UNIVERSIDAD DE LA REPÚBLICA

SEGURIDAD EN REDES DE SENSORES INALÁMBRICOS

Proyecto de grado presentado por
Mariana Segovia

SUPERVISORES
Eduardo Grampín Castro
Leonardo Vidal

Montevideo, Junio de 2014

Índice general

Índice de figuras

Índice de cuadros

CAPÍTULO

1

Introducción

Este capítulo presenta la motivación, el problema planteado y el contexto de la investigación que fue llevada a cabo en este proyecto. Luego se explican las contribuciones que se lograron y por último se presenta la estructura del resto del documento.

1.1 Motivación

Los recientes avances en las comunicaciones inalámbricas y la electrónica han permitido el desarrollo de nodos sensores de bajo consumo, bajo costo y tamaño pequeño que se comunican de forma inalámbrica en distancias cortas. Esto ha permitido el diseño de aplicaciones en las que un grupo de estos sensores forman una red de sensores inalámbricos o WSN (*Wireless Sensor Network*) y cooperan para poder monitorizar su entorno. Las WSN representan una mejora significativa sobre los sensores tradicionales y están cobrando gran importancia en la actualidad utilizándose en variadas aplicaciones para distintos propósitos como pueden ser la medicina, la industria o el medio ambiente.
Una red de sensores puede desplegarse en una gran área por lo que no todos los nodos sensores pueden comunicarse directamente con la estación base. Para superar esta limitación, los nodos cooperan para retransmitir los paquetes hasta que lleguen a la estación base utilizando un protocolo de enrutamiento. Por lo que, los nodos cumplen dos funciones principales: recolectar datos a través de sus sensores y transmitir los datos de otros nodos para que lleguen a la estación base.
Por otro lado, la seguridad es uno de los retos clave para la creación de una red de sensores robusta y confiable. Los protocolos de enrutamiento son blanco de una gran variedad de ataques. Estos ataques son diferentes de los ataques contra las redes cableadas. En general, explotan el hecho de que la comunicación es inalámbrica y que los sensores no están físicamente protegidos, por lo que son suceptibles de ser capturados, examinados o modificados. La modificación de un nodo legítimo o la incorporación de nuevos nodos puede ser utilizada de forma malintencionada para dañar el funcionamiento de la red.

1.2 Planteo del problema

En las redes convencionales, la autenticidad, integridad y confidencialidad de los mensajes es obtenida mediante un mecanismo de seguridad extremo-a-extremo, como por ejemplo SSH, SSL o IPSec. Esto ocurre debido a que el patrón de comunicación es extremo-a-extremo, es decir, los routers intermedios solo necesitan ver el cabezal de los paquetes, sin ser necesario ni deseable que puedan acceder al contenido del mismo.

Esto no ocurre en las redes de sensores, ya que el patrón dominante de comunicación es muchos-a-uno, es decir, muchos nodos comunican sus lecturas de sensores o los eventos de red sobre una topología multi-salto hasta una estación base central. Sin embargo, nodos vecinos en la WSN usualmente observan los mismos eventos u observan eventos correlacionados. Si cada nodo envía un paquete a la estación base en respuesta al evento se crea tráfico redundante, desperdiciando energía y ancho de banda dentro de la red. Para reducir estos mensajes redundantes, la red de sensores utiliza técnicas de procesamiento dentro de la red como por ejemplo agregación de datos y eliminación de duplicados. Este procesamiento dentro de la red requiere que nodos intermedios accedan, modifiquen y/o eliminen el contenido de los mensajes enviados por otros nodos. Por lo tanto, los mecanismos de seguridad extremo-a-extremo entre los nodos y la estación base no son viables. A su vez los mecanismos de seguridad extremo-a-extremo son vulnerables a ciertos ataques de denegación de servicio. Por ejemplo, si la integridad del mensaje solamente se chequea en el destino final, la red va a enrutar paquetes inyectados por un atacante varios saltos antes de detectar el problema, consumiendo recursos de la red [21].
Por otro lado, las WSN son especialmente vulnerables a ataques de seguridad debido a diversas características que presentan, por ejemplo, utilizan un medio de transmisión broadcast y los nodos generalmente se ubican en medios hostiles donde además no se cuenta con protección física. Además los escasos recursos de este tipo de dispositivos dificultan la tarea de proteger la información que estos nodos envían. Incluso los mecanismos de seguridad para redes inalámbricas ad hoc pueden no ser directamente aplicables a redes inalámbricas de sensores debido a esta reducida capacidad de los nodos. Por lo que, apesar de que comparten propiedades respecto a la infraestructura, los protocolos ad hoc de enrutamiento pueden no ser una buena solución para las WSNs.
Debido a estas carencias, diferentes mecanismos de seguridad han sido propuestos y muchos más están siendo elaborados en la actualidad. Por tal motivo, es importante conocer las debilidades de los protocolos actuales y contar con herramientas para facilitar el desarrollo de nuevos mecanismos para proteger este tipo de redes.
La idea de construir herramientas de *ethical hacking* que permitan testear la seguridad de una infraestructura es bastante común en las redes tradicionales. Actualmente existen númerosas herramientas y de distintos tipos con este propósito, como por ejemplo: Nmap, Ncrack, Maltego, Yersinia, Dsniff, Ettercap, ike-scan, Nessus, entre muchas otras. Sin embargo, para las redes de sensores inalámbricos se ha realizado poco trabajo en esta área. Actualmente, las únicas herramientas de este tipo que existen son Spysense y Sensys, ambas propietarias. En este contexto es que se propone, como objetivo de este proyecto, crear una herramienta de *ethical hacking* para realizar testing de los mecanismos de seguridad, a la vez que permite el estudio y comprensión de las vulnerabilidades actualmente existentes.

1.3 Contribución

Las principales contribuciones de este trabajo son:

- Un estudio exhaustivo de los distintos tipos de ataques que pueden ocurrir en las redes de sensores inalámbricos.
- La identificación de vulnerabilidades existentes en los protocolos: GTS, CSMA/CA y CTP.
- La implementación de ataques que explotan esas vulnerabilidades identificadas en el marco de la herramienta de *ethical hacking* objetivo de este trabajo.
- Un estudio del impacto y comportamiento de cada uno de los ataques implementados que resalta la necesidad de mejores mecanismos de seguridad.
- Crear una herramienta de *ethical hacking* que integra los ataques implementados. También permite analizar tráfico capturado desde sensores y brinda una interfaz amigable para analizar los datos recabados en simulaciones realizadas con el simulador Avrora.

La herramienta construida permite la monitorización pasiva de datos así como la ejecución de varios tipos de ataques. Esto permite comprometer la funcionalidad de la red, revelar debilidades existentes que hacen que las redes de sensores sean susceptibles de amenazas y también permiten estudiar los efectos de tales amenazas en la propia red. Una mejor comprensión de las vulnerabilidades permite mitigarlas y diseñar mecanismos de seguridad más resistentes. Así como conducir al desarrollo de aplicaciones más seguras y mejores mecanismos de detección y prevención.
Las pruebas realizadas y los ataques disponibles en la herramientas se puede ejecutar en simulaciones y en dispositivos reales. Para este proyecto se utilizaron las plataformas TelosB y Shimmer; así como los simuladores Cooja y Avrora. Integarar nodos reales en las pruebas es un punto muy importante ya que las simulaciones tienen el desafío de modelar con precisión la interferencia de la capa física y además no tienen en cuenta las limitaciones impuestas por el hardware, el sistema operativo y las aplicaciones, lo que puede conducir a la simplificación de los escenarios de ataque. Esto es especialmente importante en dispositivos de recursos limitados como es este caso. Por otro lado, cuando se cuenta con una red de sensores desplegada y funcionando se puede utilizar esta herramienta para dejar al descubierto configuraciones no adecuadas de las aplicaciones instaladas o generar planes de contingencia frente a vulnerabilidades conocidas que se decidan aceptar. Sin embargo, las simulaciones también son de gran utilidad para la evaluación del comportamiento, los efectos de los distintos ataques y la realización de pruebas de escalabilidad. A la vez que también sirven para el desarrollo y testeo de nuevos mecanismos de seguridad.
Es importante señalar que para los ataques desarrollados en este proyecto no hay implementaciones de referencia. Por otro lado, esta herramienta permite ejecutar los ataques en una red física y real con dispositivos desplegados. Esto es posible a través del uso de un nodo o un transmisor de radio que utilice IEEE 802.15.4 y el sistema operativo TinyOS v2.
El nodo transmisor de IEEE 802.15.4 funciona como atacante instalando software construido a partir de los protocolos de red modificados para explotar ciertas vulnerabilidades existentes e identificadas durante este proyecto. De esta forma, un nodo comercialmente disponible brinda la infraestructura necesaria para testear la seguridad de una red desplegada. Este nodo puede incluso ser uno de los nodos desplegados en la red. Esto se logra simplemente grabando en la memoria flash del nodo los binarios correspondientes a los ataques implementados durante este proyecto.

1.4 Organización del documento

El resto del documento se organiza de la siguiente forma. En el capítulo 2 se introduce el concepto de red de sensores inalámbricos explicando cuáles son sus características, aplicaciones, principales plataformas y tecnologías de radio. En el capítulo 3 se presenta una revisión de los principales ataques de seguridad en este tipo de redes así como una clasificación de los mismos. En el capítulo 4 se describe el ambiente de desarrollo y pruebas, así como también la arquitectura de la herramienta desarrollada. En el capítulo 5 y 7 se presentan los ataques implementados para los protocolos de capa MAC (GTS y CSMA/CA) y red (CTP) respectivamente. Para todos los protocolos se describe su funcionamiento, las vulnerabilidades encontradas y como fueron explotadas en la implementación. En los capítulos 6 y 8 se muestran los resultados obtenidos para los ataques a los protocolos de capa MAC y red respectivamente. Finalmente, el capítulo 9 contiene las conclusiones, un resumen de los principales resultados y el trabajo a futuro.

CAPÍTULO

2

Redes de sensores inalámbricos

2.1 Introducción

Este capítulo ofrece una visión general del mundo de las redes inalámbricas de sensores. Se comenzará introduciendo este concepto y sus principales aplicaciones. A continuación, se mencionan los principales sistemas operativos utilizados por este tipo de dispositivos. Finalmente, se presentan las plataformas de nodos sensores que actualmente se encuentran disponibles en el mercado.

2.2 Concepto de redes de sensores inalámbricos

Una red de sensores inalámbricos (WSN - *Wireless Sensor Network*) está compuesta de un gran número de nodos sensores densamente desplegados (el número de dispositivos puede ser del orden de cientos o miles de nodos) que tienen reducidas dimensiones, bajo consumo de energía y bajo costo. Estos nodos son dispositivos inalámbricos auto configurables capaces de detectar eventos o capturar señales como por ejemplo contaminación del aire, temperatura ambiente, presión, señales cardíacas y encefálicas, etc. Los nodos se despliegan en distintos ambientes como la tierra, el aire, en vehículos, dentro de las edificaciones, etc, para captar datos y trasmitirlos utilizando un canal de comunicación inalámbrico hasta una estación base o *sink* como se puede observar en la figura 2.1. La estación base recolecta la información de todos los sensores, luego una aplicación los analiza y puede tomar decisiones en base a estos datos. A su vez, esta información puede ser accedida a través de otro tipo de redes (como por ejemplo Internet) y utilizada para múltiples propósitos. Los nodos generalmente utilizan tecnologías de bajo consumo para comunicarse entre sí. En el Anexo A.3 se presentan las principales tecnologías de radio que se utilizan. Por otra parte, en el Anexo A.2 se describe como es la pila de protocolos que se emplea en este tipo de dispositivos.
Estos sensores reciben usualmente el nombre de *mote* (mota, pequeña partícula) y presentan ciertas limitaciones en cuanto a reservas de energía, memoria, ancho de banda y procesamiento disponible (en el Anexo A.1 se presenta la arquitectura de un *mote*). Dadas las limitaciones mencionadas estos nodos no pueden realizar un gran procesamiento de los datos recolectados ni almacenarlos por un largo período de tiempo.
Las redes de sensores inalámbricos se han previsto para una amplia gama de áreas de aplicación. La mayor parte de los despliegues de este tipo de redes cuentan una estación base que normalmente es estática, aunque pueden existir múltiples estaciones bases o incluso, estas pueden ser móviles. Esta estación base recoge la información obtenida por los sensores. Luego analiza y procesa la información

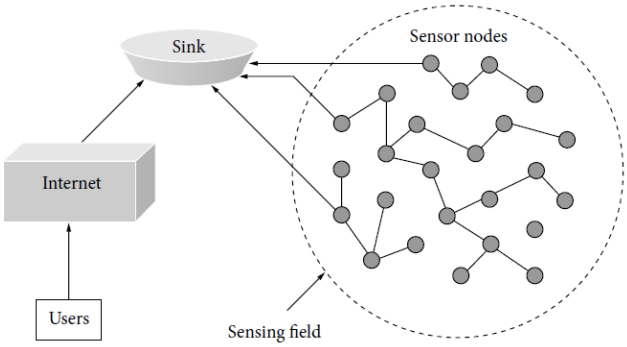


Figura 2.1: Estructura de una red de sensores [1].

para aplicaciones específicas. Podría estar conectada a Internet a través de comunicaciones inalámbricas o cableadas de tal manera que un usuario remoto sea capaz de acceder a los datos obtenidos en cualquier momento o desde cualquier lugar [22].

2.3 Aplicaciones

Las WSN pueden recoger datos de los nodos en tres modos diferentes: captura de eventos, capturas periódicos e informes bajo demanda. En el modo de presentación de informes por captura de eventos, los nodos sensores detectan e informan datos solo si se produce un evento en su entorno. Por ejemplo, si una WSN se despliega en un bosque para controlar los incendios forestales, los nodos sensores se reportará a la estación base solo si se produce un incendio en el bosque. La presentación de informes periódicos se utiliza cuando los datos no son urgentes o cuando la información debe ser enviada cada cierto intervalo de tiempo, por ejemplo, un electrocardiograma que monitoriza un paciente. En esta modalidad, el informe de los datos detectados se presenta a la estación base cada períodos de tiempo predefinidos. En la captura bajo demanda, se envían solicitudes de información a los nodos sensores para que envíen sus datos detectados. Por ejemplo, una aplicación que supervisa contaminantes químicos en el agua puede enviar una petición a los nodos de sensores para detectar el nivel de contaminantes y reportar los valores a la estación base cuando se requiera.
A continuación, se presentan las aplicaciones más comunes de las WSN[23]. Este relevamiento de aplicaciones será utilizado posteriormente para la elección de los escenarios sobre los cuales se harán las pruebas.

Aplicaciones Militares

Dentro de esta categoría las WSN se utilizan para recolectar información militar como por ejemplo rastrear a un enemigo, vigilar el campo de batalla o rastrear y clasificar objetivos en movimiento [24] [25]. También se utilizan para protección de propiedades, vigilancia y control de las fronteras. Un ejemplo puede verse en la figura 2.2.
Por ejemplo, en [26], los autores presentan un sistema de vigilancia de una reja o límite que comprende un robot, una cámara y dos tipos de nodos sensores: nodos que se colocan en el suelo y nodos que se utilizan en las rejas. La red informa de los datos adquiridos a la estación base la cual envía mensajes de control a la cámara para que enfoque el lugar donde el evento detectado ha ocurrido. Adicionalmente se puden enviar comandos a los robots móviles que amplían la distancia de comunicación del sistema.

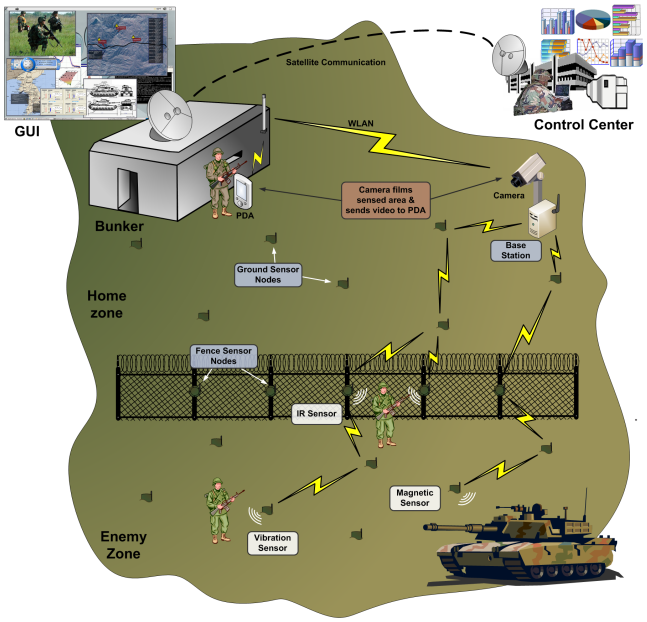


Figura 2.2: Arquitectura de una aplicación militar para vigilancia utilizando una WSN [2]

Monitorización ambiental en ambientes cerrados

La capacidad de utilizar un sensor para medir la temperatura, la luz, el estado de puertas y ventanas, los flujos y la contaminación del aire puede ser utilizada para realizar un óptimo control de ambientes cerrados. Se pueden utilizar sensores para ayudar en el uso de calentadores, ventiladores, entre otros equipos, de una manera razonable para evitar gastos innecesarios, por ejemplo, por calentamiento o enfriamiento innecesario de una habitación.

Otras aplicaciones en las que se pueden utilizar sensores son para mitigar fuegos y terremotos. Hoy en día los detectores de humo son comunes en edificios y permiten disparar alertas. Sin embargo, una WSN podría en estos casos guíar a las personas atrapadas en el incendio a través de la ruta de escape más segura haciendo uso de luces en el techo y paredes u otro método. Sha et al. [27] proponen FireNet, una arquitectura de WSN para realizar aplicaciones que ayuden en los rescates de incendios. Además señalan las carencias en los sistemas anti-incendios actuales que se podrían mejorar. Por otro lado, una WSN podría ser integrada con otros componentes como por ejemplo un sistema de información geográfica (GIS – *Geographic Information System*) que permitiera tener más información de la situación real.

Otra aplicación de estas redes se muestra en la figura 2.3 y consiste en permitir inspeccionar la estructura de una construcción (edificios, puentes, aviones, barcos, etc.) basándose en las vibraciones que esta tiene. Para esto se pueden incorporar sensores inalámbricos en unidades estructurales específicas a modo de grabar las vibraciones. La inspección de un edificio, por ejemplo luego de un terremoto, utilizando este sistema se puede basar en datos computando las vibraciones promedio y máximas de cada sensor. De este modo se puede determinar rápidamente los daños y fisuras para repararlos de forma más precisa. También se podrían utilizar para estudios de otro tipo como por ejemplo realizar pruebas de esfuerzo de una construcción o evaluar las vibraciones causadas por el viento u otro factor. Xu et al. [28] propone Wisden, un sistema de WSN para obtener información estructural. Wisden guarda información de forma contínua utilizando una WSN multi-salto y luego almacena esta información en una estación base.

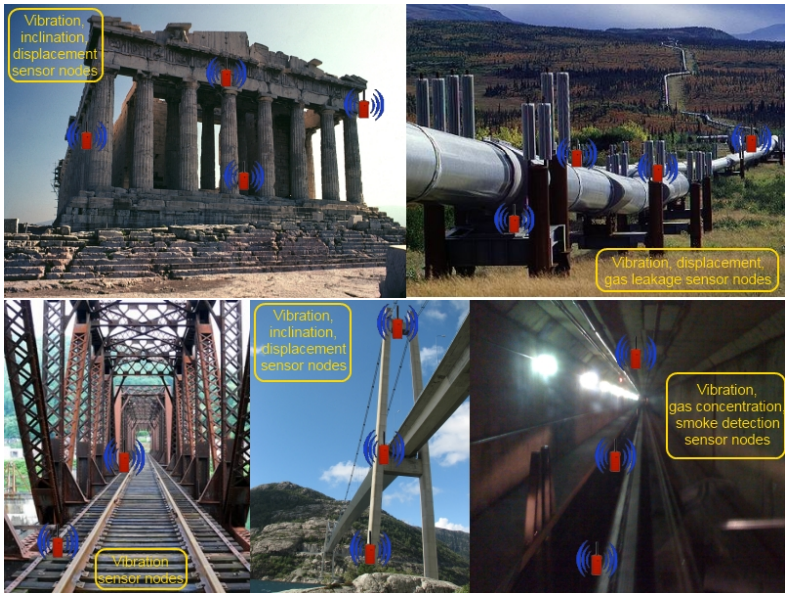


Figura 2.3: Aplicación de una red de sensores para monitorizar la estructura de una construcción [3] [4]

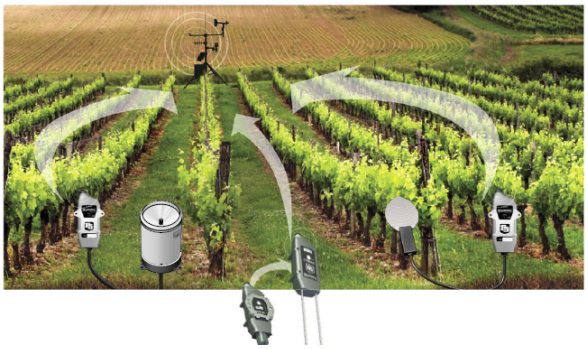
Monitorización ambiental al aire libre

Otra área de aplicación de las WSN es la monitorización ambiental. Por ejemplo, para el rastreo de animales como se muestra en la figura 2.4a. En [29] se menciona una aplicación donde desplegaron una WSN de 150 nodos para estudiar la anidación de una especie particular de aves. Otros ejemplos son presentados en [29] donde se utilizaron WSNs para el estudio de lobos. Se colocaba un nodo en cada animal para poder recabar información acerca de sus condiciones de vida y su comportamiento. De forma similar, en reservas de Kenia se han desplegado WSNs para estudiar las cebras. Su objetivo era colocar sensores en distintas especies de animales terrestres para comprender la interacción e influencia de estas, comprender patrones de migración de los animales salvajes y como estos son afectados por el cambio climático y la influencia humana.
Otras aplicaciones relacionadas con la monitorización ambiental que se han desarrollado son aplicaciones para observaciones del medio ambiente, estudio de fenómenos naturales y pronóstico del clima. Como por ejemplo el sistema ALERT (*Automated Local Evaluation in Real Time*) desarrollado por el *National Weather Service* que es el precursor de las redes de sensores modernas utilizadas para monitorizar las predicciones de lluvias e inundaciones de California y Arizona. También se ha investigado la posibilidad de desplegar WSNs para monitorizar la actividad volcánica, como puede verse en la figura 2.4c, de modo de determinar el origen y la ubicación de la erupción, diferenciando la verdadera erupción del ruido y otras señales no deseadas [32].

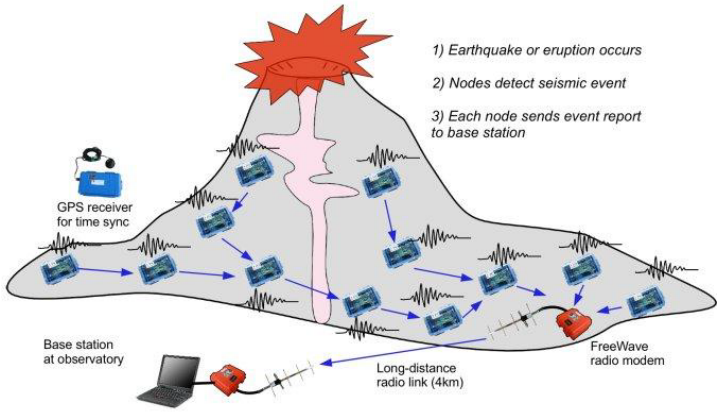
Agricultura

Las propuestas también se han desarrollado para desplegar WSNs en la agricultura como puede verse en la figura 2.4b para mejorar la eficiencia del trabajo, mejorar el crecimiento de los cultivos, reducir los costos, reducir su impacto ambiental y aumentar la calidad de los productos. Las WSN pueden ayudar a monitorizar campos, viñedos y huertas, ayudando a los agricultores a prevenir daños en sus cosechas y aumentando la producción. La automatización en la agricultura provoca una contribución fundamental a lo que hoy se conoce como agricultura de precisión. La agricultura de precisión es la técnica de aplicar la cantidad correcta de elementos (agua, fertilizantes, pesticidas, etc) en el lugar correcto y en el momento adecuado para aumentar la producción y mejorar la calidad, mientras se protege el

(a) Aplicación para estudiar el comportamiento de los animales [30]



(b) Aplicación de una WSN para monitorizar cultivos [31]



(c) Aplicación de una WSB para estudiar la actividad volcánica [32]

Figura 2.4: Aplicaciones de las redes de sensores

medio ambiente. Una aplicación de la agricultura de precisión se cumple con la utilización de una WSN capaz de controlar los parámetros relevantes (por ejemplo humedad del suelo y temperatura del aire) y transmitir estos datos de forma inalámbrica a la ubicación agricultor para tomar las medidas adecuadas o integrándolo, por ejemplo, con un sistemas de riego donde se libere la cantidad adecuada de agua para incrementar las cosechas basándose en los datos obtenidos. En [33] se utiliza una WSN en un campo de plantación de papas para recolectar datos y luego, mediante el uso de un modelo de administración se realizan cálculos matemáticos para controlar el riego y mejorar las cosechas. En [34] se utilizan los sensores en invernaderos para monitorizar factores que afectan a los cultivos como por ejemplo la temperatura, la humedad, la radiación solar y las condiciones del suelo.

Monitorización de la salud

Sistemas para el cuidado de la salud e investigaciones científicas en el área también se pueden beneficiar de las WSN ya que proveen información precisa, más detallada y en tiempo real para la toma de decisiones. Una de las aplicaciones particulares dentro del sector salud, recibe el nombre de BAN (*Body Area Network*) y puede apreciarse en la figura 2.5. Una BAN se compone en un conjunto de sensores que recolectan información sobre un paciente y un dispositivo que actúa como estación base, por ejemplo un celular, que permita enviar la información recolectada a servidores que analicen la información en busca de patrones anormales y posibles situaciones de alerta ante las cuales se deba reaccionar. La información recabada por estos dispositivos le permite a los doctores realizar un

diagnóstico más preciso y un mejor tratamiento. Esta incluye el sensado de variables fisiológicas que son señales analógicas que corresponden a actividades fisiológicas del cuerpo humano o acciones del cuerpo, como por ejemplo, electrocardiogramas, electroencefalogramas, electromiogramas, glucosa en sangre, presión sanguínea, acelerometrías, etc. Toda esta información puede ser transmitida, procesada, analizada y almacenada dentro de la historia clínica del paciente. En casos de emergencia, se pueden enviar mensajes de alarma a quien corresponda.

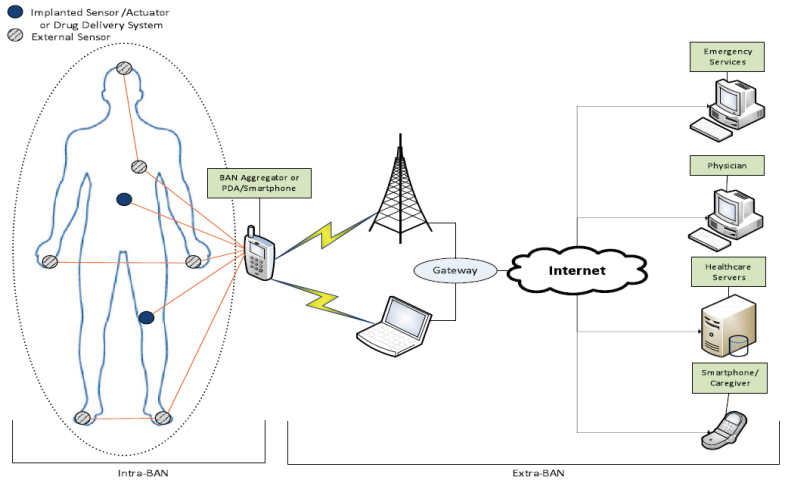


Figura 2.5: Aplicación de una red de sensores para monitorizar la variables fisiológicas de un paciente [5]

Otros

Otros posibles escenarios son:

- Computación dependiente del contexto (*context-aware computing*), por ejemplo para hogares inteligentes.
- Información sobre la movilidad, por ejemplo monitorización de tráfico de autos u ómnibus.
- Control del medio ambiente y seguimiento químico, lo cual incluye aplicaciones para detectar problemas de contaminación en al aire o agua, fuego, innundacciones, movimientos de tierra o ruido.
- En la industria, por ejemplo, para monitorizar la condiciones estructurales de las maquinarías y así saber si estas requieren mantenimiento o para automatizar procesos.

2.4 Sistemas operativos para WSN

Con el rápido crecimiento en la investigación de las WSN, varios sistemas operativos fueron desarrollados en los últimos años [35]. Los sistemas operativos para WSN son típicamente menos complejos que los de propósito general debido a que deben manejar los requisitos especiales de las aplicaciones en las que se usan y las restricciones de recursos. Además, estos sistemas no necesitan incluir el soporte de interfaz de usuario.
TinyOS [36] es el sistema operativo para WSN estándar de facto. Otros sistemas operativos disponibles son Contiki [37] y Mantis [38]. A continuación se describe estos sistemas operativos para las WSN.

Contiki

Contiki es un sistema operativo liviano para WSN de código abierto escrito en C. Una configuración normal de Contiki consume 2 KB de RAM y 40 KB de ROM. Una instalación completa de Contiki incluye: un núcleo multitarea, manejo de múltiples hilos de forma expropiativa, red TCP/IP, IPv6, interfaz gráfica, navegador web, personal web server, cliente telnet simple[39].
Este sistema operativo es altamente portable y desarrollado específicamente para su implementación en sistemas embebidos. Para lograr un sistema ligero, Contiki, se basa en eventos que cargan y descargan dinámicamente los programas en los hilos correspondientes según se va ejecutando la aplicación.

Mantis

Mantis es un sistema operativo multi-hilo reciente específico para WSN escrito en C, es liviano y eficiente en cuanto al uso de energía. Incluye la pila de protocolos de red, núcleo, planificador y manejo remoto de los nodos. También es portable a múltiples plataformas, por ejemploa PDA, PC x86 y nodos con sensores[40].
Ante el incremento de complejidad en las tareas realizadas por las WSN como comprensión, agregación y procesado de señales, los procesos múltiples en Mantis permiten interpaginar tareas complejas con tareas susceptibles de manera que se mitiguen los problemas de *overflows* en los *buffers*. Otra característica importante de Mantis es su eficiente uso de la memoria RAM (menos de 500 bytes incluyendo el núcleo, los controladores y la pila de protocolos) y de la energía (incluye un modo dormido).

TinyOS

TinyOS es un sistema operativo de código abierto específico para WSN basado en componentes desarrollado por la Universidad de Berkeley. El sistema operativo ocupa menos de 400 bytes de memoria RAM y provee una biblioteca que incluye protocolos de red, servicios distribuidos, manejadores de los sensores y herramientas para la adquisición de datos [41]. También provee bajo consumo de energía y operaciones de concurrencia intensiva. A diferencia de la mayoría de los otros sistemas operativos, TinyOS se basa en un modelo de la programación controlado por eventos en vez de multiprocesos.
El diseño del núcleo de TinyOS está basado en dos niveles de planificación: eventos y tareas. Los eventos están pensados para realizar un procesamiento pequeño, por ejemplo cuando un temporizador interrumpe o para atender las interrupciones de un sensor. Por otro lado, las tareas están pensadas para hacer una cantidad mayor de procesamiento y no son críticas en tiempo. Los eventos tiene prioridad frente a las tareas, por lo tanto si una tarea se está ejecutando y se dispara un evento, primero se atiende al evento y luego se continúa con la ejecución de la tarea. Con este diseño los eventos (que son rápidamente ejecutables) pueden ser atendidos inmediatamente e interrumpen a las tareas (que tienen mayor carga computacional en comparación a los eventos).
Actualmente, TinyOS provee un único *stack* de memoria que es compartido, es decir, no hay separación entre el núcleo del sistema operativo y el programa del usuario. Tampoco se provee protección de la memoria.
TinyOS cuenta con dos versiones, la primera versión fue liberada en 2002. Esta primera generación, a pesar de haber sido un éxito, tenía algunos defectos en cuanto al diseño por lo cual se liberó en 2006 la segunda generación. Se hicieron grandes cambios de una versión a la otra por lo que se decidió crear algo nuevo y no tratar de cambiar el código estable que ya existía de la primera versión. En consecuencia, las dos versiones de TinyOS no son compatibles entre sí.
La segunda versión de este sistema operativo fue la utilizada para el desarrollo de este proyecto. Durante este documento se hará mención a los TEPs (*TinyOS Enhancement Proposals*), estos son un conjunto de documentos que describen la estructura, los objetivos de diseño y partes de la implementación del sistema operativo TinyOS.
Tanto TinyOS como los programas escritos para él son escritos en un lenguaje de programación especial llamado NesC (*Network Embedded Systems C*) [42], que es una extensión del lenguaje de programación C orientado a sistemas embebidos.

En NesC se separa la construcción de la composición. Para ello se tienen 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.
Otra particularidad de este lenguaje es que las interfaces, que son el punto de accesos a los componentes, son bidireccionales. Estas contienen comandos y eventos. El proveedor de una interfaz implementa los comandos mientras que el que las utiliza implementa eventos.

2.5 Plataformas disponibles

Los nodos de una red de sensores varían en forma y tamaño. Normalmente tienen forma rectangular de 5 cm de lado aproximadamente o forma de disco de menos de 1 cm de diámetro. En la figura 2.6 se muestran algunos ejemplos de nodos sensores. En el mercado existen varias plataformas disponibles, entre las que se encuentran [20] [18]: TelosB/Tmote Sky, Mica2/MicaZ, SHIMMER, IRIS, Sun SPOT, entre otras. En las tablas 2.1 y 2.2 se encuentran las características de las distintas plataformas. Luego, se presentan las propiedades de cada una de ellas.

Plataforma	**Procesador**	**Bus**	**Flash**	**RAM**	**EEPROM**	**ADC**
TelosB	MSP430	16 bits	48K	10K	1M	12 bit
MicaZ	ATMega128L	8 bits	128K	4K	512K	10 bit
Shimmer	MSP430	16 bits	48K	10K	No	12 bit
IRIS	ATMega1281	8 bits	128K	8K	4K	10 bit
Sun SPOT	AT91RM920T	32 bits	4M	512K	No	12 bit
Mica2	ATMega128L	32 bits	4M	512K	No	12 bit

Cuadro 2.1: Microprocesadores utilizados y sus características [18] [19].

Plataforma	**Ancho-Largo-Altura** [mm]	**Peso (Sin batería) [g]**	**Peso (Con batería) [g]**
TelosB	32.00 x 65.53 x 6.60	14.93	63.05
MicaZ	31.75 x 57.15 x 6.35	15.70	63.82
Shimmer	20.32 x 44.45 x 12.7	4.87	10.36
IRIS	31.75 x 57.15 x 6.35	21.29	69.40
Sun SPOT	63.5 x 38.1 x 25.4	33.49	58.08
Mica2	58 x 32 x 7	18	—

Cuadro 2.2: Características físicas[20].

TelosB / Tmote Sky
TelosB [47] [48] [43] (también llamado Tmote Sky) es un nodo para WSN con un muy bajo consumo de energía diseñado para realizar un rápido prototipado de aplicaciones. TelosB cuenta con sensores de luz, temperatura y humedad. También cumple con el estándar IEEE 802.15.4, tiene una antena integrada *onboard* y proporciona una tasa de datos de 250 kbps utilizando un transmisor de radio inalámbrico de 2.4GHz ChipCon CC2420. Soporta el sistema operativo TinyOS. También tiene una interfaz USB para cargar programas y comunicarse con un PC.
Mica2/MicaZ
Mica2 y MicaZ [49] [50] son nodos sensores inalámbricos de muy bajo consumo de energía diseñado para crear rápidamente prototipos de aplicaciones. Tiene conectores de expansión para sensores de

(a) Nodo Tmote Sky[43] **(b)** Nodo Shimmer [44] **(c)** Nodo IRIS [45]

(d) Nodo Sun SPOT [46]

Figura 2.6: Plataformas de nodos sensores

temperatura, humedad, luz, presión, aceleración y movimientos sísmicos, sensores magnéticos, acústicos y otros. Ambas plataformas cuentan con un procesador Atmel ATMega128L y son alimentadas por dos baterías de tipo AA. Soporta el sistema operativo TinyOS. MicaZ tiene un transmisor de radio CC2420, cumple con el estándar IEEE 802.15.4 y opera en la banda de frecuencia de 2.4 a 2.48 GHz. Mica2 tiene un trasmisor de radio CC1000, este no es compatible con el estándar IEEE 802.15.4, sino que envía un flujo de bits en modulación FSK con codificación Manchester. Opera en la banda de frecuencia de 300 a 1000 MHz [51] [52].

