beacon).
  - Security enable: Campo que indica si hay encriptación.
  - Frame pending: Indica si se tiene que mandar más información o si es la última.
  - Ack request: Indica si es necesaria una confirmación.
  - Intra-PAN: Indica si el envío es en la misma PAN o en una externa.
  - Dest. addressing mode: Indica si la dirección destino es de 16 o 64 bits (Si el valor es 0, no es un paquete ACK o beacon, y Source addresing mode no es 0. El destino es el coordinador PAN).
  - Reserved: Zona reservada.
  - Source addressing mode: Indica si la dirección origen es de 16 o 64 bits (Si el valor es 0, no es un paquete ack, y el Dest. addressing mode no es 0. El origen es el coordinador PAN).

<table>
<tr><td>Octets: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>variable</td><td>2</td></tr>
<tr><td rowspan="2">Frame control</td><td rowspan="2">Sequence number</td><td>Destination PAN identifier</td><td>Destination address</td><td>Source PAN identifier</td><td>Source address</td><td rowspan="2">Frame payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="6">MHR</td><td>MAC payload</td><td>MFR</td></tr>
</table>

Fig.3.4: Campos de la trama general MAC.

- Campo **Sequence number:** Campo de 8 bits que indica el identificador del paquete.

- Campo **Destination PAN identifier:**
  - 0 bits: Si el campo *Dest. Addressing mode* es 0, no se incluye en la trama.
  - 16 bits: Indica el identificador PAN destino.

- Campo **Destination address:**
  - 0 bits: Si el campo *Dest. Addressing mode* es 0, no se incluye en la trama (Destino PAN coordinador).
  - 16 bits: Dirección destino de dicho tamaño.
  - 64 bits: Dirección destino de dicho tamaño.

- Campo **Source PAN identifier:**
  - 0 bits: Si el campo *Source addressing mode* es 0, no se incluye en la trama.
  - 16 bits: Indica el identificador PAN origen.

- Campo **Source address:**
  - 0 bits: Si el campo *Source Addressing mode* es 0, no se incluye en la trama (origen PAN coordinador).
  - 16 bits: Dirección origen de dicho tamaño.
  - 64 bits: Dirección origen de dicho tamaño.

- Campo **Frame payload:** Campo que varía en función del tipo de trama.

- Campo **FCS:** CRC de la trama que se realiza utilizando un polinomio de grado 16.

Una de las funciones básicas de la capa MAC es la de gestionar el acceso al medio, con el objetivo de que todos los nodos de la red compartan de una forma ordenada y justa el medio de transmisión. Para lograr este objetivo, se utiliza el algoritmo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance).*

En función de la red en la que se trabaje, se utiliza un tipo de CSMA/CA u otro. En las redes sin Beacon el CSMA/CA es no ranurado, mientras que en las redes con Beacon se utiliza CSMA/CA ranurado.

Las motas TelosB, trabajan sin Beacon, por lo que sólo está implementado el modo no ranurado, por este motivo se explica con mayor detalle esta variante.

3.1.2.1. CSMA/CA no ranurado

Este tipo de CSMA/CA se utiliza para acceder al medio en las redes sin Beacon. En la **Fig.3.5** pueden apreciarse los pasos que se siguen para acceder al medio.

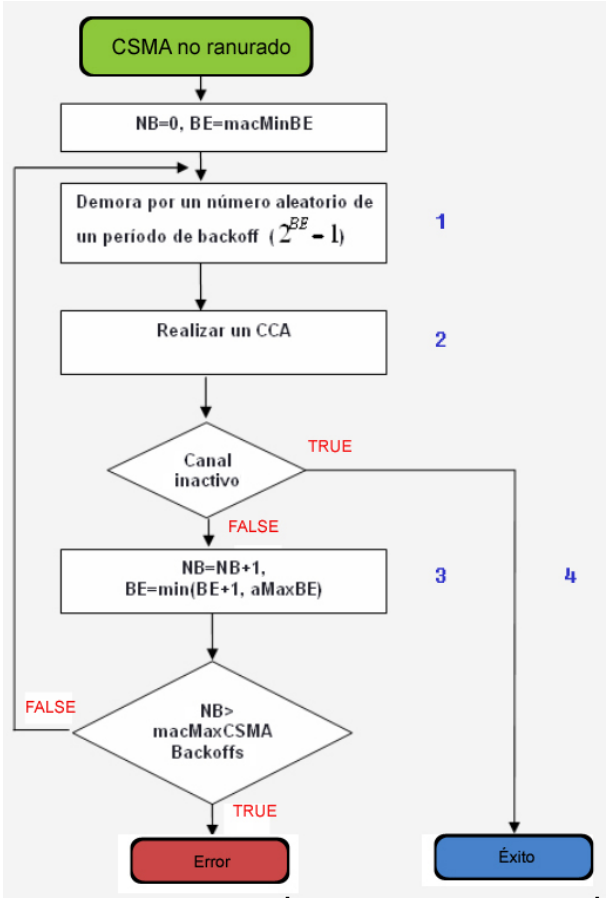

Fig.3.5: Diagrama CSMA/CA no ranurado.

- Variables que intervienen:
    - **NB:** Determina el número de veces que el algoritmo requiere el backoff para acceder al medio.
    - **BE:** es el backoff exponente, que se relaciona con el backoff que el dispositivo espera antes de evaluar el canal. En este caso se

le asigna como valor inicial macMinBE que según el estándar es 3.

- **aMaxBE** = Valor límite para determinar BE, por defecto según el estándar 5.

- **macMaxCSMABackoffs** = valor límite para determinar los intentos de transmisión, por defecto 4.

- Proceso para acceder al medio en la primera iteración:

  - **(1)** Se determina el tiempo de backoff que se tiene que esperar antes de realizar la comprobación del medio, este tiempo será un valor aleatorio de 0 a $2^{BE} - 1$. En este caso al ser BE = 3, los valores estarán comprendidos entre 0 y 7.

  - **(2)** Se pide a la capa física que realice CCA (valoración de canales desocupados).

  - **(4)** Opción 1: Canal libre y acceso al medio logrado.

  - Opción 2: Canal ocupado

    - **(3)** Se incrementa NB y BE en una unidad, cumpliendo que BE no será nunca superior a aMaxBE. En el primer paso por dicho punto, BE = min (3+1, 5); BE = 4:
      - ◊ Si NB es mayor a macmaxCSMABackoffs se desestima el continuar intentando enviar el paquete.

      - ◊ Si NB no es mayor a macMaxCSMABackoffs se volverá a calcular el tiempo de backoff con el nuevo valor de BE.

3.1.2.2. Ejemplo de transmisión con CSMA/CA no ranurado

En la **Fig.3.6** se puede apreciar de forma gráfica el proceso de envío de una serie de paquetes, mediante CSMA/CA no ranurado. Se observa que:

- DATA 1: Realiza backoff y comprobación del medio. Al estar libre transmite.

- DATA 2: Realiza backoff y comprobación del medio. Al estar ocupado vuelve a repetir el proceso, pero esta vez con resultado satisfactorio.

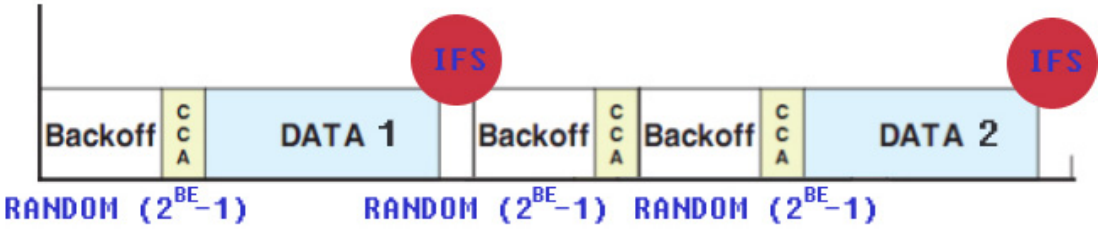

Fig.3.6: Proceso de envío de paquetes con CSMA/CA no ranurado.

En este proceso de acceso al medio interviene un tiempo adicional llamado IFS que es un tiempo finito que necesita la capa MAC para procesar la información recibida por la capa física. En el Anexo 5 se encuentra la definición que aporta el estándar 802.15.4 sobre el tiempo de IFS. El valor del IFS va en función del tamaño de la trama:

- Si se trata de tramas cortas que tiene como máximo el valor aMaxSIFSFrameSize (18 bytes), el valor IFS recibe el nombre de SIFS.

- Si se trata de tramas largas que superan al valor aMaxSIFSFrameSize (18 bytes), el valor IFS recibe el nombre de LIFS.

3.1.2.3. CSMA/CA ranurado

En las redes con Beacon se sigue el algoritmo de la **Fig.3.7** para acceder al medio, a excepción de aquellas transmisiones en las que el dispositivo tiene un tiempo garantizado gracias al coordinador de la red. Mediante una estructura llamada supertrama, el coordinador es capaz de gestionar el acceso al medio de todos los dispositivos de la red, de esta forma se deja a un lado la lucha constante que se lleva a cabo en las redes sin beacon y se pasa a competir por acceder al medio en los tiempos no asignados por el coordinador.

En este caso, intervienen más variables que en el modelo anterior como pueden ser la ventana de contención (CW) y el modo de prolongación de batería (BLE), mientras que se mantienen: BE, NB, aMaxBE y macMaxCSMABackoffs.

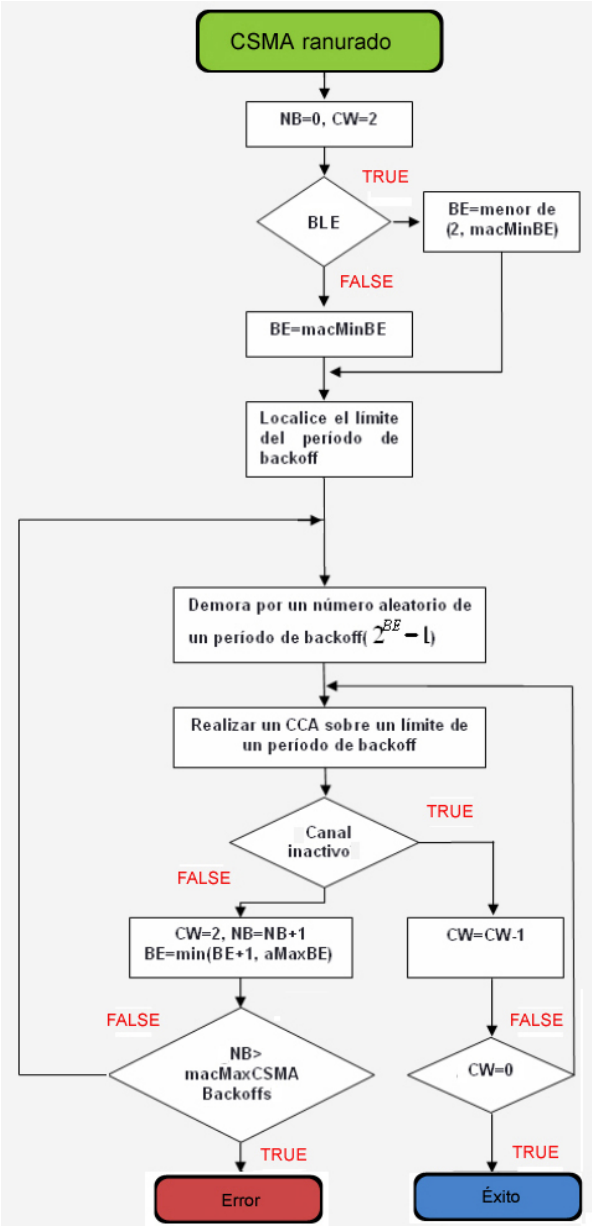

Fig.3.7: Diagrama CSMA/CA no ranurado.

3.2. ZigBee

ZigBee fue desarrollado por un grupo de empresas con el objetivo de encontrar una tecnología inalámbrica de bajo coste (ver [8]). En un principio su uso está destinado para redes WPAN, pero debido a que algunos dispositivos pueden llegar a tener un alcance superior a 75 metros, también está abarcando el campo de las WLAN.

ZigBee utiliza las especificaciones del IEEE 802.15.4 anteriormente comentadas y añade una serie de protocolos para gestionar la estructura de las redes con el objetivo de minimizar el consumo.

3.2.1. Dispositivos y topología

3.2.1.1. Dispositivos

En función del nivel en el que se analicen las redes, pueden clasificarse los dispositivos de una forma u otra.

Desde el punto de vista del estándar IEEE 802.15.4:

- **Dispositivo PAN**: Dispositivo que cumple las propiedades marcadas por 802.15.4

- **Coordinador PAN**: Existe uno por red. Es el encargado de la formación de la red y de su mantenimiento.

- **Coordinador**: Dispositivo responsable de la asociación y disociación en la red.

- **FFD (dispositivo de función completa)**:
  - Puede llevar a cabo cualquier función dentro de la red.
  - Puede utilizar más recursos y desempeñar funciones con una carga de datos importante.
  - Comunicación tanto con RFD como con FFD.

- **RFD (dispositivo de función reducida)**:
  - Funciones limitadas
  - Actúan como sensores dentro de la red, dispositivos finales.
  - Sólo pueden comunicarse con un FFD.

Desde el punto de vista de ZigBee:

- **ZigBee Coordinador (ZC):**
  - Uno sólo por red.
  - Inicia la formación de la red.
  - Actúa como 802.15.4 PAN coordinador.
  - Puede actuar como *router* una vez formada la red.

- **ZigBee Router (ZR)**:
  - Componente de red opcional.
  - Se puede asociar a un coordinador o a un *router* que forme parte de la red.
  - Actúa como 802.15.4 coordinador.
  - Participa en el encaminamiento de información.

- **ZigBee Dispositivo final (ZED):**
  - Componente de red opcional.
  - No participa en el enrutamiento.
  - No asocia otros dispositivos a la red.

3.2.1.2. Topología

Una de las grandes ventajas que presenta la tecnología ZigBee, es la capacidad de mediante una red mallada crear múltiples caminos alternativos que darán robustez ante cualquier adversidad. Pese a que este tipo de topología es la más utilizada, existen otras que se comentan a continuación y que se pueden observar en la **Fig.3.8**.

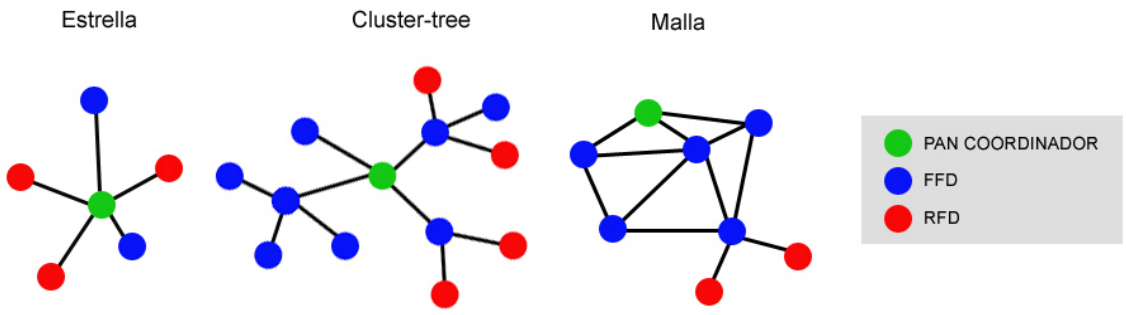

Fig.3.8: Topologías de red.

- Estrella: Las comunicaciones siempre pasan por el nodo central, que actúa como coordinador PAN. Este tipo de redes son bastante vulnerables ya que dependen del buen funcionamiento del nodo central. Ante la caída del coordinador, es imprescindible tener implementado un sistema para que la red vuelva a recalcularse.

- Malla: Como se ha comentado es la topología predilecta de la tecnología ZigBee. Se crean múltiples caminos dentro de la red que dan robustez ante la caída de uno o más nodos. La estructura mallada puede complementarse con pequeñas ramas de las que cuelgan dispositivos de funciones reducidas.

- Cluster-tree: Topología que engloba la filosofía de las dos topologías anteriores. Se crean una serie de clusters en los que un FFD adopta la función de coordinador del cluster. Cada coordinador de cluster se encarga de comunicarse con todos aquellos dispositivos que tiene conectados, creando una disposición en estrella. Como complemento, si los vértices de la estrella son FFD, estos pueden enlazarse con RFD, igual que pasa en la topología mallada.

CAPÍTULO 4. HERRAMIENTAS DE TRABAJO

Una vez conocida la tecnología sobre la que se trabaja, se enumeraran las diversas herramientas que se han utilizado en el TFC. Se aportarán las especificaciones de las motas TelosB, junto con una pequeña descripción del lenguaje de programación NesC. Se hará referencia al sistema operativo en el que se ha llevado acabo el proyecto y finalmente se hablará sobre las herramientas que se han utilizado para debugar las aplicaciones.

4.1. Motas TelosB

Plataforma experimental desarrollada por la University of California at Berkeley y Crossbow Technology Inc. Comúnmente reciben el nombre de motas (ver [7]). En la **Fig.4.1** se puede apreciar como son los dispositivos con los que se ha trabajado.

Fig.4.1: Imagen mota TelosB.

En la **Fig.4.2** se puede apreciar el diagrama de bloques de las motas, pese a que sólo se ha trabajado con aspectos relacionados con la comunicación, dejando a un lado el apartado de sensores o entradas de las mismas.

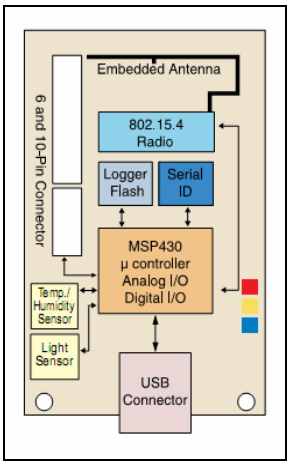

Fig.4.2: Diagrama motas TelosB.

Las características técnicas más importantes son las siguientes:

- Como transceptor RF se utiliza el chip CC2420 (ver [6]):
  - Antena integrada en la placa base.
  - Utiliza la banda de 2.4 GHz.
  - Velocidad de transmisión de 250 Kbps, modulación O-QPSK.
  - Modulación O-QPSK.
  - De los dos mecanismos de acceso al medio, CC2420 bajo TinyOS utiliza el modo sin beacon, el modo beacon no viene implementado.

- Puerto USB para poder obtener datos o programar la mota.

- Microcontrolador MSP430 de 8 MHz con 10 KB de RAM.

- Posibilidad de integrar sensores variados: luz, temperatura y humedad.

- Funciona con dos pilas AA o mediante conexión USB.

- Posibilidad de expansión mediante 16 pins.

- 3 leds y botón para interactuar con el usuario.

4.2. TinyOS

TinyOS es un sistema operativo open source basado en componentes para redes de sensores inalámbricas (ver [3]). Está diseñado para sistemas embebidos y su estructura resulta útil en sistemas con recursos limitados ya que permite reutilizar los diferentes componentes utilizados.

Por otro lado el hecho de poder crear aplicaciones ensamblando componentes hace que en algunos casos no sea necesario implementar operaciones de bajo nivel ya que existen módulos que realizan estas operaciones (ver [10]). TinyOS es un sistema operativo con un reducido núcleo multitarea y funciona a partir de eventos producidos que llamarán a funciones; esto se conoce como “event-driven”.

Cabe destacar que existen varias versiones de TinyOS que en algunos casos crean problemas de incompatibilidades. Se ha tomado la decisión de utilizar la versión 2.0 que es perfectamente estable en las motas TelosB.

El diseño del Kernel de TinyOS está basado en una estructura de dos niveles de planificación:

- Eventos: están pensados para realizar un proceso pequeño y pueden interrumpir las tareas que se están realizando.

- Tareas: están pensadas para realizar una cantidad mayor de procesamiento y no son críticas en tiempo.

Gracias a esta estructura de dos niveles de planificación se consigue un alto rendimiento en aplicaciones de concurrencia intensiva y se utiliza de manera eficiente la capacidad de la CPU.

4.3. Lenguaje de programación NesC

Para la programación de las motas TelosB se ha utilizado el lenguaje NesC que es una variación de C (ver [2]). NesC está orientado a sistemas embebidos y su estructura modular reduce el tamaño del código y elimina gran parte de fuentes potenciales de errores.

NesC ofrece:

- Separación entre construcción y composición. En NesC se puede hablar de módulos y configuraciones. Los módulos contienen el código de la aplicación mediante la implementación de interfaces. Y las configuraciones se usan para unir los componentes.

- Interfaces bidireccionales. Las interfaces son los accesos a los componentes y contienen comandos y eventos que son los que implementan las funciones. El proveedor de una interfaz implementa los comandos mientras que el que las utiliza implementa eventos.

- Unión estática de componentes mediante sus interfaces. Con esto se consigue eficiencia en tiempos de ejecución y robustez del diseño.

En NesC una aplicación se ve representada por un conjunto de componentes agrupados como puede observarse en la **Fig.4.3**.

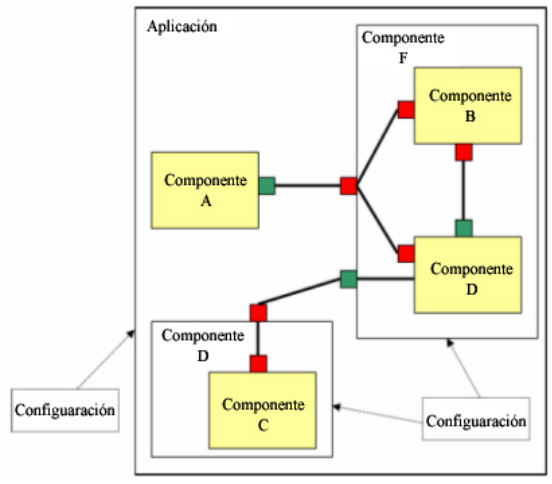

Fig.4.3: Diagrama NesC.

4.3.1. Ejemplo de wiring.

El concepto wiring cobra una gran importancia dentro de este lenguaje de programación. La traducción al castellano, vendría a ser “enganche”. Para entender dicho concepto, se ha utilizado un pequeño ejemplo que ayuda a entender su significado de una forma más clara (ver **Fig.4.4**).

App.c → Bpp.c

Fig.4.4: Ejemplo wiring.

Este enlace quiere decir que a partir de este momento, el componente Bpp proporciona la interfaz c y ésta será utilizada por el componente App.

4.4. XubunTOS

El sistema operativo sobre el que se ha trabajado es una versión de Xubuntu destinada a trabajar con TinyOS (ver [4]). En ella se incluyen todas las herramientas necesarias, debidamente actualizadas y configuradas para no tener que dedicar tiempo a adaptar nuestra versión de linux con los paquetes oficiales. La versión de XubunTOS es la 2.0, se compone de Xubuntu 7.04 y permite trabajar con TinyOS 1.x y 2.0.2, incluyendo ejemplos y tutoriales oficiales.