Shimmer

Los nodos Shimmer (*Sensing Health with Intelligence, Modularity, Mobility, and Experimental Reusability*) [44] soportan comunicaciones a través de Bluetooth y IEEE 802.15.4. Utiliza baterias recargables integradas y puede utilizarse una tarjeta de memoria micro SD como almacenamiento local. También utiliza el sistema operativo TinyOS. Estos nodos son utilizados para aplicaciones de monitorización de variables fisiológicas y de actividad. Siendo actualmente utilizados para cuidado de la salud, rehabilitación, estudios biomecánicos en deporte, control del desempeño en atletas, monitorización remoto de pacientes, etc. Puede ser integrado con distintos módulos sensores, por ejemplo, puede ser conectado a giróscopos, magnetómetros, electrocardiograma (ECG), electromiograma (EMG), conducción en la piel (GSR), GPS, temperatura, etc

IRIS

Esta plataforma cuenta con [45] un procesador Atmel ATmega1281 y un trasmisor de radio de 2.4 Ghz, compatible con IEEE 802.15.4, logrando tasas de trasmisión de datos de hasta 250 kbps. En ambientes al aire libre nodos separados hasta 500 metros pueden comunicarse entre sí sin amplificación. Utiliza dos baterías AA. Soporta el sistema operativo TinyOS. Se puede integrar con sensores para medir temperatura, luz, presión barométrica, aceleración, movimientos sísmicos, fenómenos acústicos y electromagnéticos, etc.

Sun SPOT

Este nodo cuenta con un procesador de 400MHz y un trasmisor de radio CC2420 de 2.4 GHz el cual se accede a través de un segundo bus SPI. Esta integrado con IEEE 802.15.4. Cuenta con una interfaz USB y baterías recargables 770 mAh lithium-ion. Dentro de los sensores disponibles podemos encontrar acelerómetros, sensores de temperatura y de luz, etc. Puede ser programado utilizando Java.

CAPÍTULO

3

Seguridad en redes de sensores inalámbricos

3.1 Introducción

En este capítulo se establecen cuales son las características o atributos de seguridad deseados en una WSN. Luego se destacan las amenazas de seguridad presentes en este tipo de redes. Para ello se presenta una revisión de los principales ataques clasificándolos según la capacidad del atacante y la pila de protocolos. Una selección de los ataques aquí presentados fueron implementados e incluídos en la herramienta desarrollada. Por lo que la comprensión de los mismos es importante para entender el trabajo presentado en los capítulos sucesivos. Este proyecto principalmente se centró en los ataques en capa de enlace (más específicamente en capa MAC) y en capa de red.

3.2 Atributos de Seguridad

Una WSN es un tipo especial de red, que comparte algunas características en común con una red de computadoras típica pero que también exhibe muchas características que son únicas. Los servicios de seguridad en una WSN deben proteger la información que se comunica a través de la red, los recursos y el mal comportamiento de los nodos. Los requisitos de seguridad más importantes en WSN se enumeran a continuación [53]:

- Confidencialidad: solo personas autorizadas tienen acceso a información sensible. Los mecanismos de seguridad deberían asegurar que un mensaje en la red no sea entendido por nadie excepto el receptor del mensaje.
- Integridad: solo personas autorizadas pueden modificar, borrar, crear o replicar información sensible. Este requerimiento previene que los datos expuestos sean alterados o destruidos de forma accidental o maliciosa.
- Autenticación: asegura que los nodos que se están comunicando son realmente quienes dicen ser. Un atacante puede inyectar un flujo de paquetes falsos por lo que es muy importante que el receptor tenga un mecanismo para verificar que los paquetes recibidos provengan realmente de quien dicen venir.

- Disponibilidad: La información es accesible y usable bajo demanda de entidades autorizadas, incluso aunque la red esté en presencia de un ataque interno o externo como por ejemplo de denegación de servicio.

- Frescura: Esto implica que los datos enviados sean recientes y que un atacante no pueda reenviar mensajes viejos. Este requerimiento es especialmente importante cuando los nodos dentro de la WSN usan claves compartidas ya que un potencial atacante puede realizar un ataque de repetición (donde mensajes viejos son guardados y luego repetidos por el nodo atacante). Generalmente, un *nonce* o contador específico de tiempo es agregado a cada paquetes para poder chequear que el mismo sea reciente.

- No repudio: consiste en tener información irrefutable y verificable por una tercera parte de modo que un emisor o un receptor no puedan negar el hecho de haber transmitido o recibido un mensaje.

- Trazabilidad: información de auditoría tiene que ser guardada y protegida para que acciones que afecten la seguridad del sistema puedan ser trazadas hasta el responsable.

3.3 Ataques de seguridad en WSN

Existen una gran cantidad de ataques de seguridad que pueden ser utilizados para perjudicar las WSN, aquí presentamos un conjunto de estos ataques clasificándolos según las capacidades del atacante y en la pila de protocolos [54][55].

3.3.1 Basado en la capacidad del atacante

A continuación se presenta una clasificación de los ataques en WSN basada en la capacidad del atacante, es decir, si es externo o interno, si cuenta con dispositivos iguales o más potentes que el resto de los nodos de la red y si el ataque es activo o pasivo.

a. **Ataques internos VS. Ataques externos:** Los ataques internos ocurren cuando nodos legítimos de la WSN se comportan de forma no autorizada o no intencional. Mientras que los ataques externos son ataques que provienen de nodos que no pertenecen a la WSN [54]. Los ataques internos pueden ser montados a partir de nodos comprometidos los cuales corren código malicioso o a partir de atacantes que hayan robado las claves criptográficas, código y datos de nodos legítimos, que luego utilicen laptops o dispositivos equivalentes para atacar la red [55].

b. **Ataques con sensores VS. Ataques con laptop:** En un ataque con sensores el atacante cuenta con algunos nodos de capacidades similares a los que forman parte de la WSN. En los ataques con laptop el adversario cuenta con dispositivos más poderosos como una laptop u otro dispositivo equivalente. Estos dispositivos tienen un rango de transmisión mayor, más capacidad de procesamiento y más energía que los nodos de la WSN. Por lo tanto, un atacante de este tipo puede realizar más cosas de las que podría hacer con solo nodos sensores ordinarios [54]. Por ejemplo, un nodo solo podría interferir en las ondas de radio que están en su proximidad, mientras que un ataque con laptop puede crear interferencias en la WSN en su totalidad usando un transmisor más potente. Del mismo modo, un solo ataque de laptop podría escuchar toda la red, mientras que un solo sensor tendría un rango limitado. Por otro lado, un ataque de laptop podría tener mayor ancho de banda y un canal de comunicación de baja latencia, los cuales no están disponibles en nodos comunes, y que permiten al atacante coordinar sus esfuerzos [55].

c. **Ataques activos VS. Ataques pasivos:** Ataques activos involucran modificaciones en el flujo de datos o la creación de un flujo de datos falsos. Los ataques pasivos incluyen escuchas y monitorización de paquetes dentro de la red. Por ejemplo, un atacante con una antena receptora poderosa puede fácilmente recolectar todo el flujo de datos, pudiendo ubicar físicamente a los nodos si los mensajes contienen esta información o teniendo acceso al contenido de los mensajes [56]. También se pueden realizar ataques de análisis de tráfico, en una WSN todo el tráfico se dirige a una estación base, por lo que las comunicaciones siguen un patrón de muchos-a-uno o muchos-a-pocos. Un adversario puede obtener información de la topología de red así como también de la ubicación de la estación base y otros nodos estratégicos observando el tráfico y ciertos patrones [1]. Deng et al. [57] muestra dos ataques que pueden identificar la estación base de una red con alta probabilidad incluso sin conocer el contenido de los paquetes, es decir, con los datos cifrados. Los ataques que presenta son de dos tipos: *rate monitoring attack* y *time correlation attack*. En el ataque *rate monitoring* el adversario monitoriza la tasa de envío de paquetes de los nodos que tiene en su proximidad y se va moviendo más cerca de los nodos que tienen tasa de envío más alta. Esto se basa en el principio de que nodos más cerca de la estación base reenvían más paquetes que los nodos que están lejos de la estación base. En el ataque de tipo *time correlation*, un adversario genera eventos físicos que están siendo monitorizado por los sensores y monitoriza a que nodos se envían sus paquetes. Para ello el atacante observa la correlación entre el tiempo de envió de un nodo a su vecino, el cual se supone que debe reenviar el mismo paquete. De este modo se puede deducir el camino siguiendo cada operación de reenvió mientras el paquete se propaga por la red hasta llegar a la estación base. En [58] se proponen métricas para medir el nivel de anonimato de la estación base. Además existen varias propuestas [59] [60] [61] para aumentar el anonimato de la estación base frente a ataques de análisis de tráfico.

3.3.2 Basado en la pila de protocolos

A continuación se presenta una clasificación de los ataques en WSN basada en la pila de protocolos. Para ello se dividen los posibles ataques en las capas: física, enlace, red, transporte y aplicación.

3.3.2.1 Capa física

La capa física es responsable de seleccionar la frecuencia, de detectar las señales, modularización y cifrado de los datos [53].

i. **Manipulación:** La manipulación física puede ser dividida en dos tipos [62]:

A. Ataques invasivos: En este caso el adversario gana completo control sobre un nodo sensor accediendo de forma directa y física al mismo, de este modo el adversario puede fácilmente extraer las primitivas criptográficas y obtener acceso ilimitado a la información almacenada en la memoria, destruir el nodo, modificar el código de los programas o incluso reemplazarlo con un nodo malicioso con el potencial de causar un daño substancial a todo el sistema. Este tipo de ataque es fácil de realizar dado que los nodos no están físicamente protegidos ya que generalmente se encuentran en un ambiente al aire libre. Algunas contramedidas para este ataque pueden encontrarse en el Anexo A.6.1.

B. Ataques no invasivos: En este tipo de ataques los nodos no son desarmados, un ejemplo es el ataque de canal lateral que se refiere a cualquier ataque que se basa en la información recopilada a partir de la implementación física de un criptosistema, en contraste con las vulnerabilidades en el algoritmo sí mismo [62]. Por ejemplo, el atacante controla el consumo de energía o la emanación electromagnética de dichos dispositivos criptográficos, y luego analiza los datos recogidos para extraer la clave de cifrado. Ataques de este tipo son [1]:

- Análisis simple de potencia (SPA, *Simple Power Analysis*): esta es una técnica que implica interpretar el consumo de energía medido durante operaciones criptográficas. El análisis realizado puede proporcionar información acerca de las operaciones realizadas por el dispositivo así como de la clave utilizada.
- Análisis de poder diferencial (DPA, *Differential Power Analysis*): un adversario puede monitorizar la energía consumida por dispositivos criptográficos y luego analizar estadísticamente los datos recogidos para extraer la clave.
- Análisis simple electromagnético (SEMA, *Simple Electromagnetic Analysis*): con esta técnica un adversario es capaz de extraer información comprometedora de una sola muestra electromagnética.
- Análisis electromagnético diferencial (DEMA, *Differential Electromagnetic Analysis*): el atacante monitoriza las emanaciones electromagnéticas de los dispositivos criptográficos, y luego se utiliza el mismo análisis estadístico que en DPA sobre los datos electromagnéticos recogidos.

Algunas contramedidas para este ataque pueden encontrarse en el Anexo A.6.2

ii. **Interferencias (*Jamming*):** Este es un tipo de ataque de denegación de servicio [1] , que ocurre debido a una debilidad inherente a la capa física que afecta directamente al medio de transmisión y puede ser tanto intencionado como accidental. Este ataque consiste en la transmisión de señales de radio que interfieren con la frecuencia de radio que la red de sensores está utilizando. En entornos industriales, la posibilidad de interferencia debido a maquinaria y otros dispositivos es muy elevada. Por lo que es importante implementar un mecanismo que garantice un medio libre de ruidos. Por otro lado, la emisión intencionada de interferencias puede provocar que se agoten las baterías de los nodos debido al intento de retransmisión de las tramas [63]. Los ataques de este tipo pueden ser constantes (corrompiendo los paquetes mientras estos son transmitidos), falsos (envían un flujo constante de bytes y lo hacen parecer tráfico legítimo), aleatorio (aleatoriamente se alternan períodos de interferencia y de descansos para ahorrar energía) y reactivo (se transmite una señal de interferencia cuando se detecta tráfico) [54].
Los efectos de este ataque puede dejar completamente inutilizable una red. Law, et al. [64] simula un ataque de interferencia utilizando una red montada en la misma área que la red víctima donde logran eliminar el 100 % de los mensajes, es decir, no hay comunicación posible entre los nodos y la estación base cuando el atacante está activo.
Existen varias propuestas para contrarrestar los efectos de este ataque, algunas de ellas se pueden encontrar en el Anexo A.6.3.

iii. **Escuchas:** Este ataque consiste en que receptores no deseados intercepten y tengan acceso a mensajes que corresponden a otro destino. Las comunicaciones inalámbricas son suceptibles a este tipo de ataque debido a la difusión de los mensajes. Para ello basta con que el atacante sintonice la frecuencia adecuada [65]. Un atacante con una antena poderosa y bien diseñada puede escuchar un flujo de datos. Luego podría obtener la localización física de los nodos para destruirlos, observar datos especifícos de la aplicación, etc. Para minimizar esta amenaza fuertes técnicas de cifrado deben ser usadas [56].

[1] **Ataque de denegación de servicio (*Denial of Service* - DoS):** Un ataque DoS incluye cualquier evento que disminuye o elimina la capacidad de una red para realizar su función esperada por fallos de hardware, errores de software, agotamiento de recursos, señales maliciosas de radiodifusión de alta energía, etc. Los sistemas de comunicación podrían quedar completamente inutilizable si el ataque tiene éxito [1]. El ataque DoS más simple trata de agotar los recursos disponibles en el nodo víctima enviando paquetes innecesarios e impidiendo que usuarios legítimos de la red puedan acceder a los servicios o recursos [6].

3.3.2.2 Capa enlace

La capa de enlace es responsable de multiplexar los flujos de datos, detección de las tramas de datos, acceso al medio y control de errores [53].

i. **Colisión:** Una colisión ocurre cuando dos nodos intentan transmitir simultáneamente en la misma frecuencia, teniendo como consecuencia que cuando paquetes colisionan estos deben ser descartados. Esto sucede debido a que los datos del paquete cambian causando errores en la suma de verificación [54]. Un ataque de colisión puede ser fácilmente logrado usando un nodo comprometido o un nodo hostil ya que basta con que ese nodo no siga con el protocolo de control de acceso al medio y cause colisiones con las transmisiones vecinas enviando paquetes cortos y ruidosos. Este ataque no consume mucha energía en el atacante pero puede causar grandes problemas a la red. Una defensa típica para este ataque es utilizar códigos de corrección de errores. Sin ambargo, estos códigos agregan procesamiento adicional y sobrecarga en la comunicación.

ii. **Agotamiento o acceso continuo al canal:** Algunos protocolos de capa de enlace tratan de retransmitir repetidamente ante una colisión. Un nodo malicioso puede interrumpir continuamente la comunicación entre dos nodos forzando al emisor a retransmitir continuamente. Esto puede provocar posposición indefinida en el uso del canal y además consume la energía de los nodos que intentan retransmitir reiteradamente [54]. Algunas medidas para mitigar este ataque son controlar la admisión MAC, así la red puede ignorar solicitudes excesivas y prevenir el gasto de energía debido a repetidas transmisiones. Otra técnica consiste en utilizar multiplexación por división de tiempo donde cada nodo es asignado a una partición del tiempo en el cual puede transmitir. Sin embargo, esto es aún vulnerable a colisiones.

iii. **Interrogación (*Interrogation*):** Este tipo de ataque explota el *handshake* RTS/CTS (*Request To Send/ Clear To Send*) usado por varios protocolos MAC para mitigar el problema de la estación oculta. Un atacante puede agotar los recursos de un nodo enviando repetidamente mensajes RTS para producir respuestas CTS de un nodo vecino objetivo. Para prevenir este tipo de ataque un nodo puede limitar la cantidad de conexiones que acepta de una misma identidad o utilizar protección anti repetición y autenticación fuerte en la capa de enlace [54].

iv. **Ataque de denegación del modo dormido (*Denial of Sleep Attack*):** Debido a las restricciones de energía que existen en los nodos sensores, es usual que los nodos periódicamente cambien a modo dormido, de forma de extender el tiempo de vida de la red. Esta forma de conservar energía se basa principalmente en los protocolos MAC, ya que una de las principales razones por la que se pierde energía es debido a las sobre-escuchas, lo que significa que un nodo recibe paquetes que están destinados a otros nodos, teniendo su transmisor de radio mayor tiempo prendido. La mayoría de los protocolos MAC de las WSN tratan de reducir estas sobre-escuchas tratando de asegurarse de que un nodo solo está despierto cuando existe el tráfico destinado para él [66].

   Un ataque de denegación de modo dormido provoca que los nodos permanezcan despiertos por mayores intervalos de tiempo, donde se suponía que estos deberían de estar dormidos [67]. El transmisor de radio consume la mayor parte de la energía de un nodo, por lo que se explota, la funcionalidad receptora de este haciendo que se agoten las baterías y acortando el tiempo de vida de toda la red. Un ataque de este tipo podría reducir el tiempo de vida de la red de años a tan solo días si afecta a un gran porcentaje de los nodos de la red o a unos pocos nodos críticos [68].

   El ataque de interrogación presentado anteriormente puede ser visto como un tipo de este tipo de ataque, otra técnica basada en los protocolos de capa MAC para evitar que un nodo entre en estado dormido es colocar una portadora en el canal para que el nodo víctima crea que a continuación se enviará un paquete válido.

   Los ataques de denegación de modo dormido también pueden darse en otras capas, como por ejemplo utilizando interferencias en capa física o ataques de repetición [66]. Algunas técnicas para prevenir este ataque son: utilizar autenticación a nivel de capa de enlace y protección anti-repetición [67].

3.3.2.3 Capa red

Los ataques sobre la capa de red son llamados ataques de ruteo. En una WSN cada nodo actúa como un *router* por lo que las funciones de enrutamiento y reenvío de datos son tareas importantes para los nodos sensores. Los protocolos de enrutamiento tiene que ser eficientes en cuento al uso de energía y memoria, pero el mismo tiempo tienen que ser robustos frente a ataques y fallos en los nodos, lo que implica que cada receptor autorizado debe recibir todos los mensajes que le fueron enviados y debe ser capaz de probar la integridad de cada mensaje así como también la identidad del emisor. Algunos ataques en esta capa son los que se mencionan a continuación.

i. **Ataque Blackhole:** En este ataque el nodo malicioso descarta los mensajes que recibe de sus nodos hijos, como puede observarse en la figura 3.1. Si se realiza esto con todos los paquetes se puede dejar inalcanzable un destino o la red inutilizable [1] [6]. Algunas de las contramedidas propuestas para este tipo de ataque se pueden encontrar en el Anexo A.6.4.

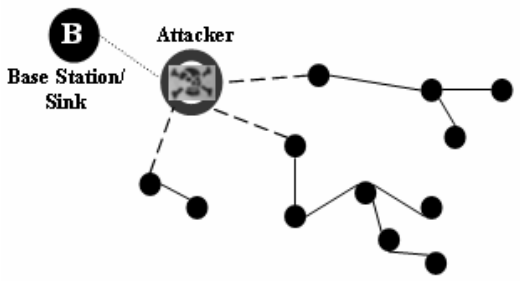
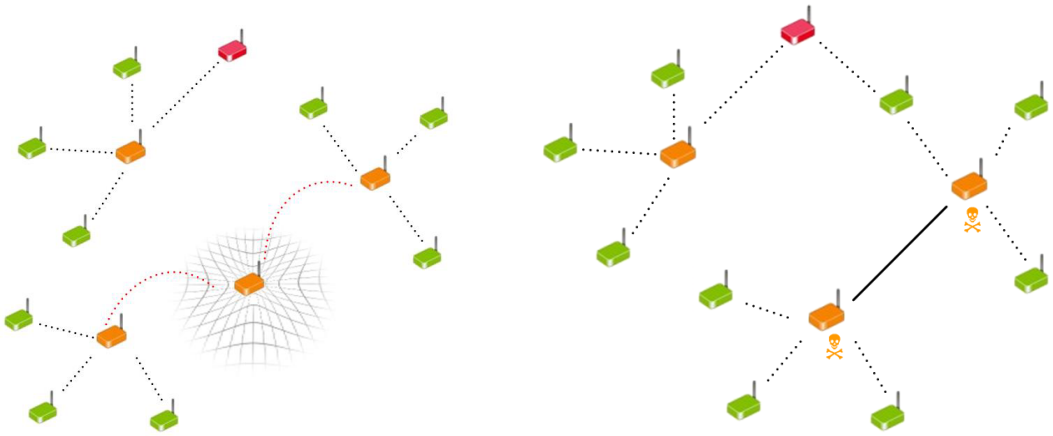


Figura 3.1: Ataque *blackhole* [6].

ii. **Ataque de falsificación, alteración, o repetición (*Spoofed, altered, or replayed attack*):** En este ataque un adversario puede guardar mensajes viejos y luego re enviarlos. Como los mensajes son viejos estos pueden, por ejemplo describir una topología que ya no existe más y provocar que los nodos actualicen su tabla de ruteo con datos falsos. Un atacante podría de este modo crear *loops* de ruteo, atraer o alejar tráfico hacia cierta área de la red, generar mensajes de error falsos, particionar la red, incrementar la latencia extremo a extremo, etc [55].

iii. **Ataque de reenvío selectivo (*Selective forwarding*):** En este ataque un nodo malicioso puede descartar de forma selectiva ciertos paquetes. Este ataque es más efectivo cuando los nodos atacantes están dentro de la ruta del flujo de datos [1]. El ataque blackhole es un tipo de este ataque donde todos los paquetes son descartados. Este ataque puede ser selectivo, es decir descartar paquetes para determinado destino, descartar paquetes cada cierto intervalo de tiempo o eliminar paquetes de forma aleatoria, descartar los paquetes que corresponden a un nodo específico o los que son de determinado tipo. Dentro de las medidas presentadas para afrontar este tipo de ataques se encuentran las mencionadas en el Anexo A.6.5.

iv. **Ataque *sinkhole*:** En este ataque se intenta atraer tanto tráfico como sea posible hacia nodos comprometidos. En la figura 3.2a se observa como el nodo señalado con la grilla atrae el tráfico. Estos ataques intentan engañar a los nodos vecinos de un nodo comprometido haciéndolo parecer atractivo con respeto al algoritmo de ruteo, convirtiéndose en un sumidero de información. Este ataque normalmente facilita el inicio de otros ataques sobre el tráfico enrutado por los nodos bajo el dominio del atacante. Por ejemplo, este tipo de ataque permite mejorar el efecto de los ataques de reenvío selectivo ya que el tráfico de una gran área de la red pasa por nodos comprometidos. Pueden ser muy difíciles de detectar, sobre todo si se hace una selección de la información que es retenida [1]. En el Anexo A.6.6 se mencionan algunas contramedidas para este ataque.



(a) Ataque de sinkhole [7] **(b)** Ataque de wormhole [7]

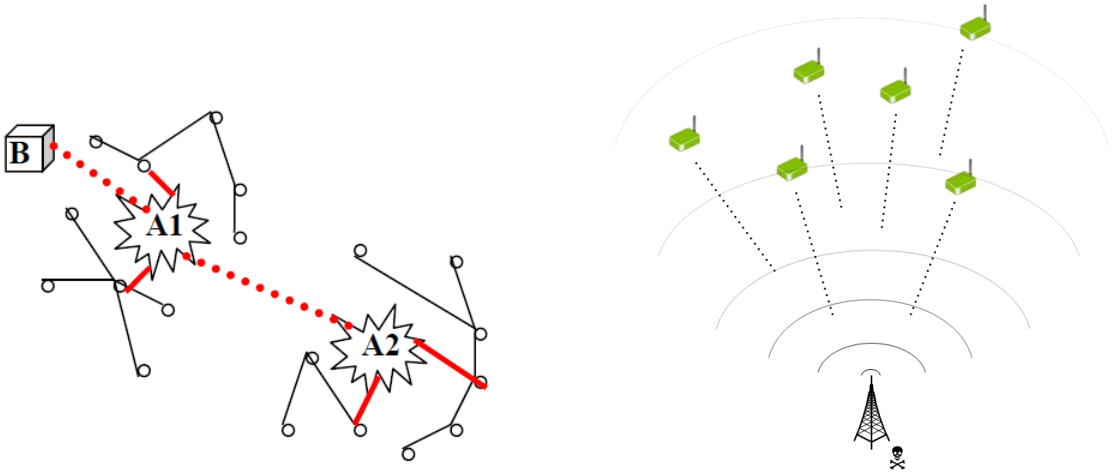
Figura 3.2: Ataques

v. **Ataque wormhole:** este ataque consiste en guardar tráfico de una región de la red y luego re enviarlo en una región diferente. Esta retransmisión de paquetes puede ser selectiva. Este ataque es difícil de tratar porque el adversario no tiene que comprometer ningún nodo, sino que puede usar una laptop u otro dispositivo de red. Sin embargo, este tipo de ataques generalmente involucra dos nodos maliciosos distantes que se envían paquetes utilizando otro canal (canal *wormhole*) disponible solo para el atacante como puede verse en la figura 3.2b. Un nodo guarda todos los paquetes que le llegan, luego se los reenvía al otro nodo atacante por el canal *wormhole* y este los retransmite en la red. Los atacantes se comunican de forma que los nodos de la red no puedan detectarlos, por ejemplo con una conexión cableada o una transmisión inalámbrica fuera de banda [69].

   Un atacante situado cerca de la estación base puede interrumpir las funciones de ruteo creando un *wormhole* estratégicamente ubicado, el cual puede convencer nodos que están a varios saltos desde la base que están solamente a uno o dos saltos de distancia. Esto también puede crear un ataque de *sinkhole* ya que otro nodo malicioso puede artificialmente proveer un "mejor" ruteo hacia la estación base, de este modo todo el tráfico del área va a ser enviado por esa ruta y no por las otras menos atractivas. En la figura 3.3a, extraída de [56], se presenta el siguiente ejemplo de un ataque de *wormhole* utilizado para crear un ataque de *sinkhole*. En la figura B es la estación base, mientras que A1 y A2 son los atacantes. Los nodos cerca de A2 creen que la estación base está más cerca a través del *wormhole* con A1. Dos nodos distantes pueden creer que son vecinos a través de la retransmisión de paquetes entre ellos. Más generalmente, los *wormholes* pueden ser utilizados para explotar condiciones de carrera en los algoritmos de enrutamiento. Una condición de carrera generalmente surge cuando un nodo toma una acción basado en la primer instancia del mensaje que recibe y luego ignora los mensajes que le llegan más tarde. Este caso, un atacante puede influir en la topología resultante. Por ejemplo, si puede provocar que los nodos reciban cierta información de ruteo antes de lo que normalmente la recibirían si esta sigue el camino multi-salto que debería. Los *wormholes* pueden lograr esto, incluso aunque la información de enrutamiento este cifrada y autenticada [55]. Su detección es potencialmente difícil cuando es usado en conjunción con un ataque *sybil* (este ataque será presentado más adelante).

   En el Anexo A.6.7 se presentan técnicas de detección para este tipo de ataque.

vi. **Ataque *hello flood*:** Muchos protocolos que utilizan paquetes *HELLO* asumen que recibir un paquete implica que el emisor está dentro del rango de radio y por lo tanto es un vecino. En este tipo de ataque, el atacante envía un mensaje *broadcast* a todos los nodos de la red con una antena de radio de largo alcance como puede verse en la figura 3.3b. Los nodos receptores de ese



(a) Ataque de wormhole para crear un sinkhole [56] **(b)** Ataque de hello flood [7]

Figura 3.3: Ataques

mensaje creen que la ruta a través del nodo enviando el *hello flood* es el camino más corto. Por lo que intentaran enviar mensajes a través de ese nodo, el cual probablemente, ni siquiera este dentro del rango de radio. También provoca el consumo de las baterías de las víctimas, ya que los nodos tratan de responder al anuncio, emitiendo señales al vacío. Este tipo de ataque puede dejar la WSN totalmente inutilizable [1].

En el Anexo A.6.8 se presenta una técnica de detección para este tipo de ataque.

vii. **Acknowledgement spoofing:** El objetivo de este ataque es hacer una suplantación o *spoofing* de un enlace caído o un nodo muerto usando los reconocimientos (ACKs) de la capa de enlace. Cuando el atacante escucha un paquete que es para un nodo que no puede responder contesta con un ACK como si fuera ese nodo, de esta forma logra hacerle creer a la víctima que el nodo está vivo.

viii. **Ataque de replicación de nodos:** En este ataque se intenta agregar un nodo a una red de sensores existente copiando o replicando el identificador de un nodo existente. El atacante también tiene que clonar otra información como por ejemplo las claves ya que de otro modo no podrá autenticarse como un nodo legítimo de la red [70]. Además se puede crear un gran número de réplicas atacantes esparcidas por la red que comparten la información extraída del nodo comprometido. Con un solo nodo comprometido se pueden crear tantos nodos como hardware disponible tenga el adversario [71]. Este ataque puede tener graves consecuencias ya que permite corromper paquetes, afectar el ruteo dentro de la red, crear lecturas de datos falsas, etc [63]. En el Anexo A.6.9 se presentan algunas contramedidas para este ataque.

ix. **Ataque sybil:** este ataque es definido en [72] como un nodo malicioso que ilegítimamente toma múltiples identidades. Esto puede observarse en la figura 3.4 donde el nodo malicioso toma tres identidades distintas. En muchos casos, los nodos de una WSN necesitan trabajar juntos para realizar una tarea, por lo tanto, subtareas son distribuidas entre los nodos [6]. Es esta situación un nodo puede pretender ser más de un nodo usando otras identidades y así dañar la red. Anunciando por ejemplo una identidad falsa o tomando la de otros nodos legítimos dentro de la red, invalidando la información de los nodos legítimos y modificando la información de ruteo.

Este ataque puede afectar a distintos protocolos tales como los que se presentan en el Anexo A.6.10. Toda red *peer-to-peer* (P2P), en especial, la redes inalámbricas ad hoc son vulnerables

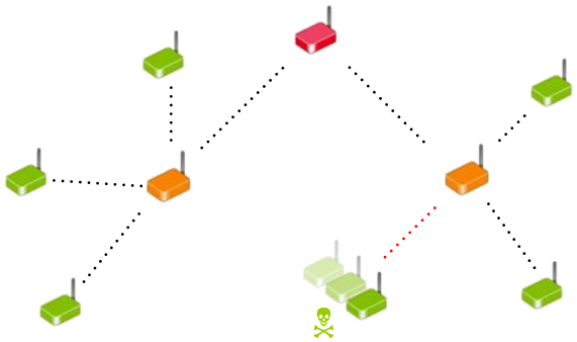


Figura 3.4: Ataque *sybil* [7]

a los ataques *sybil*. Sin embargo, como las WSN utilizan estaciones base este ataque puede ser prevenido utilizando protocolos eficientes [6]. En el Anexo A.6.10 se pueden encontrar algunas contramedidas para este ataque.

x. **Redirección:** En este ataque un nodo malicioso dentro de una ruta puede reenviar los paquetes que llegan en una dirección equivocada a través de la cual el destino sea inalcanzable [54].

xi. ***Homing***: Utilizando análisis de patrones de tráfico es posible identificar nodos con responsabilidades especiales por ejemplo la cabeza de un *cluster* o los nodos encargados de administrar las claves criptográficas. Luego otros ataques pueden ser realizados como por ejemplo interferencias o acceso físico para destruirlos. Para prevenir esto generalmente se utiliza cifrado en el cabezal [54].

xii. **Ataque *jellyfish*:** El ataque *jellyfish* demora los paquetes por un período de tiempo antes de reenviarlos. También puede cambiar el orden es que estos paquetes son recibidos y enviados de forma aleatoria, en vez de en un orden FIFO (*First In - First Out*) de los paquetes recibidos. Esto afecta los mecanismos de control de flujo realizado para realizar una transmisión confiable. Este ataque provoca un aumento significativo de la latencia extremo a extremo, degradando la calidad del servicio y afectando a las aplicaciones en tiempo real [2].

xiii. **Ataque *rushing*:** La multidifusión conserva el ancho de banda de la red enviando un solo flujo de datos a múltiples receptores. Los paquetes solo son duplicados en los puntos donde hay más de un camino a seguir. Por lo que un atacante puede explotar este mecanismo de supresión de duplicados reenviando rápidamente paquetes de descubrimiento de rutas de forma de ganar acceso dentro de una ruta [3]. Si el descubrimiento de ruta reenviado por el atacante es el primero en alcanzar cada nodo vecino de un nodo objetivo, estos vecinos descartarán los siguientes paquetes transmitidos por otros nodos. Para lograr reenviar rápidamente este paquete, al atacante puede saltarse pasos en el procesamiento del mensaje o en el reenvío que de otro modo debería hacer. Para ello podría, por ejemplo, utilizar una antena más potente para incrementar el rango de transmisión y alcanzar el nodo destino más rápido que con una ruta legítima.

xiv. ***Blackmailing*:** En un ataque de este tipo, los nodos atacantes acusan a un nodo inocente de ser un nodo malicioso. Este ataque puede ser efectivo contra protocolos distribuidos que establecen una lista que nodos buenos y malos basados en una evaluación de los nodos participantes en la red. Es decir, se hace una votación donde la mayoría decide si un nodo es malicioso y por tanto debe ser excluído de la red. Pero si los nodos maliciosos son suficientes o se utiliza un ataque *sybil* se puede acusar a un nodo bueno de malicioso o hacer que en la votación de un nodo malicioso de como resultado que ese nodo es legítimo en la WSN [2].

xv. ***Neighbor Attack*:** Cuando un nodo recibe un paquete, este escribe su identificador antes de reenviarlo al nodo siguiente. Un atacante, podría simplemente reenviar ese paquete sin escribir su identificado en ese paquete. Esto hace que dos nodos que no están dentro del mismo rango de comunicación crean que son nodos vecinos, resultando en una ruta falsa [3].

3.3.2.4 Capa transporte

Los problemas de seguridad en la capa de transporte se deben mayormente a fallas en los protocolos. Un diseño eficiente de los protocolos de capa de transporte pueden evitar amenazas.

i. **Flooding:** Un atacante crea nuevas conexiones reiteradamente hasta que se agoten los recursos requeridos por cada conexión o que se alcance el límite máximo de conexiones [54].

ii. **Ataque de desincronización:** En este ataque se envían repetidamente mensajes a uno o ambos nodos dentro de una comunicación, forzando a que estos soliciten retransmisiones de las tramas perdidas. Por lo que estos mensajes son nuevamente transmitidos. Esto provoca que los nodos no puedan intercambiar información, gastando sus reservas de energía [54].

3.3.2.5 Capa aplicación

i. **Ataque *Overwhelm*:** Un atacante puede intentar abrumar los nodos de la red con estímulos en los sensores, haciendo que la red reenvíe grandes volúmenes de tráfico a la estación base. Con esto se logra consumir el ancho de banda de la red y las reservas energéticas de los nodos [54].

ii. **Ataque de reprogramación:** Los sistemas de programación de la red permiten reprogramar los nodos remotamente. Si este proceso de reprogramación no es seguro un atacante puede utilizar este proceso para tomar control de una gran parte de la red [54].

iii. **Ataques a los protocolos de sincronización de reloj:** Los protocolos de sincronización de reloj proveen mecanismos para sincronizar los relojes locales de cada nodo dentro de la red de sensores. Esta sincronización es muy importante dentro de un conjunto de aplicaciones y servicios dentro de una red de sensores [62]. También son utilizados por ciertos mecanismos para detectar/evitar algunos ataques como por ejemplo el *wormhole*. Estos ataques pueden ser realizados capturando físicamente algunos nodos e inyectando mensajes falsos de actualización de sincronización de reloj.

iv. **Ataques de software:** En este tipo de ataque un adversario puede intentar modificar el código en la memoria o explotar vulnerabilidades conocidas en el código. Un ejemplo bien conocido de este tipo de ataque es un desbordamiento de *buffer* (*buffer overflow*). Un ataque de desbordamiento de *buffer* ocurre cuando un proceso intenta almacenar datos más allá de los límites de la longitud fija del *buffer*. Esto da lugar a que los datos adicionales sobrescriban las ubicaciones de memoria adyacentes [1]. Otros ataques de este tipo son la inyección de parámetros y los *memory leaks*. Los ataques de software pueden ocurrir fácilmente en TinyOS ya que la implementación actual del mismo no proporciona ningún control del acceso a memoria, es decir, no hay ninguna función para controlar qué usuarios o procesos acceden a qué recursos del sistema y qué tipo de derechos de ejecución tienen [62]. Para ver un estudio más en detalle de este tema y algunas contramedidas propuestas para brindar seguridad en TinyOS consultar el Anexo A.5.

CAPÍTULO

4

Infraestructura de pruebas

4.1 Introducción

En este capítulo se describe el entorno de trabajo utilizado durante el proyecto, incluyendo el hardware y software utilizados. También se presenta la herramienta construída, sus principales funcionalidades y la arquitectura de la misma.

4.2 Entorno de trabajo

En esta sección se presenta el hardware y software utilizado para el desarrollo y las pruebas.

4.2.1 Hardware

- 3 nodos TelosB
- 3 nodos Shimmer [1]
- Plataforma Shimmer SPAN
- Sniffer AVR de Atmel
- 3 Shimmer docks y sus respectivos cables USB
- Una placa para conectar los nodos TelosB al puerto USB de la computadora

Las especificaciones de los nodos TelosB y Shimmer se mencionaron en la sección 2.5.

Plataforma Shimmer SPAN

La plataforma SPAN permite incorporar, en un dispositivo que cuente con puerto USB, tecnologías de radio de bajo consumo a través del módulo de radio 2.4 GHz IEEE 802.15.4. Este dispositivo se ilustra en la figura 4.1, cuenta con un procesador MSP439 y un transmisor de radio CC2420. Soporta un *stack*

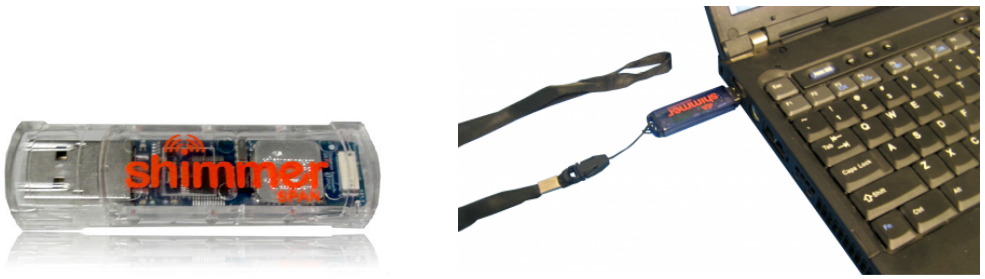


Figura 4.1: Plataforma Shimmer SPAN [8]

Figura 4.2: Shimmer Dock [9]

de radio *IP-over-802.15.4* y el *stack* de radio CC2420 de TinyOS, por lo que se puede embeber con código implementado en NesC de forma análoga a como se realiza con los nodos sensores.
En los experimentos realizados, se utilizó este dispositivo para poder comúnicar una laptop con los nodos sensores utilizando IEEE 802.15.4. Este dispositivo es el que se programará para ser el atacante en las pruebas realizadas. Sin embargo, también podría haber sido un nodo común conectado al puerto USB.

Sniffer AVR de Atmel

Este dispositivo es específico para realizar escuchas de paquetes IEEE 802.15.4 y Zigbee, por lo que se utilizó para realizar las pruebas sobre el protocolo GTS. En conjunto con la herramienta Wireshark se capturaron y analizaron las tramas enviadas por este protocolo y el nodo atacante.

Shimmer Dock y placa para TelosB

El Shimmer Dock se muestra en la figura 4.2. Este permite prender, apagar y resetear un nodo sensor Shimmer. También, a través de este dispositivo, se pueden programar, cargar su batería y conectarlos a una computadora a través de un cable USB. Cuando un nodo Shimmer tiene una tarjeta microSD y se conecta a una computadora a través del Shimmer Dock, el sistema operativo de la computadora instala la tarjeta flash de almacenamiento como se hace comunmente con cualquier otra memoria flash USB. El Shimmer Dock también contiene un módulo GPS, que provee la localización precisa y una medida del tiempo global sincronizado. Esta información se la envía al nodo sensor a través del conectector UART, de forma que el nodo pueda utilizar esta información para sincronización o localización.
De forma similar, se utilizó una placa para poder conectar los nodos TelosB al puerto USB de la computadora y así poder programarlos. Este tipo de nodos se alimentan a través de dos pilas de tipo AA por lo que no se requiere un dispositivo adicional para cargarlos.

[1] Los drivers de los nodos fueron extaídos de [73]

4.2.2 Software

Dentro de esta sección se detallan los componentes software utilizados: herramientas, protocolos, lenguajes de programación y simuladores.

Herramientas

- *Wireshark*: software que permite analizar tráfico y capturas de paquetes. Disponible en [74].
- *Base Station*: aplicación escrita en NesC que viene por defecto en TinyOS que recibe un paquete a través de la radio y lo retransmite a través del puerto USB al que está conectado el nodo. Esta aplicación también retransmite la comunicación inversa, es decir, recibe paquetes a través del USB y los envía utilizando el transmisor de radio.
- Máquina virtual Ubuntu: Para desarrollar se utilizó una maquina virtual brindada por el proveedor de los sensores Shimmer, equipada con una versión de Ubuntu como sistema operativo y con todas las herramientas necesarias. La versión de Ubuntu es la 10.04.4. El desarrollo realizado fue sobre la versión 2.x de TinyOS.

Protocolos

- CTP
- CSMA-CA
- GTS

Lenguajes de programación utilizados

- Java v1.6
- NesC v1.3.2

Simuladores

- Cooja
- Avrora v1.7.115

Los simuladores Cooja y Avrora se encuentran disponibles en [37] y [75] respectivamente.
Dentro de las funcionalidades de la herramienta construida se agregó la capacidad de analizar las tramas del simulador Avrora. Esto fue utilizado para probar los ataques a los protocolos de enrutamiento. Esto se debe a que los nodos físicos con los que contabamos no eran sufiecientes para realizar las pruebas necesarias. Para la elección del simulador se consideraron y probaron la mayoría de los simuladores actualmente disponibles, algunos de ellos fueron NS-2, TOSSIM, OMNeT++, J-Sim, ATEMU, JProwler. Sin embargo, no se encontró ninguno que permitiera al mismo tiempo realizar lo siguiente: ejecutar código TinyOS, ejecutar códigos diferentes en distintos nodos, ver los paquetes enviados, realizar filtrado de los paquetes y ver un grafo con los enlaces de comunicación utilizados durante la simulación. Este último punto es necesario para observar la topología de la red y evaluar los cambios producidos por cada uno de los ataques.
Los simuladores que cumplen con la mayoría de esos requerimientos son Cooja y Avrora. Sin embargo, Cooja no permite filtrar los paquetes capturados, por lo que en el grafo de la topología se muestran tanto los paquetes de datos como los de enrutamiento. Dado que los paquetes de enrutamiento se envían por *broadcast* en el grafo resultante no se puede observar cuál es realmente la estructura de la red y cuál es

el padre elegido por cada nodo. Por otro lado, el simulador Avrora no tiene interfaz gráfica ni filtrado de paquetes. En las simulaciones se muestra un *dump* del contenido de las tramas que fueron enviadas. Dada esta limitante para realizar las pruebas sobre los ataques a los protocolos de capa de red implementados, se decidió integrar la herramienta que se iba a implementar con el simulador Avrora. De esta forma, se permite recibir la salida de una simulación para visualizarla de una forma más amigable, realizar filtrados potentes sobre el contenido de las tramas (filtros típicos como por ejemplo, nodo origen o destino y filtros complejos como por ejemplo, expresiones regulares). También se puede visualizar un grafo de la estructura de la red considerando los paquetes capturados y los fitros configurados.
De todas formas, el simulador Cooja fue utilizado para realizar algunas pruebas sobre la implementación de los ataques al protocolo GTS.

4.3 Herramienta construída

En esta sección se describen las principales funcionalidades de la herramienta construída y la arquitectura de la misma.

4.3.1 Funcionalidades

En la figura 4.3 se muestra un diagrama de lo que sería el funcionamiento general de la herramienta. Una computadora con Java y un nodo con TinyOS que transmita utilizando el estándar IEEE 802.15.4 forman lo que sería el atacante. La herramienta es ejecutada en la computadora y dependiendo de la funcionalidad elegida, modifica el software del nodo apropiadamente para poder atacar a una red de sensores desplegada físicamente con dispositivos reales.

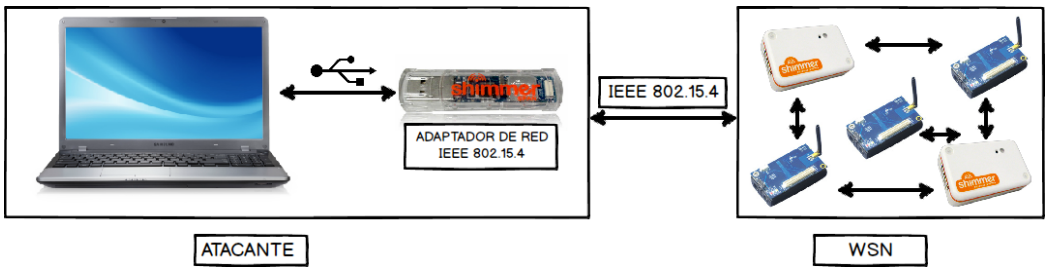


Figura 4.3: Diagrama de comunicación de la herramienta.

Las principales funcionalidades incluidas en esta herramienta son:

- *Sniffer* y análisis de datos capturados
- Análisis de tramas obtenidas mediante simulaciones con Avrora
- Ataques en capa MAC
- Ataques en capa de red

Una de las amenazas a la confidencialidad de las WSN es el análisis del tráfico y la detección de puntos claves de la red. Por tal motivo, la herramienta de ataque cuenta con un *sniffer* para escuchar el tráfico de red. También permite analizarlo mediante filtros complejos sobre la carga útil de los paquetes y visualizar el grafo de la topología de la red sobre el que viajan los paquetes. De esa manera un adversario puede procesar paquetes transmitidos con el fin de extraer información vital, como el identificador de un nodo o datos de la topología. En este caso, la información de red recolectada depende del adaptador de red utilizando. Si se emplea un dispositivo con una antena potente se podrá

capturar una mayor cantidad de tráfico y se tendrá más información sobre la red. Si el adaptador de red no es tan potente, solo se tendrá conocimiento de una porción más pequeña de la red. Con la herramienta también se puede obtener información de la cantidad de paquetes enviados a través de cada uno de los enlaces de la red. Por lo que esto permite observar patrones de tráfico de la red de sensores, por ejemplo, para deducir la ubicación de la estación base o otros nodos situados estratégicamente dentro de la red.
A nivel de capa MAC se permiten realizar ataques de denegación de servicio para los protocolos CSMA/CA y GTS. A nivel de la capa de red se implementaron los siguientes ataques: repetición, *sinkhole, blackhole, selective forwaring, sybil, ack spoofing* y *jellyfish*.

4.3.2 Arquitectura

En la figura 4.4 se puede observar la arquitectura de la herramienta construida. En la PC corre código implementado en Java, mientras que en el adaptador de red IEEE 802.15.4 corre código implementado en NesC. Estos dos componentes se comunican a través del puerto USB. La aplicación Java que se ejecuta en la PC se compone de dos móludos principales: el módulo controlador de ataques y el módulo encargado del *sniffing* y el análisis de datos. También se tiene una capa de presentación con una interfaz gráfica de tipo *swing* y el controlador del adaptador de red. Este último es el que se encarga de actualizar el software del adaptador de red IEEE 802.15.4 según la funcionalidad que se desee ejecutar.
En el adaptador de red IEEE 802.15.4, para las funcionalidades de análisis de datos se instala la aplicación *Base Station* y se utilizan los protocolos de red por defecto de TinyOS. En el caso de los ataques, en el adaptador de red se instala código malicioso para poder atacar a la red víctima. En general este código se construye a partir de la implementación por defecto de TinyOS de los protocolos MAC o de red (según el ataque seleccionado) que fueron modificados para explotar alguna vulnerabilidad detectada. A su vez, para cada ataque se crea una aplicación sencilla, que haga uso del protocolo modificado.

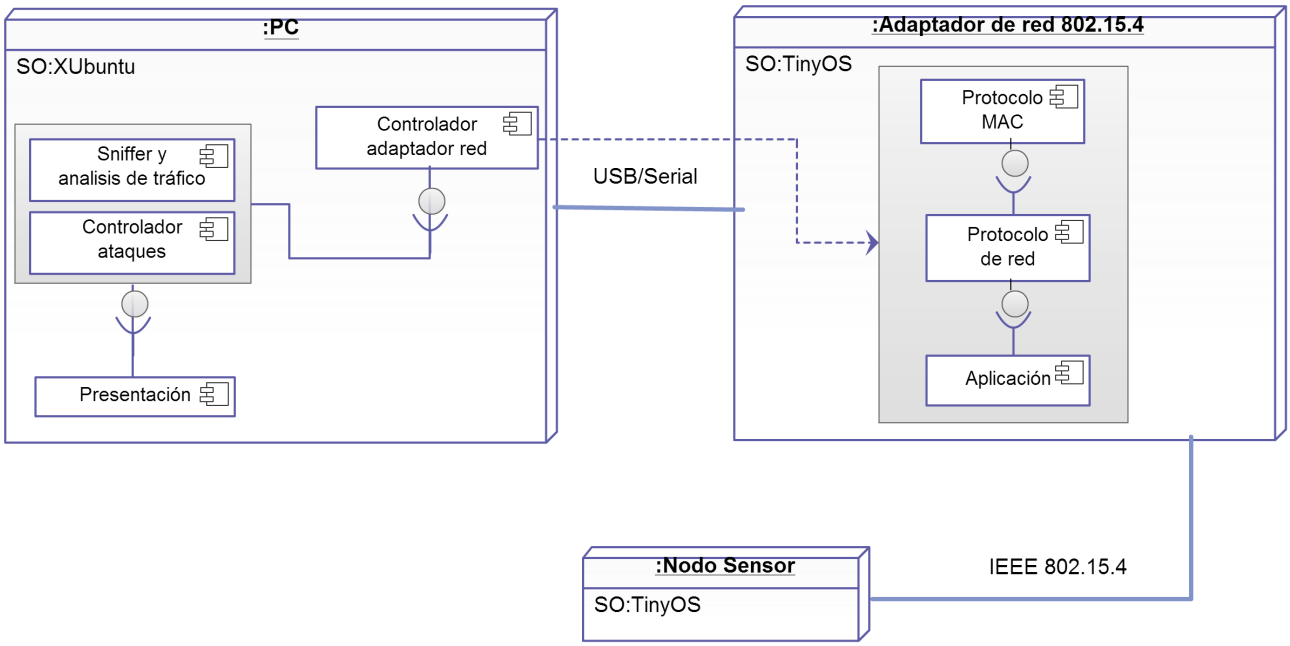


Figura 4.4: Diagrama de despliegue.

CAPÍTULO

5

Ataques en capa MAC: GTS y CSMA/CA

En esta sección se presentan dos protocolos de acceso al medio utilizados por el estándar IEEE 802.15.4: CSMA/CA y GTS. A modo de introducción se presenta en la sección 5.1 algunos aspectos generales del estandar IEEE 802.15.4 que involucran a los protocolos CSMA/CA y GTS. Luego, en la sección 5.2 se describe el funcionamiento del protocolo CSMA/CA, se explican las vulnerabilidades encontradas, como se pueden explotar y como se llevó a cabo el ataque. De forma análoga se realiza lo mismo para el protocolo GTS en la sección 5.3.
Los ataques implementados en este caso no se pueden utilizar en el controlador de radio CC1000 ya que este no implementa el estándar IEEE 802.15.4. Sin embargo, el controlador de radio CC2420 si se basa en este estándar. Por otro lado, las pruebas fueron realizadas sobre nodos reales de las plataformas TelosB y Shimmer.

5.1 Estándar IEEE 802.15.4

El estándar IEEE 802.15.4 define la capa física y la capa MAC. Este es comúnmente usado como protocolo de comunicación de baja velocidad de datos, bajo consumo de energía.
En el estándar [76] se definen tres tipos de nodos que pueden utilizarse dependiendo de la topología de la red. Los tipos de nodos del estándar son los que se presentan a continuación.

- *Coordinador PAN (Personal Area Network)*, es el responsable de gestionar la actividad de la red. Estos nodos suelen requerir un consumo mayor de energía que los demás nodos por lo que se suelen conectar a la red eléctrica.
- *Coordinador*, se utilizan en topologías tipo *cluster-tree*. En este tipo de redes existen tres niveles de jerarquías, por cada *cluster* hay un coordinador y además existe un coordinador PAN que es el coordinador de toda la red.
- *Nodo simple*, son los dispositivos más simples y menos costosos. Tienen bajos consumos de energía.

Dado que el trabajo realizado se centrará en aspectos relacionados con la capa MAC, la descripción del estándar se centrará en este punto dejando de lado los aspectos definidos para la capa física.

El protocolo MAC soporta dos modos de operación:

- ***Non Beacon-enabled mode***: Cada nodo simplemente utiliza CSMA/CA (no ranurado) para enviar sus tramas. Este modo tiene la ventaja de su simplicidad, lo que lo hace más escalable. Pero tiene entre sus desventajas que no puede dar garantías de tiempo para entregar las tramas. Este modo está orientado a redes P2P (*Peer-To-Peer*) donde todos se comunican entre todos sin la intervención de un coordinador.

- ***Beacon-enabled mode***: El coordinador PAN genera periódicamente *beacons* para que los dispositivos de la red se puedan sincronizar. En este modo se define la posibilidad de asignar ciertas ranuras de tiempo para aquellas transmisiones que requieran ciertos niveles de latencia garantizados. El protocolo GTS define como se lleva a cabo la solicitud, asignación y desasignación de estas ranuras, así como la transmisión en las mismas.

5.2 Protocolo CSMA/CA - Carrier Sense Multiple Access with Collision Avoidance

Los sensores inalámbricos que operan en la banda 2.4 GHz ISM, como es el caso de las tecnologías que utilizan el estándar IEEE 802.15.4, pueden verse afectados por las interferencias de radio de otros dispositivos. Por ejemplo, el canal 1 del estándar IEEE 802.11 comparte el espectro con los canales 11, 12, 13 y 14 del IEEE 802.15.4. Estas interferencias generalmente se generan debido a distintas redes que conviven en un mismo lugar. Sin embargo, estas interferencias pueden ser generadas maliciosamente, para afectar el funcionamiento de una red víctima.
En esta sección se presenta un mecanismo para atacar el protocolo CSMA/CA a través de la generación de colisiones produciendo una denegación de servicio.
En esta sección primero se realiza una breve descripción del protocolo CSMA/CA explicando el funcionamiento para acceder al medio. Luego se detallan los pasos seguidos para la implementación del ataque de colisiones.

5.2.1 Funcionamiento del protocolo CSMA/CA

Normalmente los radios de los nodos, como por ejemplo CC2420, utilizan el protocolo CSMA/CA, que proporciona un acceso aleatorio al canal. Para ello se basa en que cada estación sondea el canal antes de transmitir y se abstiene de hacerlo cuando detecta que el mismo está siendo ocupado por otra estación. Este protocolo no intenta detectar las colisiones, sino que intenta evitarlas. Esto se debe principalmente a dos causas. Primero a la dificultad para enviar una señal y escuchar el canal al mismo tiempo para detectar las colisiones. Segundo a los problemas de estación oculta y desvanecimiento que ocurren en las redes inalámbricas debido a las características propias del medio. Es por esto, que una vez que la estación comienza a transmitir una trama la transmite en su totalidad.
El protocolo CSMA/CA funciona como se describe a continuación [10] y se ilustra en la figura 5.1:

1. Cuando la estación o nodo posee una trama para transmitir, primero sondea el canal. Si este está inactivo, se transmite la trama después de un período corto de tiempo llamado DIFS (*Distributes Inter-Frame Space*).

2. Si el canal está activo, la estación selecciona un valor de espera aleatorio, llamado *backoff*, y efectúa una cuenta regresiva con este valor mientras detecta que el canal está inactivo. Cuando dectecta que el canal está ocupado, el valor del contador permanece congelado.

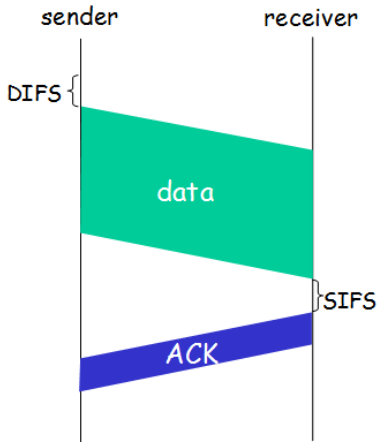


Figura 5.1: Protocolo CSMA/CA [10].

3. Cuando el contador llega a cero (lo cual puede suceder solo mientras el canal está inactivo), la estación transmite la trama completa y luego espera recibir un reconocimiento (ACK).

4. Si se recibe la trama de reconocimiento, la estación transmisora sabe que su trama ha sido recibida correctamente en la estación destino. Si la estación tiene otra trama que enviar, comienza de nuevo con el protocolo CSMA/CA desde el paso 2. Si no se recibe un reconocimiento, la estación transmisora supone que ha ocurrido un error y vuelve a retransmitir la trama. Para ello, entra en la fase de *backoff* del paso 2, seleccionando el valor aleatorio de espera dentro de un intervalo más largo. Si no se recibe un ACK después de un número fijo de retransmisiones, la estación se da por vencida y descarta la trama.

Cuando la estación destino recibe una trama, comprueba el código CRC para verificar si la trama fue recibida correctamente, espera un pequeño período de tiempo conocido con el nombre de SIFS (*Short Inter-Frame Spacing*) y luevo envía la trama de reconocimiento.

5.2.2 Implementación del ataque

La estrategia del nodo atacante consiste en desactivar el protocolo CSMA/CA y crear una aplicación que envíe continuamente un flujo de paquetes. Por lo que el atacante mandará tramas independientemente de lo que realicen los nodos vecinos. Sin sondear el canal y sin utilizar tiempos de *backoff*. Sin embargo, los nodos vecinos si realizan la detección de la portadora normal, por lo que se abstendrán de enviar sus tramas o si deciden enviarlas se generarán colisiones. Por lo que las tramas recibidas en el destino no serán válidas.
Para lograr esto se desactivo el mecanismo para detectar si el canal está libre y se configuraron en cero los valores de *backoff* del protocolo. Este ataque fue implementando modificando el software controlador del transmisor de radio CC2420. De acuerdo al TEP correspondiente al controlador CC2420 [77] el protocolo CSMA/CA es implementado mediante dos mecanismos que fueron los modificados para implementar el ataque: *Clear Channel Assessment* (CCA) y *Radio Backoff*.

Clear Channel Assessment (CCA)

Por defecto el transmisor CC2420 ejecuta el mecanismo CCA antes de transmitir. De esta forma, como fue explicado anteriormente, detecta si el canal está libre y en caso de no estarlo entra en *backoff* por determinado período de tiempo.

Para deshabilitar el mecanismo CCA se modificó el componente CC2420CsmaP. Mediante la invocación del comando send de la interfaz Send se envía las tramas. A su vez, este comando invoca al comando send del componentes CC2420Transmist donde se pasa la trama y si se utiliza el mecanismo CCA habilitado o no. Ahora se pasa como parámetro el valor *false* para que no se solicite CCA como se muestra en el cuadro 1.

Cuadro 1 Desactivar mecanismo CCA

```
command error_t Send.send( message_t* m_msg, uint8_t len ) {

        (...)
        ccaOn = FALSE;
        call CC2420Transmit.send(m_msg, ccaOn);
        (...)
}
```

Por otro lado, si se quiere deshabilitar el mecanismo CCA antes de la transmisión, se debe interceptar el evento requestCca de la interfaz RadioBackoff y llamar al comando setCca de la interfaz RadioBackoff con *false* para indicarle que no utilice el mecanismo, como se muestra en el cuadro 2.

Cuadro 2 Desactivar mecanismo CCA (2)

```
default async event void RadioBackoff.requestCca(message_t *msg) {
        call RadioBackoff.setCca(FALSE);
}
```

Radio Backoff

Cuando la radio necesita realizar un *backoff*, esta puede elegir entre tres tipos: initialBackoff, congestionBackoff, y lplBackoff, que son provistos por la interfaz RadioBackoff.
En InitialBackoff es el solicitado en el primer intento de transmisión. Es el período de *backoff* más corto. El CongestionBackoff es utilizado cuando el canal se encuentra ocupado. Finalmente, el LplBackoff es utilizado cuando los paquetes son enviados utilizando la técnica LPL (*Low Power Listening*), que busca reducir el consumo de la radio mientras se mantiene la confiabilidad de la entrega de paquetes. En está tecnica el emisor envia un preámbulo cuando desea enviar un paquete y el receptor se despierta cada cierto intervalo de tiempo para chequear si algún nodo quiere transmitir algún paquete.
Cuanto menor es el período de *backoff*, más rápida es la retransmisión y es más probable que el transmisor acapare el canal. Para ajustar estos períodos se pueden utilizar los comandos de la interfaz RadioBackoff como se muestra en el cuadro 3. En este caso configuramos el *backoff* inicial en cero ya que será el atacante.

Cuadro 3 Desactivar backoff

```
default async event void RadioBackoff.requestInitialBackoff(message_t *msg) {
        call RadioBackoff.setInitialBackoff(0);
}
```

5.3 GTS - Guaranteed Time Slot

En esta sección se presenta un estudio de las principales vulnerabilidades que presenta el mecanismo GTS (*Guaranteed Time Slot*) previsto en el estándar IEEE 802.15.4. Debido a estas deficiencias se

pueden dar distintos ataques de denegación de servicio. En esta sección se presenta las vulnerabilidades encontradas así como una implementación que las explota. Para lograr implementar el ataque fue necesario realizar un estudio detallado de las tramas intercambiadas entre los nodos cuando se utiliza el mecanismo GTS.

5.3.1 Funcionamiento del protocolo GTS y sus vulnerabilidades

El estándar IEEE 802.15.4 tiene la posibilidad de garantizar ciertos niveles de calidad de servicio (QoS), por ejemplo con el protocolo GTS. Éste permite que los nodos soliciten al coordinador PAN la asignación de un conjunto de ranuras de tiempo. Si el pedido es aceptado por el coordinador (lo cual depende de la capacidad actual de los recursos disponibles), los nodos pueden enviar sus paquetes en ciertas ranuras de tiempo dedicadas y libres de colisiones. Por lo que este mecanismo es bastante atractivo para aplicaciones sensibles al tiempo, ya que es posible garantizar el retardo extremo a extremo de los paquetes. Las ranuras GTS se otorgan según una política FCFS (*First Come, First Serve*).

El esquema de reserva de GTS es similar al esquema TDMA (*Time Division Multiple Access*) en términos de reserva de intervalos de tiempo. Sin embargo, para el mecanismo TDMA, el tiempo se divide en tramas de duración fija y cada trama se divide en un número fijo de intervalos de tiempo. Estos intervalos de tiempo están dedicados para el uso de una conexión de modo que se garantiza un cierto grado de ancho de banda. Por otro lado, el protocolo GTS del estándar IEEE 802.15.4 no tiene longitud fija. Además, es compatible con cambios dinámicos de topología tales como el fallo de un nodo y los nuevos nodos que entran en la red [78].

En la figura 5.2 se muestra la estructura de la supertrama utilizada en el modo *beacon-enabled*. Como se observa, la supertrama está delimitada por dos *beacons* (que usan los nodos para sincronizarse) y está compuesta por una parte activa (AP-*Active Period*) y una inactiva (IP-*Inactive Period*). Durante la parte inactiva (que es opcional) el coordinador y todos los dispositivos pueden entrar en un modo de bajo consumo. La parte activa consiste en 16 time slots y está dividida en dos períodos: CAP (*Contention Access Period*) y CFP (*Contention Free Period*).

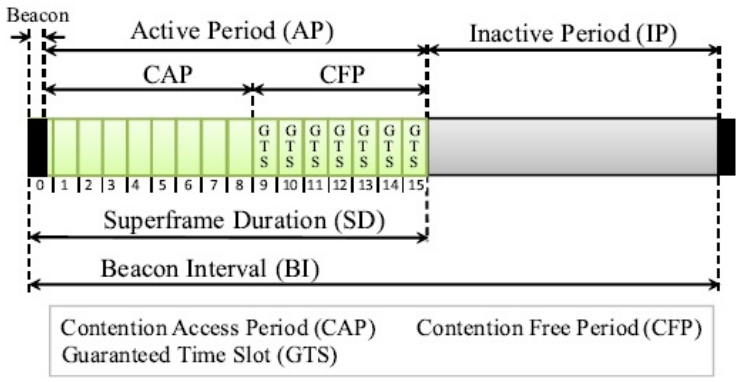


Figura 5.2: Estructura de la supertrama

Durante el CAP los nodos pueden comunicarse utilizando CSMA/CA ranurado y es en este período donde se envían las tramas con comandos MAC, por ejemplo una solicitud de *GTS allocation*. Por otro lado, las ranuras de tiempo reservadas están situadas dentro del CFP (en otras palabras, durante el CFP sólo "hablan" los nodos que tienen ranuras asignados mediante el protocolo GTS que se detallará a continuación). En algunos casos, el CFP puede no existir (por ejemplo, si ningún nodo tiene ranuras

asignados). En esta situación el CAP ocupara toda la duración de la supertrama (SD-*Superframe duration*). El CAP tiene un largo mínimo definido en la variable *aMinCAPLength*, de modo de asegurarse que los comandos MAC siempre podrán ser enviados, incluso cuando todos los GTSs estén asignados [78].

Esta estructura de supertrama es vital para el protocolo GTS, durante el CAP, los nodos acceden al canal compitiendo con los demás nodos mediante el algoritmo CSMA/CA ranurado. En cambio, durante el CFP, los nodos previamente autorizados, transmiten en las ranuras asignados (sin utilizar CSMA/-CA).

Los nodos pueden solicitar la asignación y la desasignación (una vez asignados) de las ranuras de tiempo dedicadas. Esto se realiza mediante el envío de tramas denominadas *GTS allocation/deallocation request*. La estructura de esta trama se puede observar en el Anexo A.7, estas se envían durante el CAP utilizando el protocolo CSMA/CA ranurado.

Cuando el coordinador PAN acepta una solicitud, responde con un *beacon* indicando: la lista de los nodos a los que se les aceptó la solicitud, la ranura de tiempo de la supertrama en el que deberá comenzar la transmisión de cada uno, y la cantidad de ranuras asignadas. La estructura de un *beacon* se puede consultar en el Anexo A.7.
En las Figuras 5.3 y 5.4 se muestra la secuencia de mensajes necesarios para la gestión de GTS con éxito. En particular en la primera se observan los pasos de *allocation* de un conjunto de ranuras dedicadas. Para eso el nodo envía una trama de tipo *GTS request* (este evento es disparado cuando la capa superior llama al comando *request* de la interfaz MLME_GTS). Cuando el coordinador PAN recibe esa trama responde con una trama de ACK y se lo indica a la capa superior disparando el evento *indication* de la interfaz MLME_GTS. Luego se envía un *beacon* con un nuevo *GTS descriptor* que indica los datos de la asignación del GTS pedido. Cuando el nodo recibe ese *beacon*, la capa MAC se lo indica a la capa superior a través del evento *confirm* de la interfaz MLME_GTS.

La segunda figura muestra como ocurre un *deallocation* de los GTS asignados. Esto puede ser originado por el nodo mismo o por el coordinador de la red (casos a y b respectivamente de la figura 5.4). Cuando el *deallocation* es solicitado por el nodo, este envía una trama de tipo *GTS request* (que se logra cuando la capa superior llama al comando *request* de la interfaz MLME_GTS). Cuando el coordinador PAN recibe esa trama responde con una trama de ACK y se lo indica a la capa superior disparando el evento *indication* de la interfaz MLME_GTS. Cuando el nodo recibe el ACK, la capa MAC se lo indica a la capa superior a través del evento *confirm* de la interfaz MLME_GTS.
Si un nodo tiene asignado un GTS pero no lo usa durante un período determinado de tiempo, el coordinador puede tomar la decisión de quitárselo, en ese caso, la capa MAC del coordinador se lo indica a su capa superior mediante el evento *indication* de la interfaz MLME_GTS y a continuación envía un *beacon* donde se indica que el GTS fue quitado. Cuando el nodo recibe el *beacon*, se dispara el evento *indication* de la interfaz MLME_GTS.

Como se explicó, el coordinador PAN avisa quienes son los nodos a quienes se les asignó algún GTS mediante el envío de un beacon. Esta trama de gestión se manda en *broadcast* y sin cifrar, por lo tanto, todos los nodos pueden escucharla. Esta es una gran vulnerabilidad del protocolo GTS. Otra característica que hace más débil la seguridad del mecanismo, es que el coordinador sólo verifica el número de secuencia de la trama *GTS-request* (no se hace otro tipo de verificación), por lo tanto puede ser fácilmente evadido por un atacante.

5.3.2 Ataques al protocolo GTS

En [11] se puede encontrar un resumen de posibles ataques que pueden ocurrir en el protocolo GTS. Uno de los ataques presentados se basa en que el nodo malicioso escucha los *beacons* para identificar las ranuras GTS solicitadas y así poder enviar un *GTS deallocation* haciéndose pasar por el nodo

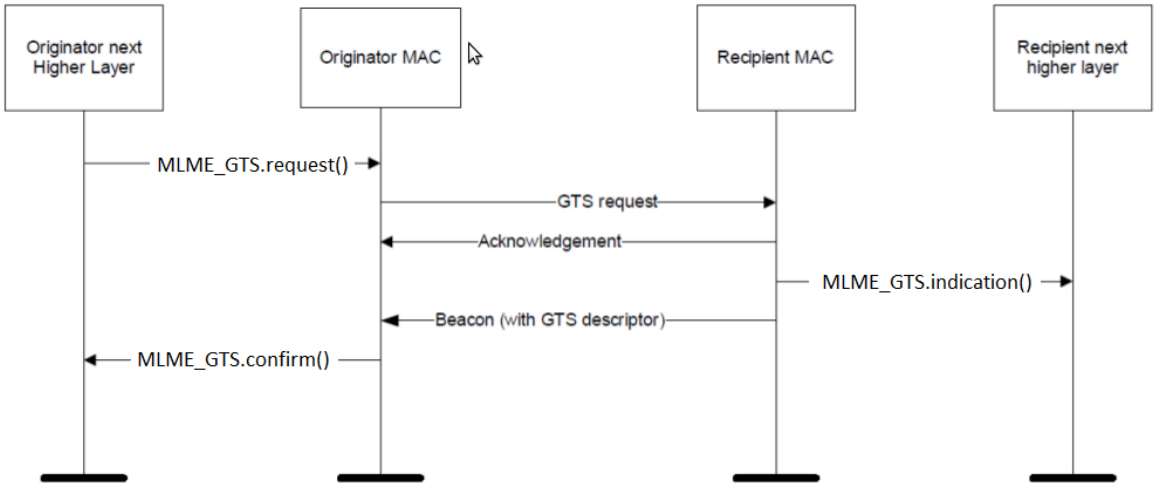


Figura 5.3: Mensajes intercambiados - GTS allocation.

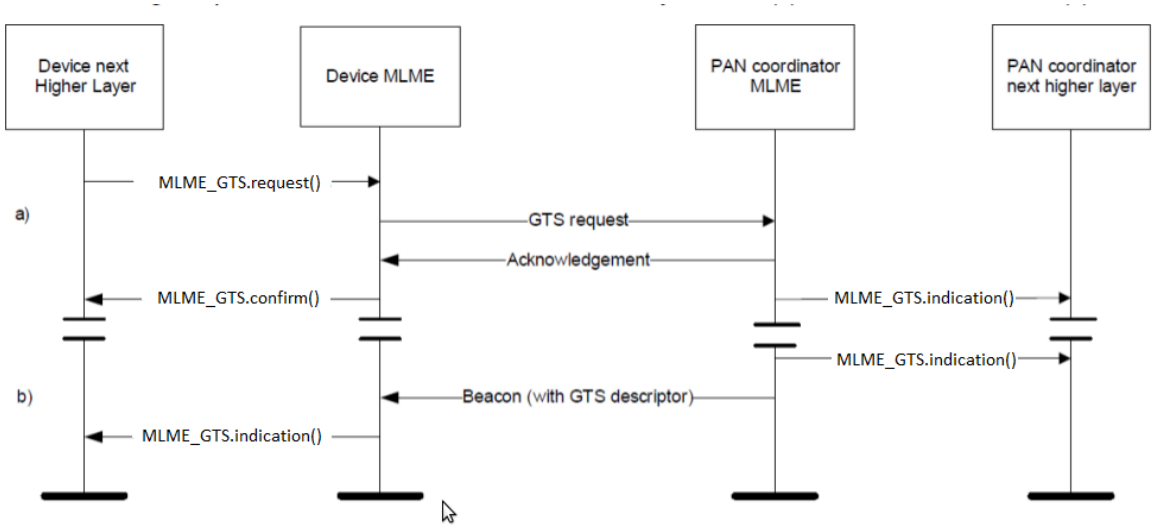


Figura 5.4: Mensajes intercambiados - GTS deallocation.

legítimo que tenía asignadas las ranuras de tiempo dedicadas. Este tipo de ataques se conoce con el nombre de denegación de servicio mediante el envío de *deallocations*. Ver figura 5.5.

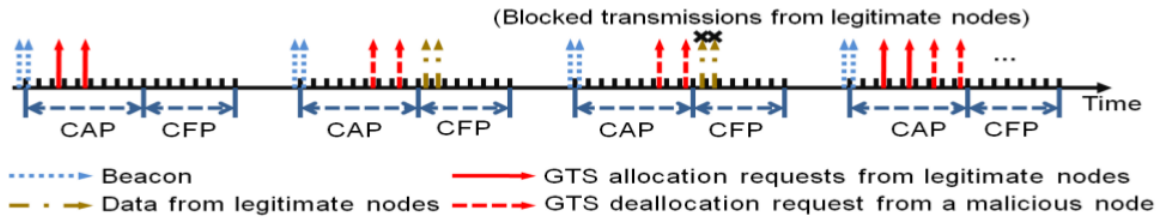


Figura 5.5: Ataque por denegación de servicio mediante el envío de deallocations [11].