4.5. Herramientas para debugar

Uno de los problemas que presenta la programación de las motas TelosB, es el poco margen que existe para debugar y comprobar que realmente se están realizando las funciones programadas. No existe un entorno de programación o emulación como el que pueden presentar .net o java, por lo que hay que buscar alternativas para asegurarse de que las aplicaciones realizan la tarea asignada.

Para poder debugar a lo largo del proyecto, se han utilizado tanto los 3 leds presentes en las motas, como tres pequeñas aplicaciones diseñadas en java para controlar las comunicaciones entre dispositivos.

Para poder capturar la información que transmiten las motas, se ha utilizado un nodo que realiza la función de sniffer llamado estación base. Una vez capturados los datos que se encuentran en su radio de alcance y en el mismo canal, transmite por el puerto serie la información y posteriormente ésta es interpretada por una aplicación java. Para poder cubrir las necesidades del trabajo, se han diseñado 3 aplicaciones que en cada momento muestran por pantalla la información necesaria.

A lo largo de este documento se utilizarán en gran medida capturas como la que se puede apreciar en la **Fig.4.5**. Simplemente se limita a mostrar por pantalla Byte a Byte el paquete recibido. Mostrando tanto datos de la cabecera como del campo de datos.

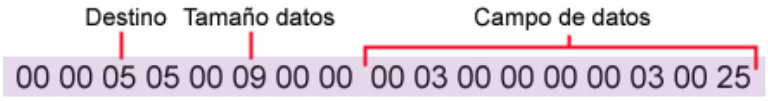

Fig.4.5: Captura estación base V1.0.

Para la presentación final del proyecto y en algún punto de este trabajo, se utilizarán capturas que proporcionan la información siguiendo las pautas de la **Fig.4.6,** es mucho más completa que la anterior y acumula una serie de datos de paquetes procesados que ayudan a entender mejor el escenario sobre el que se trabaja.

```
Tiempo recepción en nanosegundos (10-9 segundos): 321174846000.
Identificador inicial de la mota: 5, Identificador en la red: 5, Destino: 3.
Número de paquetes total: 75000.0 ; Separado: Nodo 3: 12185.0 ; Nodo 2: 24084.0 ; Nodo 4: 26327.0 ; Nodo 5: 12404.0.
Porcetaje: Nodo 3: 16.246666% ; Nodo 2: 32.112% ; Nodo 4: 35.102665% Nodo 5: 16.538666%.
Paquete o ACK perdido: Nodo 2: 3456, Nodo 4: 5157.
```

Fig.4.6: Captura estación base V2.0.

Para poder representar el RSSI también se ha creado una pequeña aplicación que muestra el valor que se considera correcto y el valor leído, como se puede apreciar en la **Fig.4.7**.

```
Identificador en la red: 5.
Diferencia de tiempo con paquete anterior: 14972000.
Valores RSSI +(-45 dBm). RSSI fijo: -13. RSSI leido: -6. Errores : 4.
```

Fig.4.7: Captura valor RSSI.

CAPÍTULO 5. APLICACIÓN Y PRUEBAS

Una vez contextualizado el proyecto y conocidas la tecnología y herramientas que se han utilizado, se dedicará este capítulo a implementar un ataque en una red de sensores, siendo el objetivo principal el acceso al medio y las posibles vulnerabilidades.

Los primeros pasos se centrarán en estudiar la viabilidad de la idea inicial planteada, desarrollando de forma modular los pasos a seguir. En función del resultado obtenido, se reajustará la idea inicial y finalmente se implementará un método de detección y respuesta ante el ataque desarrollado.

5.1. Idea inicial

La idea inicial, sobre la que se ha trabajado, se centra en que dos motas ocupen el medio de forma constante intercambiándose los identificadores y aprovechando la prioridad acumulada en la transmisión anterior. A continuación se detalla el ataque paso a paso:

1. Una mota inicia una comunicación, enviando una ráfaga de paquetes y ocupando el medio durante un tiempo determinado.

2. Una vez se finaliza la transmisión, la mota receptora adopta la identidad de la mota que inicialmente transmitía para aprovecharse de los privilegios que esta tenía a la hora de acceder al medio. De esta forma no tendrá que competir con el resto de nodos de la red, porque su nuevo identificador ya tiene ganado el medio.

3. La mota que en un principio transmitía también cambiará su identificador, de esta forma las dos motas atacantes mediante este intercambio de identificadores pueden apoderarse del medio de forma constante sin tener que competir por él.

Una vez explicados los objetivos y los pasos que se tienen que seguir, se tiene que analizar si las motas TelosB, el mecanismo de acceso CSMA/CA y las posibilidades de NesC permiten llevarlo a acabo.

Los puntos clave para poder desarrollar el ataque se centran en:

- Poder enviar en ráfaga ocupando el medio de una forma constante.
- Poder intercambiar los identificadores que intervienen en las comunicaciones, más en concreto en el acceso al medio.
- Poder deshabilitar el CSMA/CA.

5.2. Ocupar el medio

Uno de los problemas clave en el desarrollo del TFC, es que no existe una forma predefinida para mandar ráfagas de paquetes ocupando el medio de forma constante. Cada paquete tiene que pasar por el proceso de CSMA/CA y competir por el medio, sin acumular ningún privilegio por haber enviado anteriormente.

Por este motivo se ha intentado modificar con las variables que proporciona el acceso al medio con el objetivo de lograr ocupar el medio el máximo tiempo posible. Concretamente dando a la variable BE el valor inicial de 0. Para que todas las motas sigan las mismas reglas, este cambio se ha realizado en todas ellas.

A continuación se muestra un estudio de probabilidad para saber si realmente modificando esta variable se puede lograr el objetivo.

Cabe mencionar que algunas de las variables que participan en el acceso al medio aumentan exponencialmente y además hay que analizar el proceso durante varios instantes de tiempo. Por este motivo, se han realizado dos tipos de análisis:

- Detallado: Se ha realizado un estudio exhaustivo en uno de los puntos más favorables para lograr ocupar el medio de forma constante.

- Aproximado: En el resto de puntos se ha realizado un estudio más aproximado, pero igual de necesario, para tener una visión global

5.2.1. Análisis detallado

Antes de comenzar, hay que advertir que este estudio se basa en la **Fig.An.19** del Anexo 7. Por desgracia las dimensiones son demasiado grandes (6873 x 7319 píxeles) y es imposible apreciar con detalle la figura. No obstante en el Anexo 7 se indica la dirección de un enlace externo en su resolución original.

Condiciones iniciales del estudio:

- Se ha trabajado con dos nodos.
- Se han despreciado probabilidades del orden de 10^{-5}.
- Se han calculado las probabilidades hasta el primer caso donde BE = 4, con sus respectivos 16 posibles valores de backoff.
- Nodo 1 inicia el proceso de transmisión de datos, nodo 2 realiza su primera comprobación del medio cuando nodo 1 comienza a transmitir (128μs en la línea de tiempo).
- Se ha trabajado con paquetes de 18 Bytes. Éste es el valor límite para seguir utilizando SIFS. Realmente este parámetro es el único que se puede variar, siempre que se respete la correspondencia entre tamaño del paquetes y elección de tiempo IFS (SIFS/LIFS).

- Al realizar la comprobación del medio, CCA, sólo se devuelve libre si durante todo el tiempo que se escucha el canal éste está libre.

Probabilidades estudiadas:

- Probabilidad de que nodo 1 mande 1 paquete.
- Probabilidad de que nodo 1 mande 2 paquetes seguidos.
- Probabilidad de que nodo 1 mande 3 paquetes seguidos.
- Probabilidad de que nodo 1 mande 4 paquetes seguidos.
- Probabilidad de colisión.

5.2.1.1. Probabilidad de que nodo 1 mande 1 paquete

Como ya se ha comentado, el intento de acceder al medio del nodo 2, se realiza una vez el nodo 1 ya tiene el medio ganado y empieza a transmitir sus datos.

Por este motivo, siempre será posible:

P(Nodo 1 mande 1 paquete) = 1 = 100%.

5.2.1.2. Probabilidad de que nodo 1 mande 2 paquetes seguidos

En este caso, el nodo 1 tiene que ser capaz de mandar 2 paquetes seguidos. Hay que tener en cuenta que cuando se produce una colisión, los paquetes han sido enviados. No se ha valorado si la información llega o no, solamente si el nodo 1 ha podido enviar su paquete.

P(Nodo 1 mande 2 paquetes seguidos) = 1 – P(Nodo 1 NO mande 2 paquetes seguidos)

P(Nodo 1 N0 mande 2 paquetes seguidos) =

$$\left[\left(\frac{1}{2}*\frac{1}{4}*\frac{1}{8}\right)+\left(\frac{1}{2}*\frac{1}{4}\right)+\left(\frac{1}{2}*\frac{1}{4}\right)\right]=\left[\frac{1}{64}+\frac{1}{8}+\frac{1}{8}\right]=0{,}26565$$

P(Nodo 1 mande 2 paquetes seguidos) = 1 – 0,26565 = 0,73435 = 73,43%

Para obtener estos resultados se ha evaluado la **Figura An.19** teniendo en cuenta todos aquellos casos en los que no se pueden mandar 2 paquetes seguidos por parte del nodo 1. Una vez obtenido el resultado, se ha restado el valor máximo de probabilidad (1 ó 100%) y de esta forma se obtiene el valor que se buscaba inicialmente.

5.2.1.3. Probabilidad de que nodo 1 mande 3 paquetes seguidos

En este caso, el nodo 1 tiene que ser capaz de mandar 3 paquetes seguidos.

P(Nodo 1 mande 3 paquetes seguidos) = 1 – P(Nodo 1 NO mande 3 paquetes seguidos)

P(Nodo 1 N0 mande 3 paquetes seguidos) =

$$\left[\left(\frac{1}{2}*\frac{1}{4}\right)+\left(\frac{1}{2}*\frac{2}{4}\right)+\left(\frac{1}{2}*\frac{1}{4}*\frac{1}{8}\right)+\left(\frac{1}{2}*\frac{1}{4}*\frac{1}{8}*\frac{1}{16}\right)*9\right]=$$

$$\frac{1}{8}+\frac{2}{8}+\frac{1}{64}+8{,}07890*10^{-3}=0{,}3988$$

P(Nodo 1 mande 3 paquetes seguidos) = 1 – 0,3988 = 0,6011 = 60,01%

En este caso se ha repetido el procedimiento del caso anterior, pero sobre tres paquetes seguidos.

5.2.1.4. Probabilidad de que nodo 1 mande 4 paquetes seguidos

En este caso, el nodo 1 tiene que ser capaz de mandar 4 paquetes seguidos.

P(Nodo 1 mande 4 paquetes seguidos) = 1 – P(Nodo 1 NO mande 4 paquetes seguidos)

P(Nodo 1 NO mande 4 paquetes seguidos) =

$$\left[\left(\frac{1}{2}*\frac{1}{4}\right)+\left(\frac{1}{2}*\frac{2}{4}\right)+\left(\frac{1}{2}*\frac{1}{4}*\frac{1}{8}\right)+\left(\frac{1}{2}*\frac{1}{4}*\frac{1}{8}*\frac{1}{16}\right)*28\right]=$$

0,125 + 0,25 + 0,01565 + 0,02734 = 0,41799

P(Nodo 1 mande 4 paquetes seguidos) = 1 – 0,3988 = 0,5820 = 58,20%

Para obtener el resultado, se ha repetido el procedimiento pero sobre 4 paquetes consecutivos.

5.2.1.5. Probabilidad de colisión

En el estudio hay varios puntos en los que tanto nodo 1 como nodo 2 realizan la escucha del medio en el mismo tiempo. Realmente esto en un caso práctico es bastante improbable, ya que tienen que coincidir en la misma unidad de tiempo, pero en todo caso se ha calculado la probabilidad.

Hay que recordar que en este caso nodo 1 no deja de transmitir. Volver a recuperar la información que se ha perdido en la colisión o no, dependerá del mecanismo concreto implementado. Este objetivo queda fuera del ámbito del presente estudio.

$$\text{P(Nodo 1 y Nodo 2 colisionen)} = \left[\left(\frac{1}{2} * \frac{1}{4} * \frac{1}{8}\right) * 5\right] = 0{,}0781 = 7{,}81\%$$

5.2.1.6. Conclusiones

Como puede observarse en los resultados finales (ver **Fig.5.1**), es complicado asegurar una ráfaga de paquetes ocupando el medio, ya que en el mejor de los casos (2 paquetes) de cada 4 intentos existe 1 caso no favorable.

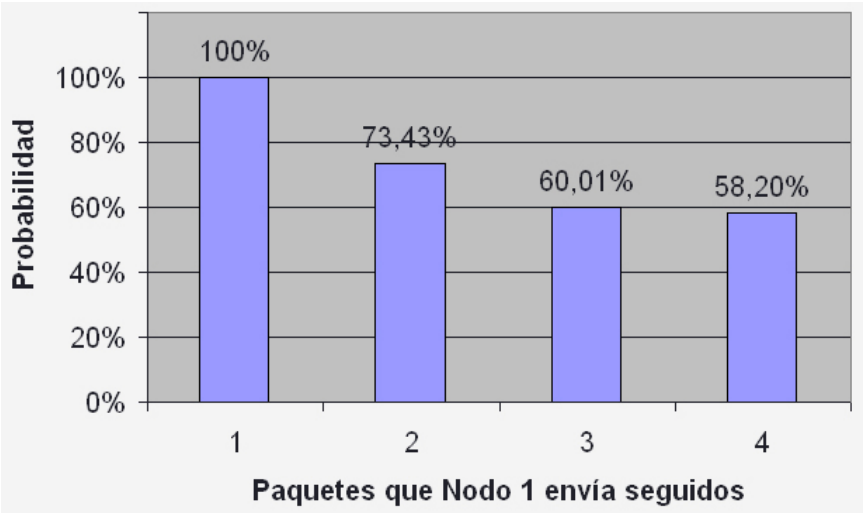

Fig.5.1: Probabilidad de tx seguidas por parte del nodo 1.

5.2.2. Análisis global aproximado

En el caso anterior, se ha realizado el estudio partiendo de un instante de tiempo determinado, pero realmente existen múltiples instantes en los que el nodo 2 puede empezar a transmitir. Por este motivo, se han analizado varios puntos dentro del tiempo de transmisión para tener una visión más global de la situación. En este estudio sólo se ha tenido en cuenta la probabilidad de que el nodo 1 mande 2 paquetes seguidos.

En la **Fig.5.2** se puede apreciar la probabilidad de que el nodo 1 realice 2 transmisiones seguidas en función de cuando accede el nodo 2 al medio. Se ha evaluado la probabilidad en diversos momentos, como el momento en que el nodo 1 realiza la comprobación del canal (CCA), cuando transmite la información o cuando tiene que esperar el tiempo SIFS. Las líneas rojas indican los diferentes instantes en los que nodo 2 prueba de acceder al medio y lo relacionan con la probabilidad de que el nodo 1 pueda enviar dos paquetes seguidos. Los cálculos para obtener estos resultados se adjuntan en el Anexo 7.

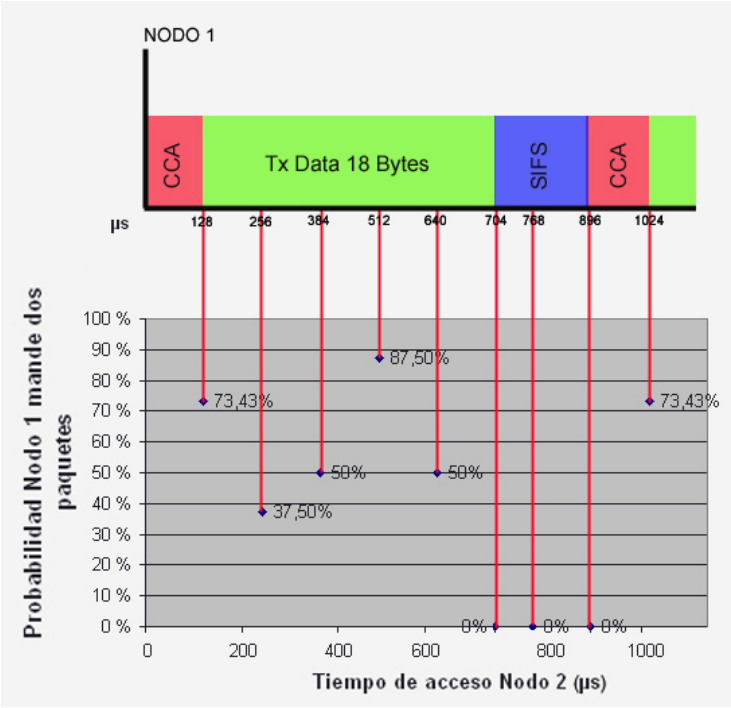

Fig.5.2: Probabilidad de que nodo 1 envíe dos paquetes seguidos.

Como se puede apreciar en estos resultados, el valor 73,43% del análisis detallado que se ha realizado anteriormente, es un valor que no se mantiene ya que existen varios puntos en los que la probabilidad que tiene el nodo 1 de mandar 2 paquetes seguidos, es inferior al 50%. También se aprecian ciertos puntos críticos en los que el nodo 2 siempre podrá ganar el medio.

5.2.3. Prueba práctica realizada con las motas TelosB

Una vez finalizados los estudios teóricos de los puntos anteriores, se ha realizado una parte práctica para observar si los resultados concuerdan. Se han utilizado dos motas enviando paquetes constantemente con una potencia de transmisión de 0 dBm y con el backoff inicial 0. Como se aprecia en la **Fig.5.3** el nodo 1, de un total de 1000 transmisiones, realiza el 63% enviado 1 solo paquete seguido, mientras que en el 31% de los casos es capaz de realizar una ráfaga de 2 paquetes. Estos resultados prácticos están por debajo de los teóricos, ya que la probabilidad teórica de que el nodo 1 mande dos paquetes seguidos se encuentra entre un 40% y 45%.

El motivo por el que se han producido estos resultados, es que las motas TelosB no cumplen exactamente lo marcado por el estándar al seleccionar el backoff de espera. En la primera iteración si que se respeta el valor 0 que se ha asignado, pero si se encuentra el medio ocupado, en la segunda iteración y posteriores no se sigue la ecuación $2^{BE} - 1$. Las motas determinan un nuevo valor de backoff llamado CongestionBackoff que es seleccionado mediante la interfaz Random y no cumple las pautas que se han explicado en la parte teórica.

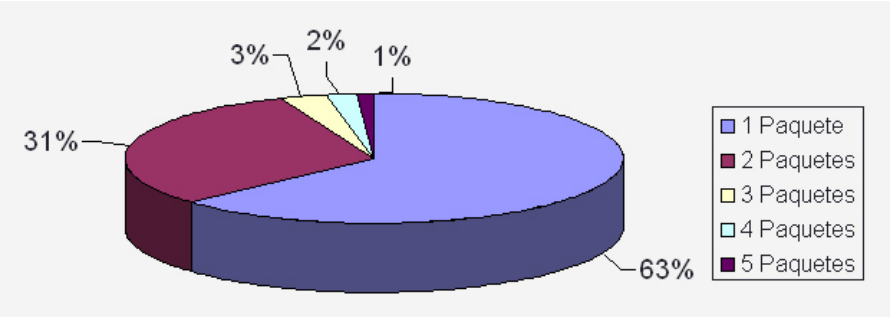

Fig.5.3: Probabilidad de tx seguidas por parte del nodo 1.

5.2.4. Conclusiones

Como conclusiones finales de esta parte del trabajo realizado, puede asegurarse que esta "técnica" para intentar apoderarse del medio no es eficiente y por tanto no es un método adecuado para implementar el ataque que se pretende estudiar.

Por este motivo, la idea original en la que un nodo estaba ocupando el medio de forma constante, ha sido ligeramente reajustada, como puede verse en el punto 5.6, para llevar a cabo el ataque de forma efectiva.

5.3. Intercambio de identificadores

Un punto clave del trabajo, ha sido determinar cual es el identificador único que tiene cada mota al entablar una comunicación (nivel MAC) y buscar el modo de modificarlo. De esta forma los dos nodos atacantes pueden ir intercambiándose las direcciones. En el punto 5.1 se encuentra explicado el ataque a nivel teórico.

Inicialmente se consideró que el identificador único que interviene en las comunicaciones de cada mota correspondía a una variable almacenada en memoria RAM que recibe el nombre TOS_NODE_ID. Algunos tutoriales oficiales de TinyOS pueden provocar que el usuario final tenga esta percepción. Al encontrarse alojada en memoria RAM, las tareas para poder modificarla, resultaban demasiado complicadas para un sistema que permite la configuración remota.

Tras examinar los archivos que definen las comunicaciones, puede determinarse que TOS_NODE_ID, es la variable que se introduce en el momento de compilar la aplicación y a partir de esta asignación, dicho valor se guarda secuencialmente en: TOS_NODE_ID → TOS_AM_ADDRESS → addr. Siendo addr el identificador buscado. En la **Fig.5.4** se aprecia la línea de comando utilizada para compilar la aplicación donde se observa como en ésta se indica el valor del identificador.

```
...
daniel@TFC:~$ make telosb reinstall.3 bsl/dev/ttyUSB0
/*El 3 indica el valor del identificador*/
...
```

Fig.5.4: Línea de comando para cargar código en mota.

Una vez determinado el identificador que tiene que modificarse, el siguiente punto se centra en buscar entre las interfaces que proporciona TinyOS para poder realizar el cambio dentro de la ejecución del código. Concretamente la interfaz ActiveMessageAddress (ver **Fig.5.5**) proporciona métodos para poder realizar esta función.

```
...
interface ActiveMessageAddress {
/*Nos permite asignar dirección y grupo*/
async command void setAddress(am_group_t group, am_addr_t addr);
...
```

Fig.5.5: Interfaz ActiveMessageAddress.

Mediante setAddress puede modificarse tanto la dirección del nodo a nivel de comunicaciones como el grupo al que pertenece, también llamado PAN ID. Esto quiere decir que una misma mota, puede tener valores diferentes para TOS_NODE_ID y addr. En la **Fig.5.6**, se representan los efectos de realizar el cambio de identificador. Una vez realizado dicho cambio y leídos los parámetros TOS_NODE_ID y addr se puede observar como el valor de las variables ya no es el mismo.