5.3.3 Implementación del ataque

A continuación se describirá como se implementó el ataque de denegación de servicio mediante el envío de *deallocations*. La implementación se basa en las ideas mencionadas en [11]. Sin embargo, no se dispone de una implementación de referencia.

5.3.3.1 Lógica

Para el ataque se tienen tres actores: el nodo malicioso, la víctima (que en este caso es una, pero que pueden ser varias) y el coordinador de la red. En la figura 5.6 se puede ver la secuencia de tramas que se intercambian en el canal durante el ataque.

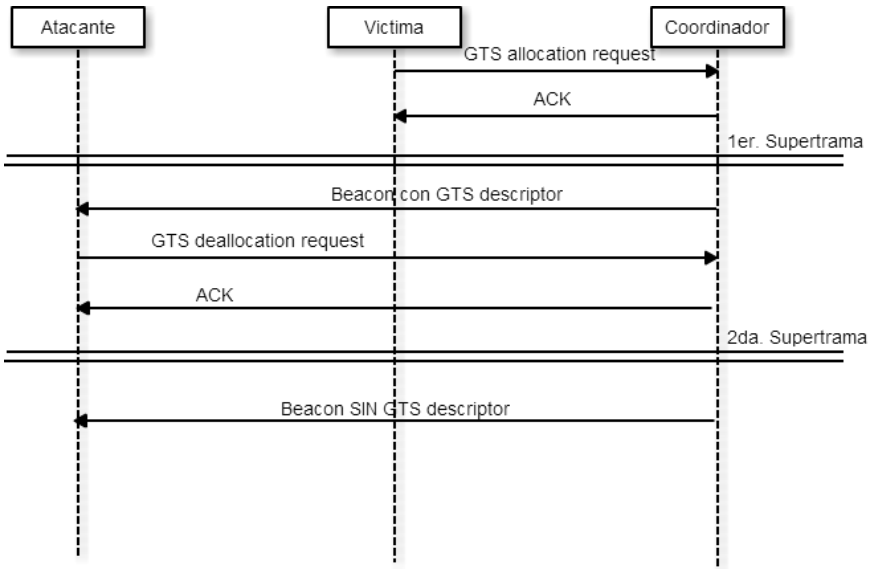


Figura 5.6: Ataque de DoS contra la transmisión de datos.

Cuando un nodo de la red envía un *GTS request* solicitando una ranura, el coordinador le responde con un ACK y envía en la siguiente supertrama un *beacon* de forma *broadcast*, donde incluye en el campo *GTS list* (ver anexo A.7) un *GTS descriptor* que contiene la información de la asignación, incluyendo

entre otras cosas la información que va a utilizar el atacante: el identificador del nodo, el largo de la ranura y si es de transmisión o recepción.

Cuando el nodo atacante, escucha ese *beacon* lo que hace es fijarse si hay algún GTS descriptor y extrae la información mencionada anteriormente. Con esos datos el atacante modifica su identificador con el identificador del nodo víctima, arma una solicitud de *GTS deallocation*, y se la envía al coordinador. Es decir, envía ese *deallocation* haciéndose pasar por el nodo víctima.

Como consecuencia, en la siguiente supertrama el coordinador manda un *beacon*, sin *GTSs descriptors*. Dado que no hay ningún GTS asignado.

En resumen, lo que va a hacer el atacante es escuchar los *beacons*, parsearlos para extraer los distintos *GTS descriptors*, modificar su identificador con un identificador falso y enviar el *deallocation request*. Notar que en este caso sólo se tiene una víctima, pero este proceso se puede repetir para todos los *GTS descriptors* en el *beacon* por lo cual se puede atacar cualquier cantidad de nodos en la red.

5.3.3.2 Programación y componentes utilizados

Como se mencionó, la implementación de este ataque se basa en la implementación del protocolo de capa MAC IEEE 802.15.4 del instituto KTH (Royal Institute of Technology – Suecia) que está disponible en tinyos-2.x-contrib/kth/tkn154-gts/ y cuya documentación puede consultarse en [79].

Para realizar el ataque se agregó una aplicación maliciosa que ataca las vulnerabilidades de GTS explicadas anteriormente. Nuestra aplicación utiliza directamente el conjunto de interfaces **MLME** que provee la capa MAC como puede apreciarse en la figura 5.7. Esta aplicación es la que tiene toda la lógica del ataque, por lo que no fue necesario modificar la implementación del protocolo GTS o las interfaces MLME. Estas interfaces puedes encontrarse en el directorio /tinyos-2.1.1/tos/lib/mac/tkn154/-interfaces/MLME. En la figura 5.8 se encuentra la máquina de estados que representa el comportamiento de la aplicación atacante creada.

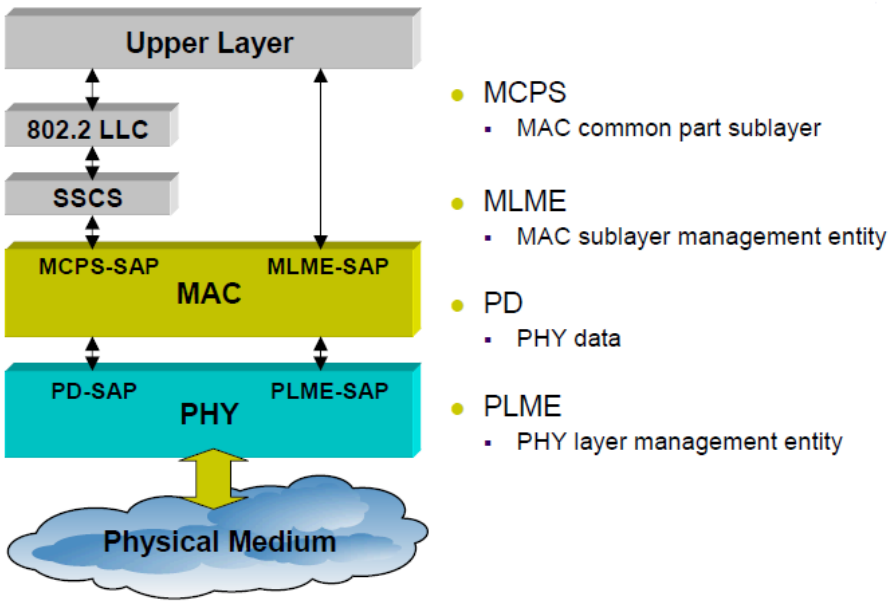


Figura 5.7: Interfaces [12].

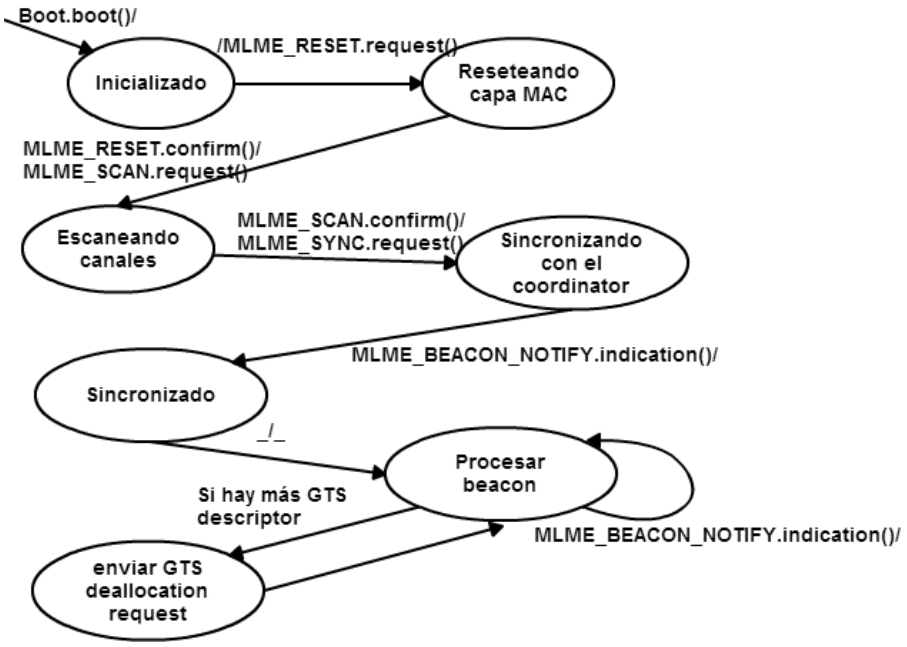


Figura 5.8: Máquina de estado con el comportamiento del nodo atacante.

A continuación se describe que funcionalidades provee cada una de las interfaces MLME invocadas desde la aplicación maliciosa creada. Conjuntamente se presenta el pseudocódigo de la implementación del componente malicioso. Recordar que las interfaces en NesC son bidireccionales. Por lo tanto, para que la aplicación maliciosa utilice el conjunto de interfaces MLME esta debe implementar los eventos para poder, a cambio, invocar la implementación de los comandos que estas proveen.

MLME_SCAN

Esta interfaz define, entre otras cosas, cómo un dispositivo puede determinar la presencia o ausencia de un coordinador en un canal de comunicación. Para ello brinda el comando *request* que inicializa un escaneo de canales en una determinada lista de canales y el evento *confirm* que informa de los resultados obtenidos de analizar los canales pedidos.

A continuación se presenta el pseudocódigo de la implementación del evento *confirm* 4.

Cuadro 4 Pseudocódigo del evento confirm()

```
event void MLME_SCAN.confirm (...){
    si encontré un canal donde hay un coordinador (recibí un beacon)
      //configurar la dirección del coordinador
      call  MLME_SET.macCoordShortAddress(...);
      // configurar el identificador de la PAN
      call MLME_SET.macPANId(...);
      //Este commando sera explicado más adelante.
      call MLME_SYNC.request(...);
      //configurar los valores de dirección destino, origen, id PAN, etc. que serán enviados en
      // la cabecera MAC de las próximas tramas a enviar.
      call Frame.setAddressingFields(...);
    } else
      Volver a escanear el canal.
    }
```

MLME_SYNC

Esta interfaz permite la sincronización con un coordinador. Para esto, ésta define el comando *request* que permite sincronizarse con el coordinador a través de los *beacons*.

MLME_SYNC_LOSS

Esta interfaz permite comunicar la pérdida de sincronización con el coordinador a la siguiente capa superior. Para lograr esto, esta interfaz define el evento *indication* que es disparado cuando se pierde la sincronización. En la implementación de la aplicación maliciosa este evento sólo modifica una variable.

MLME_RESET

Esta interfaz permite restablecer la subcapa MAC a sus valores predeterminados. Para esta tarea, ofrece el comando *request* que le permite a la siguiente capa superior solicitar una operación de reinicio. Este comando, también inicializa los valores de la capa MAC y debe ser llamado por lo menos una vez antes de poder usar las funcionalidades de esta capa. Por otro lado, el evento *confirm* reporta el resultado de la ejecución del comando *request*.

En la aplicación maliciosa se invocó el comando *request* durante el evento *boot* (como se muestra en el cuadro 5), para inicializar la capa MAC y una vez que el evento *confirm* es disparado comienza el escaneo en los distintos canales para buscar un coordinador como fue explicado en la sección de la interfaz MLME_SCAN.

Cuadro 5 Inicialización de la capa MAC

```
event void Boot.booted() {
        call MLME_RESET.request(TRUE);
}

event void MLME_RESET.confirm(ieee154_status_t status){
        if (status == IEEE154_SUCCESS)
                escanear los canales buscando un coordinador
}
```

MLME_GTS

Esta interfaz permite solicitar y mantener las ranuras GTS. Un dispositivo que quiera utilizar las ranuras debe hacer un seguimiento de los *beacons* que envía su coordinador PAN.

Esta interfaz ofrece los comandos *request* y *requestFromPAN*. El primero de éstos le permite a un dispositivo solicitar una asignación de ranuras GTS o devolver una que le fue asignada. En particular, esta segunda utilidad es la que se utiliza en la aplicación maliciosa ya que el atacante envía solicitudes de *GTS deallocation*, para que los GTS de la víctima sean desalojados. El comando *requestFromPAN* le permite a la aplicación que corre en el coordinador solicitarle a su capa MAC que le quite cierta ranura GTS asignado a un dispositivo.

Los eventos que se deben implementar son *confirm* e *indication*. El primero reporta el resultado de una solicitud de *GTS allocation* o *GTS deallocation* mientras que el segundo indica que un GTS ha sido concedido o que un GTS que se tenía asignado fue quitado. Ambos eventos se dejaron sin implementar en la aplicación maliciosa.

MLME_SET

Esta interfaz permite establecer distintos valores de atributos de la capa física y MAC. Esta interfaz es utilizada en la aplicación maliciosa para configurar el poder de transmisión (propiedad de capa física), la dirección del coordinador y el identificador de la PAN (propiedades de capa MAC). Sin embargo, la funcionalidad que a los efectos prácticos fue más importante es la que se muestra en 6 y que permite asignar una dirección MAC (identificador de nodo) falsa.

Cuadro 6 Asignar identificador de nodo falso

```
call MLME_SET.macShortAddress(spoofId);
```

Mediante este comando se puede solicitar a la capa MAC que establezca el identificador de nodo que se le pasa por parámetro. Esto permite enviar paquetes haciéndonos pasar por otro nodo legítimo de la red o un nodo inexistente en la red (ataque *sybil*). De este modo, se pueden enviar paquetes de *deallocation* para un nodo legítimo de la red. El identificador de este nodo víctima es obtenido a partir de un *beacon* enviado por el coordinador PAN.

MLME_GET

Esta interfaz permite leer distintos valores de atributos configurados en la capa física y MAC.

MLME_BEACON_NOTIFY

Esta interfaz sólo tiene el evento *indication* el cual es disparado cuando un *beacon* es recibido. Es en este evento donde se concentra la mayor parte de la lógica del atacante. En el cuadro 7 se presenta un pseudocódigo del mismo.

Cuadro 7 Evento MLME_BEACON_NOTIFY.indication(...)

```
event message_t* MLME_BEACON_NOTIFY.indication (message_t* frame){
    si se recibió el beacon durante el escaneo del canal
      se encontró una red a la cual atacar
    sino
      //ya se había encontrado esa red en un  beacon anterior
      //obtener el payload y el largo del mismo de la trama MAC
      uint8_t* macPayload = (uint8_t*) call Frame.getPayload(frame);
      uint8_t payLen = call Frame.getPayloadLength(frame);
      Si el beacon tiene una lista de GTS descriptors
         Obtengo la cantidad de GTS descriptors
         Para cada GTS descriptor
            //Extraer los datos del nodo víctima: id del nodo y largo del GTS
            spoofId = macPayload[5+(3*i)];
            spoofId = (spoofId<< 8) | macPayload[4+(3*i)];
            largoGTS = macPayload[6+(3*i)] >> 4;
            //hacer spoofing de la identidad del nodo víctima
            call MLME_SET.macShortAddress(spoofId);
            //Enviar deallocation request
            if ((macPayload[3] & (1<<i)) == 0x00) {
                    //slot de transmision
                    call MLME_GTS.request(call GtsUtility.setGtsCharacteristics(largoGTS,
                    GTS_TX_ONLY_REQUEST, GTS_DEALLOCATE_REQUEST) ,NULL);
            } else {
                    //slot de recepcion
                    call MLME_GTS.request(call GtsUtility.setGtsCharacteristics(largoGTS,
                    GTS_RX_ONLY_REQUEST, GTS_DEALLOCATE_REQUEST) ,NULL);
             }
}
```

CAPÍTULO

6

Análisis de resultados en capa MAC

En este capítulo se presentan los resultados obtenidos para cada uno de los ataques implementados en capa MAC. En la sección 6.1 se presentan las pruebas realizadas y los resultados obtenidos para el ataque de colisiones implementado para el protocolo CSMA/CA. Luego en la sección 6.2 se realiza lo mismo para el ataque de denegación de servicio mediante el envío de *deallocations* implementado para el protocolo GTS.

6.1 Resultados para CSMA/CA

En esta sección se presenta el diseño de las pruebas realizadas, los resultados obtenidos luego de su ejecución y finalmente las conclusiones.

6.1.1 Diseño de las pruebas

A continuación se detallan las aplicaciones ejecutadas, los actores participantes en las pruebas, los escenarios, las métricas y algunas consideraciones a tener en cuenta durante el desarrollo de las pruebas.

6.1.1.1 Aplicaciones, actores y métricas

Para realizar las pruebas se utilizaron tres actores: emisor, receptor y atacante.El emisor y el atacante fueron ejecutados en nodos sensores Shimmer. Mientras que el código del receptor se ejecutó en el adaptador de red Shimmer SPAN.
El emisor ejecuta una aplicación que envía 1000 paquetes de datos. El atacante ejecuta el código malicioso que fue detallado en el capítulo anterior. El receptor ejecuta la aplicación *base station* para recibir paquetes a través de su transmisor de radio y enviarlos el puerto USB. Luego, para calcular la cantidad de paquetes recibidos, se conecta el receptor a la computadora que ejecuta una aplicación Java creada para realizar estas pruebas. La aplicación JAava cuenta la cantidad de paquetes recibidos desde el puerto USB, los cuales fueron capturados a través de la aplicación *base station*.
Esta información es utilizada para calcular el porcentaje de pérdida de paquetes que se emplea como métrica para evaluar el desempeño del ataque. Para realizar las mediciones, se corrió tres veces cada prueba y se realizó el promedio de las ejecuciones para obtener la medida para cada escenario.

6.1.1.2 Consideraciones

Durante las pruebas a realizar se debe considerar que IEEE 802.15.4 y IEEE 802.11 utilizan la misma banda ISM de 2.4GHz. El estándar IEEE 802.15.4 define dieciséis canales dentro de la banda del espectro (númerados desde el 11 hasta el 26) donde cada canal es de 2 MHz de ancho y están separados por una banda de 3 MHz entre ellos. Cada canal IEEE 802.11 ocupa 22 MHz y están separados por bandas de 5 MHz. Un canal IEEE 802.11 puede superponerse con hasta cuatro canales de IEEE 802.15.4 como puede verse en la figura 6.1. Observar que solo los canales 15, 20, 25 y 26 del estándar IEEE 802.15.4 no se solapan con ningún canal del estándar IEEE 802.11 [13].

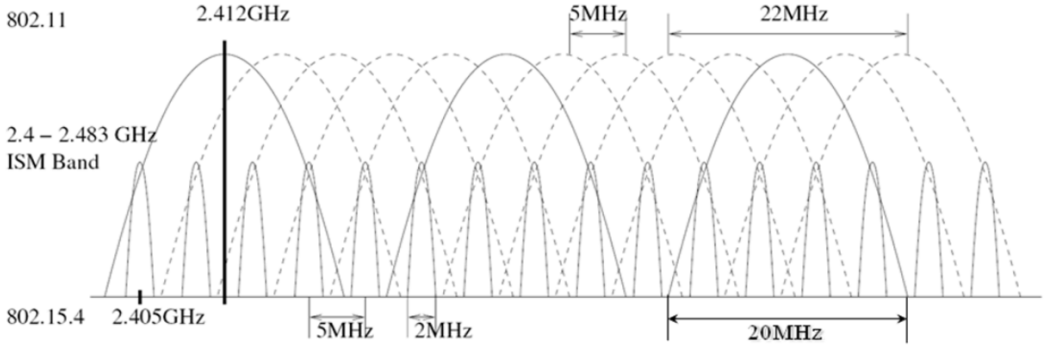


Figura 6.1: Canales de los estándares IEEE 802.15.4 y IEEE 802.11 [13].

Por otro lado, una sola transmisión IEEE 802.15.4 ocupa sólo una parte del ancho de banda de un canal IEEE 802.11 y su potencia de transmisión es muy bajo en comparación con las transmisiones de IEEE 802.11 (1/10 – 1/100). Por lo tanto, en la mayoría de los casos, el dispositivo IEEE 802.11 no puede detectar con eficacia las transmisiones de IEEE 802.15.4, mientras que el dispositivo IEEE 802.15.4 puede detectar las de IEEE 802.11. Por lo tanto, un dispositivo que transmita con el estándar IEEE 802.11 no pospondrá la transmisión incluso en la presencia de tráfico IEEE 802.15.4. Esto provoca que cuando los nodos transmiten a través de los canales que se solapan con los del estándar IEEE 802.11 logren una tasa de envío de paquetes menor que a través de los canales que no se solapan. Esto debe ser considerado en las pruebas ya que serán realizadas en un ambiente donde hay redes WiFi presentes por lo que el desempeño de los nodos puede variar dependiendo del canal utilizado.

6.1.1.3 Escenarios

Para realizar las pruebas se consideró la variación de distintos parámetros entre los que se encuentran los siguientes.

- Distancias entre emisor y receptor: las distancias consideradas serán 0.5, 1, 5 y 10 metros.
- Distintos canales (frecuencias) de transmisión: se considerarán los canales 11 y 26. Observar que el canal 11 colisiona con las transmisiones WiFi, mientras que el canal 26 no. El canal puede ser configurado en los nodos en tiempo de compilación agregando al Makefile la siguiente directiva “CC2420_CHANNEL=11” donde 11 es el canal.
- Distintas posiciones relativas del atacante con respecto a la pareja emisor-receptor. En particular se considerarán las siguientes.
  - El atacante cerca del receptor: dada la distancia entre emisor y receptor, el atacante se colocará a un 5 % de esta distancia a partir del receptor. Este caso será mencionado como A1 durante las pruebas realizadas.

- El atacante a mitad de distancia entre emisor y receptor: en el caso A2 el atacante se coloca a un 50 % de la distancia entre emisor y receptor.
- El atacante cerca del emisor: este caso es análogo al caso A1 pero el nodo malicioso se coloca a un 5 % de la distancia a partir del emisor. Este será el caso A3.

6.1.2 Resultados obtenidos

Primero se analizó la influencia de la redes WiFi (IEEE 802.11) en las transmisiones IEEE 802.15.4. Para ello se midió el porcentaje de paquetes recibidos correctamente, de los 1000 paquetes enviados por la aplicación emisora. Esto se realizó para los canales 11 y 26. Las transmisiones en el canal 11 colisiona con las transmisiones WiFi, mientras que las transmisiones en el canal 26 no lo hacen. Los resultados obtenidos se pueden apreciar en la figura 6.2. Como era esperable en el canal 11 que interfiere con las transmisiones WiFi llega una menor cantidad de los paquetes enviados.

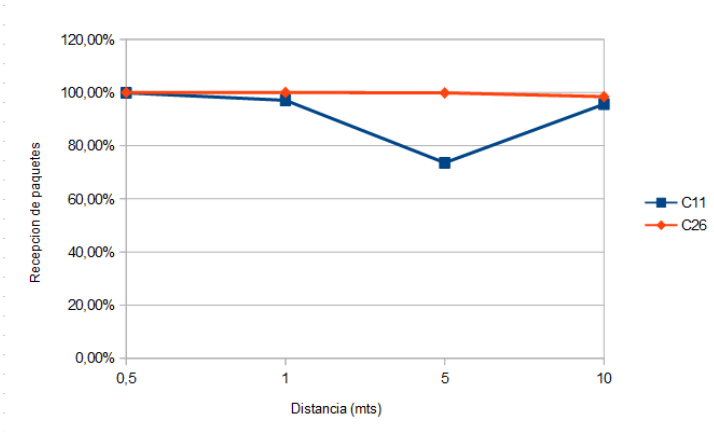
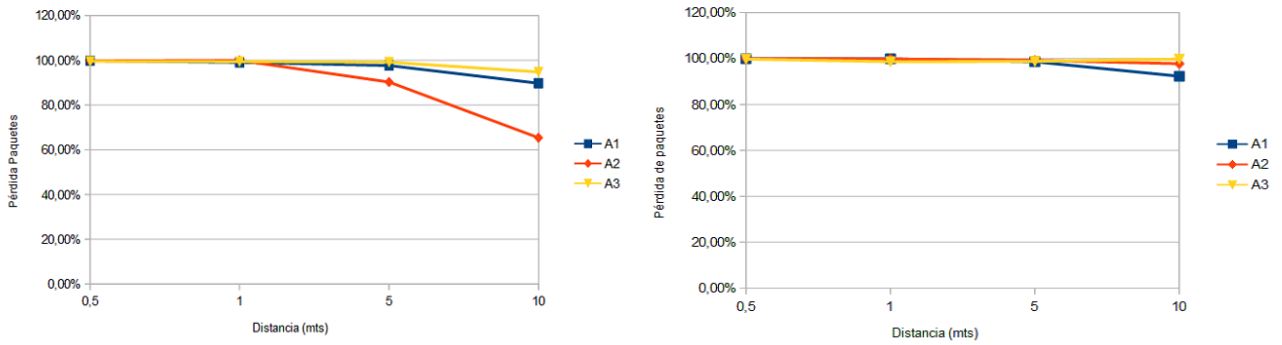


Figura 6.2: Porcentaje de recepción de paquetes para los canales 11 (interfiere con WiFi) y 26 (no interfiere con WiFi).

Luego se analizó la efectividad del ataque de colisiones, para eso se consideró como métrica el porcentaje de paquetes perdidos para los 2 canales y distintas posiciones relativas del nodo atacante entre el emisor y receptor. Los resultados obtenidos se muestran en las figuras 6.3a y 6.3b.



(a) Canal 11 **(b)** Canal 26

Figura 6.3: Pérdida de paquetes-ataque de colisiones para CSMA/CA

Para el canal 11 se observa que el escenario A3 donde el atacante está cerca del emisor se obtiene los mejores resultados. Luego está el escenario A1 donde el atacante se sitúa cerca del receptor. Finalmente, en el escenario A2 donde el atacante está a mitad de distancia entre emisor y receptor se obtienen los peores resultados. Se aprecia una significativa diferencia con los dos casos anteriores.
Por otro lado se observa que para distancias cortas la distancia relativa entre el emisor y el receptor no es importante, ya que la pérdida de paquetes en todos los casos es similar. Además, la efectividad del ataque es mucho mayor, logrando hasta un 99.87 % de pérdida de paquetes. A mayores distancias, la diferencia en los resultados para los distintos escenarios comienza a hacerse visible. A una distancia de 10 mts entre emisor y receptor, para el caso A3 se obtuvo un 94.78 % de pérdida de paquetes. Mientras que para los escenarios A1 y A2 se obtuvo 89.77 % y 65.40 % respectivamente.
En el canal 26, las diferencias entre los resultados de los distintos escenarios son menores que para el canal 11. El escenario A3, donde el atacante está cerca del emisor continúa teniendo los mejores resultados. Por otro lado, en los escenarios donde las distancias entre emisor y receptor son pequeñas el ataque tiene mayor impacto, alcanzando el 99.93 % de pérdida de paquetes. A una distancia de 10 mts entre emisor y receptor, para el caso A3 se obtuvo un 99.80 % de pérdida de paquetes. Mientras que para los escenarios A1 y A2 se obtuvo 92,31 % y 97,76 % respectivamente.

6.1.3 Conclusiones

A la hora de configurar una WSN es importante considerar que ciertos canales colisionan con las redes WiFi. Por lo que, de ser posible es conveniente elegir uno de los canales que no colisiona con este tipo de transmisiones.
En los análisis realizados se observó que la cantidad de paquetes perdidos no depende de la posición del atacante entre el emisor y el receptor para nodos que se sitúan cerca. Por otro lado, en este caso los efectos son mayores que para nodos que se encuentran más distantes. En general, el efecto es mayor cuando el nodo malicioso está situado cerca del emisor.
Se logró demostrar que el impacto de este ataque puede ser muy grave ya que en varias situaciones se obtuvieron pérdidas muy cercanas al 100 % de los paquetes. Por lo que, bajo el efecto de este ataque, una WSN puede quedar incomunicada.
Por otro lado, sería interesante estudiar porque es que no llegan los paquetes al receptor. Si es porque el nodo emisor escucha al atacante y no envía sus paquetes entrando en *backoff*, o si es porque el nodo emisor decide envíar los paquetes (por ejemplo, luego de abstenerse durante determinado período de tiempo) pero los paquetes enviados colisionan con la señal emitida por el atacante.
Otro factor que se podría considerar es la variación de la potencia de transmisión del atacante. Una señal más fuerte debería influir de forma negativamente en la transmisión de los nodos victíma. Los valores de potencia válidos que se pueden configurar entán entre 1 y 31, donde 1 equivale a -25 dBm y 31 a 0 dBm. En las pruebas realizadas la potencia de transmsión se mantuvo en su valor por defecto.

6.2 Resultados para GTS

Las pruebas realizadas para este ataque fueron de tipo funcional. En esta sección se presenta el diseño de las mismas, los resultados y las conclusiones obtenidas.

6.2.1 Diseño de las pruebas

Para verificar el funcionamiento del ataque se realizaron pruebas en escenarios. En todos los casos, se prendieron primero todos los nodos de la red víctima, y luego de que estos están funcionando normalmente, se prende el nodo malicioso. En otras palabras, primero se prende el coordinador y éste empieza a enviar periódicamente los *beacons*. Luego, se prende/n el/los nodo/s. Estos se sincronizan e inmediatamente mandan las tramas de *GTS request allocation*. Una vez que el coordinador les acepta la solicitud se comienza el intercambio de datos. Luego de este momento es que se enciende el nodo malicioso.

Escenario 1
Actores:

- coordinador PAN
- nodo (víctima)
- atacante

En este primer escenario el nodo víctima tiene el identificador 0x0002. Éste solicita una ranura de tiempo dedicada para transmisión y otra para recepción.

Escenario 2
Actores:

- coordinator PAN
- dos nodos (víctimas)
- atacante

En este escenario los nodos víctimas se configuraron con los identificadores 0x0002 y 0x0003 respectivamente. Cada uno solicita distintas cantidades de ranuras dedicadas. El nodo 0x0003 solicita tres ranuras para transmisión. Por otro lado el nodo 0x0002 solicita una ranura para recepción y cinco para transmisión.

Aplicaciones
Para establecer la red víctima se tomaron las aplicaciones de prueba que vienen por defecto con la implementación del protocolo GTS. En el nodo coordinador PAN de la red se instaló la aplicación TestCoordReceiverC que se puede encontrar en el directorio GTS/tkn154-gts/apps/beacon-enabled/TestGts/coordinator. Para los nodos se utilizó la aplicación TestDeviceSenderC del directorio GTS/tkn154-gts/apps/beacon-enabled/TestGts/device. Mientras que para el atacante se instaló la aplicación maliciosa desarrollada que fue descrita en la sección 5.3.

6.2.2 Verificación del funcionamiento

Escenario 1
En las pruebas realizadas sobre el escenario 1 se obtuvo la captura que se muestra en la figura 6.4a. En las lineas 7 y 11 están las tramas de *GTS allocation request* enviados por el nodo. En la figura 6.4b se muestra en mayor detalle el contenido de la trama número 7, como puede observarse es una ranura (campo *GTS length*), para recepción (campo *GTS Direction*), y es un pedido de *allocation* (campo *characteristic type*). Por más información de los campos de esta trama consultar el Anexo A.7.
Luego el coordinador de la red asigna estas ranuras solicitadas, lo cual lo anuncia en el *beacon* que se muestran en la figura 6.5. Este *beacon* contiene dos *GTS descriptors*, uno de transmisión y otro de recepción, ambos de largo uno y para el nodo 0x0002.
Luego se muestran las líneas 47 y 55 de la misma captura de la figura 6.6a, en esta figura se pueden observar las tramas de *GTS deallocation request* enviados por el nodo malicioso haciéndose pasar por el nodo víctima (observar que el nodo origen en la captura es el 0x0002 pero este nodo no envió el pedido de *deallocation*). En la figura 6.6b se muestra en mayor detalle el contenido de la trama número 47, como puede observarse el campo *characteristic type* indica que es un pedido de *deallocation* de una ranura para recepción. Es decir, para la trama solicitada en la trama número 7.
A continuación el coordinador le quita las ranuras y lo informa a través del *beacon* mostrado en la figura 6.7. Observar que este *beacon* no tiene *GTS descriptors*, por lo tanto, ningún nodo de la red tiene ranuras asignadas.

No.	Time	Source	Destination	Protocol	Length	Info
4	2.812949000	0x0000		IEEE 802.15.4	27	Beacon, Src: 0x0000
5	3.749926000	0x0000		IEEE 802.15.4	27	Beacon, Src: 0x0000
6	4.687997000	0x0000		IEEE 802.15.4	27	Beacon, Src: 0x0000
7	4.691880000	0x0002		IEEE 802.15.4	25	GTS Request
8	4.693904000			IEEE 802.15.4	19	Ack
9	5.624908000	0x0000		IEEE 802.15.4	31	Beacon, Src: 0x0000
10	5.625908000	::	ff02::	IGMP	56	Traceroute Request[Malformed Packet]
11	5.631911000	0x0002		IEEE 802.15.4	25	GTS Request
12	5.632958000			IEEE 802.15.4	19	Ack
13	6.562945000	0x0000		IEEE 802.15.4	34	Beacon, Src: 0x0000

(a) GTS allocation request

```
7    4.691880000   0x0002                                IEEE 802.15.4   25   GTS Request

Frame 7: 25 bytes on wire (200 bits), 25 bytes captured (200 bits) on interface 0
Ethernet II, Src: f2:3a:3b:3c:3d:3e (f2:3a:3b:3c:3d:3e), Dst: IPv6mcast_00:00:80:9a (33:33:00:00:80:9a)
IEEE 802.15.4 Command, Src: 0x0002
    Frame Control Field: Command (0x8023)
    Sequence Number: 88
    Source PAN: 0x1234
    Source: 0x0002
    Command Identifier: GTS Request (0x09)
    GTS Request
        .... 0001 = GTS Length: 1
        ...1 .... = GTS Direction: True (Receive)
        ..1. .... = Characteristic Type: True (Allocate GTS)
    FCS: 0xfd2f (Correct)
[Frame Length: 11]
```

(b) Contenido de la trama 7

Figura 6.4: Escenario 1 - ataque GTS - allocation request

Escenario 2

En la captura de tráfico realizada para este escenario se pueden observar que las tramas de *GTS allocation request* se encuentran en las líneas 12, 56 y 64. Las figuras correspondientes a estas tramas se pueden consultar en el Anexo A.11. En la trama número 12, el nodo con identificador 0x0003 solicita tres ranuras para transmisión. Luego, en las tramas 56 y 64 el nodo 0x0002 solicita una ranura de recepción y cinco de transmisión respectimente.
En el siguiente *beacon* que se envía (trama número 72), que se muestra en la figura A.23 del Anexo A.11, se puede observar que el coordinador PAN de la red realizó tres asignaciones, por lo que se envían tres *GTS descriptors*:

- El primero es de transmisión y está asignado al nodo con identificador 0x0003, comienza en la ranura 13 y es de largo tres. Por lo que abarca las ranuras 13, 14 y 15. Esta asignación se corresponde con la solicitud de la trama número 12.

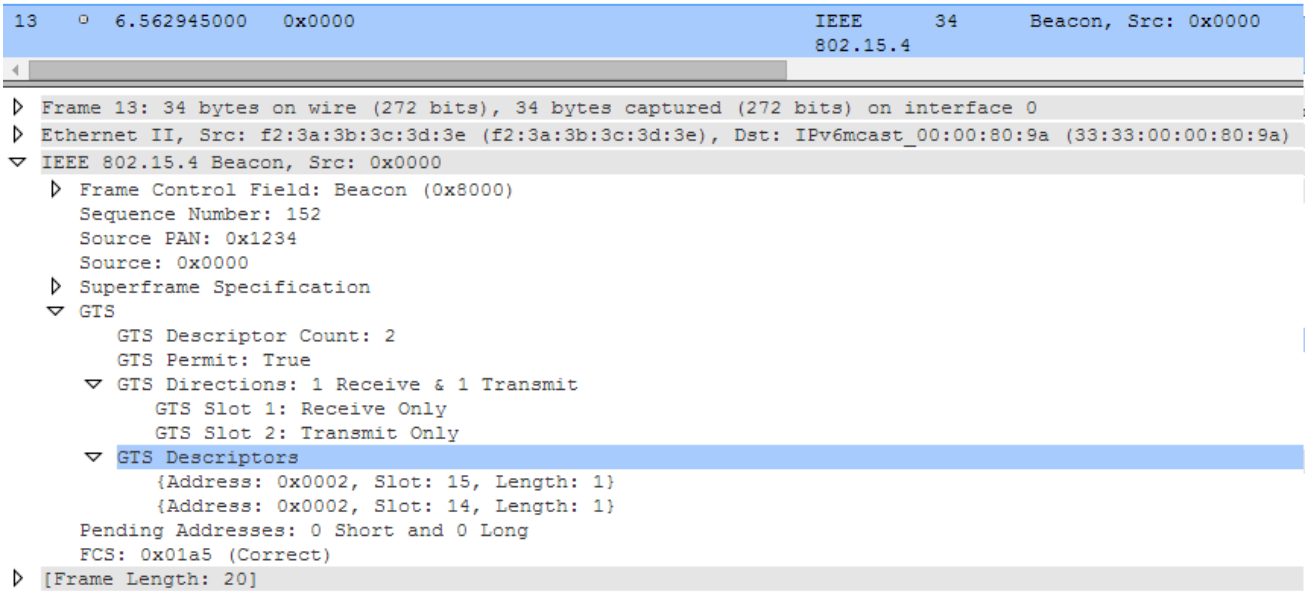


Figura 6.5: Escenario 1 - ataque GTS - beacon

- Luego se asigna una ranura de recepción para el nodo con identificador 0x0002 que comienza en la ranura 12 y es de largo uno. Esta asignación se corresponde con la solicitud de la trama número 56.
- Finalmente, se asigna una ranura de transmisión para el nodo con identificador 0x0002 que comienza en la ranura 7 y es de largo cinco. Por lo que abarca las ranuras 7, 8, 9, 10 y 11. Esta asignación se corresponde con la solicitud de la trama número 64.