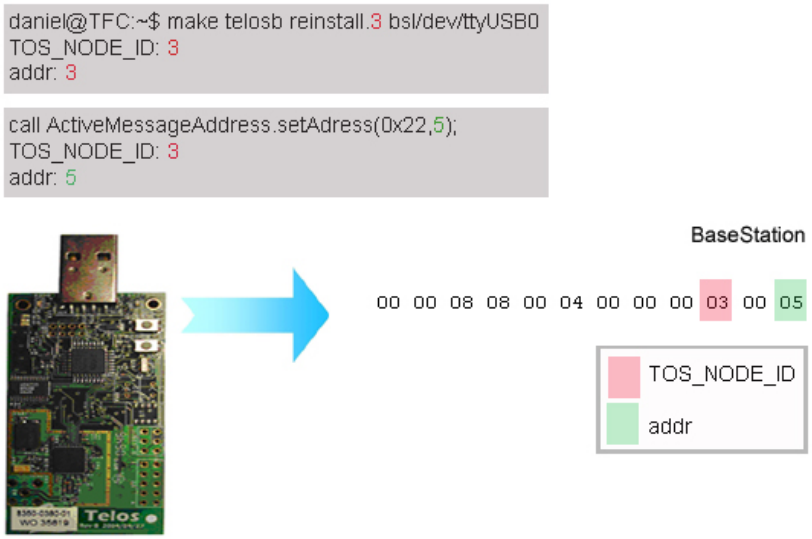

Fig.5.6: Valores TOS_NODE_ID y addr diferentes.

En el Anexo 6 se ha adjuntado una aplicación que corrobora lo descrito en este punto. En dicha aplicación se realiza un cambio de identificador tras un tiempo determinado y de esta forma la mota que ha realizado el cambio comienza a recibir datos de otros nodos.

5.4. Deshabilitar CSMA/CA

Este punto trata de explicar el método que se ha utilizado para que los nodos no implementen el protocolo de acceso al medio CSMA/CA en el momento de la transmisión.

CSMA/CA se caracteriza entre otras cosas por buscar un acceso al medio equitativo entre todos los nodos que lo implementan. Si se respetan los mismos parámetros de configuración en todos los nodos, el número de accesos al medio durante un periodo de tiempo acaba siendo el mismo para cada mota.

En el momento que uno de los nodos modifica dichos parámetros de configuración, sin que se realicen los mismos cambios en el resto o deja de implementar el algoritmo CSMA/CA, se desajusta lo comentado en el párrafo anterior.

Para demostrar estas afirmaciones teóricas, se ha diseñado un pequeño escenario sobre el que se han evaluado los efectos antes de deshabilitar CSMA/CA y tras eliminar dicho algoritmo.

El escenario está formado por tres nodos, con identificadores 2, 3 y 4, junto a una estación base que capta todo el tráfico de la red.

Desarrollo paso a paso:

1. Nodo 2 envía un mensaje broadcast cada 300 ms.
2. Nodo 3 y nodo 4, implementan el mismo código: en el momento que procesan el mensaje de nodo 2, pasan a emitir un mensaje.
3. La estación base ayudará a determinar si ha emitido primero nodo 3 o nodo 4.
4. Las distancias de nodo 3 y nodo 4 respecto a nodo 2 son las mismas (1 metro).
5. Todas las motas transmiten con una potencia de 0 dBm.

5.4.1. CSMA/CA implementado en todos los nodos

En este caso, se ha trabajado en igualdad de condiciones, es decir, tanto el nodo 3 como el nodo 4 tienen que realizar el proceso de CSMA/CA para poder enviar su mensaje. En la **Fig.5.7**, se aprecian las dos situaciones que se puedan dar; que acceda primero el nodo 3 y el nodo 4 se encuentre el medio ocupado, y viceversa.

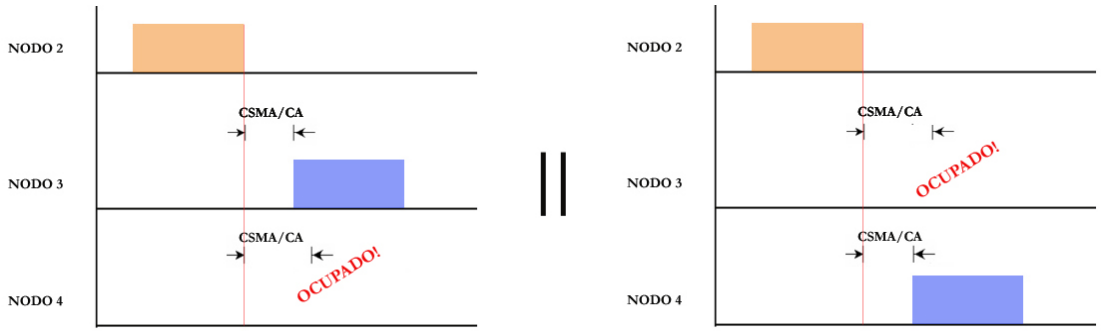

Fig.5.7: Posibles situaciones en el acceso al medio.

Para ver la probabilidad que tienen nodo 3 y 4 a la hora de acceder primero al medio se ha realizado un estudio con 1000 muestras. Si en la estación base, después del mensaje broadcast del nodo 2 aparece el mensaje del nodo 3, indicará que dicho nodo ha sido el que primero ha accedido al medio. Esta situación es equivalente si se habla del nodo 4.

Los resultados obtenidos son los mostrados en la **Fig.5.8**.

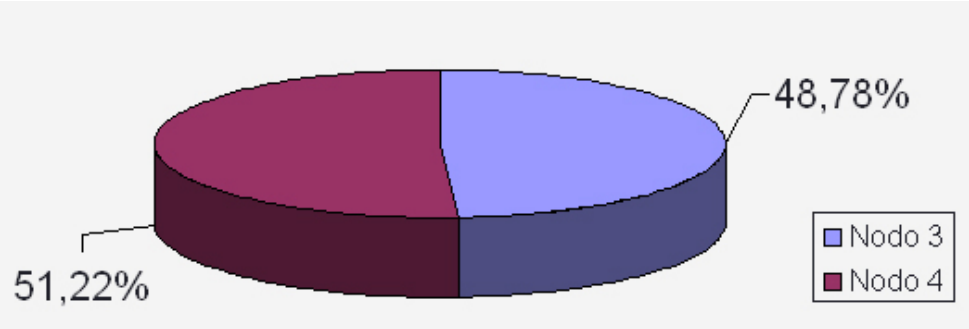

Fig.5.8: Probabilidad de acceder al medio.

Como puede observarse la probabilidad es bastante similar para cada nodo, ajustándose cada vez más al teórico 50% en el que cada nodo realmente tiene las mismas posibilidades que el otro para acceder antes al medio, ya que no se han realizado modificaciones en el acceso al medio.

5.4.2. CSMA/CA deshabilitado en un nodo

En este punto se ha tratado de encontrar un método en el que nodo 4 sea capaz de enviar siempre antes que nodo 3. La forma más eficiente es saltándose el proceso de CSMA/CA, de esta forma tendría que suceder lo mostrado en la **Fig.5.9**; que el nodo 4 ganase el acceso al medio.

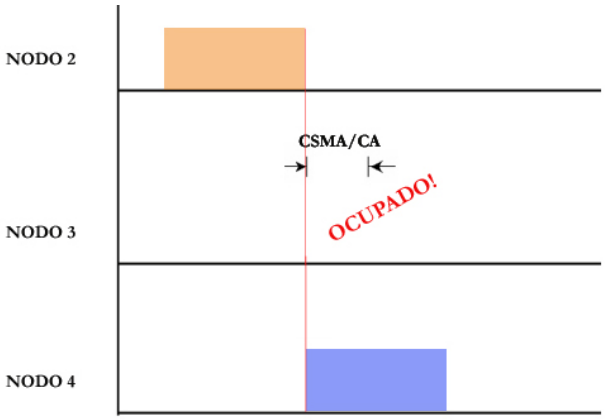

Fig.5.9: Nodo 4 gana el acceso al medio.

Como ya se ha explicado en este TFC (ver punto 3.1.2.1) el CSMA/CA no ranurado se centra en un tiempo de backoff y en una comprobación del medio antes de transmitir. El objetivo es establecer el backoff inicial a 0 y que no se realice la comprobación del medio, simplemente que se envíe la información. Estas dos tareas se pueden realizar modificando el archivo CC2420CsmaP.nc o accediendo mediante eventos.

Para establecer el backoff inicial a 0, se ha accedido al archivo CC2420CsmaP.nc y se ha modificado el backoff inicial asignándole el valor 0 como puede observarse en la **Fig.5.10**.

```
                                        …
async event void SubBackoff.requestInitialBackoff(message_t *msg) {
    call SubBackoff.setInitialBackoff ( /*call Random.rand16()
        % (0x1F * CC2420_BACKOFF_PERIOD) + CC2420_MIN_BACKOFF*/
0);
/*Se elimina todo el cálculo de Backoff y se le asigna el valor 0*/
                                        …
```

Fig.5.10: Modificación valor backoff.

De esta forma en la primera y única iteración no existe backoff. Una vez superado este paso, la mota tiene que ser capaz de enviar sin realizar el CCA. Para lograr este objetivo se han realizado una serie de cambios que ayudan a capturar el evento.

Mediante la interfaz RadioBackoff es posible establecer si se quiere utilizar o no CCA en la transmisión (ver **Fig.5.11**). Por defecto las motas lo implementan, pero asignando el valor 0 a la variable booleana CCA queda totalmente deshabilitado. En el módulo creado se ha capturado el evento en el que se reclama si se utiliza o no el CCA y mediante la función setCca, se ha asignado el valor FALSE.

```
                                        …
async event void CcaOverride.requestCca(message_t *msg){
/*Captura todos los métodos que preguntan si se tiene que usar CCA*/
call CcaOverride.setCca(FALSE);
  }
                                        …
```

Fig.5.11: Capturar evento CCA y asignación valor FALSE.

Como no se produce comprobación del medio, el paquete es enviado.

Si se repite el estudio anterior con 1000 muestras, los resultados son los esperados y el nodo que no implementa el CSMA/CA gana el acceso al medio cerca del 100% de los casos, como se puede observar en la **Fig.5.12**.

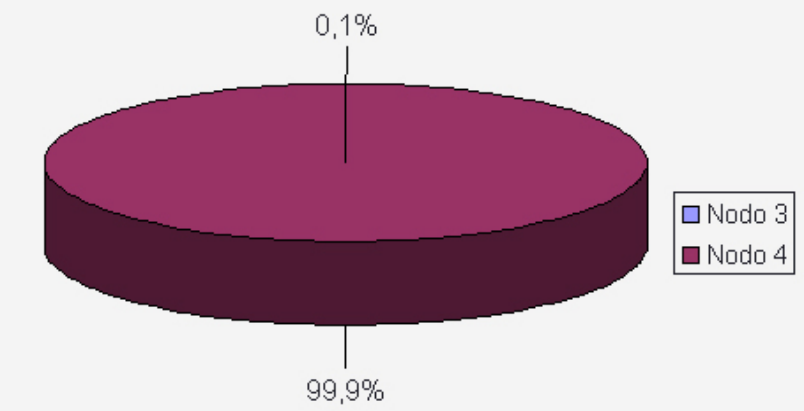

Fig.5.12: Nodo 4 gana el medio prácticamente el 100% de las veces.

Para concluir este apartado, se han capturado en la estación base los instantes de tiempo en los que se procesan los paquetes enviados por el nodo 3 y 4, tras recibir el paquete del nodo 2. En la figura **5.13** se han representado las 300 primeras pruebas de las 1000 totales, obteniendo un diagrama de puntos que nos muestra el tiempo de recepción de los nodos 3 y 4 en cada uno de los experimentos.

Los paquetes enviados por el nodo 4, tardan de media unos 600 µs tras la transmisión del nodo 2 (referencia 0 segundos), mientras que el nodo 3, que sí implementa CSMA/CA, lo hace 18 ms después. Estos parámetros de tiempo no se pueden utilizar para establecer tiempos de transmisión entre nodos, ya que el timestamp se captura en la estación base y no en las motas correspondientes. Simplemente se utilizan para confirmar que el nodo 4 envía primero, ya que la estación base implementa una cola FIFO que asegura que lo primero en entrar es lo primero en mostrarse en la aplicación java.

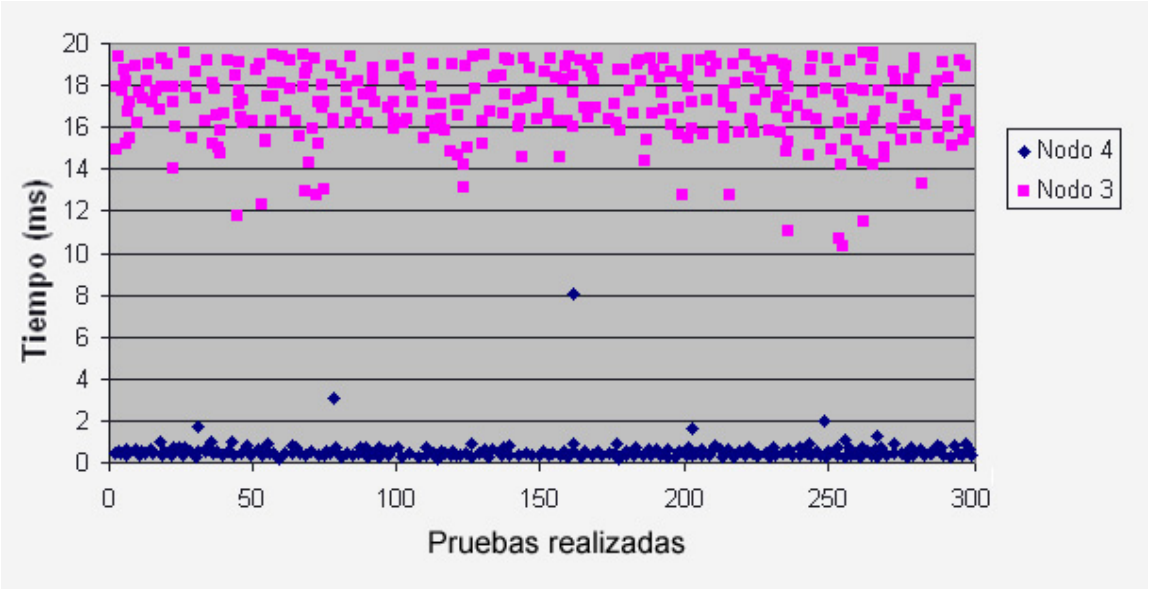

Fig.5.13: Tiempo de tx nodos 3 y 4 respecto nodo 2.

5.5. Deshabilitar CSMA/CA dentro de una red.

Una vez explicado el procedimiento para poder saltarse el CSMA/CA y acceder al medio el siguiente objetivo es ver los resultados dentro de una red simple. Como se aprecia en la **Fig.5.14**, la red está formada por dos nodos atacantes (nodo 2 y nodo 4) y dos nodos que cumplen las reglas del CSMA/CA. Uno de ellos (nodo 3) que intenta establecer una comunicación con otro nodo (nodo 5) que simplemente actúa de receptor. La distancia mayor entre nodos nunca supera los 4 metros y todas las motas transmiten con una potencia de 0 dBm.

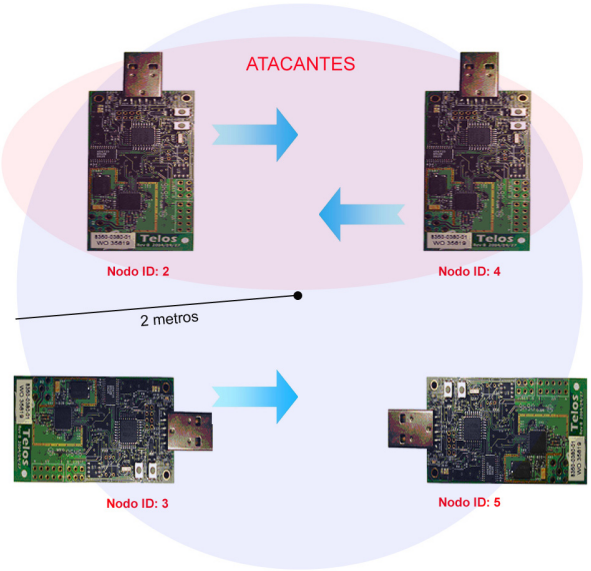

Fig.5.14: Nodos de nuestro escenario.

El objetivo de los dos nodos atacantes, es apoderarse el máximo tiempo posible del canal impidiendo que el nodo 3 pueda transmitir. Un punto importante a destacar es que todos los nodos están programados para enviar de forma continuada sin ningún tipo de retraso.

En un principio, se había pensado que tanto el nodo 2 como el 4, no implementasen reconocimientos. Pero al establecer una comunicación de dependencia en la que el nodo actúa en función de si le ha llegado o no el mensaje, es obligatorio utilizar ACK para asegurar que los paquetes han sido mandados y recibidos. Si no se implementa ACK cualquier posible colisión que provoque la pérdida del paquete, dejaría a estos dos nodos en stand-by. Por este motivo en el código de las motas se ha implementado la evaluación de ACK. En la **Fig.5.15** se puede apreciar la parte de código que se refiere a la evaluación del ACK.

```
                                                …
  event void AMSend.sendDone(message_t *msg, error_t error) {

    if(call PacketAcknowledgements.wasAcked(msg)) {
      call Leds.led0Toggle();
      //call ActiveMessageAddress.setAddress(red,4);
    }
else
    {
      call Leds.led1Toggle();
      post send();
    } }
                                                …
```

Fig.5.15: Evaluación ACK.

Para poder comparar los resultados, primero se han realizado las pruebas en una situación normal, es decir, en aquella situación en la que ningún nodo ha sido modificado. En este caso, los resultados concuerdan con lo esperado (ver **Fig.5.16**) y no se crea ninguna situación en la que uno o más nodos se apoderen del medio.

```
00 00 02 02 00 09 00 00 00 04 00 00 00 00 04 00 EC
00 00 04 04 00 09 00 00 00 02 00 00 00 00 02 00 BD
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 09
00 00 02 02 00 09 00 00 00 04 00 00 00 00 04 00 ED
00 00 04 04 00 09 00 00 00 02 00 00 00 00 02 00 BE
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 0A
00 00 02 02 00 09 00 00 00 04 00 00 00 00 04 00 EE
00 00 04 04 00 09 00 00 00 02 00 00 00 00 02 00 BF
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 0B
00 00 02 02 00 09 00 00 00 04 00 00 00 00 04 00 EF
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 0C
```

ID nodo ttx

Fig.5.16: Captura de tráfico sin modificaciones.

Una vez implementadas las modificaciones en los dos nodos atacantes, sí que se aprecia como se apoderan del medio más veces respecto al nodo 3. En la **Fig.5.17** se puede observar como el nodo 3 es incapaz de trasmitir a la misma tasa que lo hacia en el caso anterior.

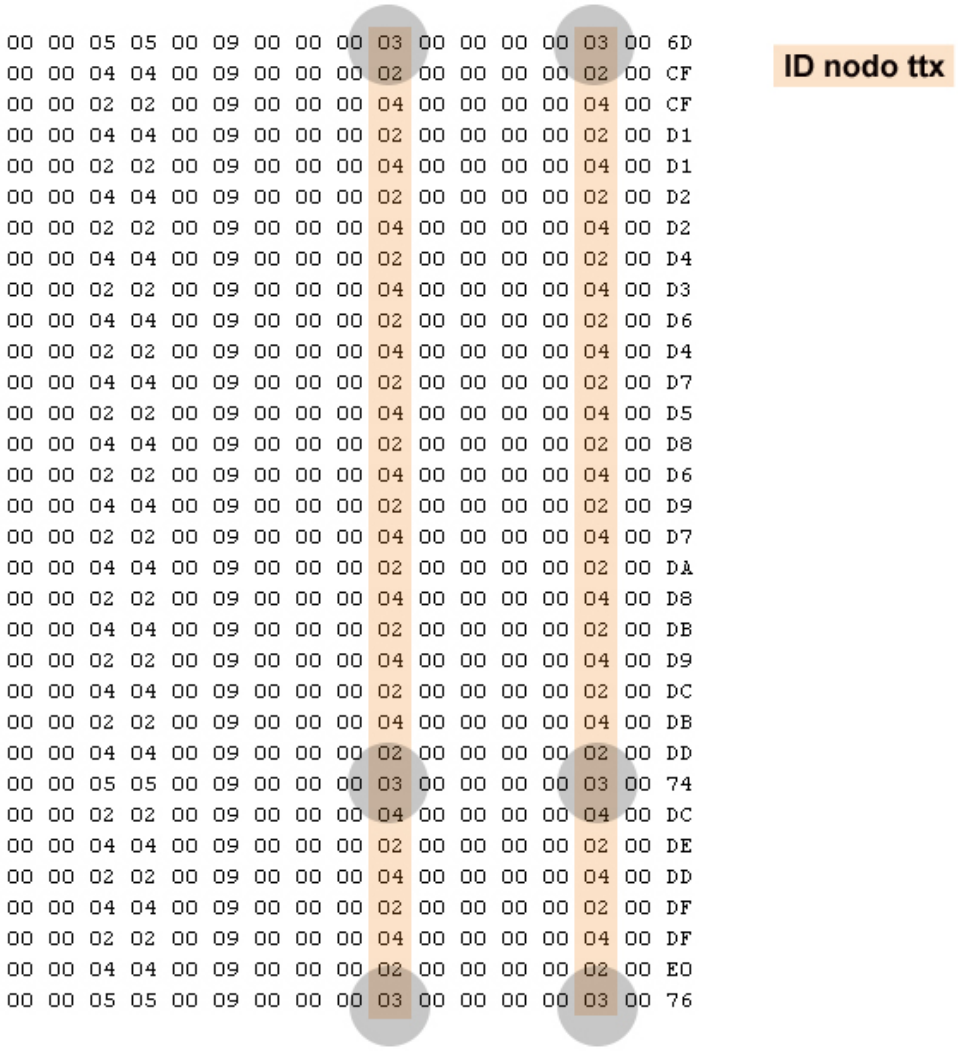

Fig.5.17: Captura de tráfico, nodos atacantes modificados.

Como se aprecia en la imagen anterior, los nodos atacantes cumplen su cometido, pero el número de transmisiones que pueden hacer mientras ocupan el medio varía. Es decir, según el ejemplo aportado la primera vez que el nodo 2 y nodo 4 ocupan el medio pueden enviar 12 y 11 paquetes respectivamente antes de que el nodo 3 mande el suyo, mientras que la segunda vez que se han apoderado del medio tanto el nodo 2 como el 4 sólo pueden mandar 3 paquetes antes de que vuelva a trasmitir el nodo 3. En la **Fig.5.18** se muestra un estudio para determinar el número de paquetes que los dos nodos atacantes son capaces de transmitir en el tiempo en el que el nodo 3 es incapaz de acceder al medio. Los cálculos se han realizado sobre 2000 accesos totales.