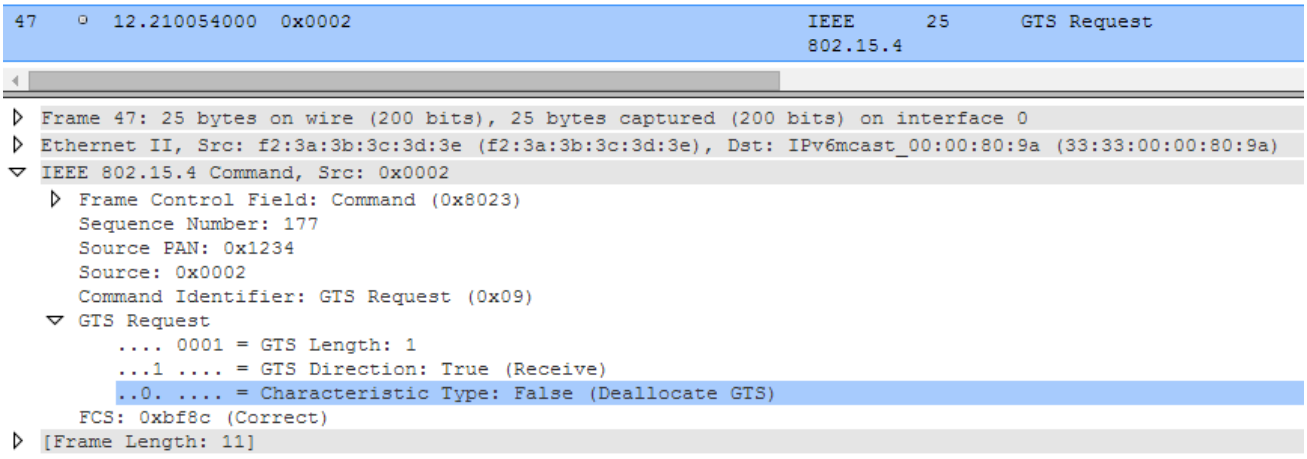
Luego de esto se enciende el nodo atacante que utiliza esta información enviada en el *beacon* para realizar el ataque. En las tramas número 116, 125 y130 el atacante envía las solicitudes de *GTS deallocation* haciéndose pasar por los nodos víctima. Estas tramas son similares a las que se mostraron para el escenario 1, pueden consultarse las mismas en las figuras A.24, A.25 y A.26 del Anexo A.11. Luego de esto el coordinador PAN les quita las ranuras asignadas, por lo que en los próximos *beacons* no hay *GTS descriptors* como puede verse en la figura A.27 del Anexo A.11.

6.2.3 Análisis de resultados

Para obtener una métrica del impacto de este ataque, se desea comparar en una ventana de tiempo dada, cuántos paquetes logra mandar el nodo víctima sin y con el nodo malicioso presente en la red. Para ello, se necesita que el nodo víctima pida nuevas ranuras GTS una vez que estas le fueron desasignados por culpa del nodo atacante.
Se observó (realizando capturas y utilizando la herramienta *printf* de TinyOS) que el nodo víctima, una vez atacado con el nodo malicioso, queda en un estado que se podría definir como “congelado”, donde ningún otro comando de los provistos por la interfaz de capa MAC pueden ser utilizados, entre ellos, el que permite realizar los GTS request para solicitar una nueva ranura y continuar enviando datos. Esto a priori se trata de un bug de la implementación ya que en la implementación solo se consideran dos casos de *deallocation*: cuando el nodo solicita la desasignación y cuando el coordinador decide quitárselo (como se podía observar en la figura 5.4). Cuando el nodo es atacado, se encuentra con una secuencia de eventos que no espera, dado que recibe un *acknoledgement* de un paquete que el no envío (con número de secuencia equivocado) y luego recibe un *beacon* donde se le informa que ya no tiene los GTS asignados.

No.	Time	Source	Destination	Protocol	Length	Info
45	12.186881000	0x0000		IEEE 802.15.4	34	Beacon, Src: 0x0000
46	12.188865000	::	ff02::	IPv6	59	[Malformed Packet]
47	12.210054000	0x0002		IEEE 802.15.4	25	GTS Request
48	12.211373000			IEEE 802.15.4	19	Ack
49	13.010892000	0x206f	0x6c6c	ZigBee	43	Reserved Frame Type, Dst: 0x6c6c, Src: 0x206f[Malformed Packet]
50	13.012966000			IEEE 802.15.4	19	Ack
51	13.068874000	0x0000	0x0002	ZigBee	26	[Malformed Packet]
52	13.069892000			IEEE 802.15.4	19	Ack
53	13.124882000	0x0000		IEEE 802.15.4	31	Beacon, Src: 0x0000
54	13.125882000	::	ff02::	IPv6	56	[Malformed Packet]
55	13.139896000	0x0002		IEEE 802.15.4	25	GTS Request
56	13.140906000			IEEE 802.15.4	19	Ack

(a) GTS deallocation request



(b) Contenido de la trama 47

Figura 6.6: Escenario 1 - ataque GTS - deallocation request

Utilizando la herramienta printf se pudo comprobar que no se ejecuta ninguno de los eventos de la interfaz MLME_GTS que se dispararían en los casos normales de *deallocation* y que se muestran en la figura 5.4 (*confirm* en el caso (a), o *indication* en el caso b). A su vez, el comando que permite mandar datos falla y ni siquiera intenta mandarlos. Lo cual indica que de alguna forma el nodo nota a través del *beacon* que le fueron quitados las ranuras pero no sabe como recuperarse ante ese evento.

Para lograr obtener una métrica de cuán efectivo es el ataque, se trabajó sobre el código del nodo víctima y se logró una implementación que permite realizar lo deseado. Lo que se hace para remediar manualmente este fallo producido por el ataque, es resetear los valores de la capa MAC a sus valores por defecto a través del comando *request* de la interfaz MLME_RESET. Esta no es la situación ideal ya que luego de ésto se debe volver a escanear los canales en busca de los coordinadores que hay y una vez que se escuchan los *beacons* del coordinador de la red buscado, debe sincronizarse con él y enviarle el *GTS request*. Esto toma mucho más tiempo del que llevaría solamente volver a enviar un *GTS request*, haciendo que el ataque sea aún más efectivo.

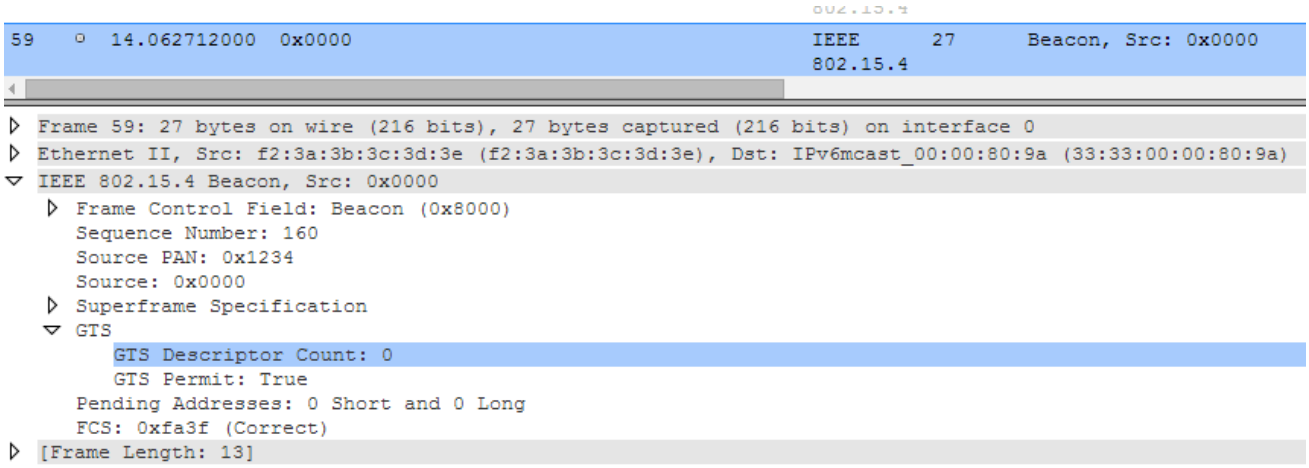


Figura 6.7: Escenario 1 - ataque GTS - beacon

En un intervalo de un minuto, un nodo ejecutando la aplicación del nodo víctima logra transmitir 59 paquetes de datos. Sin embargo, el mismo nodo, bajo la influencia del atacante solo logra mandar con éxito 6 paquetes. Con lo cual se puede concluir que el ataque reduce la tasa de transmisión a un 10 % de lo que puede transmitir normalmente.

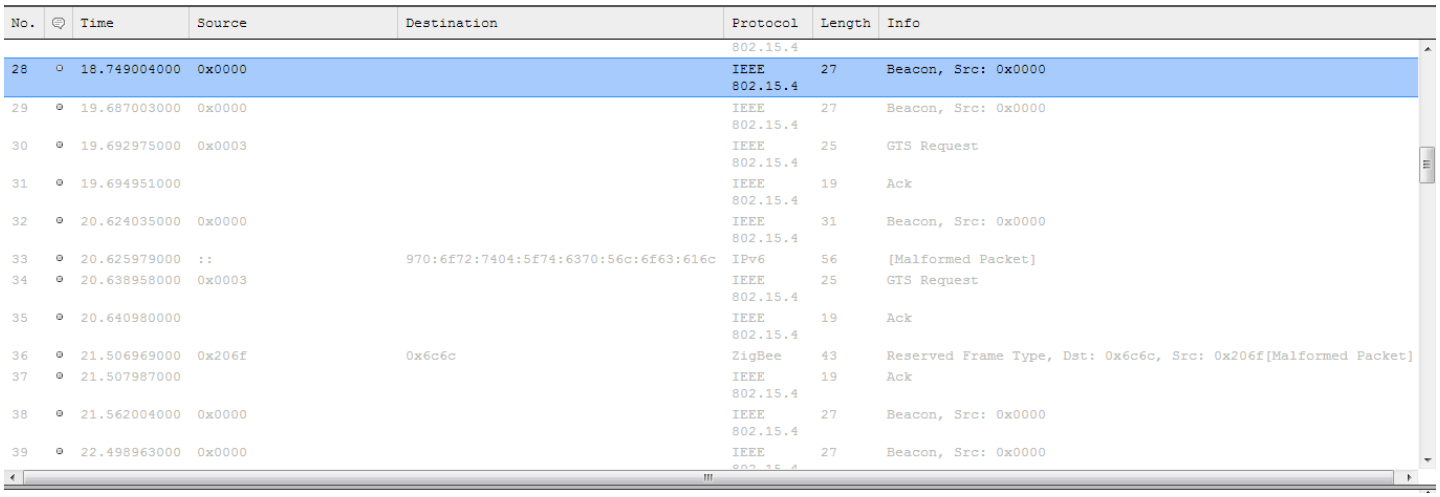


Figura 6.8: Captura de los paquetes intercambiados durante el ataque.

En la figura 6.8 se puede observar el intercambio de paquetes realizados durante el ataque. Primero hay una secuencia de *beacons* que el coordinador de la red envía, luego el nodo víctima envía un *GTS request* solicitando una ranura para enviar datos, a lo que el coordinador le responde con un ACK y envía un *beacon* con el *GTS descriptor* indicando la asignación. Cuando el atacante escucha ese *beacon* envía un *GTS request* de *deallocation*, pero el nodo víctima puede mandar un paquete de datos (paquete 36) antes de que le sea quitada la ranura en el siguiente *beacon* (paquete 38). Este desfasaje, entre que el atacante envía el *deallocation request* y que este se le quita al nodo víctima, puede verse con mayor claridad en el diagrama de la figura 6.9. Por lo tanto, cada vez que un nodo pide una ranura, logra enviar un paquete de datos antes de que el atacante logre quitarle la asignación. Es por esta razón

que la tasa de transmisión del nodo víctima se reduce considerablemente pero nunca llega a 0 %.

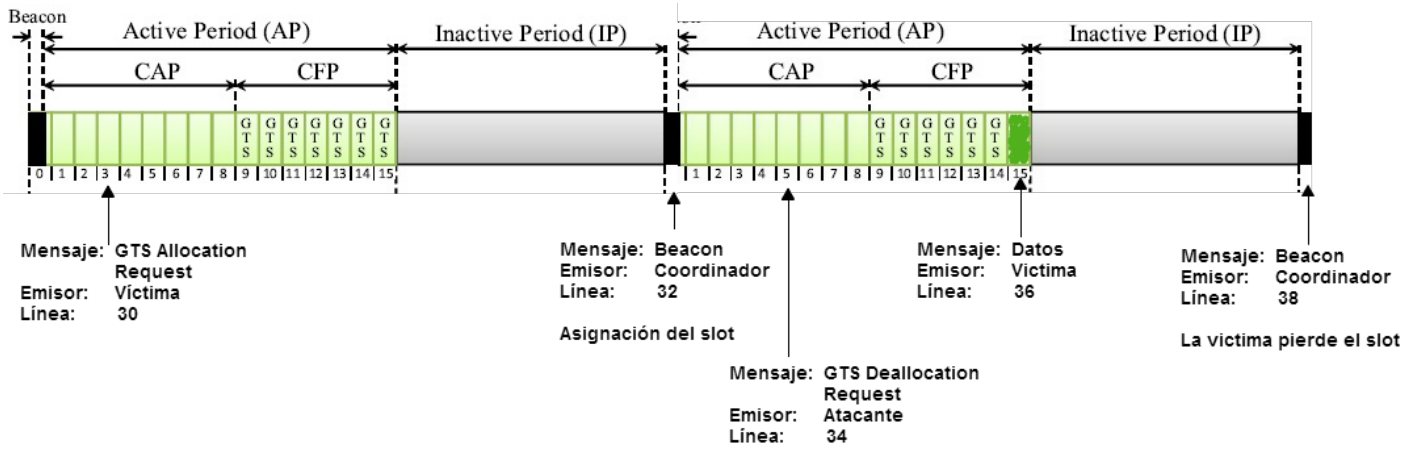


Figura 6.9: Distribución de las tramas enviadas por el coordinador, el nodo víctima y el atacante dentro de la estructura de supertrama.

6.2.4 Conclusiones

En todos las capturas realizadas se observó las tramas esperadas. El nodo malicioso logra hacerse pasar por los nodos (usando sus identificadores) y manda las solicitudes de *deallocation* correspondientes (con la información necesaria: largo y dirección) que extrae de los *beacons* que escucha en el canal. Luego el coordinador, una vez procesados los *deallocations*, actualiza la lista de *GTS Descriptors* del *beacon*. Esto se hace hasta que no quedan ningún nodo con ranuras dedicadas asignadas (ya que el malicioso envía un *deallocation* por cada uno de los nodos que tienen algo asignado).

CAPÍTULO

7

Ataques en capa de red: CTP

Las WSNs son, generalmente, redes multi-salto que requieren de las funciones de reenvío y enrutamiento para que los nodos puedan enviar sus paquetes a la estación base. Diversos protocolos de reenvío y enrutamiento han sido propuestos para este tipo de redes, siendo más convenientes unos u otros dependiendo de la aplicación específica de la red.
En este caso, estudiaremos el protocolo Collection Tree Protocol (CTP), que es un protocolo de recolección que forma una topología de árbol en base al costo de envío calculado a partir del número de transmisiones esperadas (ETX) para que un mensaje pueda llegar a una estación base.
Las estaciones base se anuncian como las raíces del árbol para que los otros nodos le envíen sus paquetes. CTP no se basa en una dirección de destino, es decir, un nodo no envía un paquete a una raíz en particular, sino que se determina implícitamente cuando elige su próximo salto o nodo padre.
Este capítulo presenta al funcionamiento del protocolo CTP así como vulnerabilidades en su diseño y en la estimación de la calidad de los enlaces que permiten a un nodo malicioso explotarlas. La especificación de este protocolo está disponible en el TEP 123 [80] y su implementación está disponible en la distribución TinyOS 2.x.
En este capítulo primero se presenta las características del protocolo CTP, los módulos que lo componen y como estos interaccionan entre sí para lograr las funciones de enrutamiento y reenvío. Esto será vital para comprender mejor el funcionamiento del protocolo, implementar el ataque y validar su correcto funcionamiento. Luego se detallan uno a uno los ataques que fueron implementados, las vulnerabilidades que estos explotan y los pasos seguidos para lograr su implementación.
En el próximo capítulo se muestran los resultados obtenidos en las pruebas realizadas para validar su desempeño y se analizan los efectos que estos ataques pueden tener en una red de sensores.

7.1 Funcionamiento de CTP

CTP utiliza varios mecanismos para incrementar la confiabilidad pero es un protocolo “mejor esfuerzo”. CTP cuenta con una estimación de la calidad del enlace de sus nodos vecinos, esto le permite estimar la cantidad de transmisiones que le llevará a un nodo enviar un paquete unidifusión para que se reciba correctamente el correspondiente ACK.
CTP es un protocolo híbrido basado en la calidad del enlace, esto significa que computa las calidades de los enlaces de forma bidireccional para algunos nodos y de forma unidireccional para otros. La estimación de la calidad de los enlaces bidireccionales es computada de forma independiente, sin la cooperación de los nodos de la red

Algunos de los problemas comunes a los que CTP hace frente son los bucles de enrutamiento y los paquetes duplicados. Por otro lado, este protocolo utiliza el envío adaptativo de *beacons* para no generar tráfico innecesario en el cómputo de las rutas.

Bucles de enrutamiento

En general, estos se producen cuando un nodo elige un nuevo camino que tiene un ETX significativamente mayor que el ETX del camino anterior, por ejemplo, por la pérdida de conectividad con su padre. Si el nuevo camino incluye un nodo que anteriormente era un hijo, entonces se forma un bucle. CTP utiliza dos mecanismos para evitar los bucles. Primero cada paquete CTP de datos contiene el valor del ETX actual total para llegar a la raíz. De modo que si un nodo recibe un paquete con un valor de ETX inferior al suyo es porque existe una inconsistencia en el árbol. De esta forma los paquetes de datos son utilizadas para analizar la topología y detectar problemas de enrutamiento. Para resolver esta inconsistencia se envia un *beacon* solicitándole (a través del bit de *pull*), a los demás nodos que también envíen sus datos de estimación. El segundo mecanismo de CTP consiste en no tener en cuenta las rutas con un ETX mayor a una cierta constante razonable.

Paquetes duplicados

Cuando un nodo recibe un paquete de datos y transmite un ACK que no es recibido por su destinatario puede ocurrir la aparición de paquetes duplicados en la red, debido a la retransmisión de los paquetes de datos.
Los bucles de enrutamiento dificultan la supresión de duplicados, ya que pueden provocar que un nodo reciba un paquete más de una vez pero que este paquete sea el mismo que había sido reenviado anteriormente.
Por lo tanto, si un nodo suprimiera duplicados (basado únicamente en la dirección de origen y en el número de secuencia) un paquete que recorre un ciclo y vuelve sería descartado como duplicado.
Para solucionar este problema, los paquetes de datos CTP llevan un campo con el tiempo que hayan vivido (THL), el cual se incrementa en cada salto de enrutamiento. De esta forma, en una retransmisión el THL tiene el mismo valor para los dos paquetes enviados, mientras que en una repetición por bucle es poco probable que el THL sea el mismo.

Beacons adaptativos basado en el algoritmo Trickle

La implementación adapta la tasa de envío de *beacons* basándose en el dinamismo de la red. Enviando pocos *beacons* cuando la topología es estable para no congestionar la red con este tipo de información pero con una adaptación más rápida cuando ocurren cambios. De esta forma cuando la red detecta un bucle o inconsistencia, los nodos comienzan a enviar tráfico de control más rápidamente para reparar la topología. Para lograr esto, el tiempo de envío de *beacons* se selecciona de forma aleatoria dentro de un intervalo de tiempo que crece exponencialmente hasta llegar a un máximo [81]. Este tiempo es reseteado al mínimo cuando ocurre uno de los siguientes eventos:

1. La tabla de ruteo está vacía.
2. El ETX de la ruta elegida se incrementa en un valor mayor a uno.
3. El nodo escucha un *beacon* con el bit de *pull* en uno.

La implementación del protocolo CTP puede ser encontrada en el directorio tos/lib/net/ctp de TinyOS 2.0. Esta implementación se encuentra estructurada en tres componentes:
1) Motor de reenvío: mantiene una cola de los paquetes a enviar. Este decide si mandarlos y cuando hacerlo.
2) Motor de enrutamiento: utiliza las estimaciones de los enlaces e información de la red para decidir a que nodo vecino elegir como siguiente salto.
3) Estimador de enlaces: es responsable por estimar el ETX de los enlaces hacia los vecinos.

Estos tres módulos interactúan entre sí y residen en la capa de red de la pila de protocolos. También interactúan con otras capas a través de interfaces bien definidas. A continuación se detalla en mayor profundidad las responsabilidades y particularidades de cada uno de estos tres componentes.

7.1.1 Motor de reenvío

El componente tos/lib/net/ctp/CtpForwardingEngineP implementa el motor de reenvío que tiene cinco responsabilidades:

1. Enviar los paquetes hacia el próximo salto, retransmitirlos si es necesario y comunicarle al estimador de enlaces información sobre los ACKs.
2. Decidir cuando transmitir un paquete y cuando no.
3. Detectar inconsistencias en el enrutamiento e informárselo al motor de enrutamiento.
4. Mantener una cola de los paquetes a enviar, que incluye los paquetes a reenviar y los generados localmente por el propio nodo.
5. Detectar duplicados.

Las cuatro funciones principales del motor de reenvío son las siguientes:
1) Recepción de paquetes (a través del evento SubReceive.receive()). Esta función decide si el nodo debe enviar o no un paquete. Comprueba si hay duplicados utilizando un pequeño caché de los paquetes recibidos recientemente. Si se decide que un paquete no es un duplicado, se llama a la función forwad.
2) Reenvío de paquetes (a través de la función forward()). Esta función da formato al paquete para el reenvío. Primero, se comprueba el paquete recibido para ver si hay posiblemente un bucle en la red, luego se comprueba si hay espacio en la cola de transmisión. Si no hay espacio, se descarta el paquete y se establece el bit de congestión en uno. Si la cola de transmisión estaba vacía, entonces se lanza la tarea sendTask.
3) Transmisión de paquetes (a través de la tarea sendTask()). La tarea de envío toma el primer paquete de la cola de reenvío, establece como destino el siguiente salto (cuya dirección se la pide al motor de enrutamiento), y lo envía a la capa AM (*Active Message*) que es la capa que se encuentra debajo en la pila de protocolos.
4) Decidir qué hacer después de una transmisión de paquetes (a través del evento SubSend.sendDone()). Cuando el envío se completa, se dispara el evento sendDone que examina el resultado del envío del paquete. Si se recibió el ACK del mismo, se quita el paquete de la cola de transmisión. Si el paquete fue generado localmente, se dispara el evento sendDone( ) al cliente anterior. Si se ha reenviado, se devuelve el paquete al *pool* de mensaje disponibles para reenvío.
Si existen paquetes restantes en la cola, se inicia un temporizador aleatorio que vuelve a disparar el proceso de reenvío. Este temporizador se utiliza para limitar la tasa de envío de CTP de modo que no envíe paquetes tan pronto como le sea posible con el fin de prevenir las colisiones a lo largo de la ruta.

7.1.2 Motor de enrutamiento

Este componente es el que selecciona el próximo salto en la ruta a la estación base, para ello se mantiene una tabla con los ETXs de los caminos que se pueden alcanzar a través de los nodos vecinos. El ETX hasta la raíz es la suma del ETX del enlace con el padre (ETX_{1-HOP}) más el ETX del padre hasta la raíz ($ETX_{multi-HOP}$). El nodo elegido como padre es el que tiene menor ETX para todo el camino hacia la estación base, es decir, menor valor de la suma $ETX_{1-HOP} + ETX_{multi-HOP}$.
Las principales funciones del motor de enrutamiento son:
1) updateRouteTask que se llama periódicamente para eligir un nuevo padre.
2) sendBeaconTask transmite la información de la ruta actual a los vecinos.
3) el evento que permite la recepción de *beacons* y actualiza la tabla de vecinos.
El componenete tos/lib/net/ctp/CtpRoutingEngineP es el que implementa el motor de enrutamiento.

7.1.3 Estimador de enlaces

El estimador de calidad de enlace en CTP es el ETX (*Expected Transmission*), que se informa a los nodos vecinos mediante los *beacons* y de los paquetes de datos.
El ETX de un nodo vecino se calcula a partir de información del enlace de subida como del enlace de bajada. Esta estimación se coloca en una tabla estimadora de enlaces. En esta tabla se guardan los datos de los diez mejores vecinos de un nodo.
El ETX de un nodo es igual al ETX de su padre mas el ETX del enlace que lo comunica con su padre. Esta medida es aditiva por lo que en una ruta con múltiples saltos, la métrica es la suma de las ETX de los saltos individuales.
Las raices del árbol tienen un ETX de cero y si un nodo no tiene una ruta, anuncia un ETX infinito representado con 0xFFFF. CTP elige la ruta que tenga el menor valor de ETX y representa los valores de ETX como un número real de 16 bits de punto fijo con una precisión de décimas. Por ejemplo, si se recibe un *beacon* con un valor de ETX igual a 45, entonces este representa un ETX de 4.5, mientras que un valor de ETX 10 representa un ETX de 1.
CTP puede utilizar dos mecanismos de estimación: el protocolo LEEP (*Link Estimation Exchange Protocol*) y el protocolo 4bitLE (*4 bit Link Estimator*).
El estimador bidireccional del enlace ETX utiliza *beacons* periódicas enviados en *broadcast* para medir las tasas de recepción entrante y los ACKs de los paquetes de datos enviados de forma *unicast* para medir la tasa saliente. El estimador solo produce estimaciones para los nodos vecinos a un salto de distancia (ETX_{1-HOP}), el $ETX_{multi-HOP}$ es calculado por el motor de enrutamiento y se produce un valor de ETX cada 5 transmisiones, en el caso de que ninguna transmisión haya sido exitosa se establece el ETX con un valor de 6.

7.1.3.1 Protocolo LEEP

Este protocolo esta definido en el TEP 124 [82]. La calidad del enlace entre un par de nodos dirigidos A y B es la probabilidad de que un paquete transmitido por A sea recibido con éxito por B. Mientras que la calidad bidireccional del enlace entre un par de nodos no dirigido (A, B) es el producto de la calidad del enlace de (A, B) y (B, A). Esta definición asume las pérdidas de enlace independientes e incluye el caso cuando la calidad del enlace de (A, B) y (B, A) son diferentes.
Por eso se manejan los conceptos de calidad de enlace *in-bound* y *out-bound* que pueden verse en la figura 7.1.

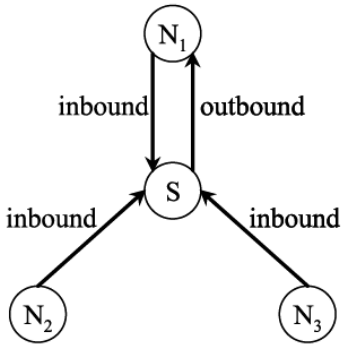


Figura 7.1: Tipos de calidad estimada por LEEP[14].

En un par de nodos (A, B) la calidad del enlace *in-bound* es un valor en el rango de 0 a 255 que describe la calidad del enlace de A a B estimado por el nodo B a partir de la cantidad de paquetes recibidos con éxito de A. Se consideran todos los paquetes transmitidos, esto incluye los *beacons* y los ACKs en respuesta a datos enviados. También se pueden utilizar otros indicadores de calidad de enlace

tales como LQI (*Link Quality Indicator*) y RSSI (*Received Signal Strength Indicator*) proporcionadas por la radio en el nodo B.
Por otro lado, en un par de nodos (A, B), la calidad del enlace *out-bound* se define como la calidad del enlace de B a A. B puede determinar la calidad del enlace *out-bound* si A anuncia su calidad *in-bound*. Por lo que para computar la calidad del enlace bidireccional, la calidad de los enlaces de tipo *in-bound* debe ser compartida entre los nodos vecinos.
Este protocolo mantiene un número de secuencia que es incrementado en uno para cada paquete LEEP que se envía. Este número se incrementa aunque la capa de enlace deba retransmitirlo. Esto se debe a que este número de secuencia es el utilizado para contar la cantidad de paquetes perdidos para estimar la calidad *in-bound* desde el nodo emisor.
Por otro lado, para estimar la calidad *out-bound*, los nodos comparten un conjunto de *Link Information Entries* que desciben la calidad *in-bound* de los enlaces hacia un subconjunto de nodos vecinos. Es decir, una *Link Information entry* creada por el nodo K es una tupla (N, Q) donde Q es la calidad *in-bound* del enlace desde el nodo N a K. De esta forma, los nodos pueden extraer de las tramas LEEP la calidad *out-bound* del enlace que lo comunica con el origen del *beacon*, buscando una *Link Information entry* que contenga su identificador de nodo.
La implementación trata de enviar una *Link Information entry* para todos los vecinos en su tabla de vecinos enviando el mayor largo de trama posible. Si no hay espacio suficiente para enviar todas las *Link Information entries*, se utiliza una política *round-robin* para seleccionar las entradas que se enviarán. Las tramas LEEP se transmiten cuando se envían los *beacons* CTP, enviados como carga útil LEEP, se envían.
El componente tos/lib/net/le/LinkEstimatorP implementa este estimador de enlaces.
Algunos de los campos importantes de la tabla estimadora (estructura *neighbor_table_entry_t* del archivo LinkEstimator.h) de este protocolo son los siguientes:

- ll_addr: identificador del nodo vecino.
- lastseq: número de secuencia del último *beacon* recibido desde ese vecino.
- rcvcnt: número de *beacons* recibidos luego de la última actualización de estimación disparada por un *beacon* de ese vecino. Esto ocurre cada una cantidad de *beacons* configurable en la variable BLQ_PKT_WINDOW, que por defecto es 3.
- failcnt: número de *beacons* perdidos luego de la última actualización de estimación disparada por un *beacon*. Esto se calcula utilizando los números de secuencia, en caso de recibir *beacons* con «huecos» en la numeración o asignando un valor.
- data_success: número de paquetes de datos recibidos correctamente desde ese vecino particular (es decir, para los cuales se recibió un ACK) desde la última actualización de estimación disparada por un paquete de datos. Esto ocurre cada una cantidad configurable de paquetes definida en DLQ_PKT_WINDOW, que por defecto es 5.
- data_total: número de intentos de transmisión de datos desde determinado nodo desde la última actualización de estimación disparada por un paquete de datos.

Observar que la actualización a través de *beacons* se realiza cada vez que se llega a una cantidad BLQ_PKT_WINDOW de *beacons* recibidos o perdidos. Es decir, se detecta un aumento de BLQ_PKT_WINDOW unidades en el número de secuencia. Si no se recibe ningún *beacon*, no se puede calcular cuantos se perdieron y en consecuencia no se actualiza el ETX por esta condición. De todos modos, se puede actualizar la estimación utilizando otros datos o en el peor de los casos luego de cierto tiempo esas entradas ya no son consideradas.
En el caso de la actualización a través de paquetes de datos siempre se «sabe» si llegaron o se perdieron, ya que en caso de no recibir un ACK en un tiempo razonable se da por perdido el paquete y se contabiliza apropiadamente.

Como ya se dijo, este estimador calcula la calidad del enlace, en base a los datos/ACKs, a los *beacons* recibidos de otros nodos y a las estimaciones que los otros nodos anuncian. El cálculo que se realiza para actualizar el ETX en base a los datos/ACKs recibidos puede apreciarse en el algoritmo 8. Este se basa en que, si se enviaron t paquetes de datos y se recibieron s ACKs, entonces la calidad del enlace es $\frac{t}{s}$. Si no se recibieron ACKs entonces la calidad del enlace es igual al número de intentos fallidos de transmisión desde la última transmisión exitosa.

Cuadro 8 $ActualizarDLQ(DataLinkQuality)$

si data_total $\geq$ DLQ_PKT_WINDOW **entonces**
  si data_success = 0 **entonces**
    q = 10 x data_total
  si no
    q = $\frac{10 \times data\_total}{data\_success}$
    data_success = 0
    data_total = 0
  fin si
  etx = $\frac{ALPHA \times ext+q}{10}$
fin si

El cálculo que se realiza para actualizar el ETX en base a los *beacons* recibidos es análogo al realizado para los datos/ACKs como puede apreciarse en el algoritmo 9. Este se basa en que, si se recibieron s *beacons* y enviaron t en total, entonces la calidad del enlace es $\frac{s}{t}$.

Cuadro 9 $ActualizarBLQ(BeaconLinkQuality)$

total = rcvcnt + failcnt
si total $\geq$ BLQ_PKT_WINDOW **entonces**
  newEst = $\frac{250 \times rcvcnt}{total}$
  inQuality = $\frac{ALPHA \times inQuality+newEst}{10}$
  q = $\frac{2500}{inQuality}$
  rcvcnt = 0
  failcnt = 0
  etx = $\frac{ALPHA \times ext+q}{10}$
fin si

Notar que en ambos algoritmos el ETX se calcula utilizando un filtro de suavizado exponencial y el nuevo calculo de la calidad del enlace tiene una contribución regulada por la variable ALPHA, que por defecto toma el valor 9. Por otro lado, observar que los calculos realizados evitan las operaciones en punto flotante que son costosas.

7.1.3.2 **Protocolo 4bitLE**

Este estimador se basa en el anterior e introduce los conceptos encontrado en [15]. El principal cambio, radica en que este protocolo estima la calidad de los enlaces utilizando información de la capa física, la capa de enlace y la capa de red. Esto se debe a que la visión de cada capa es parcial y puede dar lugar a errores. Su nombre se debe a que utiliza 4 bits como puede verse en la figura 7.2:

1. Un bit de la capa física, llamado *white bit*, que mide la calidad del canal durante el envío de un paquete. Un paquete con *Bit Error Rate* (BER) bajo es probable que provenga de un mejor enlace que otro con muchos errores de bit. Este bit no es considerado en el estimador LEEP.

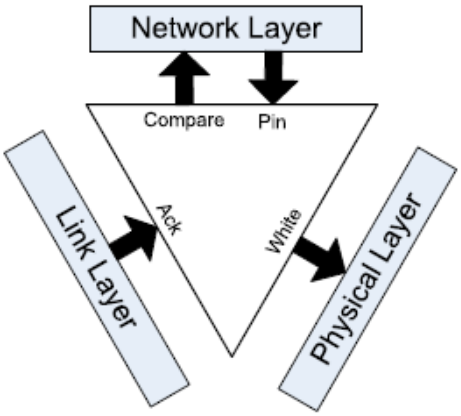


Figura 7.2: Estimador 4bitLE [15].

2. Un bit de la capa de enlace, llamado ACK bit. Si un enlace se cae y en consecuencia ocurren pérdidas de paquetes el estimador físico no lo reflejará. Por lo que se utiliza el bit de ACK que indica si el nodo ha recibido un ACK en respuesta a una transmisión.

3. Dos bits de la capa de red, conocidos como *pin bit* y *compare bit*. En *pin bit* le indica al estimador que no retire un enlace de la lista de estimaciones porque está en uso, de esta forma se puede ver cuáles son los enlaces más valiosos para el desempeño de las capas superiores. El *compare bit* puede ser utilizado por el estimador para preguntarle a la capa de red si un enlace se ve prometedor, de esta forma se pueden detectar enlaces que forman bucles o que desconecten la red.

El formato de los paquetes en análogo al del protocolo LEEP. Sin embargo, a pesar de que 4bitLE envía las *Link Information entry* este no utiliza esta información para calcular la calidad del enlace. La calidad *in-bound* del enlace se computa a partir de los *beacons* recibidos, mientras que la calidad *out-bound* se computa únicamente a partir de los ACKs recibidos. Notar que esta última solamente se puede estimar para el nodo padre. Por lo que algunos enlaces solo tendrán una estimación unidireccional mientras que para otros esta será bidireccional.
El componente tos/lib/net/4bitle/LinkEstimatorP implementa este estimador de enlaces.

7.1.4 Interacción entre módulos

Los módulos de CTP no funcionan de forma independiente sino que interactúan a través de un conjunto de interfaces bien definidas. El motor de enrutamiento obtiene la métrica ETX 1-hop del estimador de enlaces para calcular el ETX multi-hop. El motor de reenvío debe obtener el identificador del padre actual y comprobar con el motor de enrutamiento el estado de congestión de los vecinos.
La figura 7.3 muestra la interacción entre los distintos módulos de CTP y cual es el componente que envía los distintos tipos de paquetes.
Los paquetes de datos se envían a través del motor de transmisión, quién le agrega su encabezado (*CTP data frame header*). Los paquetes de enrutamiento o *beacons* son enviados por el motor de enrutamiento a través del estimador de enlaces. Para ello, el motor de enrutamiento agrega su encabezado CTP (*CTP Routing Frame header*) y luego es enviado al estimador de enlaces que le agrega un encabezado (*LE header*) y un pie (*LE footer*).
En resumen, los paquetes de datos son manejados únicamente por el motor de reenvío. Mientras que los paquetes de enrutamiento se generan y procesan tanto por el motor de enrutamiento como por el estimador de enlaces.

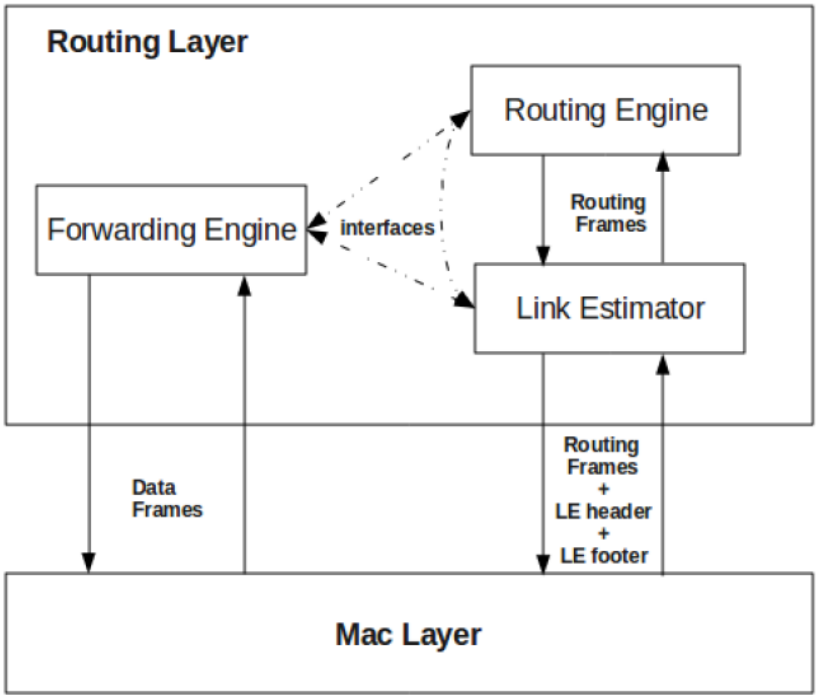


Figura 7.3: Interacción entre los módulos CTP y flujo de mensajes [16].

En el Anexo A.9 se muestra la estructura de las tramas que contienen los paquetes de datos y enrutamiento CTP, mas los datos agregados por la capa física y MAC (para un transmisor de radio siguiendo el estándar IEEE 802.15.4). Luego, en el Anexo A.10 se presentan las estructuras que representan esos paquetes CTP en la implementación del protocolo y la verificación que se realizó mediante capturas para comprobar que los formatos de los paquetes CTP sean realmente como se describieron. En este Anexo también se muestra la estuctura del resto de los bytes que se transmiten en las tramas capturados o en las simulaciones Avrora. Se consideran las tramas enviadas por los transmisores CC2420 y CC1000. Recordar que el transmisor CC1000 no cumple con el estándar IEEE 802.15.4, mientras que CC2420 si lo hace.

7.2 Ataque sinkhole

En esta sección se presenta la lógica y la implementación realizada para el ataque *sinkhole* en el protocolo CTP.

Lógica

En este ataque se utilizaron varías estrategias que se mencionan a continuación para poder engañar a los nodos vecinos y así atraer el tráfico.

Estrategias 1 y 2

Como fue explicado anteriormente, un nodo estima el ETX hasta la raíz como la suma del ETX_{1-HOP} del enlace con el padre más el $ETX_{multi-HOP}$ del nodo padre. Por otro lado, un nodo puede determinar

la calidad del enlace *out-bound* (es decir, de su enlace de subida) a partir de lo que los otros nodos anuncian como estimación *in-bound* hasta ese nodo. De esta forma, un nodo malicioso puede manipular la estimación *out-bound* de los demás nodos enviando valores de ETX falsos en los *beacons*.
Dos de las estrategias que podría utilizar un nodo malicioso serían mentir sobre:
(E1) La calidad *in-bound* que estima hacia un nodo vecino afectando de esta forma el valor de la calidad *out-bound* de los nodos víctima. Observar que de esta forma solo se afecta el cómputo de la calidad bidireccional correspondiente al enlace que comunica al nodo víctima con el atacante. Es decir, solo se afecta el ETX_{1-HOP}.
(E2) El $ETX_{multi-HOP}$ hacia la raíz.
Observar que en el cálculo del ETX hasta la raíz es esperable que la estrategia (E1) tenga una contribución mucho menor que la de la estrategia (E2), en especial, cuando el atacante se encuentra a varios saltos de la estación base.

Estrategia 3

Otra estrategia posible sería afectar la estimación de la calidad *in-bound* que un nodo víctima realiza del enlace con el nodo atacante. Esta idea fue extraída de [14]. La calidad *in-bound*, como se dijo, es estimada a partir de la cantidad de paquetes recibidos con éxito considerando todos los paquetes transmitidos, esto incluye los *beacons* y los ACKs en respuesta a datos enviados. Por lo tanto, se puede utilizar el envío de *beacons* para influenciar el cálculo que los nodos vecinos realizan [14]. En esta estrategia (E3) el nodo malicioso explotará los intervalos adaptativos y el hecho de que cada una cierta cantidad de *beacons* se computa el ETX.
CTP utiliza intervalos adaptativos, es decir, el intervalo de tiempo entre el envío de dos *beacons* debe duplicarse. El nodo atacante transmitirá los *beacons* utilizando siempre el mínimo intervalo posible. De esta forma los nodos vecinos recibirán *beacons* del nodo malicioso con más frecuencia que desde los otros nodos de la red.
En la figura 7.4 se muestra como son los intervalos de transmisión de *beacons* para un nodo 'bueno'y para un nodo malicioso. Se puede observar que si el nodo malicioso envía los *beacons* con el menor intervalo de tiempo posible, este envía siete *beacons* mientras que un nodo 'buenoénvía solamente tres. Esto indica que la calidad *in-bound* para el nodo malicioso va a ser estimada dos veces mientras que para el resto de los nodos solo se computa una vez.
Este comportamiento no se ve sospechoso, ya que en CTP, los nodos utilizan el intervalo mínimo de transmisión de *beacons* si no tienen rutas a la estación base. También podría darse el caso donde un nodo en el rango de radio del nodo que envía con el mínimo intervalo esté solicitando *beacons* (con el bit P) pero que los otros no lo escuchen por el problema de la estación oculta, y ese sea el motivo por el que el nodo envía los *beacons* con el mínimo intervalo posible.

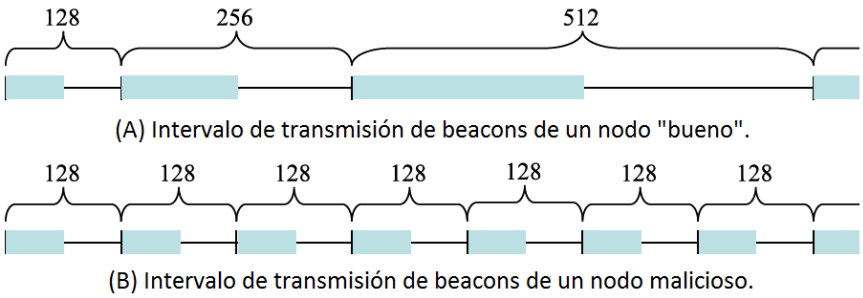


Figura 7.4: Intervalos de transmisión de *beacons* para un nodo malicioso y un nodo 'bueno'de la red [14].

Por otro lado, hay que notar que si se pierden muchas tramas en el enlace entre el atacante y sus víctimas, esta no es una buena estrategia, ya que se aceleraría el computo de una mala calidad de enlace.

Estrategia 4

Cuando se envían *beacons* conteniendo información falsa sobre las estimaciones, estos valores no deben enviarse en los paquetes de datos. Por lo que se continuarán realizando los calculos de las estimaciones reales para ser enviadas en este tipo de paquetes. De otro modo, cuando el nodo padre reciba un paquete con un valor de ETX menor al suyo detectará una inconsistencia en el árbol y se solicitará que todos los nodos envíen *beacons* para volver a calcular las rutas a la raíz, generando inestabilidad en la estructura de la red.

Programación y componentes utilizados

En esta sección se presenta como fue implementado el ataque en base a las estrategias anteriormente presentadas. Para implementar este ataque se modificaron los componentes CtpRoutingEngineP y LinkEstimatorP.

Cuadro 10 Modificaciones en el módulo estimador (LinkEstimatorP)

```
am_addr_t padre;

command error_t LinkEstimator.clearDLQ(am_addr_t neighbor) {
        (...)
        padre = neighbor;
}

uint8_t addLinkEstHeaderAndFooter(message_t * ONE msg, uint8_t len) {
        (...)
        for (i = 0; i < NEIGHBOR_TABLE_SIZE && j < maxEntries; i++) {
                neighbor_stat_entry_t * COUNT(neighborCount) neighborLists;
                neighborLists = TCAST(neighbor_stat_entry_t * COUNT(maxEntries), footer->neighborList);

                if ((NeighborTable[i].flags & VALID_ENTRY) && (NeighborTable[i].flags & MATURE_ENTRY)) {
                        neighborLists[j].ll_addr = NeighborTable[i].ll_addr;

                        if (neighborLists[j].ll_addr == padre) {
                                neighborLists[j].inquality = 0;
                        }
                        else {
                                neighborLists[j].inquality = NeighborTable[i].inquality;
                        }
                }
    }
        (...)
}
```

La implementación del protocolo CTP se modifica en el nodo atacante para explotar las vulnerabilidades encontradas. Como se muestra en el cuadro 10, se agrega una variable en el módulo estimador que lleva registro de cual es el nodo padre actual (este módulo normalmente no tiene conocimiento de esta información debido a que no se encuentra entre sus tareas conocerlo).
El comando LinkEstimator.clearDLQ() es invocado cada vez que se cambia de nodo padre, de forma de limpiar la estimación de la calidad del enlace realizada a través de los paquetes de datos que se calculó. Es en este comando donde se asigna a la nueva variable la dirección del nodo padre.
Por otro lado, el método addLinkEstHeaderAndFooter, agrega el cabezal y el pie LEEP a la trama de enrutamiento generada por CTP. En este método se recorren una a una las entradas de la tabla que contiene las estimaciones calculadas para los nodos vecinos. Si una entrada es válida se agrega al pie LEEP el identificador del nodo y la estimación. En particular, si es el nodo padre, se agrega el valor cero. Es decir, el atacante anuncia que conoce un camino de costo cero hasta la raíz del árbol.
Por otra parte, dentro del módulo de enrutamiento CTP se encuentra el método decayInterval() que se muestra en el cuadro 12, este es uno de los métodos que participa en la implementación del mecanismo de envío adaptativo de *beacons*. Cada vez que este método es invocado se duplica el intervalo actual que se espera entre dos *beacons* consecutivos. El crecimiento de este intervalo de tiempo entre *beacons* está acotado por el valor *maxInterval*. Dentro del intervalo actual, *currentInterval*, el instante en el que

se envía el próximo *beacon* es calculado de forma aleatoria. Para quitar el mecanismo adaptativo de envío de *beacons* simplemente se quita la línea que duplica el intervalo actual.

Cuadro 11 Mecanismo de envío adaptativo de *beacons* del módulo de enrutamiento (CtpRoutingEngineP)

```
void decayInterval() {
        //currentInterval *= 2;

        if (currentInterval > maxInterval) {
                currentInterval = maxInterval;
        }
        chooseAdvertiseTime();
}
```

Cuadro 12 Envío de *beacons* en el módulo de enrutamiento (CtpRoutingEngineP)

```
task void sendBeaconTask() {
        (...)
        beaconMsg->parent = routeInfo.parent;

        if (state_is_root) {            //si es el nodo raíz
                beaconMsg->etx = routeInfo.etx;
        }
        else if (routeInfo.parent == INVALID_ADDR) {
                beaconMsg->etx = routeInfo.etx;
                beaconMsg->options |= CTP_OPT_PULL;
        } else {
                beaconMsg->etx = 0;     //routeInfo.etx + call LinkEstimator.getLinkQuality(routeInfo.parent);
        }

        call BeaconSend.send(AM_BROADCAST_ADDR, &beaconMsgBuffer, sizeof(ctp_routing_header_t));
```

La tarea sendBeaconTask() mostrada en el cuadro 11 es la encargada de armar la trama de enrutamiento CTP, cargando los datos adecuados en cada caso. Si el nodo es la raíz, se envían los datos de la estructura routeInfo. Por ejemplo, si la raíz es el nodo de identificador 1, en el *beacon* se informa que el nodo 1 llega a su padre, que es el nodo 1 en costo cero. Cuando el nodo no es una raíz y no conoce una ruta para llegar a la misma también se envían los datos de la misma estructura pero en este caso se envía como identificador del nodo padre la dirección de broadcast (0xFFFF) y como costo cero. Finalmente si el nodo no es una raíz y conoce un camino a la raíz se envía la estimación de la ruta (en la estructura routeInfo) más la estimación de la calidad del enlace hacia el nodo padre. Este valor se solicita al módulo estimador. En el nodo atacante, esto fue modificado para que siempre se envíe el costo cero. Luego mediante la interfaz BeaconSend, se envía el *beacon* construído al módulo estimador para que le agregue, como se vio anteriormente, el cabezal y el pie LEEP con el método addLinkEstHeaderAndFooter().

7.3 Ataque blackhole

En esta sección se presenta la lógica y la implementación realizada para el ataque *blackhole* en el protocolo CTP.

Lógica

Para implementar este ataque se quitaron las funcionalidades que se encargaban de reenvíar los paquetes recibidos. Sin embargo, se continúan enviando los ACKs correspondientes para los paquetes que llegan correctamente al nodo.

En la sección 7.1.1 se mencionó que el motor de reenvío de CTP recibe los paquetes desde la capa inferior a través del evento SubReceive.receive(), este evento, a su vez invoca a la función forward() que se encarga del reenvío de los paquetes, luego esta función deja pendiente de ejecución la tarea sendTask() que se encarga de la transmisión de los paquetes, para ello se lo envía a la capa inferior que se encarga de mandarlo invocando al comando SubSend.send(), una vez que la capa inferior manda el paquete dispara el evento SubSend.sendDone() para indicarlo y es en este último evento donde se puede decidir qué hacer después de la transmisión.
Esta secuencia de eventos y acciones puede observarse en la figura 7.5. Este flujo de acciones también puede ser desencadenado por el comando Send.send(), que es invocado desde la capa superior, que hace uso de los servicios del protocolo CTP, en este caso, este permite el envío de los paquetes de datos generados por el nodo atacante mismo.

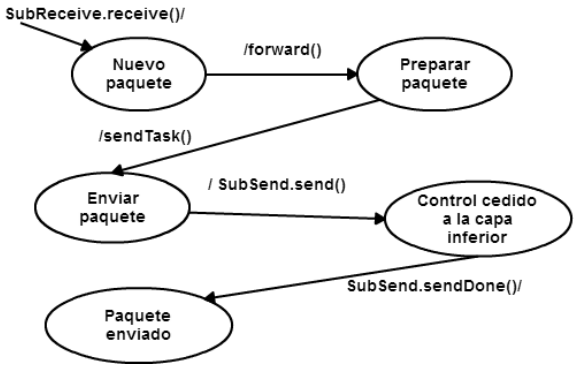


Figura 7.5: Secuencia de eventos y funciones realizadas para reenvíar un paquete en CTP.

Programación y componentes utilizados

Para implementar este ataque se partió de la implementación del ataque de *sinkhole* explicada anteriormente y se modificó el componente CtpForwardingEngineP del módulo de reenvío para generar el ataque de *blackhole*. Combinando ambos ataques se tiene un efecto mayor sobre la red, ya que se atraen más nodos para dejarlos incomunicados.
CTP reenvía paquetes que otros nodos le envían, pero también envía los paquetes que el mismo nodo genera, incluyendo los paquetes de datos y los *beacons*. Por lo que, a la hora de modificar el módulo de reenvío no se debe afectar la operativa que permite el envío de *beacons*. De otro modo, los nodos vecinos, no "conocerían" al nodo atacante y este no recibiría ningún paquete para reenviar.
Sin embargo, los *beacons* son generados por el módulo de enrutamiento, quien se los pasa al estimador de enlace, que a su vez se comunica directamente con la capa inferior que efectivamente realiza el envío. Esto lo podemos ver en los archivos de configuración que realizan el "*wiring*" de las interfaces (ver cuadro 15) y en la declaración de que interfaces cada módulo provee y usa. Esto sae puede observar respectimentente en los cuadros 13 y 14.
Por otro lado, los paquetes de datos son enviados al módulo de reenvío quien también se comunica con la capa inferior para realizar el reenvío. Esto lo podemos ver en los archivos de configuración que realizan el "*wiring*" de las interfaces en el cuadro 17 y en la declaración de que interfaces del módulo que pueden verse en el cuadro 16.
Por lo tanto, el envío de *beacons* y el de datos es independiente, por lo que se pueden deshabilitar por completo las funcionalidades de envío de paquetes del módulo de reenvío.
Para deshabilitar el reenvío de paquetes se quito la función forward, quitando también su invocación en el evento SubReceive.receive(). Sin embargo, esto no es suficiente ya que esto solo afecta a la funcionalidad de reenvío de paquetes. Pero el nodo puede continuar mandando sus propios paquetes de datos a través del comando Send.send().

Cuadro 13 Interfaces usadas por el estimador de enlaces (LinkEstimatorP)

```
module LinkEstimatorP {
  provides {
            interface AMSend as Send;
            (...)
      }
  uses {
            interface AMSend;
            (...)
      }
}
```

Cuadro 14 Interfaces usadas por el motor de enrutamiento (CtpRoutingEngineP)

```
generic module CtpRoutingEngineP(...) {
    provides {
        (...)
    }
    uses {
        interface AMSend as BeaconSend;
        (...)
    }
}
```

Cuadro 15 Wiring de interfaces en CtpP para LinkEstimatorP y CtpRoutingEngineP

```
        //Componentes
        components new CtpRoutingEngineP(TREE_ROUTING_TABLE_SIZE, 128, 512000) as Router;
        components LinkEstimatorP as Estimator;
        components new AMSenderC(AM_CTP_ROUTING) as SendControl;

        //Wiring
        Estimator.AMSend -> SendControl;
        Router.BeaconSend -> Estimator.Send;

        (...)
```

Cuadro 16 Interfaces usadas por el motor de reenvío (CtpForwardingEngineP)

```
generic module CtpForwardingEngineP() {
        provides {
                interface Send[uint8_t client];
                (...)
        }
        uses {
                (...)
                interface AMSend as SubSend;
        }
}
```

Cuadro 17 Wiring de interfaces (archivo de configuración CtpP)

```
        //Componentes
        components new CtpForwardingEngineP() as Forwarder;
        components new AMSenderC(AM_CTP_DATA);

        //Wiring
        Forwarder.SubSend -> AMSenderC;

        (...)
```

En los casos de prueba realizados la aplicación que corre sobre el nodo malicioso no envía ningún dato. Sin embargo, para mantener las interfaces que provee el protocolo CTP por defecto se modificó también esta parte de la implementación.
Observar en el cuadro 16 que el módulo de reenvío provee la interfaz Send para que la capa superior pueda pedirle que envie sus paquetes y usa la interfaz SubSend para solicitarle a la capa inferior que encapsule el paquete en una trama y lo envíe. Por el mecanismo *split-phase* de TinyOS, la capa superior solicita el envío de un paquete mediante el comando Send.send y aguarda por el evento Send.sendDone para saber que el paquete realmente fue enviado. De forma análoga, el módulo de reenvío le envía el comando SubSend.send() y este dispara el evento SubSend.sendDone() cuando la trama fue enviada.
A pesar de no envíar realmente el paquete, para continuar proporcionando las interfaces de CTP, se debe disparar el evento Send.sendDone() en algún momento. Las modificaciones introducidas quitan la lógica del comando Send.send() y dentro de este método se dispara el evento SubSend.sendDone que se ejecutaría cuando la capa inferior, que en este caso nunca fue invocada, termine de enviar el paquete. Por otro lado, se modifica el evento SubSend.sendDone() quitando toda su lógica y agregando el código necesario para disparar el evento Send.sendDone del cliente correspondiente para el mensaje solicitado.
La tarea sendTask(), encargada de tomar el primer paquete de la cola de reenvío y darselo a la capa inferior, es invocada desde varios lugares por lo que se deja vacía para que no haga nada. También se quita la función packetComplete() que se invocaba una vez enviado el paquete.

7.4 Ataque selective forwarding

En esta sección se presenta la lógica y la implementación realizada para el ataque *selective forwarding* en el protocolo CTP.

Lógica

El reenvío selectivo de paquetes puede realizarse tomando diferentes criterios, como por ejemplo no reenvíar los paquetes de determinado protocolo, los paquetes de determinado origen, etc. En este caso se implementaron dos tipos de reenvío selectivo, uno que descarta paquetes en base al origen del paquete (eliminando los paquetes cuyo origen tiene como identificador un número par) y otro que descarta un paquetes cada cierto intervalo de tiempo.

Programación y componentes utilizados

Selective forwarding por cliente

En este ataque solo se busca alterar el funcionamiento del renvío de paquetes, por lo que se modifica el evento SubReceive.receive() del archivo CtpForwardingEngineP que implementa el motor de reenvío. Dentro de este evento se incluye la condición mostrada en el cuadro 18 que determina si se debe reenviar el paquete o no. Si se decide no reenviarlo, entonces se retorna del evento sin seguir con la secuencia mostrada en la figura 7.5.

Cuadro 18 Reenvío selectivo en base al nodo origen

```
if ((getHeader(msg)->origin %2) == 0){
        return msg;
}
```

Selective forwarding por tiempo

Para la implementación de este ataque se modificaron los archivos CtpP y CtpForwardingEngineP. En el cuadro 19 se puede ver el código agregado en el componente CtpP donde se agrega el "*wiring*" de la nueva interfaz *SFDropTimer* que será implementada por el componente *TimerMilliC* que es un temporizador en milisegundos.

Cuadro 19 Selective forwarding por tiempo: CtpP

```
  components new TimerMilliC() as SFDropTimer;
  Forwarder.SFDropTimer -> SFDropTimer;
```

Los cambios realizados en el componente CtpForwardingEngineP se muestran en el cuadro 20. Se agrega la implementación del evento SFDropTimer.fired() que se dispara cuando expira el temporizador para indicar que se debe descartar el próximo paquete. La interfaz StdControl es un servicio que provee este módulo a la capa superior indicar cuando desea iniciar y parar los servicios provistos por la capa de red. Por lo que, cuando se invoca el comando StdControl.start() se dispara un temporizador periodico y cuando se invoca el comando StdControl.stop() se detiene. Luego, de forma análoga al caso anterior, se modifica el evento SubReceive.receive() para descartar el paquete si corresponde.

Cuadro 20 Reenvío selectivo por tiempo: CtpForwardingEngineP

```
generic module CtpForwardingEngineP() {
  provides {
                (...)
                interface StdControl;
  }
  uses {
                (...)
                interface Timer<TMilli> as SFDropTimer;
  }
}
implementation {
        enum {
                sf_drop_time = 300
        };
        bool dropNextPacket = FALSE;

        command error_t StdControl.start() {
                setState(ROUTING_ON);
                call SFDropTimer.startPeriodic(sf_drop_time);
                return SUCCESS;
        }

        command error_t StdControl.stop() {
                clearState(ROUTING_ON);
                call SFDropTimer.stop();
                return SUCCESS;
        }

        event void SFDropTimer.fired() {
                dropNextPacket = TRUE;
        }

        event message_t* SubReceive.receive(...) {
                (...)
                if (dropNextPacket){
                        dropNextPacket = FALSE;
                        return msg;
                }
                (...)
        }
```

7.5 Ataque ACK spoofing

Este ataque fue implementado para el transmisor de radio CC2420. Este provee reconocimientos tanto por software como por hardware. Por defecto se usan los reconocimientos por software.
Para implementar este ataque se modificó el componente CC2420ReceiveP del controlador del transmisor de radio CC2420 y el componente CtpRoutingEngineP del módulo de enrutamiento de CTP.
El evento RXFIFO.readDone() de la interfaz CC2420Fifo se dispara cuando se reciben bytes por el bus SPI y es implementado por el componente CC2420ReceiveP. Dentro de esta implementación se realizan chequeos para determinar si los datos recibidos son una trama válida, luego se encuentra el código mostrado en el cuadro 21, donde se utiliza al comando SACK.strobe() de la interfaz CC2420Strobe para transmitir una trama de reconocimiento por software. Como puede verse en el código, el envío del reconocimiento se encuentra condicionado a que la trama recibida sea para el nodo $(header->dest == call\ CC2420Config.getShortAddr())$ o que el destino sea la dirección de broadcast $(header->dest == AM\_BROADCAST\_ADDR)$. Para implementar el ataque, se quitan estas dos condiciones, de forma que cuando se reciba una trama válida, sin importar el destino de esta se envíe una trama de reconocimiento. Observar que este ataque se podría realizar de forma selectiva, modificando esa condición para atacar ciertos nodos específicos.
Los ACKs que maneja el protocolo CTP son a nivel de capa de enlace. Por otro lado, estos no contienen dirección de origen, solo se envía el número de secuencia del paquete que están confirmando (la estructura de un ACK se puede consultar en mayor detalle en el Anexo A.9). Por tal motivo, el atacante no debe realizar *spoofing* del identificador del nodo víctima.

Cuadro 21 Evento RXFIFO.readDone() del componente CC2420ReceiveP

```
if(call CC2420Config.isAutoAckEnabled() && !call CC2420Config.isHwAutoAckDefault()) {
        if ((((  header->fcf >> IEEE154_FCF_ACK_REQ ) & 0x01) == 1) &&
        ((header->dest == call CC2420Config.getShortAddr())|| (header->dest == AM_BROADCAST_ADDR)) &&
        (((  header->fcf >> IEEE154_FCF_FRAME_TYPE ) & 7) == IEEE154_TYPE_DATA)) {
                call CSN.set();
                call CSN.clr();
                call SACK.strobe();
                call CSN.set();
                call CSN.clr();
                call RXFIFO.beginRead(buf + 1 + SACK_HEADER_LENGTH, rxFrameLength - SACK_HEADER_LENGTH);
                return;
        }
}
```

Por otro lado se modifico el componente CtpRoutingEngineP para deshabilitar el envío de *beacons*, de forma que el nodo atacante solo esté escuchando el canal y enviando ACKs falsos sin anunciarse a los demás nodos de la red. Para lograr esto se dejó vacía la tarea sendBeaconTask().

7.6 Ataque jellyfish

En esta sección se presenta la lógica y la implementación realizada para el ataque *jellyfish* al protocolo CTP.

Lógica

El motor de reenvío almacena los paquetes de datos a enviar en una cola FIFO que se accede a través de la interfaz SendQueue y cuya longitud por defecto en la implementación de TinyOS 2.x es 12, este valor se encuentra configurado en la constante FORWARD_COUNT del componente CtpP. En esta cola se agregan los paquetes que van a ser reenviados y los paquetes creados por el mismo nodo para esperar su turno y poder ser enviados. Una vez que un paquete es agregado a esta cola, permanecerá

allí hasta que se sea enviado y se reciba su correspondiente ACK. Si no se recibe el ACK se vuelve a retransmitir luego de cierto tiempo, por defecto se retransmitirá el paquete un máximo de 32 veces antes de descartarlo.
Este ataque busca introducir retardo en la comunicación extremo-extremo. Para lograr esto, el atacante cada vez que reciba un paquete, lo va a retener cierto tiempo antes de enviarlo. Para la implementación de este ataque se utilizará una cola FIFO, que llamaremos JellyQueue, para guarde los paquetes recibidos y que serán retenidos por un determinado intervalo de tiempo que llamaremos *JELLY_TIME*. Observar que esta cola es distinta a la utilizada por defecto en CTP y provista por la interfaz SendQueue. También es necesario un temporizador que cuente ese *JELLY_TIME* e indique cuanto se debe enviar un paquete.
En la figura 7.6 se muestra la maquina de estados del comportamiento del nodo atacante, con amarillo se indican los nuevos estados que fueron agregados a lo que sería el comportamiento normal del reenvío de paquetes.

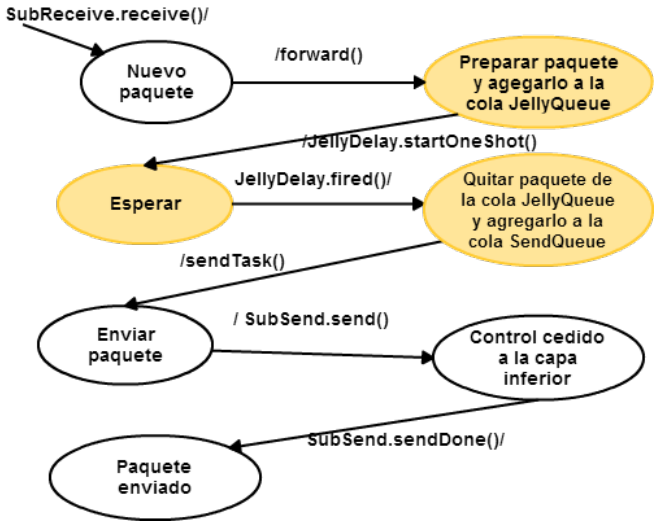


Figura 7.6: Secuencia de eventos y funciones utilizadas por el nodo malicioso en el ataque Jelly Fish.

Por cada paquete a envíar se vuelve a disparar el temporizador, por lo que, para cada paquete se introduce un retardo de al menos *JELLY_TIME* microsegundos. En algunos casos este retardo puede ser significativamente mayor porque también se incluyen otros retardos que dependen, por ejemplo, de la cantidad de mensajes que hayan llegado antes y estén aún esperando en la cola, de los retardos de procesamiento, de los retardos introducidos por el envío (sondeo del canal realizado por el protocolo CSMA-CA, retardo de trasmisión y retardo de propagación), entre otros.
Dependiendo del intervalo de tiempo elegido y del tráfico de la red, se pueden desbordar los *buffers* disponibles para almacenar paquetes. Por lo que estos son factores a tener en cuenta.

Programación y componentes utilizados

Para la implementación de este ataque se modificaron los archivos CtpP y CtpForwardingEngineP. En el cuadro 22 se puede ver el código agregado en el componente CtpP donde se agrega el "*wiring*" de dos interfaces nuevas *JellyDelay* que será implementada por el componente *TimerMilliC* y *JellyQueue* que será implementada por el componente QueueC que es una cola que almacena hasta QUEUE_SIZE elementos de tipo *fe_queue_entry_t*.
Los cambios realizados en el componente CtpForwardingEngineP se muestran en el cuadro 23. En la función forward() se recibe un paquete para reenvíar y se agrega a la cola donde esperarán los paquetes antes de ser reenviados. Por otro lado, si el temporizador no está corriendo aún, se dispara el mismo.

Cuadro 22 Ataque jelly fish: componente CtpP

```
  components new TimerMilliC() as JellyDelay;
  Forwarder.JellyDelay -> JellyDelay;

  components new QueueC(fe_queue_entry_t*, QUEUE_SIZE) as JellyQueueP;
  Forwarder.JellyQueue -> JellyQueueP;
```

Cuadro 23 Ataque jelly fish: componente CtpForwardingEngineP

```
#define JELLY_TIME 1000

generic module CtpForwardingEngineP() {
  provides {
                (...)
  }
  uses {
                interface Timer<TMilli> as JellyDelay;
                interface Queue<fe_queue_entry_t*> as JellyQueue;
        }
}
implementation{
        bool jellyTimer = FALSE;

        message_t* ONE forward(message_t* ONE m) {

                fe_queue_entry_t *qe;
                qe = call QEntryPool.get();
                qe->msg = m;

                (...)
                if (call JellyQueue.enqueue(qe) == SUCCESS) {

                        (...)
                        if (!call RetxmitTimer.isRunning()) {
                                if (!jellyTimer){
                                        call JellyDelay.startOneShot(JELLY_TIME);
                                        jellyTimer = TRUE;
                                }
                        }
                }
        }

        event void JellyDelay.fired() {

                if (!call JellyQueue.empty()){
                        fe_queue_entry_t* qe = call JellyQueue.head();

                        if (call SendQueue.enqueue(qe) == SUCCESS) {
                                if (hasState(RADIO_ON) && !hasState(SENDING)) {
                                        post sendTask();
                                }
                                call JellyQueue.dequeue();
                        }
                        call JellyDelay.startOneShot(JELLY_TIME);
                }
                else {
                        jellyTimer = FALSE;
                }
        }
}
```

En el evento fired() de la interfaz *JellyDelay*, se obtiene el primer elemento de la cola a ser reenvíado y se agrega a la cola por defecto que utiliza CTP para almacenar los paquetes que no puede envíar inmediatamente. Luego se inicia la tarea sendTask() que se encarga de envíar los paquetes y se quita de la cola utilizada para el ataque el paquete. Finalmente, se vuelve a iniciar el temporizador para reenvíar el próximo paquete. En caso de que la cola este vacía no se vuelve a iniciar el temporizador, sino que cuando llegue un nuevo paquete se iniciará toda la secuencia.
Observar que solo se guardan en la cola *JellyQueue* los paquetes que pasan por la función forward(), por lo que solo se afectan a los paquetes de datos reenviados. Si el nodo atacante enviara datos, estos serían recibidos a través del comando Send.send(), por no que pasarían directo a la cola SendQueue de reenvío, sin experimentar ningún retardo adicional.

7.7 Ataque sybil

En esta sección se presenta la lógica y la implementación realizada para el ataque *sybil* al protocolo CTP.

Lógica

El objetivo de este ataque es introducir información de enrutamiento falsa basándose en un ataque *sybil*. Para lograr esto, el atacante envía *beacons* con un identificador de nodo falso haciéndose pasar por otros nodos legítimos de la red.
El identificador de un nodo generalmente se asigna en tiempo de instalación, cuando se ejecuta algunos de los comandos que permiten instalar el software en el nodo: “make install.n” o “make reinstall.n”. Sin embargo, este puede ser modificado en tiempo de ejecución invocando al componente ActiveMessageAddressC que implementa la interfaz ActiveMessageAddress. Cuando un nodo cambia su identificador, también cambia la dirección de origen que envía en sus paquetes y cuales son los paquetes que recibe.

Programación y componentes utilizados

Para implementar este ataque se modificaron los componententes CtpP, CtpRoutingEngineP, CtpForwardingEngineP y LinkEstimatorP.
En en componente CtpP, como puede verse en el cuadro 24, se agrega el “*wiring*” donde se establece que en componente LinkEstimatorP utiliza la implementación que ActiveMessageAddressC hace de la interfaz ActiveMessageAddress.

Cuadro 24 Ataque sybil: componente CtpP

```
components LinkEstimatorP as Estimator;
components ActiveMessageAddressC;

Estimator.ActiveMessageAddress -> ActiveMessageAddressC;
```

En el cuadro 25 pueden verse los cambios realizados en el componente estimador de enlaces LinkEstimatorP. En este componente se agrega la interfaz ActiveMessageAddress que va permitirle al nodo asignarse un nuevo identificador, en este caso, el de alguno de los nodos vecinos. Como puede verse, este componente provee la interfaz Send para que le soliciten envíar un paquete. Recordando el funcionamiento de CTP, el motor de enrutamiento crea los *beacons* y se los pasa al estimador de enlaces para que le agregue un cabezal y un pie antes de pasarselo a la capa inferior para que efectivamente realice el envío del mismo. El estimador de enlaces utiliza la interfaz AMSend para pasarle los paquetes a enviar a la capa inferior.

Por lo tanto, cuando el comando Send.send() es invocado, este recibe en el parámetro *msg* un *beacon* para enviar. Dentro de su implementación, el nodo atacante va a agregar el cabezal y el pie LEEP, luego va a buscar un identificador de nodo que sepa que es válido dentro de la red. Para lograr esto, se va a buscar en la tabla de estimaciones una entrada válida (con la bandera VALID_ENTRY asignada). Si no se encuentra una, se utiliza por defecto el identificador de nodo uno. Al realizar esto, el atacante va a utilizar todos los identificadores de red que conozca utilizando una política *round-robin*.
Luego se invoca el comando setAddress de la interfaz ActiveMessageAddress que recibe dos parámetros: uno es el identificador de grupo (que se utiliza el valor que ya tenía el nodo malicioso) y el otro es el identificador de nodo (que se asigna como fue explicado anteriormente). Al ejecutar este comando es que se cambia el identificador del nodo para asignar el nuevo valor falso. Finalmente se invoca al comando AMSend.send() para que la capa inferior envíe el paquete.
En la implementación también se agrega el evento ActiveMessageAddress.changed() que debe proveerse obligatoriamente para hacer uso de la interfaz ActiveMessageAddress. Este evento se dispara cuando el identificador del grupo o del nodo es modificado.