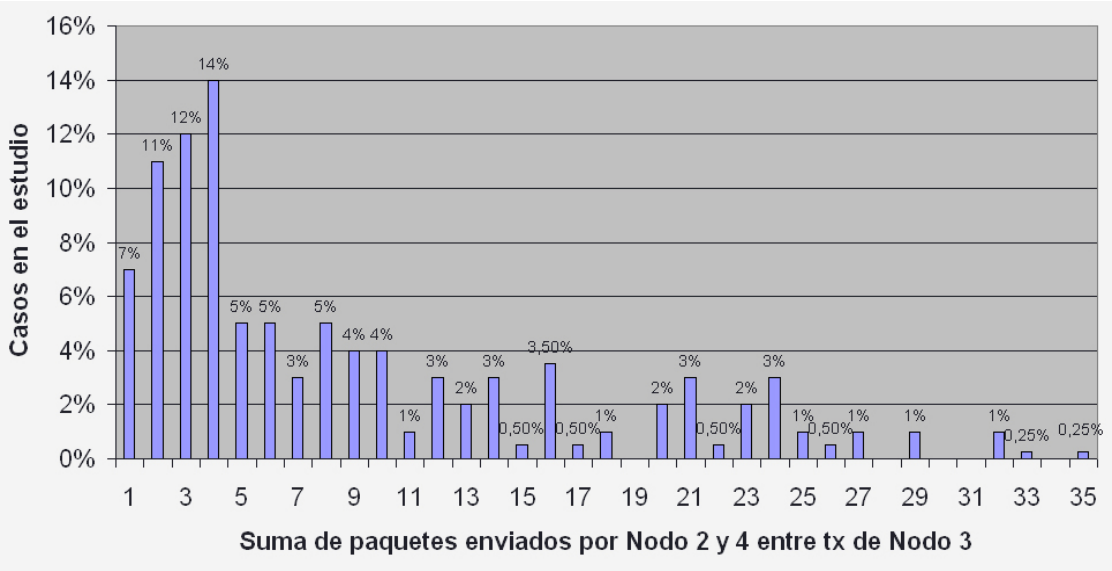

Fig.5.18: Paquetes enviados entre nodo 2 y 4 por una transmisión de nodo 3.

Durante el tiempo que ha durado la prueba, como se puede observar en la **Fig.5.19**, los nodos que no implementan el CSMA/CA realizan un 44,02% y 42,54% respectivamente de los accesos totales, mientras que el nodo 3 sólo realiza un 13,44% del total.

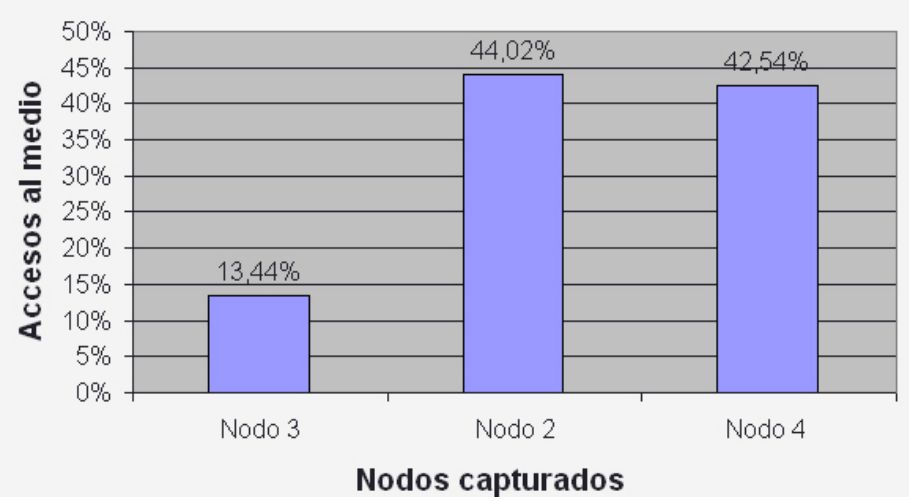

Fig.5.19: Porcentaje accesos al medio de los nodos.

Uno de los problemas más graves con el que se encuentra el nodo 3 ante esta situación, es que el algoritmo CSMA/CA no es un proceso infinito hasta que devuelve TRUE. Si al intentar mandar un paquete se supera el valor de maxCSMABackoffs (valor 4 por defecto), dicho paquete quedará descartado, por este motivo si se quiere que el paquete llegue a su destino, se tendría que implementar alguna solución que no entra dentro de los objetivos del TFC. En la **Fig.5.20** se puede apreciar la proporción de paquetes enviados y descartados del nodo 3.

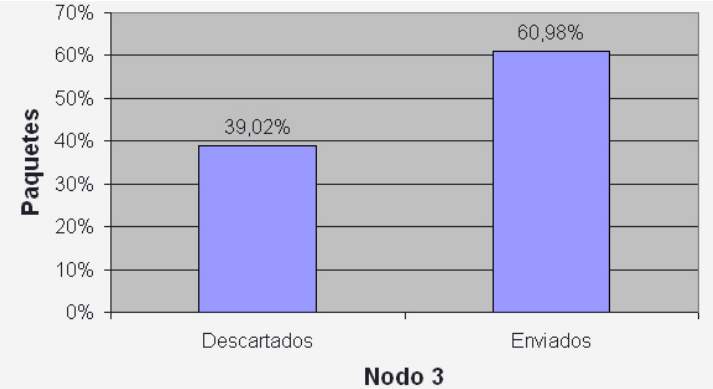

Fig.5.20: Paquetes descartados y enviados del nodo 3.

5.6. Adaptar el algoritmo de ataque

Como se ha descrito en los puntos anteriores, hay algunas limitaciones que no permiten crear el ataque tal y como estaba planificado inicialmente. Principalmente existen dos factores que son determinantes:

- No se puede mantener el medio ocupado de forma constante enviando una ráfaga de paquetes.

- Como se compite por cada paquete para acceder al medio sin acumular ningún tipo de privilegios, adoptar el identificador de otra mota realmente no tiene ningún beneficio desde este punto de vista.

Debido a todo esto, el ataque ha sido modificado para ser implementado y seguirá las pautas que se enumeran a continuación:

- Los dos nodos atacantes tienen deshabilitado el CSMA/CA de inicio, todas las transmisiones se realizan sin este algoritmo de acceso al medio.

- Se conserva el cambio de identificadores, pero en este caso con el objetivo de crear la sensación de que se está produciendo una comunicación unidireccional cuando realmente es bidireccional. Cualquier mota ajena que capture el tráfico de la red verá que existe sólo una comunicación en una dirección y de esta forma se reduce la probabilidad de que el ataque sea detectado.

5.7. Desarrollo del ataque completo

5.7.1. Ataque final

Una vez conocidas las modificaciones que se han de realizar sobre la idea inicial, se ha utilizado la red de la **Fig.5.14** como base para implementar el

ataque final. El funcionamiento de la red es el mismo que se ha explicado en el punto 5.5 con la única diferencia que ahora los dos nodos atacantes tendrán que ir intercambiándose las direcciones tras cada transmisión (ver **Fig.5.21**).

Las motas cambian su identificador en función de si reciben un ACK o un paquete. De esta forma siempre existe una sincronización perfecta para simular una comunicación en una dirección.

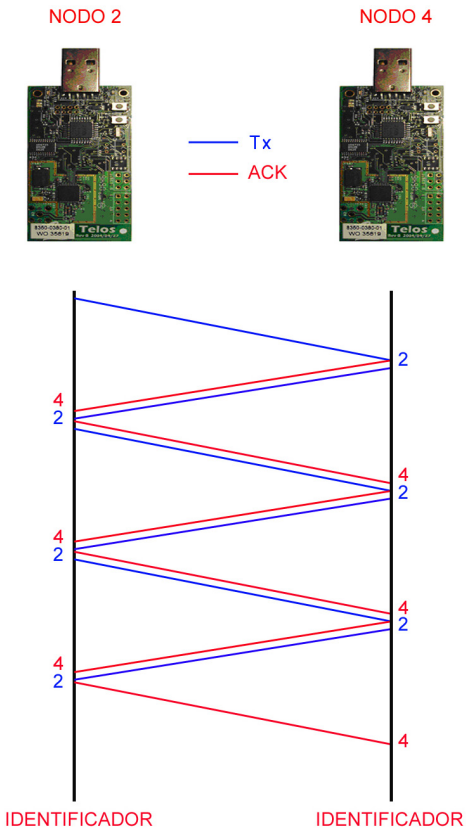

Fig.5.21: Cambio de identificadores.

A través de la **Fig.5.22** se puede apreciar como el cambio se realiza correctamente. Para saber que nodo realmente está transmitiendo, se ha utilizado un identificador propio en el campo de datos que indica la identidad real de la mota.

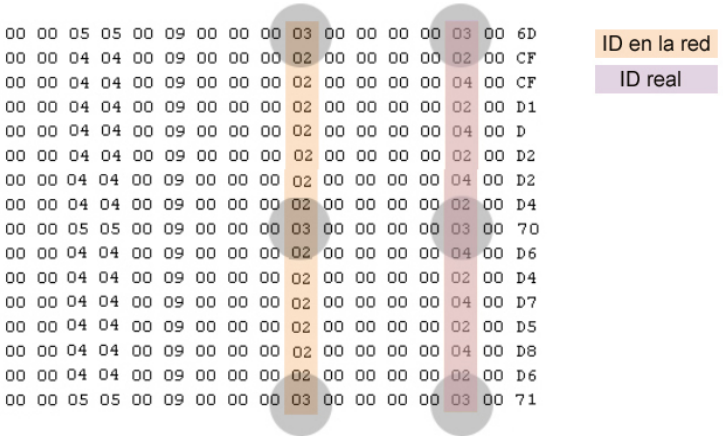

Fig.5.22: Captura de tráfico con los nodos atacantes modificados.

Uno de los problemas derivados del cambio de dirección que se produce constantemente, es que las motas invierten un tiempo determinado y de esta forma el nodo que tiene habilitado el CSMA/CA tiene más probabilidades de acceder al medio.

Se han realizado las mismas pruebas que se han llevado a cabo en el punto 5.5 para poder apreciar las diferencias entre los dos casos, realizando los cálculos sobre 2000 accesos totales. En la **Fig.5.23** se puede apreciar el número de paquetes que pueden transmitir los atacantes por cada transmisión del nodo 3.

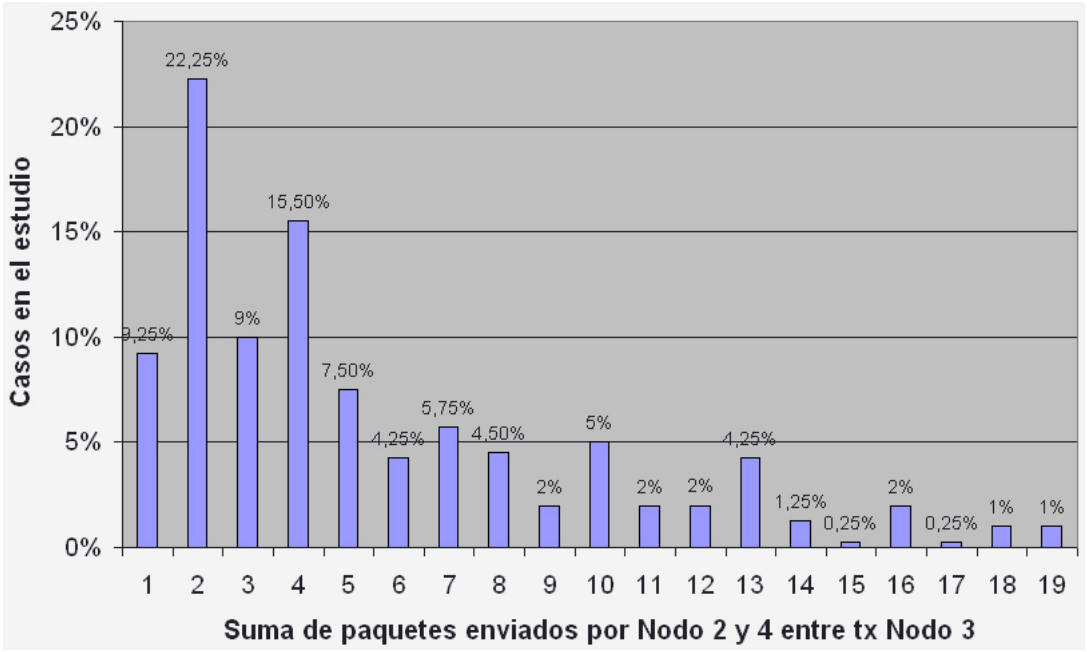

Fig.5.23: Paquetes enviados entre nodo 2 y 4 por una transmisión de nodo 3.

Durante el tiempo que ha durado la prueba, como se puede apreciar en la **Fig.5.24**, los nodos que no implementan el CSMA/CA realizan un 39,05% y 41,92% respectivamente de los accesos totales, mientras que el nodo 3 realiza el 19,03% de los accesos.

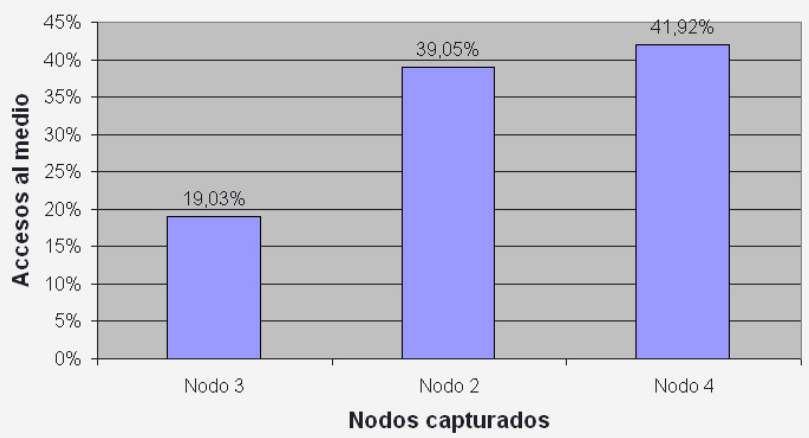

Fig.5.24: Porcentaje accesos al medio por nodo.

Por último destacar que como se aprecia en la **Fig.5.25**, la proporción de paquetes descartados es muy inferior al caso anterior.

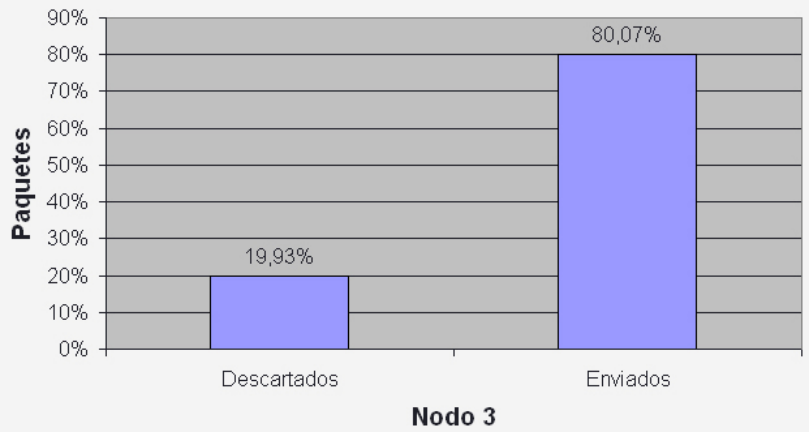

Fig.5.25: Paquetes descartados y enviados de nodo 3.

Finalmente en la **Fig.5.26** y la **Fig.5.27** se representan, a modo de resumen, el porcentaje de accesos al medio y de paquetes descartados por el nodo 3, con y sin cambio de identificador en los nodos atacantes.

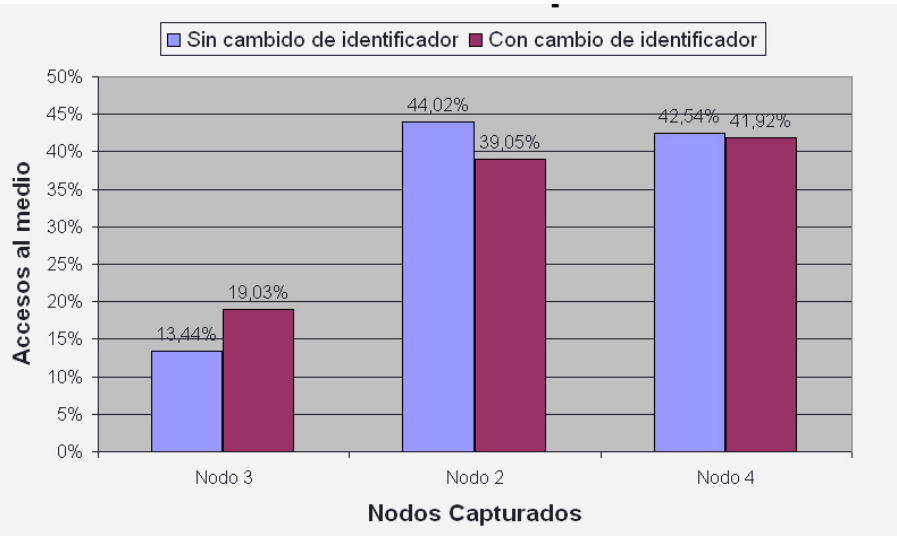

Fig.5.26: Comparación Accesos al medio.

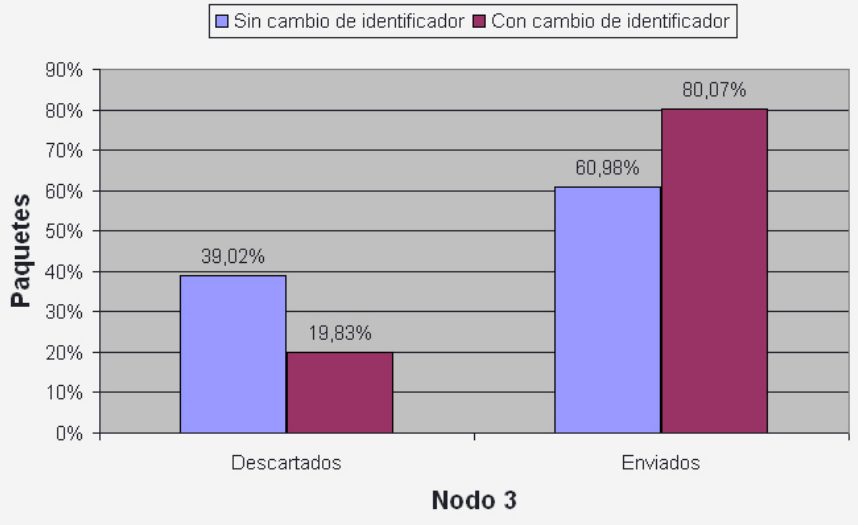

Fig.5.27: Comparación paquetes enviados y descartados del nodo 3.

Estas diferencias son consecuencia de las llamadas realizadas al método SetAddress. Dicho método requiere un tiempo de proceso importante que es aprovechado por el nodo 3 para tener más posibilidades de acceder al medio y de esta forma reducir el número de paquetes descartados.

5.7.2. Escenario con más de dos nodos

Pese a que el escenario final está compuesto de 2 nodos con CSMA/CA, por el número total de motas con las que se ha trabajado, se ha realizado un pequeño estudio para determinar la eficacia de los nodos atacantes en función del número de nodos que intentan transmitir dentro de una red. Este estudio se ha llevado a cabo sobre 2000 accesos al medio en cada escenario y con cambio

de identificador, ya que es el diseño final del ataque. Todas las motas de los diversos escenarios transmiten con una potencia de 0 dBm.

La **Fig.5.28** muestra el tanto por ciento de accesos al medio de un nodo atacante y de un nodo que cumple las reglas del CSMA/CA. Puede apreciarse que al elevarse el número de nodos dentro de la red que cumplen el CSMA/CA, se mantiene el efecto del ataque. Pese a que como es normal disminuye el porcentaje de accesos por nodo, se mantiene en todo caso una proporción relativa estable. Si se dividen los accesos al medio del nodo atacante entre los de un nodo con CSMA/CA, la proporción se mantiene estable entre 1,69 y 1,79.

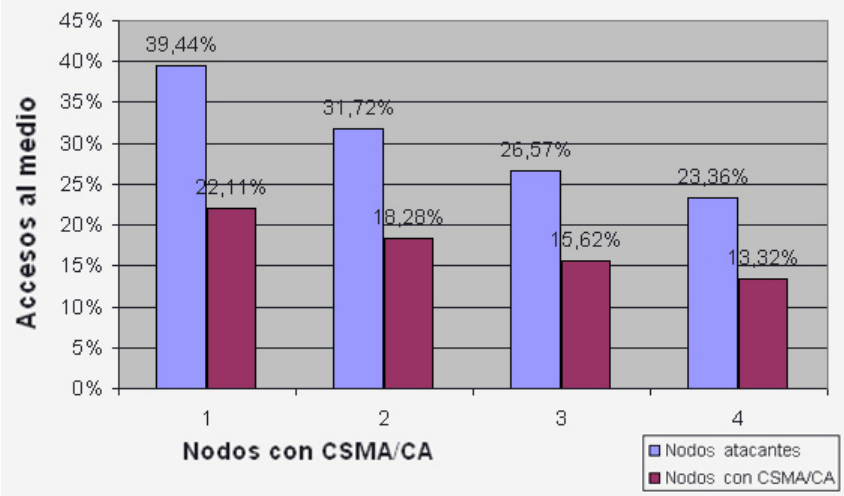

Fig.5.28: Accesos nodo atacante y nodo con CSMA/CA.

La **Fig.5.29** muestra la media de paquetes que los nodos atacantes pueden transmitir entre la transmisión de los nodos que tienen CSMA/CA activado, es decir, el número de transmisiones que los nodos atacantes pueden realizar sin ser interrumpidos por el resto de nodos que implementan CSMA/CA. Se puede observar ver que mientras más nodos coexisten en la red, menores son las ráfagas de los paquetes atacantes.

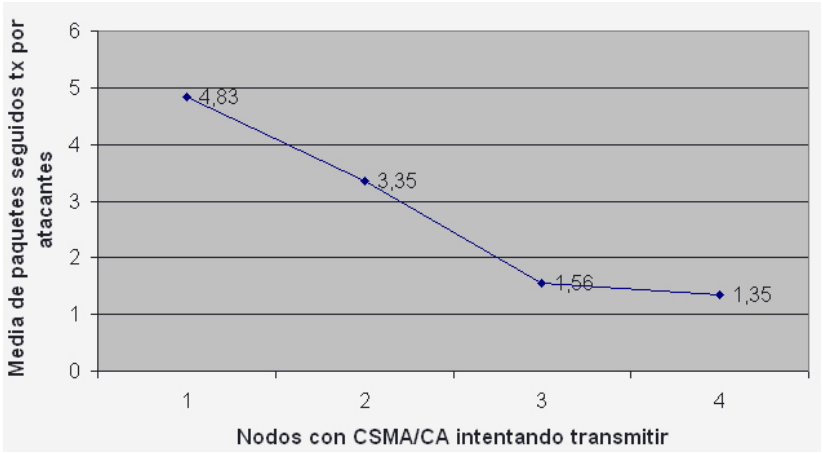

Fig.5.29: Media de paquetes transmitidos por atacantes.