Cuadro 25 Ataque sybil: componente LinkEstimatorP

```
module LinkEstimatorP {
  provides {
    interface AMSend as Send;
              (...)
  }
  uses {
    interface AMSend;
    interface ActiveMessageAddress;
  }
}
implementation {

        uint8_t addr_index = 0;

        command error_t Send.send(am_addr_t addr, message_t* msg, uint8_t len) {

                addLinkEstHeaderAndFooter(msg, len);

                uint8_t count = 0;
                am_addr_t dest_addr;

                while (!(NeighborTable[addr_index].flags & VALID_ENTRY)&&(count<NEIGHBOR_TABLE_SIZE)){
                        addr_index = (addr_index + 1) %NEIGHBOR_TABLE_SIZE;
                        count++;
                }
                if (count<NEIGHBOR_TABLE_SIZE){
                        dest_addr = NeighborTable[addr_index].ll_addr;
                        addr_index = (addr_index + 1) %NEIGHBOR_TABLE_SIZE;
                }
                else {
                        dest_addr = 1;
                }

                call ActiveMessageAddress.setAddress(call ActiveMessageAddress.amGroup(), dest_addr);

                return call AMSend.send(addr, msg);
        }

        async event void ActiveMessageAddress.changed(){}

}
```

Como se mencionó anteriormente al cambiar el identificador del nodo, también se cambia cuales son los paquetes que se reciben. Por lo que el nodo malicioso recibirá los paquetes de datos enviados al nodo víctima e intentará reenviarlos. Para evitar esta situación es que se deshabilitaron las funciones de envío y reenvío de paquetes de forma análoga a como se realizó en el componente CtpForwardingEngineP para el ataque *blackhole*.

Finalmente, para enviar *beacons* más rápido, se modifico el componente CtpRoutingEngineP desactivando el mecanismo de envío adaptativo de *beacons*. Las modificaciones son análogas a las realizadas en el ataque de *sinkhole*.

7.8 Ataque de repetición

Otro de los ataques provistos por la herramienta es el ataque de repetición, este consiste en volver a transmitir un paquete que se escuchó. Sin embargo, no se entrará en detalle sobre la implementación del mismo. Esto se debe a que este ataque no fue implementado en el nodo malicioso propiamente dicho, sino que fue implementado en Java y el código malicioso corre en la computadora. En el nodo se instala la aplicación *Base Station* para poder obtener los paquetes desde la aplicación Java que recibe el paquete y lo vuelve a retransmitir nuevamente a través de la aplicación *Base Station*.

CAPÍTULO

8

Análisis de resultados en capa de red

En este capítulo se presentan los resultados obtenidos para cada uno de los ataques implementados en capa de red. Primero se describe como se diseñaron las pruebas a realizarse, incluyendo aplicaciones ejecutadas, escenarios y métricas utilizadas. Finalmente se presentan los resultados de las pruebas donde se verifica que se cumpla con el comportamiento esperado, se analizan los cambios ocurridos en la topología de la red y el impacto en las métricas definidas.

8.1 Diseño de las pruebas

A continuación se presentan los elementos considerados durante el diseño de las pruebas, esto incluye los actores involucrados, las aplicaciones que estos ejecutan, los escenarios y las métricas utilizadas, entre otros.

8.1.1 Aplicaciones

Todas las pruebas que se presentan a continuación fueron simuladas en Avrora compilando el código con la plataforma MicaZ (transmisor CC2420). Sin embargo, las funcionalidades también fueron testeadas para la plataforma Mica2 (transmisor CC1000).
Los escenarios para probar los ataques al protocolo CTP cuentan con dos tipos de actores: nodo víctima y nodo malicioso. Los nodos víctimas ejecutan los componentes de TinyOS sin modificaciones, esto incluye los controladres del hardware (como por ejemplo el transmisor de radio) y los protocolos de red. A nivel de capa de aplicación, todos los nodos víctima ejecutan la misma aplicación. En esta luego de que el nodo *bootea* se enciende la radio. Cuando se dispara el evento indicando que la radio está lista se inicia la ejecución del protocolo CTP y se inicia un temporizador periodico que se dispara cada 2 segundos para envíar un paquete. Este paquete contiene una estructrua donde se envía el identificador del nodo y un contador que indica el número de paquete enviado. Estos paquetes podrían haber contenido lecturas de valores medidos con los sensores que los nodos poseen. Sin embargo, se optó por enviar esta estructura de datos ya que de forma rápidamente permite analizar en las capturas realizadas con el simulador la secuencia de paquetes envíados/recibidos por un nodo y el camino seguido por un paquete. A los efectos de simular el funcionamiento de una red real esto no nos afecta, por que el estudio que se quiere realizar es sobre la estructura de la red y su funcionamiento, sin importar los datos de capa aplicación capturados por los nodos y enviados.

En las pruebas se utilizó un único nodo malicioso, aunque perfectamente se podría haber utilizado más de uno. En este nodo se ejecutan los protocolos que vienen por defecto en la implementación de TinyOS con algunas modificaciones introducidas para explotar vulnerabilidades de los mismos. Los componentes y módulos que fueron modificados varían dependiendo del ataque en particular y fueron explicados en el Capítulo 7. A nivel de capa aplicación, para todos los casos, el nodo atacante está programado para que luego de *bootear* se duerma por un período de tiempo utilizando un temporizador que al disparase inicia la radio. Una vez que el nodo se despierta comienza la etapa de ataque. De esta forma, se logra que el nodo malicioso esté inactivo por un período de tiempo y le permite a la red establecer su funcionamiento normal para luego poder comparar la red antes, durante y después del ataque.
En las pruebas realizadas sobre el protocolo CTP el nodo malicioso no envía paquetes de datos. Sólo envía *beacons* para influir de alguna forma en el comportamiento de la red. Para ello, el nodo enciende el transmisor de radio con el comando RadioControl.start() y una vez que la radio está encendida inicia la ejecución del protocolo CTP con el comando RoutingControl.start().

8.1.2 Escenarios

A continuación se presentan los cuatro escenarios sobre los que se realizaron las pruebas. Estos fueron tomados de las diferentes aplicaciones de las redes de sensores inalámbricas relevadas en el estado del arte.

Escenario 1

Este escenario se corresponde con una red para monitorizar la estructura de un edificio. En la figura 8.1a se muestra la disposición de los nodos. Debido a que los nodos se disponen en una estructura tridimensional en la imagen no se ven todos los nodos. Estos se disponen en cuatro pisos colocando seis nodos por piso. La estación base se dispone en uno de los nodos de la mitad del segundo piso. Este escenario se caracteriza por tener una topología estrella con uno o dos saltos dependiendo de la cantidad de nodos en la red. Los escenarios sobre los que se experimentó fueron tomados a partir de los nodos mostrados en la figura 8.1a y seleccionando 7, 9, 13, 17, 19 y 24 nodos.

Escenario 2

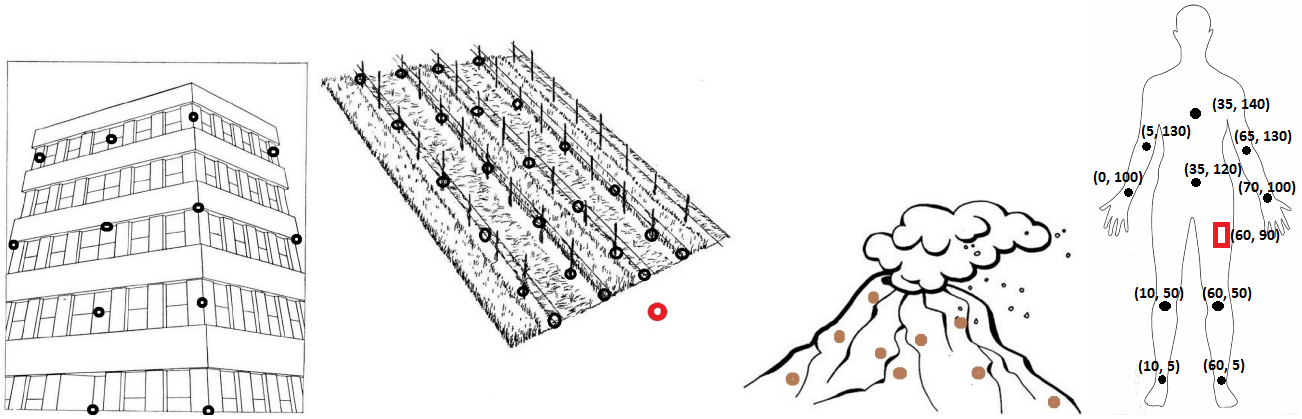
Este escenario representa una red desplegada sobre una plantación o viñedo donde los nodos se encuentran desplegados en forma de grilla. En la figura 8.1b se muestra la estrucutra de la red. En rojo se encuentra señalada la estación base que podría, por ejemplo, estar situada en algún granero u otra edificación donde sean recogidos los datos. Para las pruebas se tomaron redes de 7, 9, 13, 17, 19 y 24 nodos del escenario planteado.

Escenario 3

Para este escenario los nodos se dispusieron de forma aleatoria para representar una aplicación para monitorizar un territorio o para medir la actividad de un volcán como puede verse en la figura 8.1c. Para los escenarios utilizados se tomaron redes con 7, 9, 13, 17, 19 y 24 nodos.

Escenario 4

Este escenario se corresponde con una red de área corporal para medir datos fisiológicos de un paciente. En la figura 8.1d se muestra la topología de la red base sobre la que se introduce el nodo atacante. Los escenarios sobre los que se experimentó fueron formados a partir de dicho escenario variando la cantidad de nodos. Para este caso, se utilizaron redes de 4, 7, 10 y 12 nodos elegidos de los nodos mostrados en 8.1d. El nodo señalado en rojo representa la estación base, por ejemplo un celular, por lo que está presente en todos los casos. Este nodo será el que recolecta los datos de los demás nodos por



(a) Escenario 1 **(b)** Escenario 2 **(c)** Escenario 3 **(d)** Escenario 4

Figura 8.1: Escenarios de prueba

lo que será la raíz del árbol. Este escenario se caracteriza por tener una topología en forma de estrella con uno o dos saltos cuando está funcionando normalmente. Las coordenadas mostradas se encuentran en centímetros.

Atacante

Los escenarios 1, 2 y 3 se probaron con distintos posiciones geográficas del atacante. Se consideró el atacante situado cerca de la estación base o raíz, el atacante en medio de la red (es decir entre los nodos víctima) y el atacante situado fuera de la red a una distancia donde pudiera influir sobre algunos nodos. Estos casos serán los que en las pruebas se mencionen como A1, A2 y A3 respectivamente. Para el escenario 4, solo tiene sentido considerar el caso donde el atacante está situado fuera de la red, por ejemplo a un par de metros de la persona que tiene colocados los sensores. Por lo tanto, solo será evaluado este caso, que se denominará A1.

8.1.3 Métricas

Las métricas utilizadas para analizar el impacto de los ataques son las siguientes:

- Pérdida de paquetes: relación entre el número total de paquetes de datos recibidos por el nodo de destino (en este caso la raíz del árbol) con el ataque y el número de paquetes de datos recibidos sin el ataque.
- Retardo en el nodo atacane: tiempo que demora el nodo malicioso en reenvíar un paquete. Esta métrica es utilizada para el ataque *jellyfish*.
- Nodos en el subárbol del nodos atacante: cantidad de nodos en el subárbol del nodo atacante en relación a la cantidad total de nodos en la red.

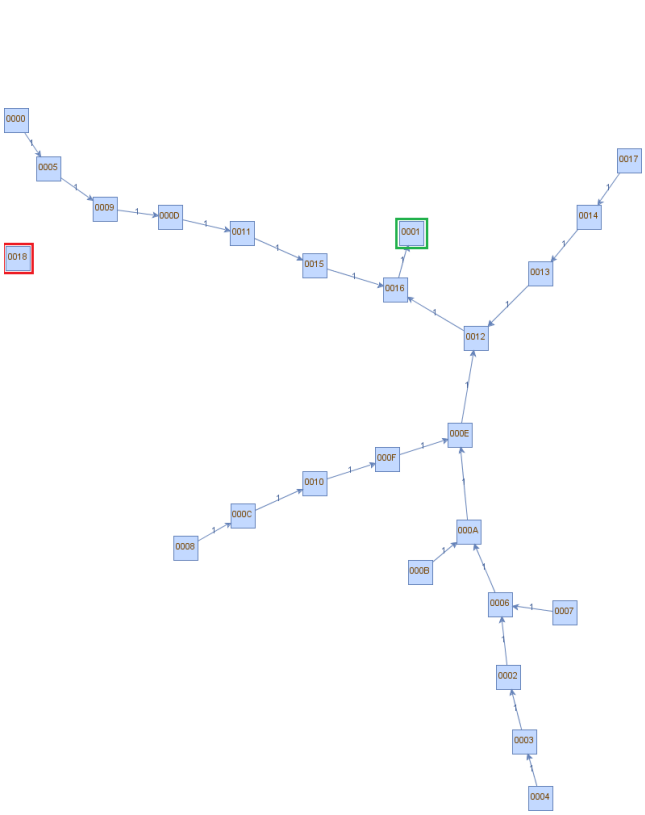
Por otro lado, para estudiar la evolución de la topología de la red se utilizaron los grafos que crea la herramienta desarrollada. Para analizar la estructura del árbol creado por el protocolo CTP se agrega un filtro para que solo se consideren los paquetes de datos (tipo AM igual a 0x71). Sino los paquetes de enrutamiento enviados por *broadcast* no permite analizar correctamente cual es el padre de cada nodo. Además, se agrega otro filtro para considerar en el grafo solamente el último paquete de datos enviado por un nodo. De esta forma, si un nodo había elegido determinado nodo como padre y luego cambia su elección, solo se muestra en el grafo el padre actual. Para analizar la evolución del grafo a lo largo de la captura de paquetes, se hace uso de la funcionalidad de la herramienta que permite considerar los paquetes recibidos hasta cierto instante.

8.2 Ataque sinkhole

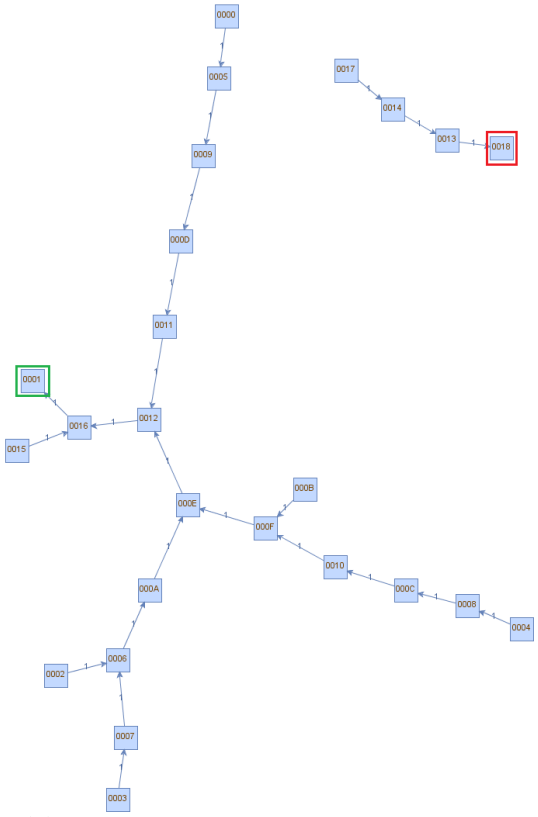
En esta sección se analizan los efectos del ataque *sinkhole*. Primero, en la sección 8.2.1 se presenta un caso donde se muestra paso a paso los cambios topológicos que sufre la red ante la presencia de este ataque. Luego, en la sección 8.2.2 se presenta un estudio del impacto que este ataque en los distintos escenarios variando la cantidad de nodos en la red y la posición del atacante con respecto a la estación base. En la sección 8.2.3 se estudia si para este ataque es conveniente enviar *beacons* en intervalos adaptativos o utilizando el intervalo mínimo. Finalmente, en la sección 8.2.4 se presentan las conclusiones sobre este ataque.
Durante este análisis de resultados se muestran figuras conteniendo las topologías de la red. Estas figuras fueron obtenidas de la herramienta implementada a partir de simulaciones realizadas con Avrora. En ellas se resalta con color verde a la estación base o raíz del árbol. El atacante se encuentra en color rojo y el resto de los nodos en azul.

8.2.1 Cambios de la topología de red

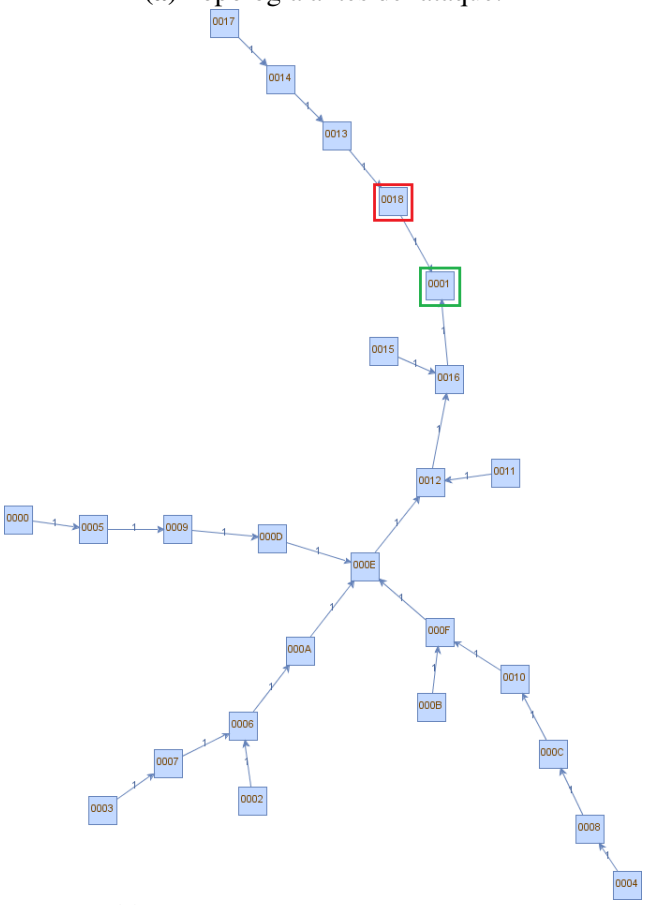
A continuación se muestra un caso donde se pueden apreciar los efectos del ataque de *sinkhole*. Para ello se presenta una secuencia de figuras donde se muestran los cambios en la topología que la red experimenta una vez empezado el ataque. Primero se muestra en la figura 8.2a la topología de la red antes de introducir al nodo atacante. Se puede observar que el nodo 0018 (en rojo) está aislado del resto de la red, ya que es el atacante y aún no envió ningún paquete (se encuentra en estado dormido).
Luego comienza la etapa de ataque. En 8.2b puede verse la topología en una instancia temprana de esta etapa. En general, inmediatamente después de introducir al nodo malicioso se genera una partición de la red, dado que el nodo atacante logra atraer el tráfico de nodos víctima incluso antes de poder determinar su propia ruta hasta la estación base. En este caso se genera una nueva componente conexa formada por los nodos 0013, 0014, 0017 y el atacante. Luego de un corto período de tiempo, en general, el nodo malicioso encuentra un camino hacia la estación base eligiendo un nodo padre y se forma una única componente conexa en el grafo como puede verse en la figura 8.2c. En este caso el nodo atacante elige como nodo padre directamente a la estación base señalada en color verde. A partir de allí, el nodo atacante puede comenzar a atrer más tráfico llegando a apoderarse, en algunos casos, de todo el tráfico de la red. En la figura 8.2d, puede observarse como finalmente para este escenario, al atacante logra atraer a todos los nodos con excepción del 0015 y 0016.
En algunos casos muy particulares, el particionamiento que se genera en la red luego de introducido el atacante (figura 8.2b) no puede ser solucionado por los nodos por lo que una porción de la red queda desconectada de la raíz. Esto ocurre cuando el nodo atacante elige como padre uno de los nodos de su propio subárbol. La situación más frecuente que ocurre es cuando el nodo padre del atacante se ve atraído por los bajos costos anunciados por el nodo malicioso, por lo que de cierta forma decide enrutar sus paquetes 'hacia atrás', dejando al nodo atacante sin un enlace mediante el cual pueda llegar a la estación base. Es decir, no conoce a ningún nodo dentro de su rango de radio mediante el cual pueda enviar sus paquetes. De esta forma, se genera un bucle de enrutamiento sin poder alcanzar la estación base. Cuando esto ocurre, los nodos que quedan desconectados detectan la situación por lo que se incremente el envío de *beacons*.
Esta no es una situación deseable, por lo que para evitar esto se tomo como estrategia en la implementación del nodo atacante no contestar con ACKs a los paquetes de datos que envíe el nodo padre. Es decir, cuando el nodo que el atacante eligió como padre elija a su vez al atacante como padre e intente enrutar paquetes de datos a través del nodo atacante. Este no le contestará con ACKs a sus paquetes. De esta forma, se afecta de forma negativa la estimación *in-bound* del enlace que el nodo padre del atacante realiza del enlace decidiendo no tomarlo en cuenta. Por lo que este intentará enrutar sus paquetes por otro camino. Esto fue implementado modificando los componentes que forman parte del controlador de radio CC1000 (no se realizó para el controlador de radio CC2420). Sin embargo, luego de probar esta estrategia en la práctica los problemas detectados no fueron totalmente solucionados. Es decir, se logró que el nodo padre no decida enrutar sus paquetes directamente a través del atacante. Pero en la mayoría de los casos el nodo padre del atacante decidía alternativamente enrutar sus paquetes por uno



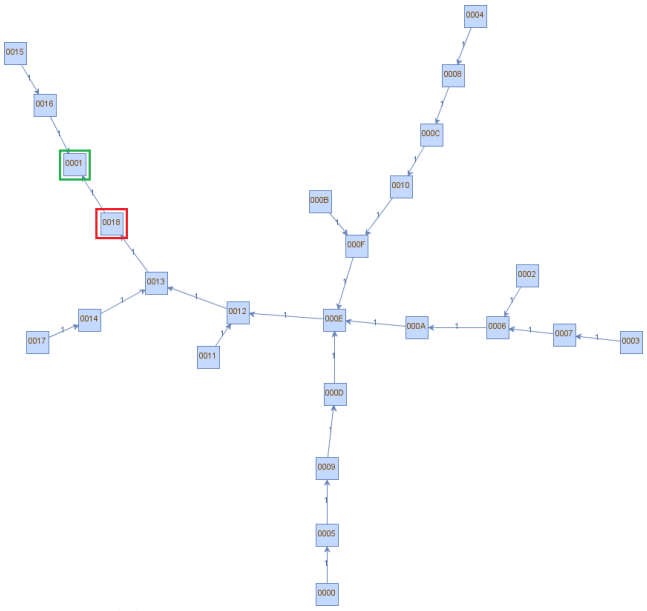
(a) Topología antes del ataque.



(b) Topología después del ataque



(c) Topología después del ataque (2)



(d) Topología después del ataque (3)

Figura 8.2: Topología de la red antes y después del ataque

de los nodos hijos del atacante, generándose bucles de 3 o más enlaces que no pudieron ser corregidos.

8.2.2 Impacto del ataque

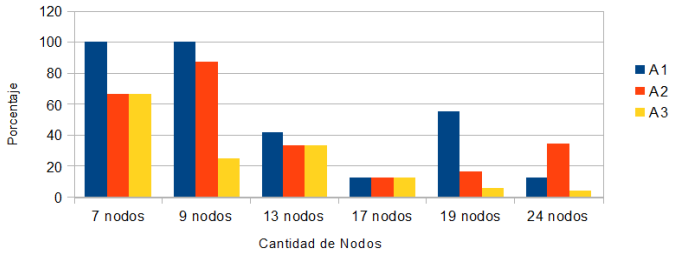
En esta sección se presenta un estudio del impacto de este ataque para cada uno de los escenarios presentados variando la cantidad de nodos en la red y la posición del atacante con respecto a la estación base. Para contar con algún dato objetivo y medible que nos permita cuantificar el impacto se toma como medida de referencia el porcentaje de nodos víctima (sin contar a la raíz) que deciden enrutar su tráfico a través del nodo malicioso. Esto implica el porcentaje de nodos que forman parte del subárbol generado a partir del nodo atacante. También para cada uno de los escenarios se eligió una de las configuraciones probadas para mostrar la topología de la red antes y después del ataque. Los resultados aquí presentados fueron realizados con la implementación del ataque que utiliza el intervalo adaptativo de *beacons* ya que, como se mostrará más adelante, esta tiene un mejor resultado que la que envía *beacons* al intervalo mínimo.

Escenario 1

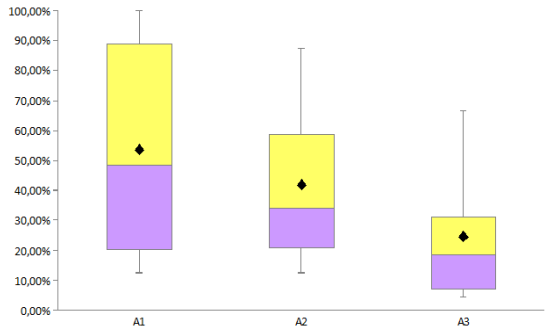
En la figura 8.3a se muestra el gráfico con el porcentaje de nodos en el subárbol del atacante para este escenario variando la cantidad de nodos en la red y la posición del atacante respecto a la estación base. En este gráfico puede observarse una tendencia a disminuir el impacto del ataque y luego a aumentar nuevamente conforme con el crecimiento de la cantidad de nodos presentes en la red.
Por otro lado, en los casos A1, donde el atacante está cerca de la raíz del árbol se logró un mejor resultado que los obtenidos cuando el atacante está más lejos de la estacion base. Esto puede verse más claramente en la figura 8.3b donde se muestra el mínimo, maximo, los cuartiles y la media para los porcentajes de nodos que participan en el subárbol del nodo atacante obtenidos en este escenario. En este gráfico se observa más claramente que el resultado mejora conforme a la proximidad del atacante a la estación base, esto se evidencia en el desplazamiento hacia arriba de las cajas. También las cajas son más grandes por lo que la dispersión de los resultados aumenta. Para el escenario A3 la mediana se encuentra en el 19 %, para el escenario A2 en 34 % y para A1 en 49 % de los nodos de la red atraídos.
Debajo se muestra en más detalle los resultados obtenidos para uno de los casos mostrados en la gráfica 8.3a. En la figura 8.3c se muestra la topología del Escenario 1 que cuenta con 19 nodos funcionando de forma normal, es decir, sin sufrir ningún tipo de ataque. Observar que la topologia de la red, se corresponde con una estructura de estrella, donde los nodos más alejados de la raíz se encuentran a dos saltos de distancia. En la figura 8.3d se muestra la topología de la misma red pero introduciendo un nodo malicioso cerca de la estación base (caso A1). Podemos ver que como resultado el nodo atacante atrae gran parte del tráfico de la red hacia él, simulando de cierta forma ser la estación base. Se puede observar que la red continúa teniendo una estructura de estrella aunque se incrementa el número de saltos que los nodos víctima requieren para llegar a la raíz.

Escenario 2

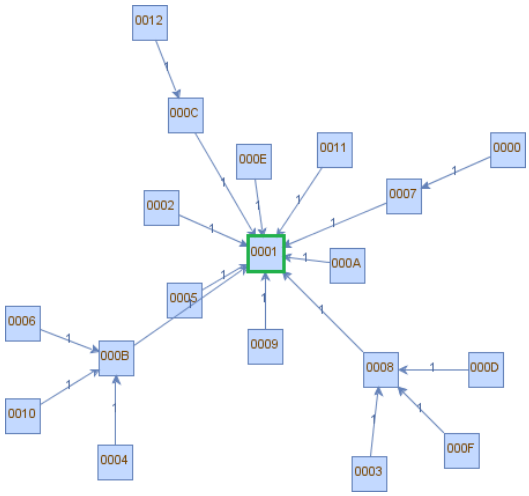
Para este escenario se puede observar el gráfico con el procentaje de nodos de la red atraídos por el atacante en la figura 8.4a. Para esta topología, donde los nodos son dispuestos en forma de grilla, los resultados obtenidos son bastante variables. Sin embargo, al igual que en el caso anterior el procentaje obtenido decrece y luego aumenta a mayor cantidad de nodos en la red.
En la figura 8.4b podemos apreciar que el caso A2 fue el que logró atraer la mayor cantidad de nodos de los tres casos probados con la mediana ubicada en el 75 % de nodos atraídos por el atacante. Luego le sigue el caso A1 donde en la mediana se tiene el 66 % de los nodos de la red y 36 % para el caso A3. Nuevamente se observa el aumento de la dispersión de los resultados al estar más cerca de la estación base.
En las figuras 8.4c y 8.4d se muestra el antes y el después del ataque realizado para una red en este escenario con 19 nodos. Observar que la topología formada en este caso es un árbol con nodos más alejados de la estación base. Para este escenario con el ataque, el nodo malicioso logra atraer el 100 %



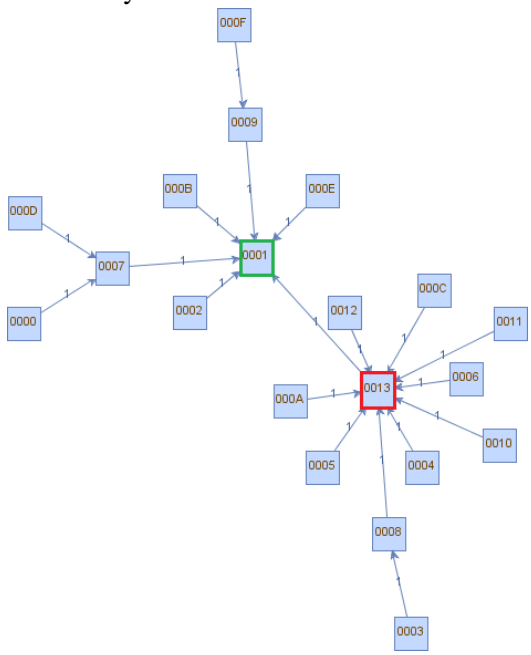
(a) Porcentaje de nodos en el subárbol del atacante.



(b) Mínimo, 1er. Cuartil, Mediana, Media, 3er. Cuartil y Máximo.



(c) Antes del ataque.



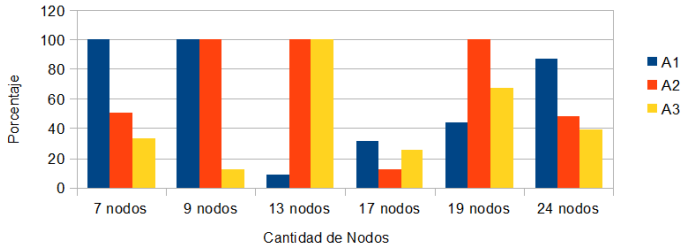
(d) Después del ataque.

Figura 8.3: Sinkhole - Escenario 1

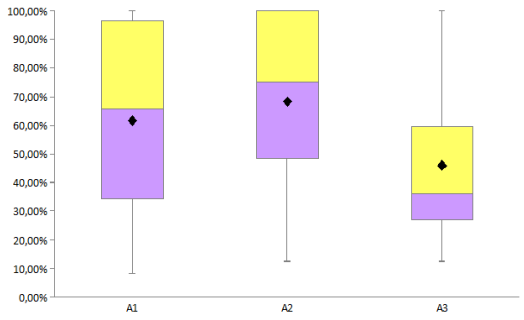
de los nodos de la red. Es decir, todos los nodos le envían de forma directa o indirecta los paquetes al atacante y este luego los reenvía a la estación base. El nodo atacante toma el total control del tráfico de la red.

Escenario 3

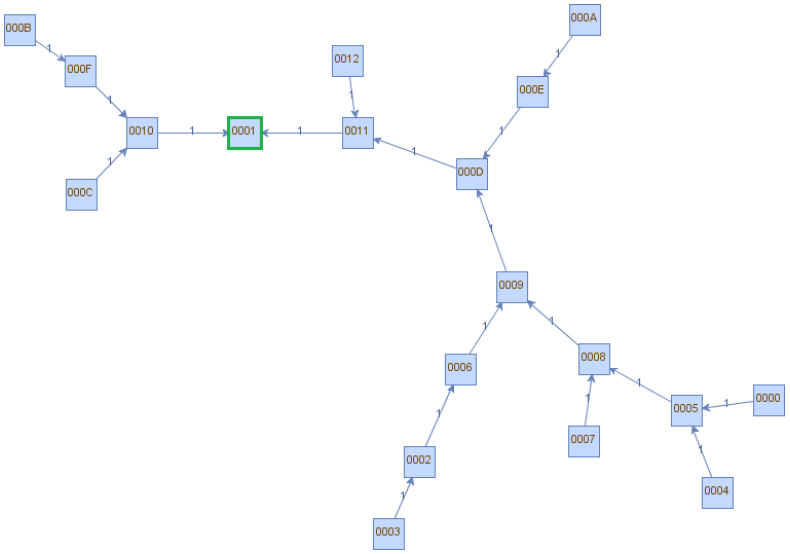
Los resultados obtenidos para este escario son similares a los anteriores. Nuevamente se nota la tendencia decreciente y luego creciente en la figura 8.5a conforme con el aumento de nodos en la red. Los mejores resultados se logran en redes pequeñas, luego la efectividad del ataque decrece para volver a crecer en redes más grandes. Por otro lado, en la figura 8.5b para los casos donde el atacante se encuentra más alejado de la estación base se obtienen resultados más pobres que cuando el atacante se sitúa próximo a la raíz o a una distancia media entre los demás nodos de la red. Nuevamente, las cajas se desplazan hacia arriba y se ensanchan a medida que el atacante se acerca a la estación base indicando



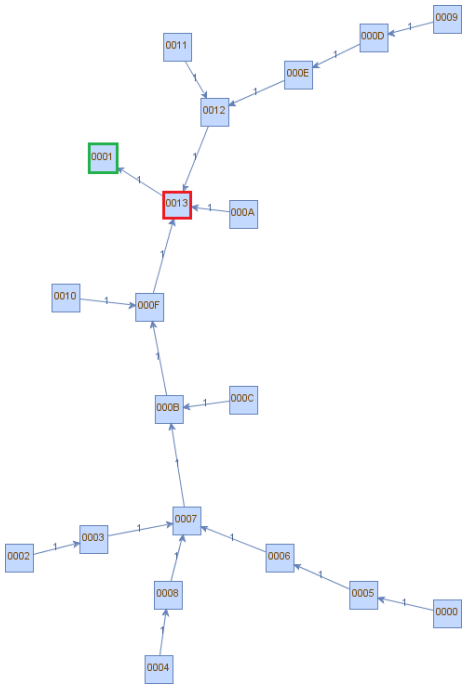
(a) Porcentaje de nodos en el subárbol del atacante.



(b) Mínimo, 1er. Cuartil, Mediana, Media, 3er. Cuartil y Máximo.



(c) Antes del ataque.



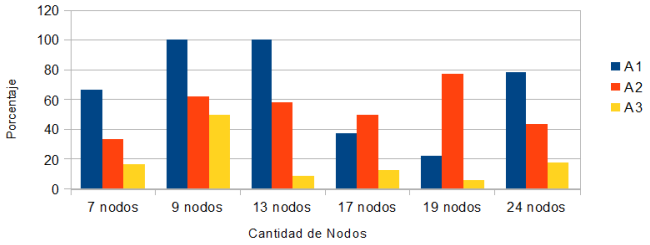
(d) Después del ataque.

Figura 8.4: Sinkhole - Escenario 2

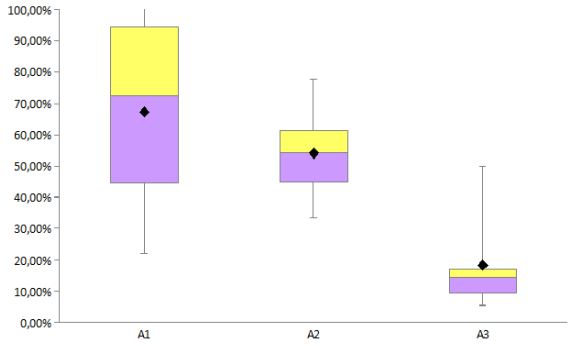
que mejoran los resultados obtenidos pero también son más dispersos. Para el caso A1 la mediana se ubica en el 72 %, para el caso A2 en 54 % y para A3 en el 15 % de los nodos de la red atraídos por el nodo atacante.
En las figuras 8.5c y 8.5d se muestra en más detalle, el antes y el después del ataque realizado para el caso con 17 nodos. Nuevamente en este escenario, el atacante logra atraer gran parte del tráfico.