5.8. Detectar el ataque

Una vez implementado el ataque, el siguiente objetivo se ha centrado en que el resto de motas de la red fuesen capaces de percatarse del desajuste provocado. Tras detectar el ataque, las motas, tienen que ser capaces de volver a la situación en la que cada nodo tiene las mismas posibilidades de acceder al medio.

Para poder detectar el ataque implementado, se han diseñado dos métodos. A continuación se ha desarrollado cada método especificando el contexto en que se puede utilizar y en las ventajas e inconvenientes que presentan.

Métodos de detección:

- Nivel local.
- Nodo dedicado dentro de la red.

5.8.1. Nivel local

Este método de detección se basa en que cada mota sea capaz de detectar cuando el acceso al medio ha dejado de ser justo. Si se aplica este método, todas las motas de la red tienen que implementar el código desarrollado ya que cada uno de los nodos puede tomar la decisión de contraatacar.

Como se ha comentado en el punto 5.5, las motas que juegan en inferioridad a la hora de acceder al medio tienen un volumen de paquetes descartados muy elevado ya que CSMA/CA permite un número limitado de intentos. Si las motas son capaces de detectar estos descartes, pueden determinar si existen problemas de acceso al medio.

Utilizando la interfaz AMSend, puede utilizarse el comando send para mandar los paquetes a la dirección que interese y mediante el evento sendDone, se puede capturar el resultado de dicho envío. Los tres posibles resultados son SUCCESS en el caso de que el envío se haya realizado con éxito, FAIL en el caso de que el envío falle y por último ECANCEL que es el caso en el que ha sido cancelado por el usuario mediante el comando cancel de la misma interfaz.

Para implementar este sistema de detección, el caso que hay que tener en cuenta es el resultado FAIL, ya que es el que asegura que el paquete no se ha enviado por descarte en el algoritmo CSMA/CA. En función del número de veces que se repita este resultado se puede determinar si alguien está ocupando o no el medio de forma ilícita. Antes de tomar la decisión de determinar si se está produciendo un ataque o no, se tienen que acumular un conjunto de muestras que garanticen que otras motas están accediendo al medio de forma privilegiada, ya que si se lanza el contraataque, al mínimo desajuste, puede que se esté produciendo un falso positivo y no un ataque real.

```
                                                    ...
event void AMSend.sendDone(message_t *msg, error_t error) {
if(error == FAIL){
        error_acceso++;
                if(error_acceso==X){
                post send_aux(); /*En este punto se inicia el contraataque*/
                }
        }
else{
        error_acceso=0;
        }
}
                                                    ...
```

Fig.5.30: Detección errores.

Como se puede apreciar en el código anterior (ver **Fig.5.30**), se considera que si se producen X intentos de transmisión fallidos seguidos ya existe un ataque y puede iniciarse el contraataque que se comentará en el punto 5.9. En cada transmisión se evalúa el comando send mediante el comando sendDone para determinar si todo se ha producido correctamente. Al descartarse el paquete y obtener como resultado de esta evaluación el valor FAIL, se incrementa el numero de errores representados por la variable error_acceso.

El valor de X no puede ser un valor aleatorio, ya que podría provocar falsos positivos. Para determinar este valor hay que observar la red antes de iniciarse el ataque y una vez iniciado, de esta forma se puede determinar las veces que no se puede acceder al medio consecutivamente cuando se produce un ataque. En las pruebas realizadas en este apartado, con 2 nodos atacantes y 2 nodos con funcionamiento normal se le ha asignado a X el valor 3, ya que sobre 200 pruebas realizadas sólo se repiten 3 o más intentos fallidos cuando hay un ataque en la red, en ningún caso si todas las motas tienen un comportamiento normal.

Ventajas de este método:

- El tiempo invertido para detectar el ataque es mínimo.
- No se depende de un nodo coordinador que determina cuando hay un ataque.
- Método recomendado si los atacantes no pertenecen a la red de la mota que implementa el sistema.

Desventajas:

- Todas las motas de la red tiene que implementarlo.
- No se puede determinar que motas están atacando ya que todo se hace en función de los intentos de transmisión

- 0% efectivo si los atacantes son de la red a la que pertenece la mota atacada ya que cuando se mande el contraataque, los nodos atacantes también reciben la orden de reajustarse.

5.8.2. Nodo dedicado dentro de la red

Este método se caracteriza por tener una sola mota dentro de la red con la función de detectar los ataques e iniciar una respuesta. Este nodo tiene que ser capaz de procesar todos los paquetes que circulan por la red, aunque la dirección destino no corresponda con la dirección de dicho nodo. Es decir, actuará como un sniffer pero también será capaz de procesar datos y tomar decisiones.

Como se ha comentado en las especificaciones del ataque, las dos motas atacantes se intercambian la dirección creando la sensación al resto de motas que sólo se está produciendo una comunicación unidireccional. Pese a que las motas atacantes pueden crear este efecto, cada paquete está asociado a una serie de valores que solamente se "rellenan" en recepción. En realidad no existen valores fijos que puedan determinar la identidad de una mota, pero si que existen valores como el RSSI o el LQI que en condiciones estables tienen un valor relativamente continuo en el tiempo.

Si se trabaja en redes en las que no existen variaciones de potencia de transmisión y la distribución geográfica de las motas es fija, puede llegar a limitarse el rango de variabilidad de estos valores, considerándolos distintivos y únicos de cada mota.

En este caso se ha trabajado con el RSSI (Receive Signal Strength Indication) (ver [9]). Como indica su nombre mide el nivel de fuerza de las señales de redes inalámbricas recibidas. Por lo tanto a mayor valor de RSSI señal recibida con más fuerza. Sobre esta herramienta se desarrollan fundamentos de la tecnología inalámbrica y se puede utilizar de modo especial para la localización.

El objetivo de este método de detección es asociar a cada dirección dentro de la red un rango viable de RSSI. En el momento que una mota se salga de su rango de valores de una forma reiterada en un intervalo de tiempo, puede considerarse que hay dos motas con diferente valor RSSI que se están intercambiando la dirección.

Para poder guardar los valores de RSSI de cada mota y el número de errores que van acumulando, se ha trabajado con variables dedicadas. En redes mayores se podrían utilizar vectores para facilitar esta tarea y reducir el volumen de código (ver **Fig.5.31**).

```
…
int8_t rssi_2_fijo=0; /*Valores RSSI fijo sobre los que comparar*/
int8_t rssi_3_fijo=0;
```

```
  int8_t rssi_4_fijo=0;
  int8_t rssi_5_fijo=0;

  int8_t rssi_2_vari=0;  /*Valores RSSI variable que comparamos*/
  int8_t rssi_3_vari=0;
  int8_t rssi_4_vari=0;
  int8_t rssi_5_vari=0;

  uint8_t contador_2=0;  /*Contador de errores permitidos*/
  uint8_t contador_3=0;
  uint8_t contador_4=0;
  uint8_t contador_5=0;

  bool acceso_2=TRUE; /*Variables para determinar si hay que modificar rssi
fijo*/
  bool acceso_3=TRUE;
  bool acceso_4=TRUE;
  bool acceso_5=TRUE;

  uint16_t nodos[4]={2,3,4,5}; /*Nodos iniciales de la red*/
                                        …
```

Fig.5.31: Variables utilizadas.

Mediante la interfaz CC2420Packet, se puede obtener el RSSI de cada paquete recibido (ver **Fig.5.32**).

```
                                        ….
rssi_2_vari=call CC2420Packet.getRssi(msg);
                                        …
```

Fig.5.32: Leer RSSI.

De esta forma, el nodo encargado de controlar el cambio de identificadores, cada vez que recibe un paquete de un nuevo nodo de la red, almacena su valor RSSI y posteriormente va comparando todos los nuevos valores RSSI con este valor inicial. En este caso es especialmente importante, ajustar los rangos de los valores que se consideran aptos y el número de veces que se permite un desajuste en el RSSI antes de iniciar un contraataque.

Esta tolerancia, varía en función de la distancia de las motas y de posibles efectos de fading. Se ha realizado un estudio de la evolución del RSSI en función de la distancia (ver **Fig.5.33**), las motas se han situado a 1,5 metros del suelo y sin ningún objeto intermedio que dificulte la comunicación. Al obtener el valor RSSI mediante el comando getRssi, tiene que pasarse a decimal y sumarle el offset del chip (-45 dBm), de esta manera se obtiene el valor real en dBm.

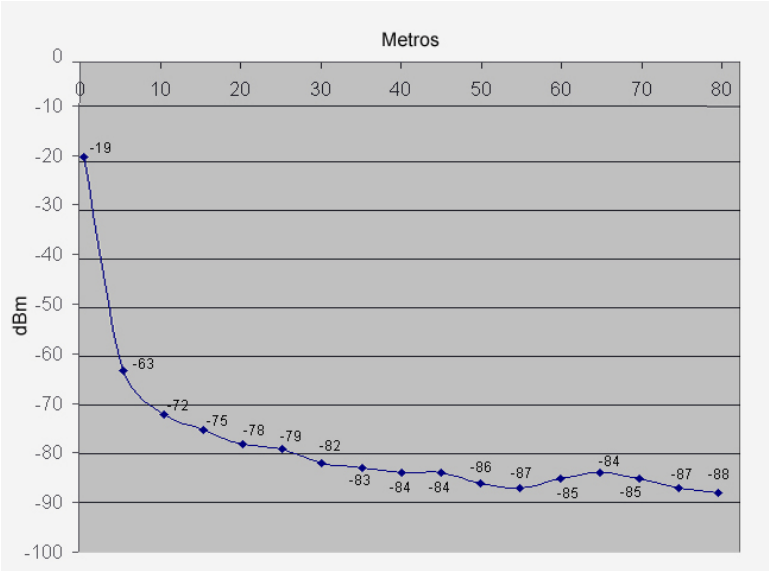

Fig.5.33: Evolución RSSI en función de la distancia

Una vez conocidos los valores de RSSI, hay que determinar el grado de variabilidad. En las pruebas realizadas se ha trabajado en distancias menores a 7 metros por lo que sólo se han determinado las mediciones con distancias inferiores a este valor. Como en el caso anterior, se han realizado las pruebas en un medio abierto y con las mismas condiciones durante todo el estudio. En la **Fig.5.34**, se ha representado el valor medio del RSSI tras 300 muestras por posición. Una vez determinado el valor medio, se han indicado los valores del muestreo que más lejos se encuentran de la media, tanto en su límite superior como en el inferior.

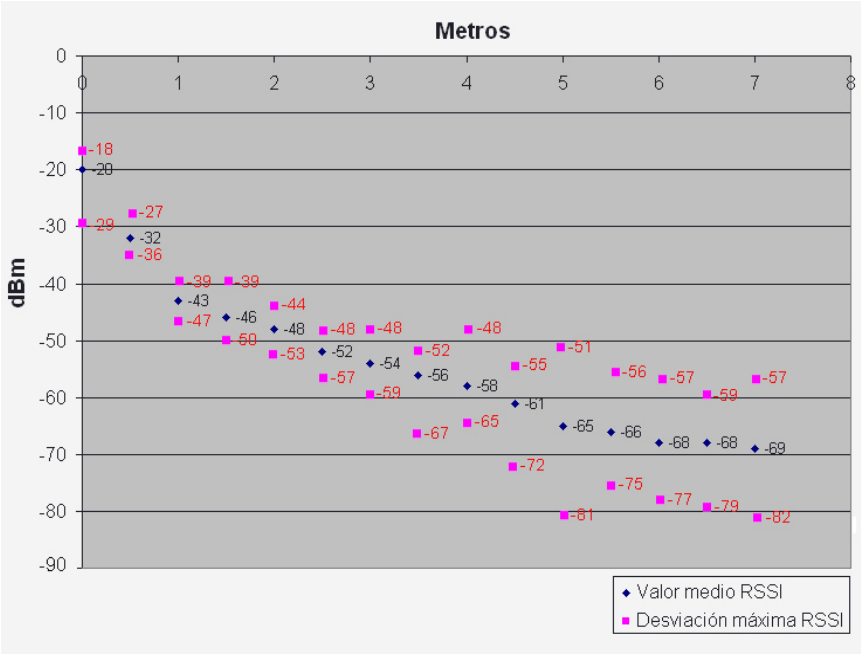

Fig.5.34: Desviaciones sobre valor de RSSI medio

Como puede observarse en la figura anterior, la desviación respecto al valor medio se hace mayor proporcionalmente con la distancia. Por este motivo cada escenario tiene que ajustar sus valores haciendo un pequeño estudio inicial de la distancia de cada mota.

En el ejemplo de la **Fig.5.35** se ha utilizado una tolerancia de ±7 para considerar que el valor entra dentro de lo permitido y no se captura el valor fijo de RSSI hasta que no han llegado 200 paquetes.

```
                                        …
 contador_acceso2++;
 if (contador_acceso2>200)
{
    if(acceso_2==TRUE){
       rssi_2_fijo=call CC2420Packet.getRssi(msg);
       acceso_2=FALSE;
      }
    else
     {
       rssi_2_vari=call CC2420Packet.getRssi(msg);
       if(rssi_2_vari<rssi_2_fijo + 7  && rssi_2_vari>rssi_2_fijo - 7){}
       else{
              contador_2 ++;
              if(contador_2>250){
               post send_aux(); /*Se inicia contraaataque*/
              }
             }
                                        …
```

Fig.5.35: Comparación valores RSSI utilizando una tolerancia.

En este caso, se ha considerado que hay un error cuando se producen 250 desviaciones del valor RSSI estimado. Tanto el rango de tolerancia como el número de errores es algo que se tiene que adecuar a las características de la red y a la tasa de transmisión de paquetes.

Ventajas:

- Permite un método de control de errores centralizado.
- Las motas de la red no se tienen que preocupar de esta tarea.
- Al realizar el contraataque se puede discriminar a las dos motas atacantes ya que se conocen sus identificadores.

Desventajas:

- Sólo se puede aplicar a redes en las que las motas tienen un valor constante de potencia y su posición no varía.

- Depender de una variable que no es constante, puede provocar que si no se cumplen los requisitos del punto anterior se inicie un contraataque sin presencia de ataque. En el apartado 5.10 se ha estudiado este punto sobre un escenario en concreto.

5.9. Contraataque

Una vez detectado el ataque por uno de los métodos explicados anteriormente, se ha buscado el método para reajustar las condiciones de igualdad dentro de la red. El principal problema es que las motas atacantes, igual que el resto de nodos, autogestionan sus mecanismos de acceso. Por esta razón, no se puede contemplar la idea de intentar modificar el comportamiento de los nodos atacantes y hay que centrase en reajustar el resto de motas de la red.

En un principio se trabajó en dotar a las motas atacadas de las mismas características que las atacantes. Es decir, eliminar su CSMA/CA para que de esta forma todas las motas volviesen a jugar en las mismas condiciones de igualdad. El problema de esta solución es que si no se implementa un control de acceso al medio, la probabilidad de colisión aumenta considerablemente, obligando a un número de retransmisiones muy elevado que aumentaría el tráfico dentro de la red.

Tras descartar esta solución, se buscó un método que fuese capaz de mantener las propiedades de las motas con CSMA/CA y permitiese descartar a aquellas que no lo utilizasen. Si se quiere tener un canal que no esté constantemente ocupado por nodos atacantes, lo mejor que puede hacerse es cambiar a uno que se encuentre libre.

Las comunicaciones con el chip CC2420 que implementan las motas TelosB se pueden realizar entre los canales 11 a 26 (26 por defecto), por lo que se dispone de un amplio abanico de canales para seleccionar. El contraataque se ha centrado en que todas las motas sean capaces de cambiar de canal cuando detecten un ataque o reciban un paquete de configuración de cambio de canal.

El método de contraataque es el mismo en los dos sistemas de detección explicados en el punto 5.7. Tras considerar que existe un ataque en la red, la mota que ha sido capaz de detectarlo, se encargará de enviar un mensaje de cambio de canal al resto de nodos (ver **Fig.5.36**).

```
…
Typedef nx_struct Topo {
Nx_uint8_t canal;
}Topo;
….
```

Fig.5.36: Mensaje de cambio de canal.

Cuando las motas de la red reciben este tipo de paquetes, almacenan el nuevo canal y mediante la interfaz CC2420Config, utilizando el comando setChannel, se realiza el cambio del canal de comunicaciones. Un punto a tener en cuenta, es que los efectos de cambiar el canal de comunicaciones no se producen hasta que no se cierran las comunicaciones de transmisión y recepción activas. Por este motivo como se aprecia en la **Fig.5.37** hay que deshabilitar y volver a habilitar las comunicaciones por Radio.

```
                                                …
  event message_t *Receive.receive(message_t *msg, void *payload, uint8_t
len) {
    call Leds.led2Toggle();
    if(len==sizeof(Topo))   /*Paquete cambiado de canal*/
     {
         Topo* btrpkt = (Topo*)payload;
         canal_p=btrpkt->canal;
         call SplitControl.stop();   /*Se para envío y recepción*/
     }
                                                …
                                                …
  event void SplitControl.stopDone(error_t error) {
    call CC2420Config.setChannel(canal_p);    /*Cambio de canal*/
    call SplitControl.start();    /*Se reinicia envío y recepción*/
  }
                                                …
```

Fig.5.37: Proceso de cambio de canal.

En este tipo de comunicación siempre hay que controlar los ACKs ya que si no se tienen en cuenta se corre el riesgo que alguna de las motas quede aislada. Por este motivo la mota que ha detectado el ataque es la última en cambiarse al nuevo canal y no realiza el cambio si el resto no le han confirmado que han recibido el paquete mediante la verificación de ACKs o si se supera un máximo de intentos (5 por nodo).

Una vez se han realizado los cambios comentados, las motas atacantes seguirán en el canal inicial mientras que el resto podrán disfrutar de un acceso al medio justo, en su nuevo canal de comunicaciones.

5.10. Pruebas finales.

Para finalizar este trabajo, se han realizado una serie de medidas sobre el escenario de la **Fig.5.38**. En dicho escenario existen dos nodos atacantes (2 y 4) y otros dos nodos que implementan el CSMA/CA e intentan transmitir constantemente (3 y 5). Para detectar el ataque y contraatacar se ha utilizado el método de nodo dedicado con identificador 14. Todas las motas del escenario trabajan con una potencia de transmisión de 0 dBm y una tasa de transmisión constante.

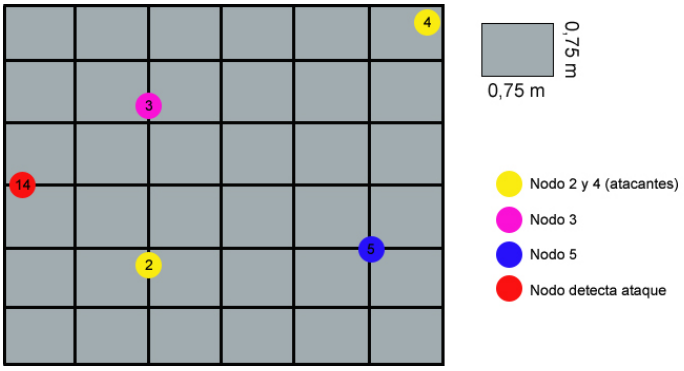

Fig.5.38: Situación de los nodos en nuestro escenario.

La primera prueba realizada, consiste en contabilizar el número de accesos al medio de cada nodo antes de activar el nodo 14. Tras 75.000 accesos totales, los resultados son los que se aprecian en la **Fig.5.40**.

```
Tiempo recepción en nanosegundos (10-9 segundos): 321174846000.
Identificador inicial de la mota: 5, Identificador en la red: 5, Destino: 3.
Número de paquetes total: 75000.0 ; Separado: Nodo 3: 12185.0 ; Nodo 2: 24084.0 ; Nodo 4: 26327.0 ; Nodo 5: 12404.0.
Porcetaje: Nodo 3: 16.246666% ; Nodo 2: 32.112% ; Nodo 4: 35.102665% Nodo 5: 16.538666%.
Paquete o ACK perdido: Nodo 2: 3456, Nodo 4: 5157.
```

Fig.5.39: Captura información mostrada por nuestra aplicación.

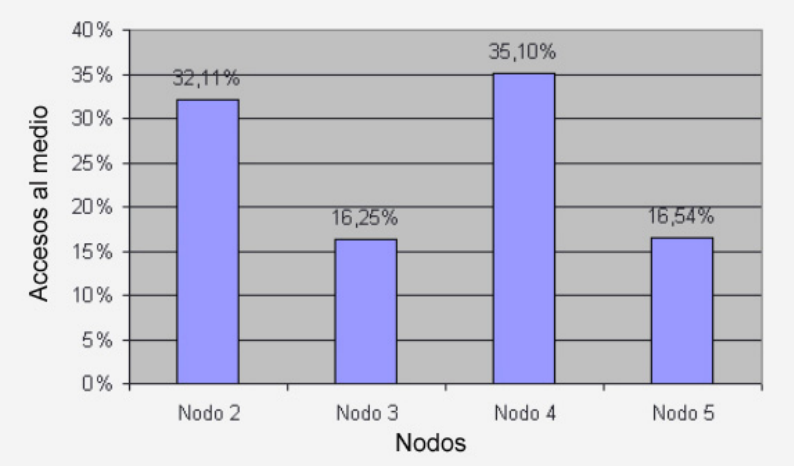

Fig.5.40: Porcentaje accesos de cada nodo antes de activar el nodo 14.

Se puede apreciar que los dos nodos atacantes son capaces de acceder al medio un número de veces mayor al resto, pese a que todos los nodos tienen la misma tasa de transmisión. Es más, como ya se ha comentado los nodos atacantes invierten tiempo de proceso en el cambio de identificadores.

En cuanto a paquetes descartados, los nodos que implementan CSMA/CA sufren descartes considerables. Mientras que los que no lo implementan no descartan ningún paquete ya que no realizan el algoritmo (ver **Fig.5.41**).

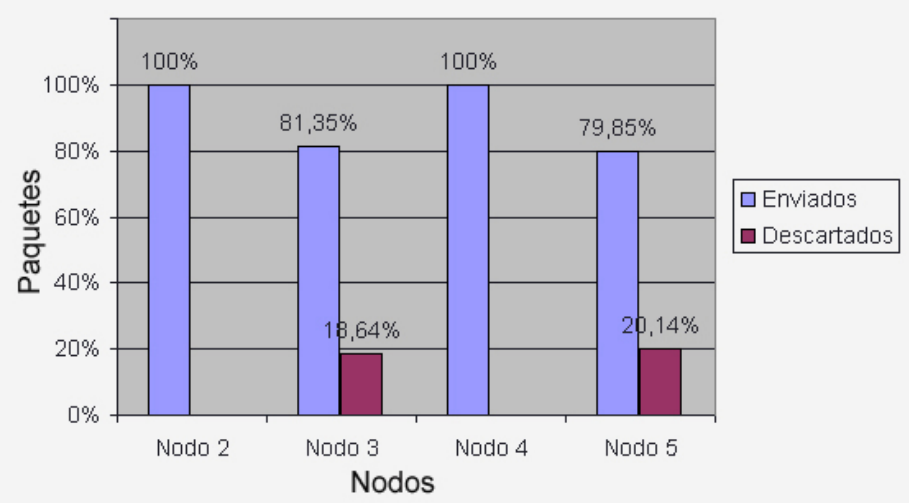

Fig.5.41: Paquetes enviados y descartados.

En el punto 5.5, se ha comentado que es obligatorio establecer un sistema para que los dos nodos atacantes se aseguren de que su paquete ha sido recibido por el otro nodo compinchado. En la **Fig.5.42** se pueden apreciar el número de retransmisiones que tienen que hacer el nodo 2 y 4. Al no utilizar el CSMA/CA ni evaluar el canal, se sufre una alta penalización en este aspecto.

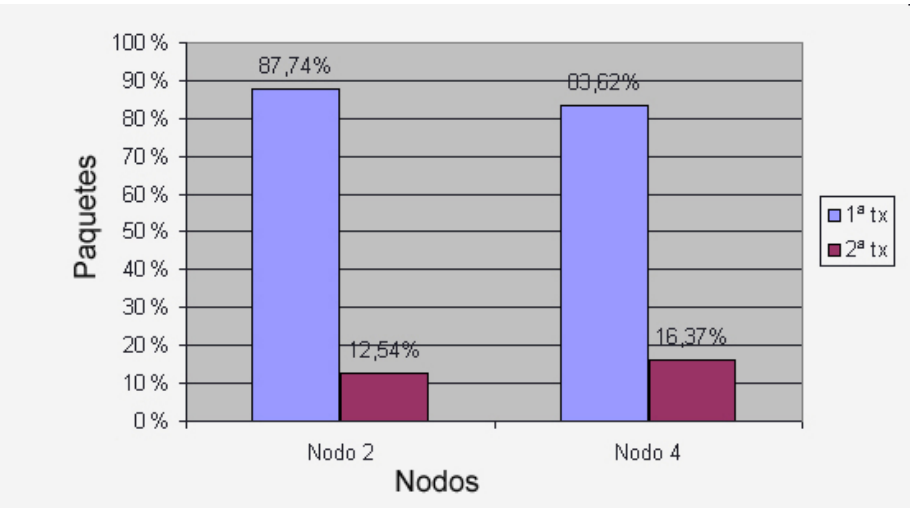

Fig.5.42: Paquetes transmitidos y retransmitidos.

Como se ha comentado en puntos anteriores, el primer paso para configurar bien el nodo detector es realizar un estudio de los valores RSSI de cada nodo respecto a dicha mota (ver **Fig.5.43**).

	RSSI (dBm)	Tolerancia (dBm)
Nodo 2	-45	+/- 6
Nodo 3	-44	+/- 7
Nodo 4	-62	+/- 10
Nodo 5	-53	+/- 13

Fig.5.43: Tabla valores RSSI de los nodos de nuestro escenario.

Una vez conocidos los valores en los que se puede mover cada nodo, el detector de ataques tiene que ajustar las tolerancias para cada nodo. Tras realizar este paso, se han realizado 500 pruebas para ver si el sistema cae en falsa alarma. Se produce una falsa alarma cuando el sistema considera que el ataque lo realiza una mota que no tiene esta intención.

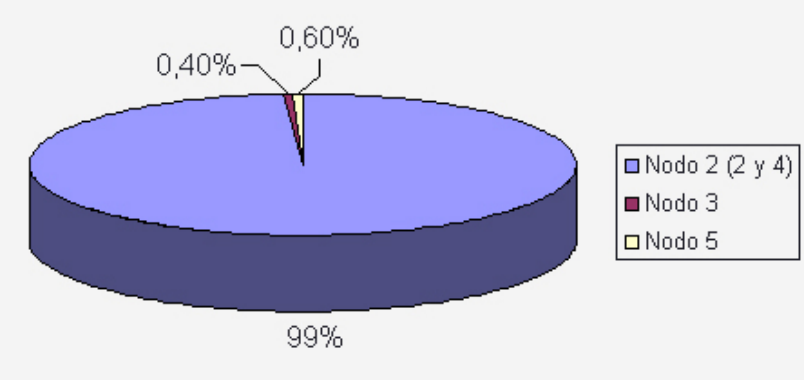

Fig.5.44: Nodo considerado atacante.

Como se aprecia en la **Fig.5.44**, la fiabilidad del sistema es cercana al 100%. En el gráfico no aparece el nodo 4 ya que éste nunca utiliza su identificador para realizar un envío, usa el identificador del nodo 2.

En la **Fig.5.45** se ha representado el tiempo que tarda el nodo 14 en detectar el ataque una vez se ha sido introducido en la red en cada una de las 500 pruebas realizadas anteriormente. El tiempo medio tras analizar las 500 pruebas es de 4,77 segundos.

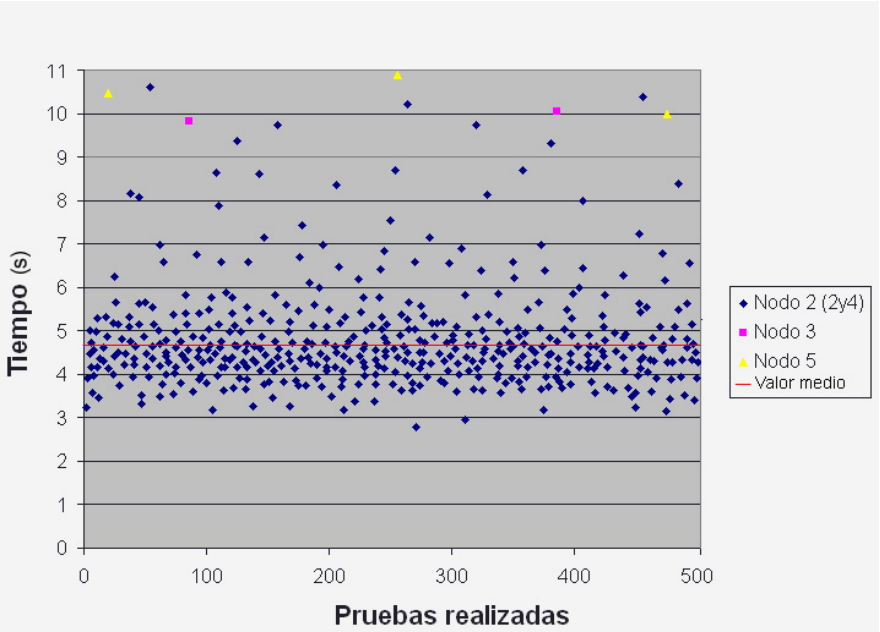

Fig.5.45: Tiempo necesario para detectar el ataque.

Una vez realizado el contraataque por parte del nodo detector. Los nodos 3 y 5 vuelven a disfrutar de un acceso al medio justo en su nuevo canal y tras 30.000 accesos, cada nodo ha realizado cerca del 50% de ellos como se puede observar en la **Fig.5.47**.

```
Tiempo recepción en nanosegundos (10-9 segundos): 289598144000.
Identificador inicial de la mota: 5, Identificador en la red: 5, Destino: 3.
Número de paquetes total: 30000.0 ; Separado: Nodo 3: 15030.0 ; Nodo 2: 0.0 ; Nodo 4: 0.0 ; Nodo 5: 14970.0.
Porcetaje: Nodo 3: 50.1% ; Nodo 2: 0.0% ; Nodo 4: 0.0% Nodo 5: 49.9%.
```

Fig.5.46: Captura de los valores obtenidos de 30.000 envíos de paquetes.

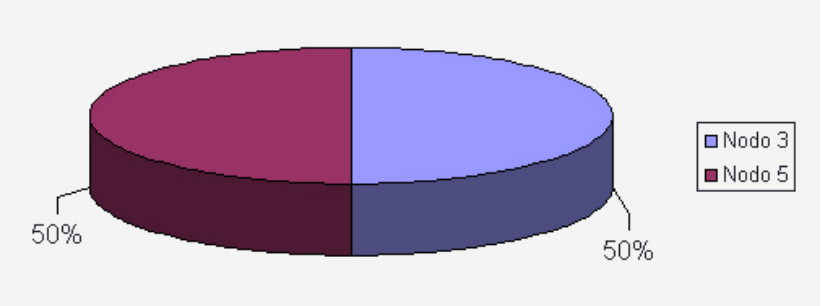

Fig.5.47: Accesos al medio en el nuevo canal.

Por otra parte los dos nodos atacantes quedan aislados en el canal inicial y siguen realizando el ataque pero esta vez sin afectar al resto de nodos de la red.

CAPÍTULO 6. CONCLUSIONES

6.1. Conclusiones

Una vez finalizado el trabajo puede corroborarse que las comunicaciones inalámbricas presentan ciertas vulnerabilidades que pueden desajustar el funcionamiento de una red. Por tanto, la seguridad es en efecto un tema importante a estudiar en este tipo de sistemas. En el caso de sistemas de código abierto, la implementación de posibles ataques es más accesible puesto que resulta viable acceder a campos y funciones con las que se puede implementar ataques o acciones ilícitas para obtener un beneficio frente al resto de nodos.

A lo largo del TFC se ha descrito la implementación realizada de un ataque para poder apoderarse del medio y poder enviar un número de paquetes mayor que el resto de motas. Una vez implementado el ataque el siguiente paso ha sido trabajar en mecanismos capaces de detectar el ataque y posteriormente realizar un contraataque.

Las soluciones implementadas para la detección presentan sus puntos negativos y positivos, pero presentan una fiabilidad muy ajustada siempre que se respete el contexto sobre el que se pueden utilizar. Finalmente, cabe destacar que el contraataque propuesto e implementado asegura completamente que los nodos podrán volver a trabajar en condiciones normales y disfrutando de un acceso al medio justo, sin sufrir los efectos de los nodos atacantes.

6.2. Estudio de ambientalización

Una de las características principales de la tecnología ZigBee, es el bajo consumo de energía. Al trabajar en este tipo de redes se optimizan las baterías y se busca un uso prolongado de las mismas.

Al realizar el ataque que se ha planteado en el trabajo, las motas que se ven afectadas dentro de la red ven como este consumo de batería aumenta, ya que la evaluación constante del medio para intentar acceder requiere un consumo importante. Tras realizar el contraataque las motas vuelven a realizar las comunicaciones como marca el estándar y el consumo se ajusta a los valores previsibles.

6.3. Líneas futuras

Más allá de las conclusiones obtenidas del trabajo, es posible realizar algunas mejoras en el futuro:

- El sistema de detección mediante un nodo único. En el caso expuesto se ha trabajado con variables ya que el número de motas con las que se ha podido trabajar ha sido bastante ajustado, pero en redes mayores se tendría que implementar un sistema más eficiente basado en vectores que reduciría el código considerablemente.

- Buscar alguna alternativa de contraataque que no sea el cambio de canal.

- Ajustar el código para redes que trabajan con TinyOS 1.x

- Realizar más ataques a otros niveles, por ejemplo nivel de aplicación.

6.4. Valoración personal

A nivel personal, consideramos que este TFC ha servido para conocer una nueva tecnología y ampliar nuestros conocimientos sobre nuevos lenguajes de programación. Trabajar con código abierto y plataformas experimentales siempre supone un reto ya que la información disponible es mínima y poco contrastada.

En un principio fueron numerosas las dificultades ya que la primera idea planteada para el proyecto resultó inviable por las posibilidades de las motas y se invirtió mucho tiempo en demostrarlo. Pese a estos problemas, finalmente hemos podido realizar un trabajo que cumple ampliamente los objetivos planteados

Para concluir, consideramos que el potencial que presenta la capa de aplicación para trabajar con ataques y vulnerabilidades es mucho mayor que el que presenta la capa MAC y en futuros estudios recomendamos trabajar en la seguridad a dicho nivel.

BIBLIOGRAFÍA

[1] Información general: www.wikipedia.org

[2] NesC: http://nescc.sourceforge.net/

[3] TinyOS: http://www.tinyos.net

[4] XubunTOS 7.04: http://toilers.mines.edu/Public/XubunTOS

[5] Estándar 802.15.4 - 2003: http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf

[6] Datasheet CC2420: http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf

[7] Datasheet TelosB: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf

[8] ZigBee alliance: http://www.zigbee.org/

[9] Estudio sobre RSSI: http://www.eecs.harvard.edu/emnets/papers/levisEmnets06.pdf

[10] Ayuda para programar con TinyOS: http://mail.millennium.berkeley.edu/pipermail/tinyos-help/

[11] Seguridad en redes inalámbricas: http://dspace.icesi.edu.co/dspace/bitstream/item/400/1/jamdrid-seguridad_redes_inalambricas.pdf

[12] Gao, Bo, Chen He, and Lingge Jiang. 2008. Modeling and analysis of IEEE 802.15.4 CSMA/CA with sleep mode enabled. Paper presented at 2008 11th IEEE Singapore International Conference on Communication Systems (ICCS).

[13] Egan, D. 2005. The emergence of ZigBee in building automation and industrial control. Computing & Control Engineering Journal 16, (2): 14-9.

[14] Tian-Wen, Song. 2008. A connectivity improving mechanism for ZigBee wireless sensor networks.

[15] Seguridad: http://dns.bdat.net/seguridad_en_redes_inalambricas/x187.html

ANEXOS

TÍTULO DEL TFC: Seguridad para protocolos MAC cooperativos en redes de sensores inalámbricas

TITULACIÓN: Ingeniería Técnica de Telecomunicaciones, especialidad en Telemática

AUTORES: Daniel Casabona Gómez
Daniel Leiva Martín

DIRECTOR: Christos Verikoukis

CO-DIRECTORES: Luís Gonzaga Alonso Zárate
David Sánchez

FECHA: 9 de julio de 2009

Anexo 1. Tipos de tramas

Trama Beacon

Sólo en redes con beacon habilitados. Enviados por el coordinador para sincronización. Permite buscar redes y direccionar dispositivos para que se envíen información.

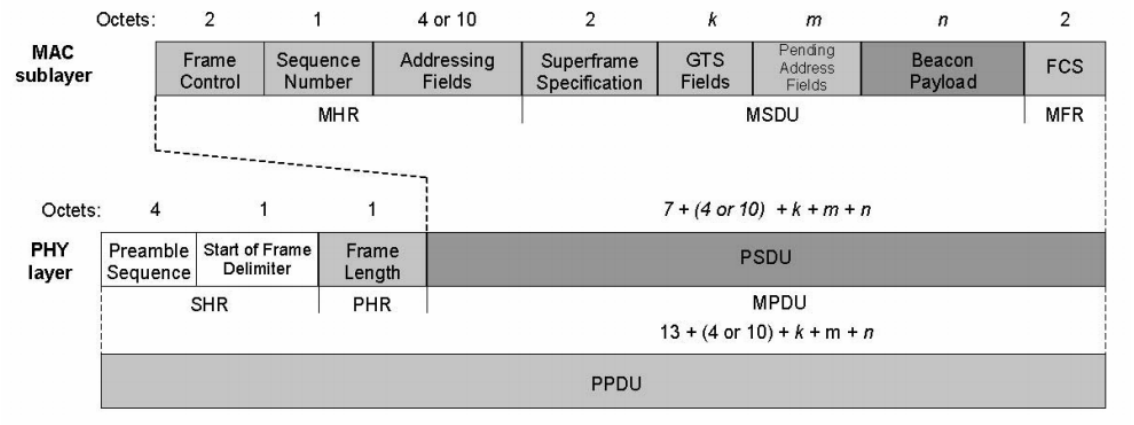

Fig.An.1: Formato trama beacon.

Trama Datos

Paquete con datos útiles que se intercambian los nodos.

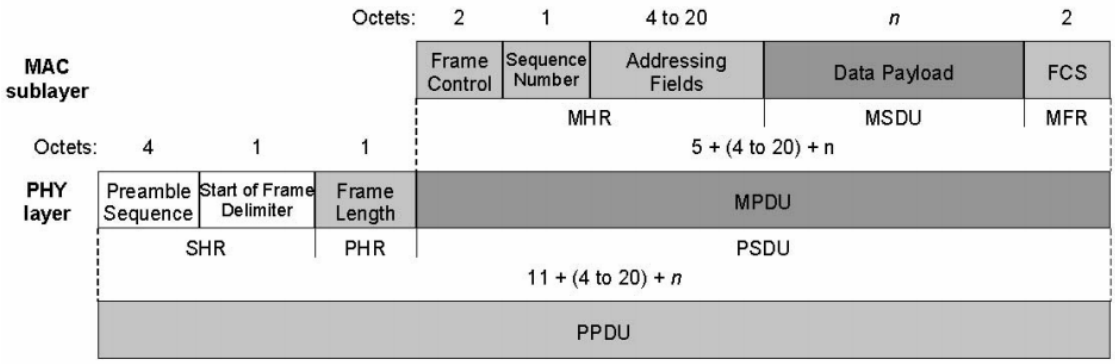

Fig.An.2: Formato trama datos.

Trama Ack

Se utiliza para confirmar la recepción de un paquete.

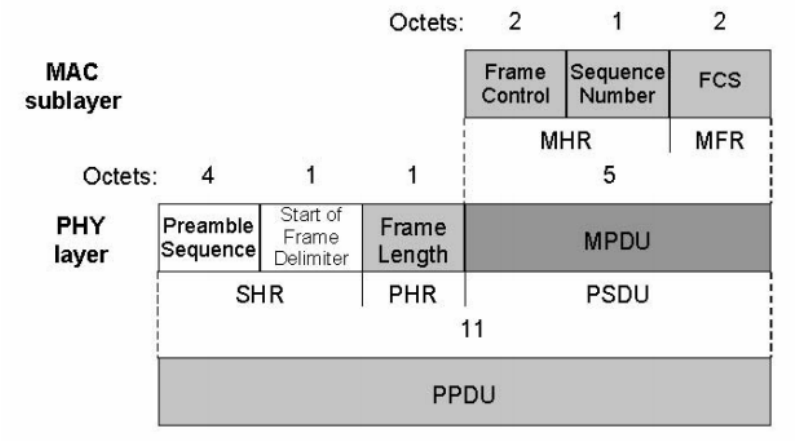

Fig.An.3: Formato trama Ack.

Trama comandos MAC

Permite el control y configuración de los nodos y la formación de la red.

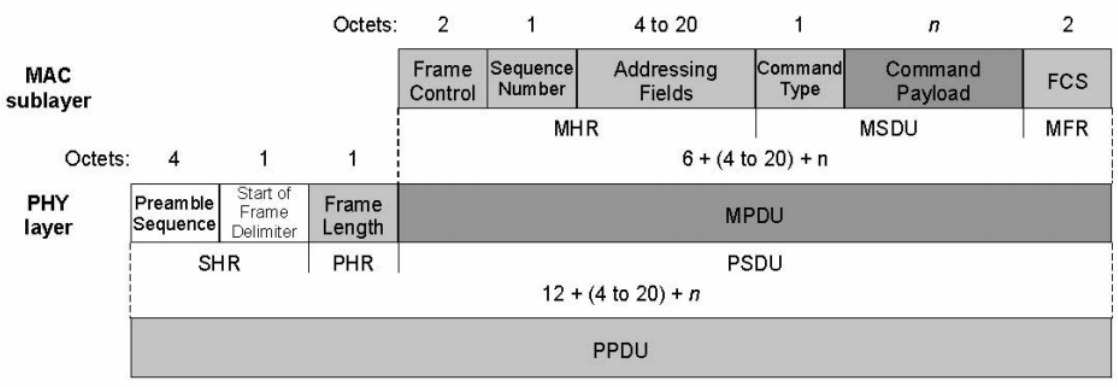

Fig.An.4: Formato trama MAC.

Anexo 2. Redes inalámbricas

Las redes inalámbricas se pueden diferenciar por el rango de frecuencias utilizado para transmitir:

- **Ondas de radio:** en este rango se encuentran las bandas que van desde la ELF hasta la UHF (3Hz – 3GHz). Se propagan por el medio omnidireccionalmente mediante antenas y al operar en frecuencias bajas no sufren atenuación por la lluvia.

- **Microondas terrestres:** son las señales que van desde 1Ghz a 300GHz. Utilizan antenas parabólicas en enlaces punto a punto donde las distancias no suelen ser muy largas. Al trabajar con frecuencias más elevadas que las ondas de radio sufren interferencias por la lluvia. Tienen el inconveniente de que los puntos que se van a comunicar tienen que estar perfectamente alineados.

- **Microondas por satélite:** Son muy parecidas a las microondas terrestres pero en este caso desde un punto de la tierra se envía una señal a una satélite que la recibe por una banda, la amplifica y la envía por otra banda a un receptor de la tierra.

- **Infrarrojos:** su rango de frecuencias va de 300GHz a 384THz. El transmisor y el receptor deben de estar alineados directamente ya que no es capaz de atravesar paredes.

Anexo 3. Ataque por fuerza bruta y solución

Introducción

El objetivo de esta aplicación es realizar un ataque en una comunicación sencilla para estudiar las interfaces y los métodos necesarios, familiarizarse con valores como el RSSI y posteriormente poder realizar un ataque más elaborado, inteligente y destinado a redes mayores y centralizadas.

Planteamiento

La red en la que se ha trabajado es la más simple posible, compuesta por dos nodos que se intercambian información cada cierto tiempo sin ningún tipo de nodo central ni estructura mallada.

El objetivo del ataque será el de poder establecer una comunicación con alguno de los dos nodos y proporcionarle información no autorizada o en su defecto congestionarlo emitiendo la máxima tasa de paquetes posibles por unidad de tiempo.

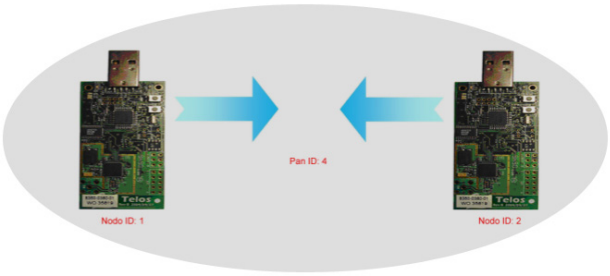

Fig.An.5: Dos motas emitiendo a la vez.

El ataque se realizará por fuerza bruta, sin ningún tipo de lógica mayor que la de ir evaluando cada situación para saber si el ataque tiene efecto o no.

Desarrollo

Un punto a tener en cuenta, es conocer el número de redes en las que se puede trabajar y dentro de cada red saber cuantas direcciones pueden utilizarse. Este punto es importante ya que habrá que evaluar cada red con sus posibles direcciones hasta encontrar la idónea. El número total de iteraciones que se necesitan para recorrer todas las direcciones posibles son 64.516.

Una vez conocidas el número de comprobaciones que se tienen que hacer, hay que encontrar una forma transparente de comprobar si el nodo está activo.

Al tratarse de un ataque, tiene que tenerse en cuenta que no se puede trabajar a nivel de aplicación en esta fase, ya que ninguno de los dos nodos de la red va a transmitir su estado a nodos no autorizados.

Por esta razón se ha trabajado a nivel MAC, para determinar si el nodo está activo. El nodo atacante manda un paquete de prueba a un nodo determinado, en una red determinada, si este nodo contesta con un ACK quiere decir que ese nodo está activo. Si no lo hace simplemente se avanzará en la iteración para evaluar otro nodo.

```
…
call PacketAcknowledgements.requestAck(&pkt);
call AMSend.send(destino, &pkt, sizeof(AckPack));
…
```

Fig.An.6: Código.

El método requestAck de la interfaz PacketAcknowledgements, se utiliza para avisar al protocolo que cuando se envíe el mensaje se utilicen reconocimientos. Este paso es importante ya que las comunicaciones también pueden establecerse sin ningún tipo de reconocimiento.

```
…
event void AMSend.sendDone(message_t *msg, error_t error) {
if(call PacketAcknowledgements.wasAcked(msg)) {
…
```

Fig.An.7: Código.

En este caso, wasAcked es el método que advierte si el paquete ha sido reconocido o no. Si devuelve FALSE, se pasará al siguiente valor dentro de la iteración. Si devuelve TRUE, se guardará el nodo destino y la red para empezar el ataque.

Uno de los grandes inconvenientes de este tipo de ataques, es el tiempo invertido. A continuación se adjunta un diagrama de puntos en el que se especifica el tiempo que tarda el nodo atacante hasta localizar el nodo activo.

En ningún caso el tiempo invertido será superior a 18 minutos y 29 segundos, que es el tiempo que se tarda en recorrer las 64.516 combinaciones posibles a la máxima velocidad que proporcionan las motas TelosB.

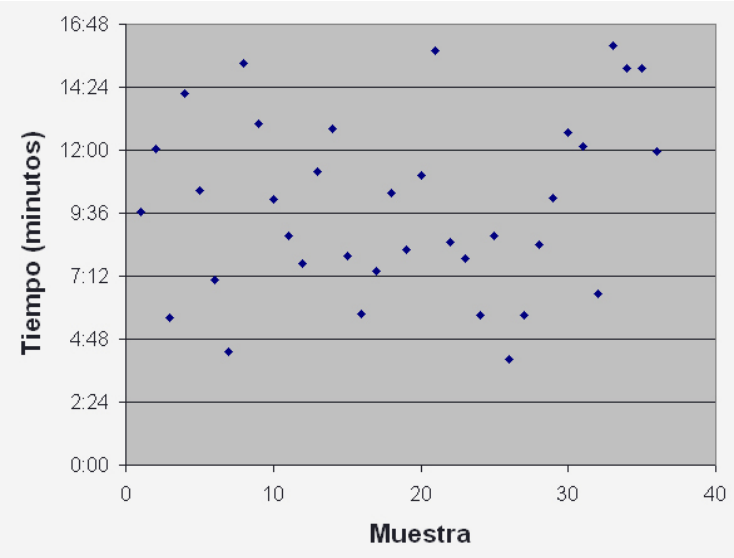

Fig.An.8: Gráfica obtención de valores.

El tiempo medido obtenido del estudio es de 9 minutos con 52 segundos. El problema para realizar un estudio con más muestras es la cantidad de tiempo tan elevado que se tarda por cada una de ellos, pero con 35 intentos ya se empieza a percibir una media bastante sólida.

Detección del ataque

Una vez realizado el ataque, uno de los dos nodos recibirá información constante del nodo atacante con identificador 3. Con dos posibles consecuencias:

- Si no hay un filtro a nivel de aplicación de los nodos autorizados, la información no autorizada puede causar desajustes tanto en la configuración como en el funcionamiento de la mota.

- Si existe un filtro a nivel de aplicación, en el que sólo se acepta información de las motas autorizadas (en este caso 1 y 2) los paquetes no serán tratados pero sí recibidos, por lo que la mota atacada estará constantemente recibiendo información y consumiendo tiempo de ejecución y batería.

La solución que se ha aportado es viable para los dos casos, pero para realizar la explicación se ha utilizado el escenario en el que las dos motas autorizadas no filtran en función del origen del paquete.

Para poder determinar si los paquetes que llegan pertenecen a una fuente autorizada, hay que buscar aquellos parámetros comunes en todos los paquetes recibidos. Es decir, si un nodo envía 100 paquetes, se tendrá que buscar un campo que sea igual en todos ellos o que por lo menos sea lo menos variable posible.

Uno de estos valores relativamente constantes es el RSSI (Receive Signal Strength Indication). Como indica su nombre mide el nivel de fuerza de las señales de redes inalámbricas recibidas. Por lo tanto a mayor valor de RSSI señal recibida con más fuerza. Sobre esta herramienta se desarrollan fundamentos de la tecnología inalámbrica. Se usa de modo especial para la localización.

La metadata es una parte del paquete que no se envía pero en la que se guardan diversos parámetros. En los paquetes pendientes de enviar, puede modificarse campos como la potencia de transmisión (tx_power) entre otros, mientras que en los recibidos, puede comprobarse por ejemplo el CRC o el RSSI. A continuación se muestra el código de la metada:

```
                                                    ...
typedef nx_struct cc2420_metadata_t {

nx_uint8_t tx_power;
nx_uint8_t rssi;
nx_uint8_t lqi;
nx_bool crc;
nx_bool ack;
nx_uint16_t time;
nx_uint16_t rxInterval;

 /** Packet Link Metadata */

#ifdef
PACKET_LINK  nx_uint16_t maxRetries;
nx_uint16_t retryDelay;
#endif
} cc2420_metadata_t;
                                                    ...
```

Fig.An.9: Código.

La idea principal de la detección del ataque, se centra en conocer siempre el RSSI respecto al otro nodo, de esta forma cuando los valores no se ajusten a lo esperado se pueden tomar medidas. En esta prueba, al tener sólo dos nodos cada una de las motas puede guardar el valor del RSSI en una variable, pero en el caso de tener más nodos en la red se tendría que recurrir a vectores.

Al trabajar con RSSI, hay que tener en cuenta que es un valor que no es constante en el tiempo. Hay diversos factores que pueden hacer variar éste valor considerablemente sin tener que tratarse de un ataque:

- A más distancia entre emisor y receptor, el RSSI es más variable.
- Si se interponen objetos entre emisor y receptor el RSSI varía considerablemente.

- Si el emisor incrementa o disminuye la potencia de transmisión, el RSSI varía considerablemente.

Ante esta dificultad, hay que vigilar cuando se inicia el contraataque ya que puede que por algún motivo de los anteriores el valor RSSI se desajuste sin que esto tenga nada que ver con la presencia de un nodo atacante.

En nuestros estudios, se considera:

- En ningún momento se modifica la potencia de transmisión de las motas.
- No se interpondrán objetos temporalmente entre emisor y receptor, todos los escenarios serán estáticos.

Al descartar estos dos parámetros, sólo habría que realizar un pequeño estudio de la variabilidad del RSSI para seleccionar los rangos en los que dicho valor se considera viable. También es importante tener en cuenta el número de veces que se permite que se produzca un error, ya que no es recomendable lanzar el contraataque ante el primer valor desajustado. En este caso se permiten 10 errores antes de considerar un ataque.

Dentro del código de detección de ataques, hay que destacar:

```
                                        …
rssi2=call CC2420Packet.getRssi(msg);
      if(!contador_rssi)
        {
          rssi1=rssi2;
                                        …
```

Fig.An.10: Código.

Gracias al método getRssi de la interfaz CC2420Packet, se puede guardar el valor RSSI de un paquete determinado, en este caso “msg”. Existen otras formas de acceder a este parámetro, como por ejemplo guardar toda la metadata en una estructura y posteriormente leer el campo de RSSI.

Contraataque

Una vez ya se considera que se está llevando a cabo un ataque, las motas tienen que ser capaces de contrarrestarlo e impedir que el nodo atacante siga comunicándose con ellos. Para aislar al nodo intruso las dos motas tienen que ponerse de acuerdo para cambiar de red en el momento adecuado. Para no crear un patrón que podría ser detectado por el atacante se ha seleccionado la nueva red mediante la interfaz Random.

Dentro del código que pertenece a la parte de contraataque, hay que destacar la tarea sendcambio:

```
                                        …
task void sendcambio(){
      Topo* btrpkt = (Topo*)(call Packet.getPayload(&pkt, NULL));
       red=call Random.rand16();
       btrpkt->red = red;


       if (call AMSend.send(destino, &pkt, sizeof(Topo)) == SUCCESS)
       {
            contador_rssi=FALSE;
             error2=0;
             rssi=0;
             call Leds.led2Toggle();
             call ActiveMessageAddress.setAddress(red,midire);
       }

}
                                        …
```

Fig.An.11: Código.

Las dos motas serán capaces de detectar el ataque y seleccionaran una nueva red para seguir compartiendo la información sin intrusos. Una vez uno de los nodos determina la nueva red, la introduce en el campo red de la estructura Topo. Las motas al recibir este tipo de paquete saben que tienen que cambiar la dirección de red.

Una vez mandado el paquete de cambio de topología, será el momento en el que la mota atacada podrá cambiarse de red, manteniendo su identificador de nodo y reseteará todas las variables que intervienen en la detección.

Secuencia completa y estación base

En la siguiente figura, se detalla paso a paso los diversos estados por los que pasan los nodos en el proceso de ataque, detección y contraataque. No se asigna un valor de red determinado para hacer el ejemplo más general.

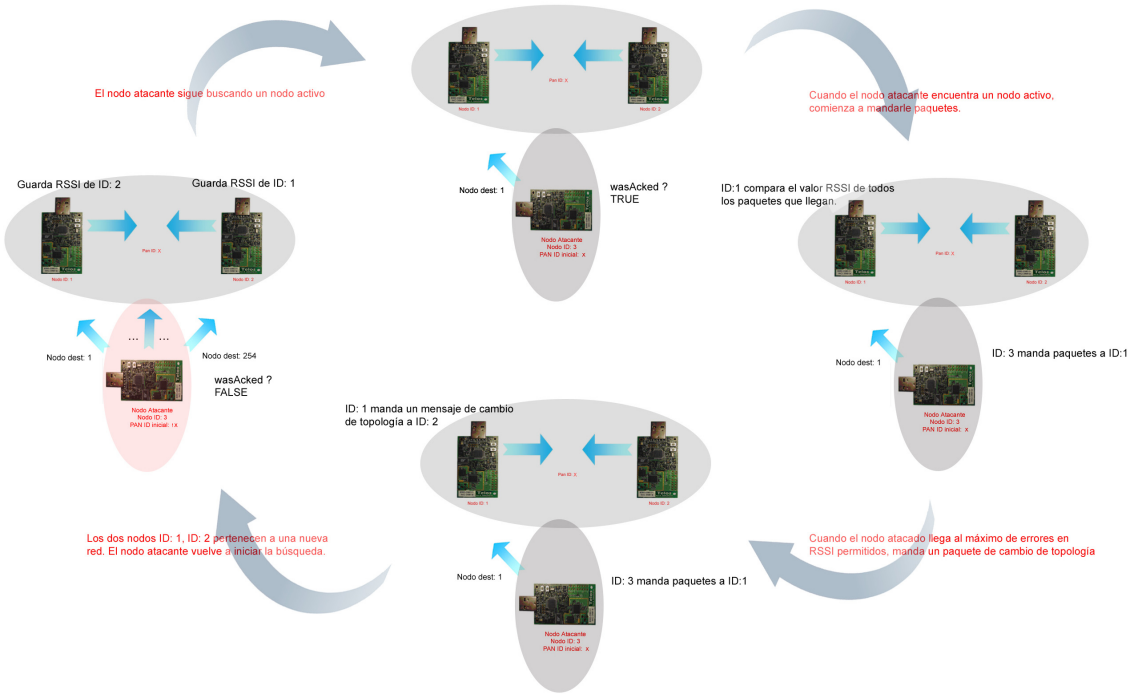

Fig.An.12: Gráfica contraataque.

En el apartado de herramientas, ya se ha hablado de la estación base, que permite ver de una forma gráfica el contenido de los paquetes que se intercambian las motas. A continuación se muestra una secuencia completa de ataque y contraataque.

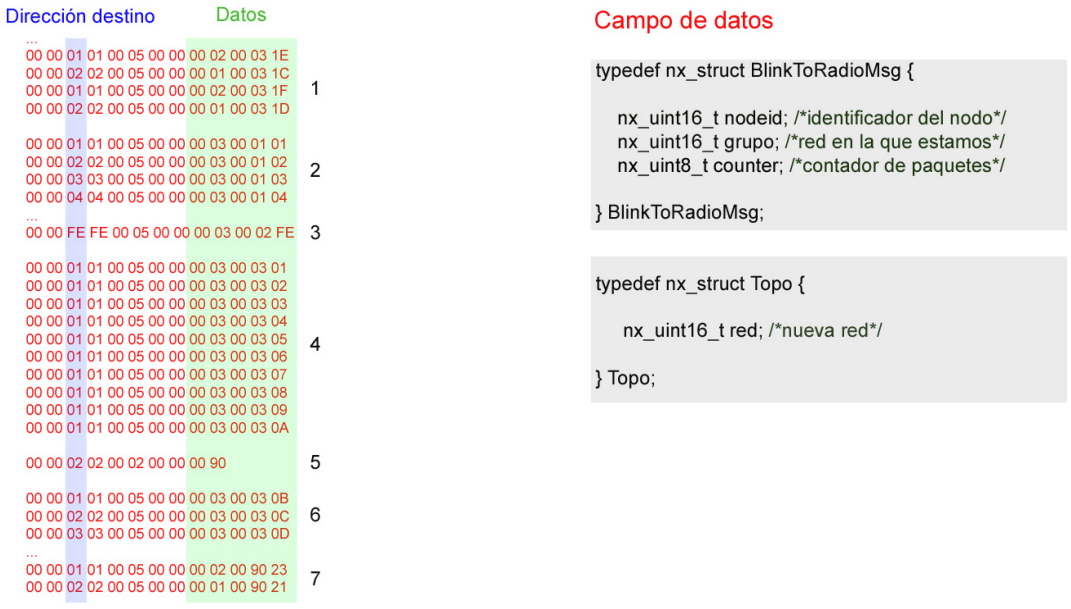

Fig.An.13: Secuencia ataque y contraataque.

Pasos en un ataque y contraataque:

1. Nodo con ID1 e ID2 mantienen una comunicación periódica en la que se intercambian el identificador de nodo, el identificador de red y un contador que indica el número de paquetes mandados.

2. Nodo atacante con ID3 manda un paquete de prueba esperando recibir un ACK. Si no lo recibe incrementa la dirección destino hasta llegar a 0xFD, una vez llega a este punto cambia de red, pone el contador de paquetes a 0 y vuelve a preguntar a todas las direcciones destino posibles.

3. Último paquete antes de recibir respuesta.

4. El nodo atacante ha recibido contestación. En este momento deja de buscar nuevos nodos y empieza a mandar mensajes indiscriminadamente al nodo con ID1.

5. Tras 10 lecturas erróneas de RSSI, el nodo con ID1 busca una nueva red y la comunica al nodo con ID2 mediante un mensaje de cambio de topología. En este caso la red seleccionada es 0x90.

6. El nodo atacante vuelve a activar el sistema de búsqueda.

7. Los nodos con ID1 e ID2 continúan con su comunicación inicial, pero en la red 0x90.

Anexo 4. Valores procedimiento deshabilitar CSMA/CA

CSMA/CA implementado en todos los nodos

Paquete nodo 2	00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 13
Primero en responder	00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 13
Segundo en responder	00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 13

Origen paquete

Fig.An.14: Ejemplo para interpretar muestras.

00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 13
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 13
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 13
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 14
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 14
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 14
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 15
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 15
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 15
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 16
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 16
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 16
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 17
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 17
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 17
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 18
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 18
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 18
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 19
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 19
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 19
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 1A
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 1A
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 1A
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 1B
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 1B
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 1C
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 1C
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 1C
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 1D
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 1D
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 1D
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 1E
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 1E

00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 1E
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 1F
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 1F
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 1F
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 20
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 20
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 20
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 21
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 21
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 21
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 22
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 22
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 22
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 23
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 23
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 23
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 24
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 24
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 24
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 25
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 25
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 25

CSMA/CA deshabilitado en nodo 4

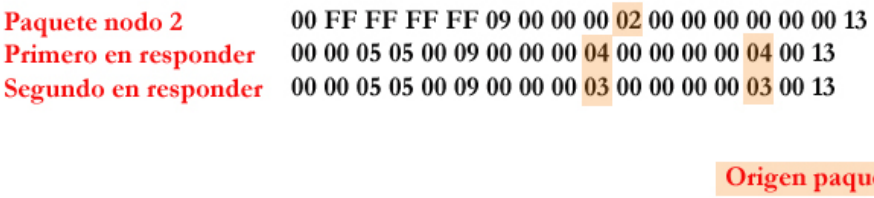

Fig.An.15: Ejemplo para interpretar muestras.

00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 13
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 13
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 13
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 14
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 14
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 14
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 15
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 15
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 15
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 16
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 16
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 16
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 17

00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 17
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 17
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 18
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 18
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 18
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 19
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 19
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 19
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 1A
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 1A
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 1A
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 1B
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 1B
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 1B
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 1C
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 1C
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 1C
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 1D
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 1D
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 1D
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 1E
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 1E
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 1E
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 1F
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 1F
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 1F
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 20
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 20
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 20
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 21
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 21
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 21
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 22
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 22
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 22
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 23
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 23
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 23
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 24
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 24
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 24
00 FF FF FF FF 09 00 00 00 02 00 00 00 00 00 00 25
00 00 05 05 00 09 00 00 00 04 00 00 00 00 04 00 25
00 00 05 05 00 09 00 00 00 03 00 00 00 00 03 00 25

Anexo 5. Tiempo IFS

The MAC sublayer needs a finite amount of time to process data received by the PHY. To allow for this, transmitted frames shall be followed by an IFS period; if the transmission requires an acknowledgment, the IFS shall follow the acknowledgment frame. The length of the IFS period is dependent on the size of the frame that has just been transmitted. Frames (i.e., MPDUs) of up to aMaxSIFSFrameSize in length shall be followed by a SIFS period of a duration of at least aMinSIFSPeriod symbols. Frames (i.e., MPDUs) with lengths greater than aMaxSIFSFrameSize shall be followed by a LIFS of a duration of at least aMinLIFSPeriod symbols.

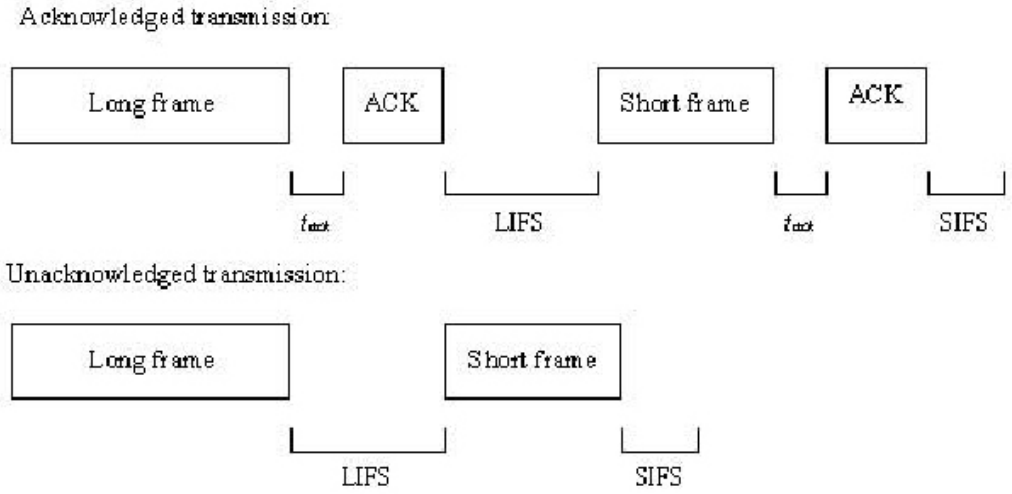

Fig.An.16: Tiempos de IFS según longitud trama.

Anexo 6. Identificador de red

Esta aplicación es efectiva para observar que se puede cambiar el parámetro que interviene en las comunicaciones a nivel MAC.

La mota con TOS_NODE_ID = 2, manda solamente un mensaje a la mota con identificador 5. Como se ha comentado en el punto 5.3, por defecto el identificador dentro de la red es el mismo valor que el presente en el TOS_NODE_ID. Por esta razón, la segunda mota que tiene una TOS_NODE_ID = 3 no recibe el paquete.

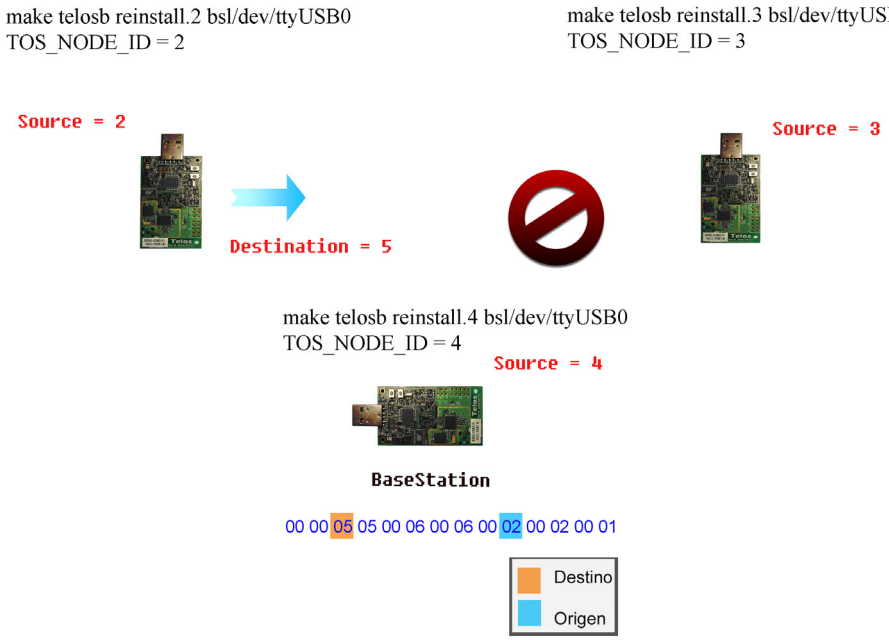

Fig.An.17: Cambio valor addr.

Para que se complete la comunicación, en la mota con TOS_NODE_ID = 3, se ha cambiado, mediante implementación de código, la dirección que interviene en las comunicaciones, asignándole el valor 5. Ahora puede observarse como sí se realiza la comunicación.

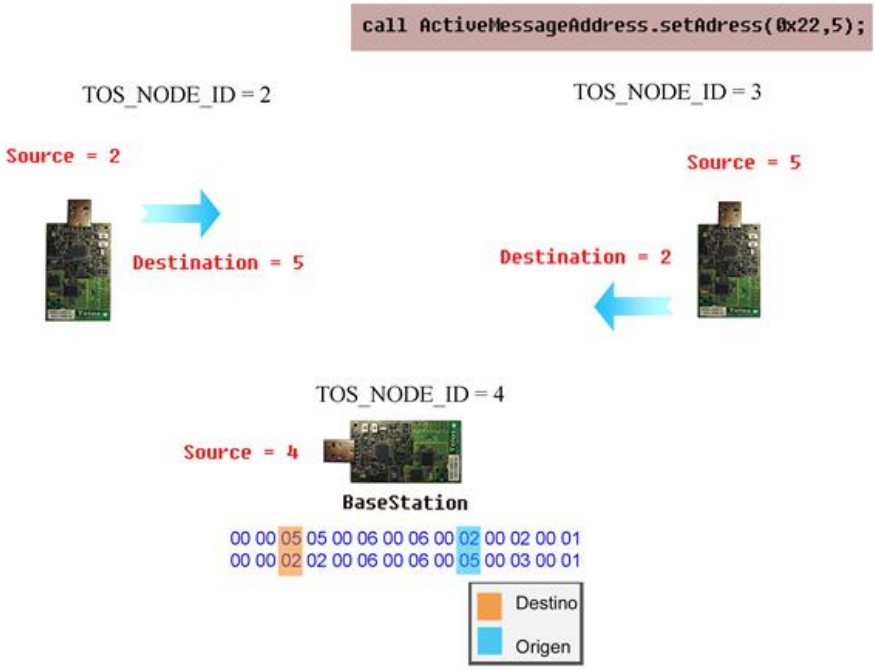

Fig.An.18: Cambio valor addr.

Anexo 7. Ocupar el medio

Estudio detallado

Para poder apreciar la figura a su máxima resolución se adjunta un enlace externo, ya que las dimensiones de la imagen hacen imposible poder reproducirla en el TFC (http://www.megaupload.com/?d=I61VQNH8).

Fig.An.19: Gráfica completa estudio de probabilidad detallado.

Leyenda

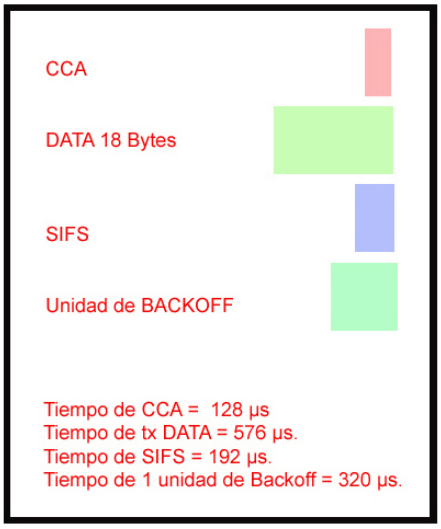

Fig.An.20: Leyenda.

Estudio global aproximado

Antes de comenzar, hay que advertir que este estudio se basa en la **Fig.An.22**. Por desgracia las dimensiones son demasiado grandes y es imposible apreciar con detalle la fotografía. Por este motivo se pone a disposición en un enlace (http://www.megaupload.com/?d=AI115O7S) externo en su resolución original.

Leyenda

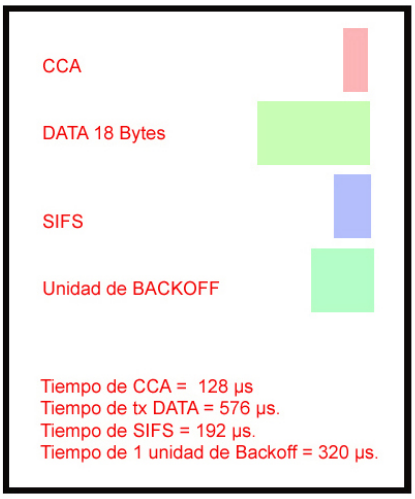

Fig.An.21: Leyenda.

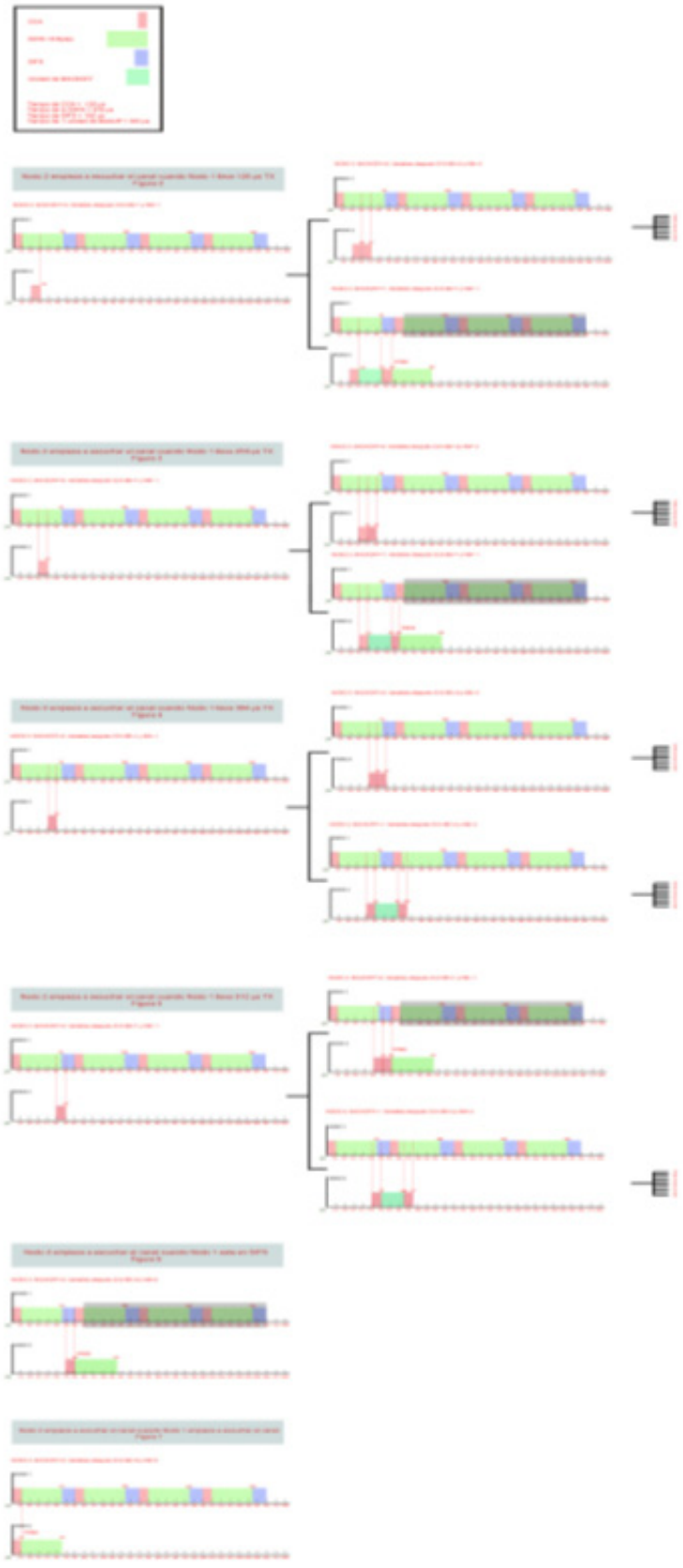

Fig.An.22: Gráfica completa estudio global aproximado.

Una vez estudiada la probabilidad en un punto determinado (Ver 5.2.1), se ha procedido a ver todo el escenario posible desde un punto de vista más global:

- Para facilitar los cálculos se ha trabajado sólo con dos nodos.
- Se han despreciado probabilidades del orden de 10^{-5}.
- Se han realizado estimaciones sólo para la probabilidad de enviar 2 paquetes ocupando el medio, siendo el primero de ellos el que ya está transmitiendo nodo 1 cuando nodo 2 realiza el CCA.
- Se ha trabajado con paquetes de 18 Bytes. El motivo es que es el valor límite para seguir trabajando con SIFS. Realmente este parámetro es el único que el usuario puede variar, siempre que se respete la correspondencia entre tamaño de paquete y elección de tiempo IFS (SIFS/LIFS).
- Al realizar la comprobación del medio, CCA, sólo se devuelve libre si durante todo el tiempo que se escucha el canal éste esta libre.

Probabilidades aproximadas

P(Nodo 1 mande 2 paquetes seguidos) = 1 – P(Nodo 1 NO mande 2 paquetes seguidos)

P(Nodo 1 N0 mande 2 paquetes seguidos) = $\left[\frac{1}{2}+\left(\frac{1}{2}*\frac{1}{4}\right)+X\right]=0{,}625+X$

P(Nodo 1 mande 2 paquetes seguidos) = $1-\left(0{,}625+X\right)=0{,}375-X=37{,}5\%-X$

P(Nodo 1 mande 2 paquetes seguidos) < 37,5%

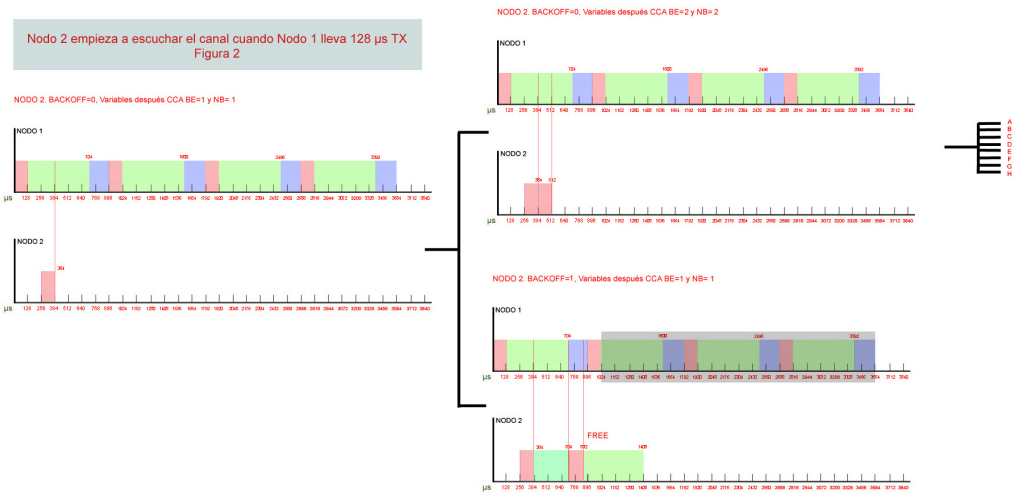

Fig.An.23: Parte 1 gráfica estudio global aproximado.

P(Nodo 1 mande 2 paquetes seguidos) = 1 – P(Nodo 1 NO mande 2 paquetes seguidos)

P(Nodo 1 N0 mande 2 paquetes seguidos) = $\frac{1}{2} + X$

P(Nodo 1 mande 2 paquetes seguidos)= $1 - \left(\frac{1}{2} + X \right) = \frac{1}{2} - X = 50\% - X$

P(Nodo 1 mande 2 paquetes seguidos) < 50%

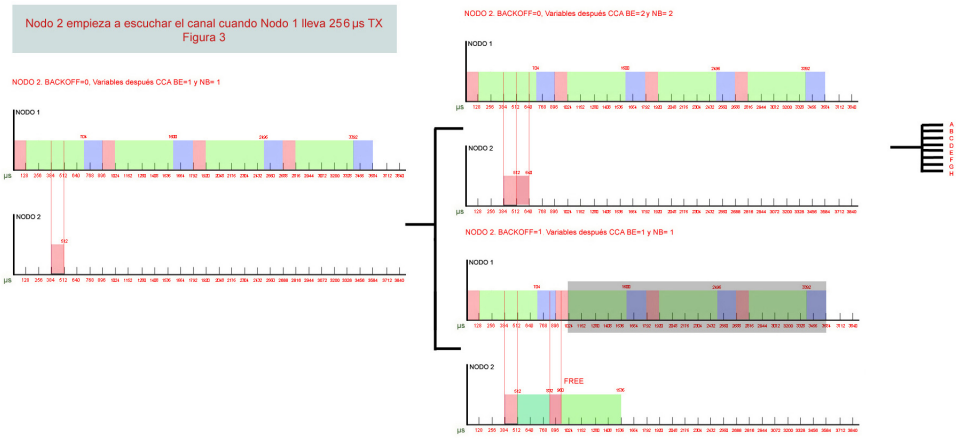

Fig.An.24: Parte 2 gráfica estudio global aproximado.

P(Nodo 1 mande 2 paquetes seguidos) = 1 – P(Nodo 1 NO mande 2 paquetes seguidos)

P(Nodo 1 N0 mande 2 paquetes seguidos) = $\left(\frac{1}{2} * \frac{1}{4} \right) = \frac{1}{8}$

P(Nodo 1 mande 2 paquetes seguidos) = $1 - \frac{1}{8} = \frac{7}{8} = 87{,}5\%$

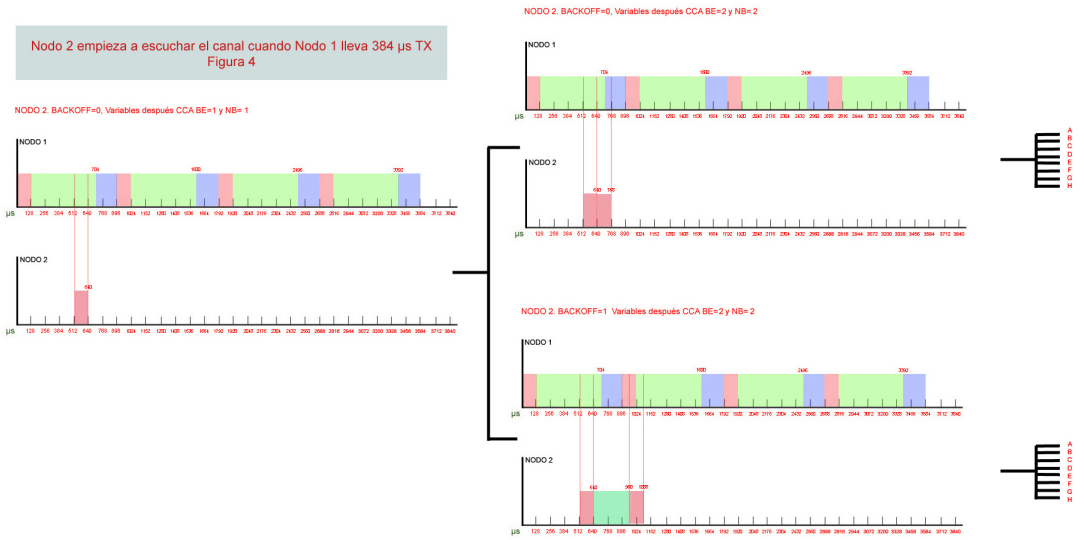

Fig.An.25: Parte 3 gráfica estudio global aproximado.

P(Nodo 1 mande 2 paquetes seguidos) = 1 – P(Nodo 1 NO mande 2 paquetes seguidos)

P(Nodo 1 N0 mande 2 paquetes seguidos) = $\frac{1}{2}$

P(Nodo 1 mande 2 paquetes seguidos) = $1-\frac{1}{2}=\frac{1}{2}=50\%$

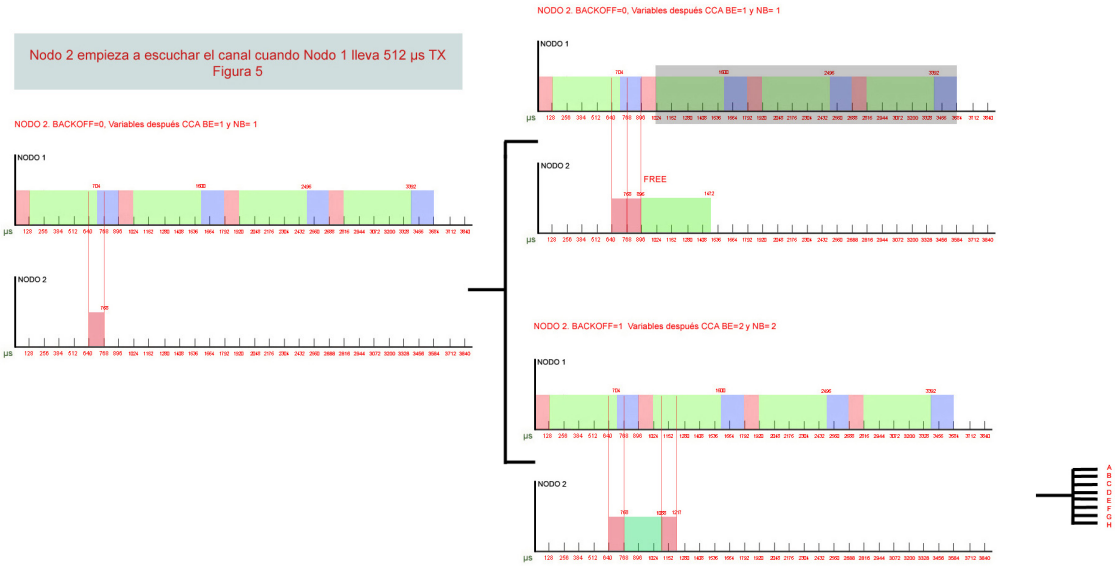

Fig.An.26: Parte 4 gráfica estudio global aproximado.

Puntos críticos

Existen varios momentos en el tiempo en el que se puede asegurar la probabilidad mínima de que no se envíe el siguiente paquete, sin depender del tamaño del paquete a enviar, siempre que se respeten los mínimos del estándar y teniendo en cuenta que se habla sólo de comprobar si se puede enviar 2 paquetes seguidos.

Momento 1 (afecta backoff 1, 2ª iteración): Si el nodo 2 realiza el primer CCA entre 384 µs y 448 µs antes de que finalice la transmisión de datos (cuadrado verde), la probabilidad de que el próximo paquete no se envíe es mínimo del 50%.

Momento 2 (afecta backoff 0, 2ª iteración): Si el nodo 2 realiza el primer CCA entre 128µs y el tiempo final de transmisión de datos (cuadrado verde), la probabilidad de que el próximo paquete no se envíe es mínimo del 50%.

Nodo 2 empieza a escuchar el canal cuando Nodo 1 esta en SIFS
Figura 6

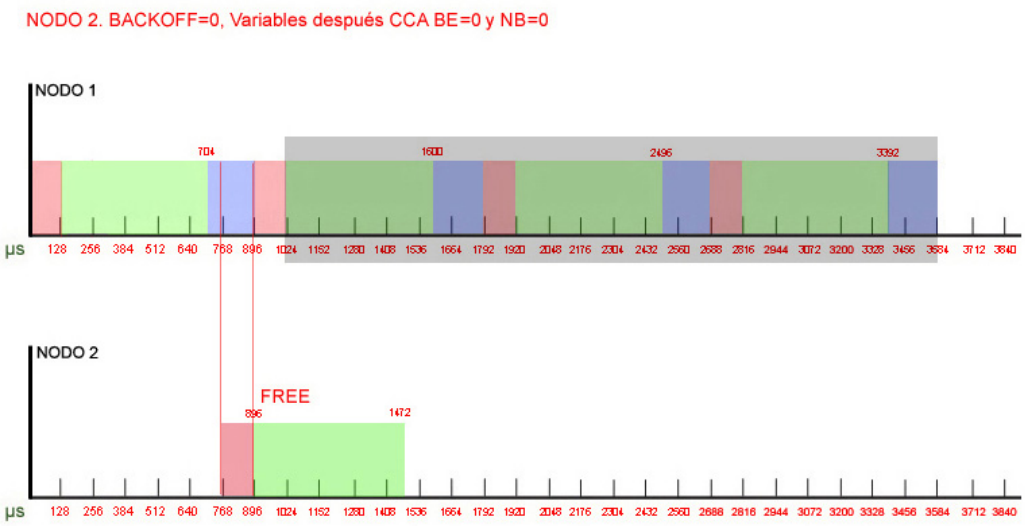

Fig.An.27: Parte 5 gráfica estudio global aproximado.

Nodo 2 empieza a escuchar el canal cuando Nodo 1 empieza a escuchar el canal
Figura 7

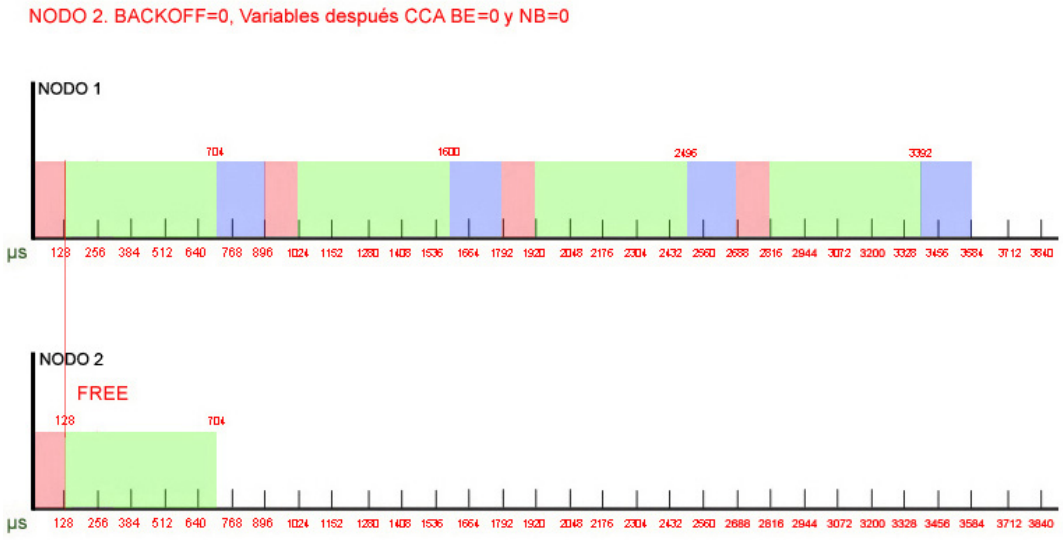

Fig.An.28: Parte 6 gráfica estudio global aproximado.