Escenario 4

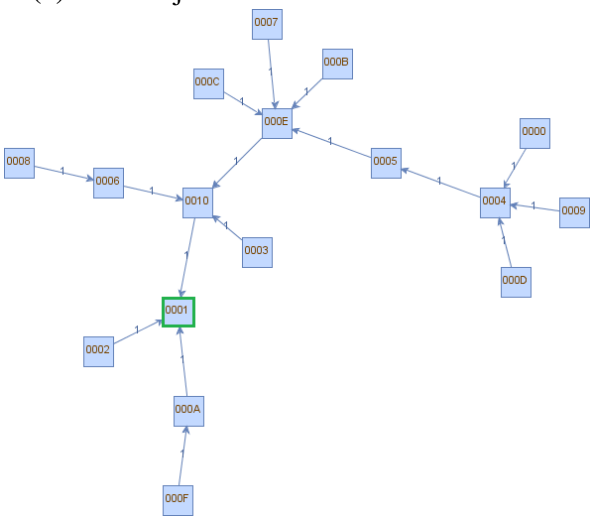
Este escenario es una red pequeña en topología estrella por lo que se reducen las posibilidades de probar los efectos de un atacante situado a distintas distancias con respecto a la raíz del árbol. Por tal motivo, sólo se considera el escenario donde el atacante está situado externamente a la red. Este es el caso donde una persona lleva una red de sensores en su cuerpo y fuera de la red hay un nodo atacante tratando de afectar el funcionamiento de la red corporal.



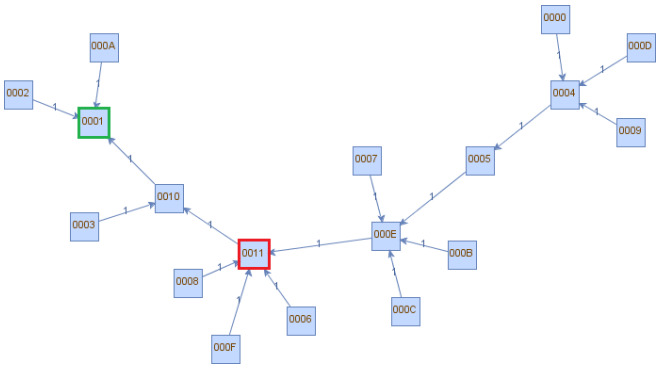
(a) Porcentaje de nodos en el subárbol del atacante.



(b) Mínimo, 1er. Cuartil, Mediana, Media, 3er. Cuartil y Máximo.



(c) Antes del ataque.



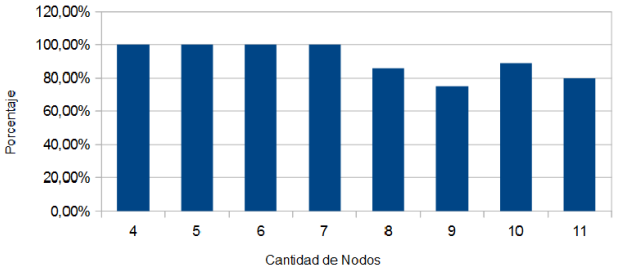
(d) Después del ataque.

Figura 8.5: Sinkhole - Escenario 3

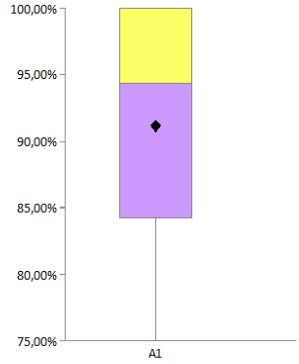
En este cuarto escenario vuelve a notarse en 8.6a la disminución del impacto del ataque y luego su aumenta conforme con el crecimiento de la red. En la figura 8.6b se observa que el máximo y el 3er. cuartil coinciden en el valor 1. Es decir, se logró atraer el 100 % de los nodos de la red y la mediana se encuentra en el 94 %.
En las figuras 8.6c y 8.6d se muestra en más detalle, el antes y el después del ataque realizado para el caso con 11 nodos. En este caso, el escenario inicial es una topología estrella donde los nodos están a un salto de la estación. Es decir, conocen un camino directo hasta la raíz que en sus *beacons* anuncia costo cero. Sin embargo, el nodo atacante logra atraer nuevamente gran parte de los nodos de la red anunciando también costo cero para alcanzar la raíz. Recordar que en este escenario los nodos de la red rodean a la estación base y el nodo atacante se coloca más alejado y por fuera de esta topología estrella. Pero de todos modos el nodo malicioso logra una gran influencia en la estructura de la red, incluso afectando a los nodos que están dentro del radio de alcance de la estación base.

8.2.3 Intervalo adaptativo VS. Intervalo mínimo

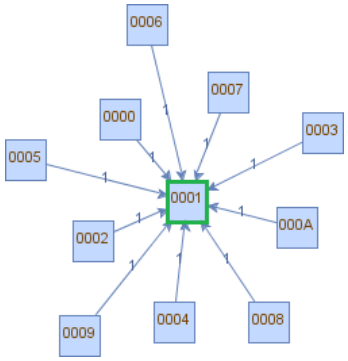
Otra de las pruebas realizadas consistió en verificar que impacto tenía la estrategia E3 del atacante sobre la red. Esta estrategía fue extraída de [14] y proponía enviar *beacons* al mínimo intervalo posible.



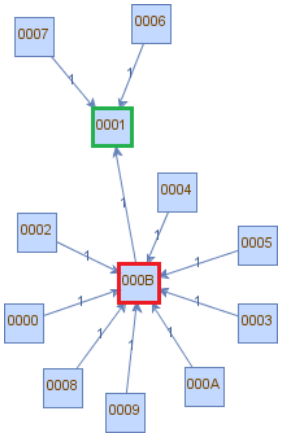
(a) Porcentaje de nodos en el subárbol del atacante.



(b) Mínimo, 1er. cuartil, mediana, media, 3er. cuartil y máximo.



(c) Antes del ataque.



(d) Después del ataque.

Figura 8.6: Sinkhole - Escenario 4

Para analizar los resultados de esta estrategia se comparó el impacto de un atacante que utiliza el tiempo adaptativo de *beacons* (mecanismo que los demás nodos de la red utilizan normalmente) y un atacante que siempre envía *beacons* al intervalo mínimo. Ambos ataques fueron realizados sobre las mismas topologías de red, donde los nodos buenos se disponen en forma de grilla. Los escenarios planteados contaban siempre con un solo nodo malicioso situado cerca de la estación base y la cantidad de nodos buenos varia en 7, 9, 13, 17 y 19 nodos situados sobre el escenario 2. Como fue explicado anteriormente, el nodo atacante comienza a ejecutar después de que los restantes nodos de la red logran establecerse y buscar las rutas al nodo raíz.

En la figura 8.7 se puede apreciar para cada una de las técnicas (intervalos adaptativos e intervalos mínimos) el porcentaje de nodos de la red que forman parte del subárbol del nodo atacante, es decir, los nodos que de forma directa o indirecta envían sus paquetes a través del atacante para que lleguen a la estación base. Se pude observar en azul el porcentaje de nodos afectados por el atacante utilizando intervalos adaptativos para el envío de *beacons* y en naranja el porcentaje de nodos afectados por el atacante utilizando el mínimo intervalo.

Del gráfico se puede concluir que la técnica que utiliza *beacons* en intervalos adaptativos tiene un mejor desempeño que la técnica que envía *beacons* utilizando el mínimo intervalo. Esta diferencia comienza a hacerse notoria en redes más grandes. El resultado aquí obtenido, contradice lo dicho en [14], que propone envíar más *beacons* para influenciar de forma favorable el cálculo que los otros nodos realizan

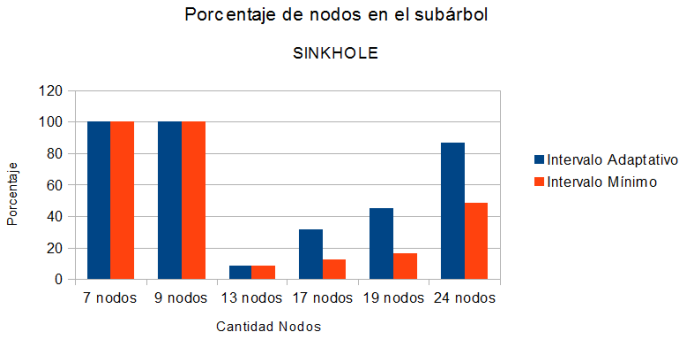
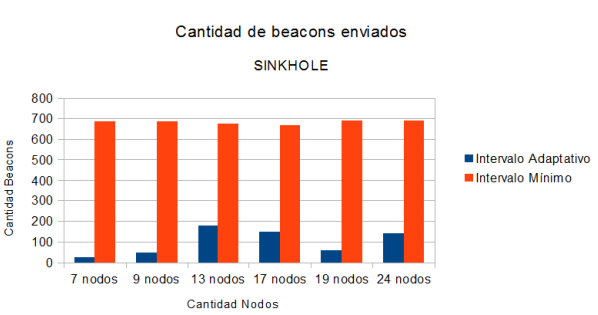
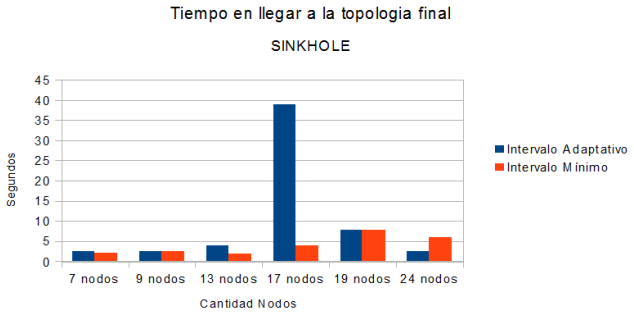


Figura 8.7: Sinkhole: Porcentaje de nodos de la red que forman parte del subárbol del nodo atacante.

de la calidad del enlace.
Este resultado podría deberse a que, como se aprecia en la figura 8.8a, el nodo atacante envía una cantidad mucho mayor de *beacons*, lo cual implica que se ocupa el canal por mayor tiempo y la probabilidad de que ocurran colisiones aumenta. Estas condiciones afectan de forma negativa el cálculo de la calidad del enlace. Como trabajo futuro se podría contar la cantidad de colisiones que ocurren para cada escenario con cada técnica de forma de confirmar la veracidad de esta hipótesis. Por otro lado, la técnica de envío de *beacons* al intervalo mínimo realiza un mayor uso de los recursos del nodo maliocos. Esto se debe a que el atacante está mayor tiempo activo por lo que gasta más energía. En los casos donde este se encuentre alimentado externamente de energía esto no presenta un problema. Sin embargo, cuando el nodo cuenta solamente con sus baterías esta no sería una característica deseable.
La propuesta de aumentar la cantidad de *beacons* enviados realizada en [14] se sustentaba en que de esta forma se computaban dos veces la calidad del enlace para el atacante mientras solo se realizaba un cálculo para los demás nodos. Considerando esto, intuitivamente se podría pensar que los efectos del ataque utilizando el mínimo intervalo se harían visibles de forma más rápida. Sin embargo, como se puede ver en la figura 8.8b esto no es completamente así. En esta figura, se muestra el tiempo que le toma a la red luego de que se introduce al atacante llegar a una topología de red estable, donde los nodos de la red eligen un nodo padre y bajan su actividad en el envío de *beacons*. En este gráfico se puede observar que generalmente, ambas técnicas demoran aproximadamente lo mismo.



(a) Cantidad de beacons del nodo atacante.



(b) Tiempo en llegar a la topología final luego del ataque.

Figura 8.8: Sinkhole: Intervalo mínimo VS. adaptativos

En el caso de la red con 17 nodos se observa una gran diferencia en los tiempos medidos. Sin embargo, en la prueba de intervalos adaptativos cuando se introdujo el nodo malicioso se generó una partición de la red de la cual los nodos no se pudieron recuperar. Al generarse un bucle de enrutamiento, los

mecanismos que tiene CTP para detectarlos provocan que los nodos continúen enviando *beacons* al intervalo mínimo sin lograr una red estable.
Como se mencionó, en algunos casos la estrategia de ataque propuesta genera una partición de la red dejando una porción de la misma incomunicada. Sin embargo, este no es el resultado deseado, dado que no se podría utilizar, por ejemplo, en casos donde se desean realizar escuchas pasivas de tráfico pasando inadvertidos o ataques que se basan en el ataque de *sinkhole*, como ser un ataque de *jellyfish*.
Por otro lado, en ninguno de los escenarios donde se aplicó el mínimo intervalo se generó una partición de la red. Posiblemente esto se debe, como se mencionó anteriormente, a que se envían más *beacons*, lo cual afecta de forma negativa la estimación de la calidad del enlace. Por lo tanto, el nodo padre del atacante decide no enrutar sus paquetes hacia el nodo atacante o sus hijos. Entonces, si bien el envío de *beacons* a intervalos mínimos logra atraer menos tráfico que la técnica de intervalos adaptativos para redes de mayor tamaño, esta también logra encontrar un camino a la estación base en más ocasiones, sin generar una partición de la red. Esta es otra de las hipótesis que se podrían comprobar como trabajo futuro.

8.2.4 Observaciones generales

En todos los casos el ataque de *sinkhole* logró engañar a sus nodos vecinos y atraer tráfico hacia él. En varios de los casos simulados el nodo malicioso logra atraer el 100 % de los nodos de la red. Este ataque permite realizar otros ataques como por ejemplo, realizar escuchas, robar la posición de la estación base, modificar mensajes o mejorar la efectividad de ataques como *selective forwarding*, *blackhole* y *wormhole*, entre otros posibles usos.
Algunas observaciones generales son que la cantidad de nodos y la posición del atacante de la red influyen en el efecto de este ataque. Respecto a la cantidad de nodos se observó en todos los casos una tendencia decreciente y luego creciente en el impacto del ataque. Observándose el menor efecto en redes de aproximadamente 17 nodos. Por otro lado, intuitivamente sería esperable que este ataque tuviera mayor efecto en nodos alejados de la estación base, ya que al estar más alejados de la raíz su estimación de la ruta hasta la estación base es la sumatoria de la estimación de más enlaces, resultando en un valor de ETX mayor. En esas circunstancias, incluir cerca de ellos un nodo que anuncia costo cero debería perturbar por completo su funcionamiento. Sin embargo, en las pruebas realizadas los atacantes situados cerca de la estación base se desempeñaron mejor. De todos modos, en el escenario 4 de la red de área corporal donde los nodos víctimas rodean a la estación base y el nodo atacante se encuentra colocado fuera de esa red se lograron muy buenos resultados.
En todos los escenarios se observaron resultados similares por lo que la topología inicial de la red, es decir topología estrella, arból o grilla, no tendría gran influencia en los resultados obtenidos por este ataque.
También se comprobó que el atacante utilizando la técnica de intervalos adaptativos atrae más tráfico que utilizando la técnica de intervalos mínimos. Sin embargo, la segunda es menos suceptible a producir bucles de enrutamiento o particionamientos de la red.
Finalmente, debido a las propiedades de las WSN, donde los nodos mismos deben enrutar sus paquetes, es que los mecanismos típicos que proporcionana confidencialidad, integridad y autenticación no son suficientes, pudiéndose generar ataques como el que fue estudiado aquí. Por ejemplo una situación donde ocurre esto podría ser una red donde se cuenta con mecanismos para que se cumplan las propiedades de confidencialidad, integridad y autenticación de los datos. Dentro de esta red podrían existir dos nodos maliciosos, uno situado cerca de la estación base y otro más alejado. Si estos dos nodos se comunican a través de un canal *wormhole*, el nodo cercano a la estación base puede guardar paquetes de enrutamiento que escucha y enviárselos al otro nodo malicioso, quien recibe estos paquetes y los vuelve a enviar a la red. Esos paquetes que de alguna forma fueron “copiados” de un sector de la red a otro siguen cumpliendo con las propiedades de seguridad iniciales. Los nodos maliciosos no necesitan saber el contenido particular de cada paquete por lo que se sigue cumpliendo con la confidencialidad. Tampoco se modifican los paquetes ni se introduce información falsa por lo que se sigue cumpliendo con la integridad. Finalmente, los datos fueron creados por un nodo legítimo de la red por lo que también se cumple con la autenticación. Sin embargo, se creará un *sinkhole* ya que los nodos que escuchen

los paquetes “copiados” creerán que conocen una buena ruta hasta la estación base a través del nodo que realmente emitió los paquetes que fueron copiados, siendo que este nodo no tiene porqué estar dentro de su rango de transmisión.

8.3 Ataque Blackhole

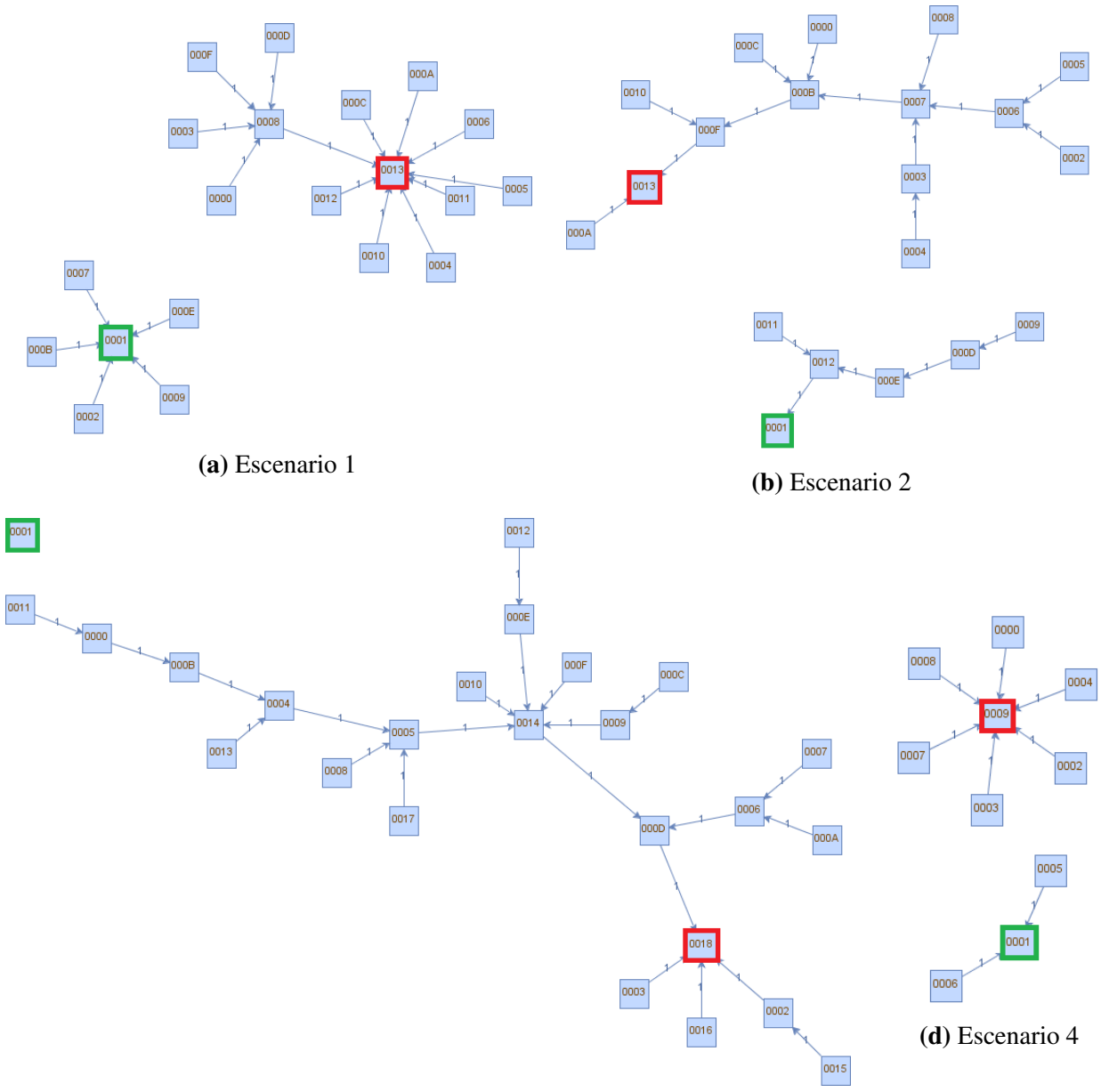
En esta sección se analizan los efectos del ataque *blackhole*. Primero, en la sección 8.3.1 se muestran los cambios en la topología producidos por este ataque. Luego, en la sección 8.3.2 se presenta un estudio del impacto de este ataque en los distintos escenarios variando la cantidad de nodos en la red y la posición del atacante con respecto a la estación base. Finalmente, en la sección 8.3.3 se presentan las conclusiones sobre este ataque.

8.3.1 Cambios de la topología de red

Para cada uno de los escenarios planteados se presentan los cambios ocurridos en las topologías correspondientes para uno de los casos. En las figuras 8.9a, 8.9b, 8.9c y 8.9d se muestran las topologías luego el ataque de *blackhole*. En ellas, se puede observar que el nodo malicioso atrae gran parte del tráfico que deja aislado de la estación base. El atancante responde con ACKs a los paquetes de datos de sus hijos como que recibió correctamente el paquete. Sin embargo, intencionalmente no vuelve a reenviarlo para que siga por la ruta que debería. En todos los casos se logra exitosamente particionar la red víctima.
En la figura 8.9c el atacante logra atraer el 100 % del tráfico de la red. Por lo que se observa, la estación base incomunicada y el nodo atacante toma su lugar como raíz del árbol.

8.3.2 Impacto del ataque

En la figura 8.10 se muestran los gráficos con el porcentaje de paquetes perdidos para los distintos escenarios variando la cantidad de nodos en la red y la posición del atacante con respecto a la estación base. Para calcularlo se determinó la cantidad de paquetes recibidos por la estación base en la red sin ataque y luego con el ataque. La implementación de este ataque se basa en el ataque de *sinkhole* para lograr un mayor efecto, por lo que en todos los casos se nota nuevamente la tendencia mencionada anteriormente donde disminuye el impacto del ataque y luego vuelve a aumentar. Observar en la figura correspondiente al escenario 2 que los resultados obtenidos para la topología en forma de grilla son muy variables.
En la figura 8.11 se muestra la media, los cuartiles, el mejor y peor caso de los resultados obtenidos para el ataque de *blackhole* considerando como medida la pérdida de paquetes.
Para el caso del escenario 1 (figura 8.11a) se observa que el intervalo entre el 1er. cuartil y la mediana es menor que el intervalo entre la mediana y el 3er. cuartil, lo cual implica una distribución sesgada a la derecha. Es decir, los datos que se encuentran en el intervalo entre 50 % y 75 % de pérdida de paquetes están más dispersos que los que se encuentran en el intervalo entre el 25 % y 50 %. Para el caso A1 la mediana se encuentra en el 60 % del tráfico de la red descartado, mientras que para los casos A2 y A3 se encuentra en 42 % y 23 % respectivamente. Podemos observar que la mediana se desplaza hacia arriba a medida que el atacante se encuentra más cerca de la estación base, lo cual implica que el ataque es más efectivo cuando el nodo malicioso está más cerca de la estación base. Sin embargo también aumenta la dispersión de los resultados obtenidos ya que las cajas son más grandes.
El escenario 2 (figura 8.11b) es el que presenta la mayor dispersión de los cuatro escenarios teniendo el valor máximo de desviación estandar de 40,69 % en el caso A1. Esto se corresponde con la variabilidad observada anteriormente. Para este caso la mediana se encuentra en 67 %, 48 % y 31 % del tráfico de la red perdido para los casos A1, A2 y A3 respectivamente.
En el escenario 3 (figura 8.11c) se observa que el intervalo entre 50 % y 100 % se desplaza hacia la arribas a la vez que se hace más pequeño, es decir, los datos están más concentrados por encima de la mediana que en el intervalo 0-50 %. Además a medida que el atancante se acerca de la estación base



(a) Escenario 1

(b) Escenario 2

(c) Escenario 3

(d) Escenario 4

Figura 8.9: Topología de la red luego del ataque

aumenta significativamente la dispersión de los datos que se encuentran por debajo de la mediana. Esta se encuentra en 83 %, 60 % y 20 % para los casos A1, A2 y A3 respectivamente.

Para el escenario 4 (figura 8.11d) la mediana de los datos se encuentra en el 97,5 %. Hacia arriba de la misma los valores están significativamente más concentrados que los valores por debajo. Por lo que la distribución de los datos es sesgada a la izquierda.

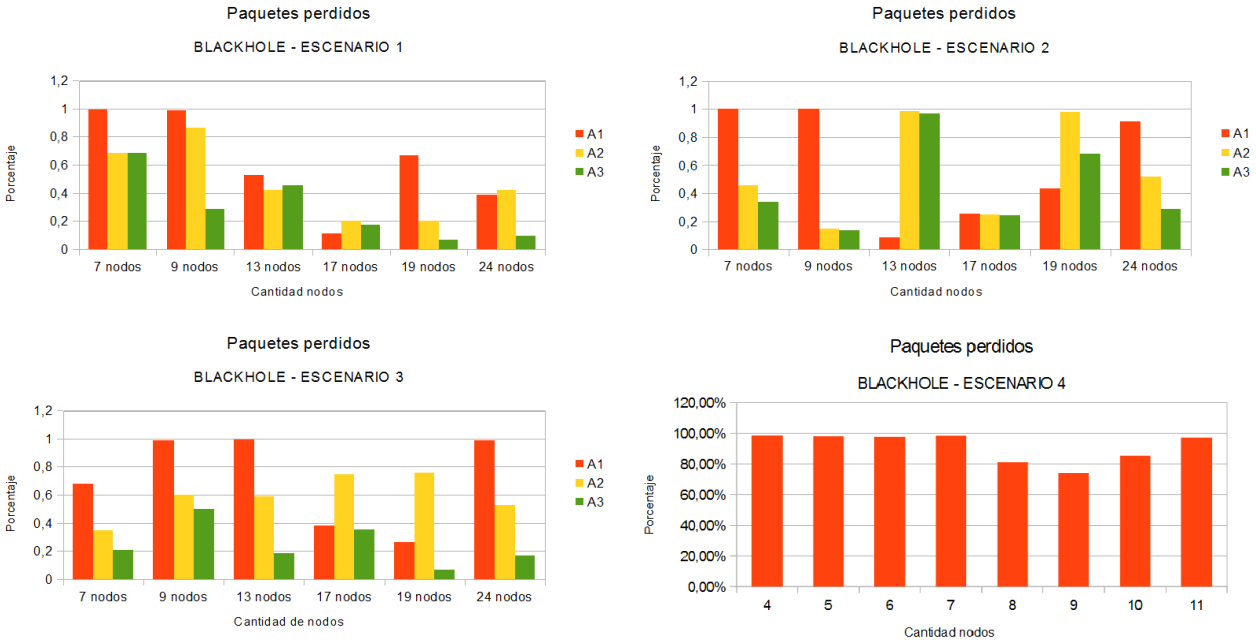
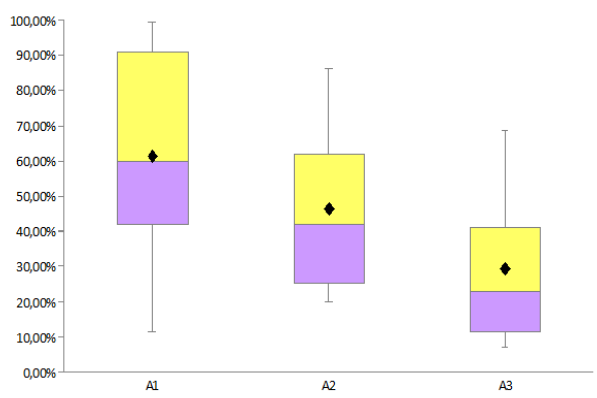


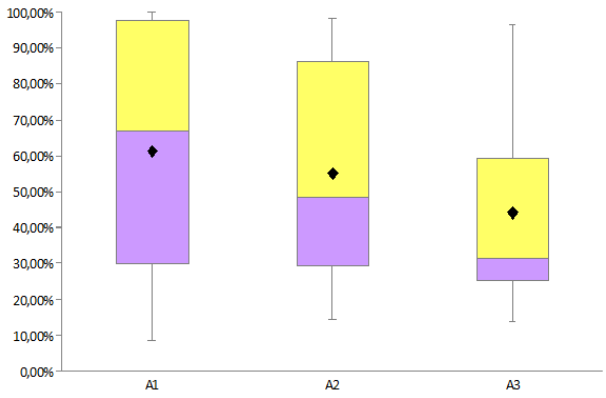
Figura 8.10: Blackhole: Porcentaje de paquetes perdidos.

8.3.3 Observaciones generales

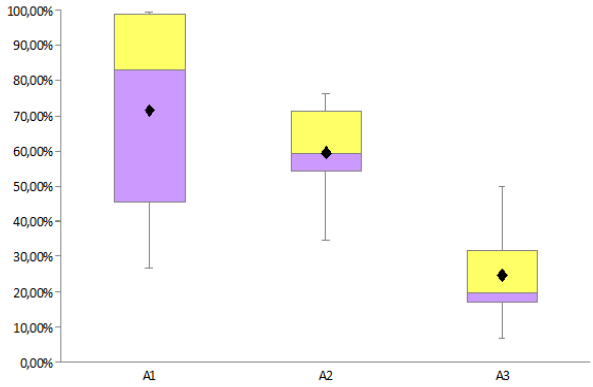
Los resultados observados fueron similares a los del ataque *sinkhole*, la cantidad de nodos y la posición del atacante de la red influyen en el efecto de este ataque. Se observó una tendencia decreciente y luego creciente en el impacto del ataque conforme con el aumento de la cantidad de nodos en la red. Por otro lado, en la figura 8.12 se muestra el promedio de paquetes perdidos para cada escenario en los casos A1, A2 y A3. En este gráfico claramente se observa que cuando se está más cerca de la estación base mayor es la influencia del atacante en el comportamiento de la red. El mejor desempeño se logró para el escenario 4 con una mediana de 97,5 % de los paquetes de la red descartados.
Este comportamiento también se observó en el diagrama de cajas (figura 8.11) con el desplazamiento de las cajas hacia la arriba ya que a medida que el atacante se acerca a la estación base el porcentaje de paquetes perdidos aumenta. Además, en ese gráfico se puedo apreciar que la diferencia entre el máximo y el mínimo es mayor, al igual que la diferencia intercuartílica cuando el atacante se acerca a la estación base. Por lo que, a pesar de ser más efectivo el ataque, los resultados presentan una mayor desviación estándar y en consecuencia son más variables.



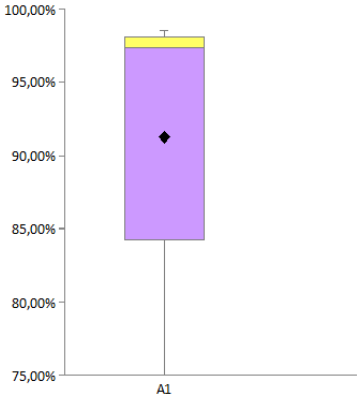
(a) Escenario 1



(b) Escenario 2



(c) Escenario 3



(d) Escenario 4

Figura 8.11: Blackhole: Porcentaje de paquetes perdidos (mínimo, 1er. cuartil, mediana, media 3er. cuartil, máximo).

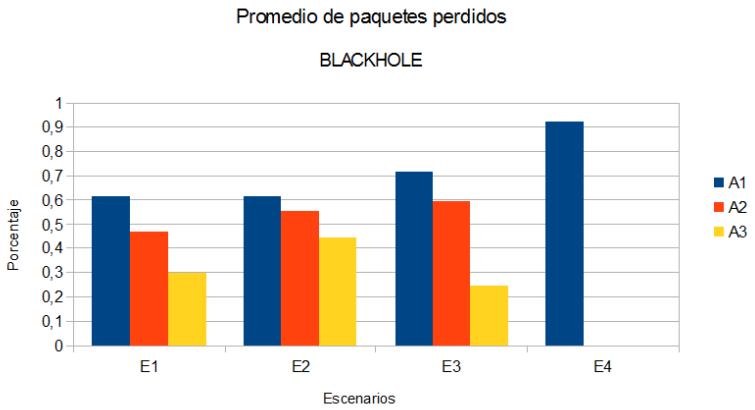


Figura 8.12: Blackhole: Promedio de pérdida de paquetes.

8.4 Ataque selective forwarding

En esta sección se analiza el funcionamiento del ataque *selective forwarding*. El impacto que este ataque tenga sobre la red esta directamente relacionado con el criterio de filtrado de paquetes elegido. El peor caso se da cuando se elige descartar todos los paquetes, que se corresponde con el ataque *blackhole* estudiado anteriormente. Por tal motivo solo se muestra el desempeño funcional de este ataque para el filtrado por cliente. El ataque *selective forwarding* con filtrado por tiempo funciona de forma similar.

8.4.1 Análisis del funcionamiento del ataque

En la figura 8.13 se muestra una captura de paquetes para este ataque. Los paquetes fueron filtrados utilizando la herramienta desarrollada para ver solamente los paquetes de datos que tienen origen o destino igual al nodo 0x000D, que es el nodo atacante. De esta forma se pueden observar cuales son los paquetes de datos que entran y salen en el nodo malicioso. Pudiendo deducir en consecuencia cuales paquetes son descartados.

Num	Destino	Origen	Largo datos	id Grupo	Tip...	Payload	Orig Sim	Tiempo	Topologia
209	000D	0008	19	0022	71	00 01 00 0C 00 0B 01 EE 00 0B 00 01	0008	0:00:05.031212	Ver
216	0001	000D	19	0022	71	00 02 00 06 00 0B 01 EE 00 0B 00 01	000D	0:00:05.046423	Ver
223	000D	0008	19	0022	71	00 01 00 0C 00 0C 01 EE 00 0C 00 01	0008	0:00:05.057241	Ver
234	000D	0008	19	0022	71	00 02 00 0C 00 05 01 EE 00 05 00 01	0008	0:00:05.085955	Ver
236	0001	000D	19	0022	71	00 03 00 06 00 05 01 EE 00 05 00 01	000D	0:00:05.095924	Ver
242	0001	000D	19	0022	71	00 03 00 06 00 05 01 EE 00 05 00 01	000D	0:00:05.126957	Ver
312	000D	0008	19	0022	71	00 07 00 11 00 09 01 EE 00 09 00 01	0008	0:00:05.841915	Ver
314	0001	000D	19	0022	71	00 08 00 06 00 09 01 EE 00 09 00 01	000D	0:00:05.846676	Ver
321	000D	0008	19	0022	71	00 08 00 11 00 03 01 EE 00 03 00 01	0008	0:00:05.869610	Ver
325	0001	000D	19	0022	71	00 09 00 06 00 03 01 EE 00 03 00 01	000D	0:00:05.877843	Ver
329	000D	0008	19	0022	71	00 09 00 10 00 02 01 EE 00 02 00 01	0008	0:00:05.891958	Ver
336	000D	0008	19	0022	71	00 0A 00 10 00 0A 01 EE 00 0A 00 01	0008	0:00:05.927478	Ver
343	000D	0008	19	0022	71	00 0A 00 10 00 07 01 EE 00 07 00 01	0008	0:00:05.958860	Ver

Figura 8.13: SELECTIVE FORWARDING - Captura de paquetes.

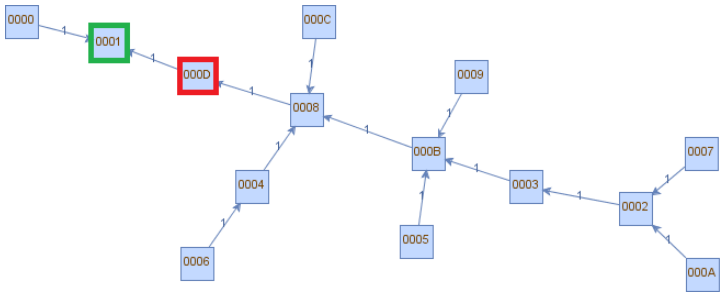


Figura 8.14: SELECTIVE FORWARDING - Topología de la red.

La implementación de este ataque descarta los paquetes cuyo identificador de nodo origen sea un número par. Observar que este identificador no es el que se muestra en la columna ‘origen’de la captura 8.13 ya que este contiene el identificador del nodo que reenvía el mensaje, es decir, es una dirección a nivel de capa de enlace. El identificador que se utiliza para descartar los paquetes es el de la cabecera CTP, es decir, es una dirección a nivel de capa de red.
Notar que en TinyOS la dirección de capa de enlace y de capa de red son la misma. Ambas capas utilizan el mismo identificador de nodo como dirección. Sin embargo, la dirección a nivel de capa de red identifica el “origen real” del paquete, es decir, al nodo que creo el mensaje original. Esta dirección no es modificada en ninguno de los saltos que el paquete debe realizar para llegar a destino. Por otro

lado la dirección a nivel de capa de enlace identifica al nodo que reenvía el paquete, es decir, esta dirección cambia cada vez que la trama es enviada a través de un nuevo enlace y es modificada tantas veces como saltos deba recorrer el paquete para llegar a destino.
Para facilitar la trazabilidad de los paquetes de datos podemos mirar los cuatro últimos bytes de la carga de datos, que intencionalmente contienen el identificador del nodo origen real del paquete y un contador con la cantidad de paquetes enviado por este origen. Por ejemplo, la carga de datos del primer paquete de la captura de la figura 8.13 termina en 00 0B 00 01, lo cual indica que es el primer paquete (00 01) enviado por el nodo con identificador (00 0B).
En la figura 8.14 se muestra la topología sobre la que se envían estos paquetes. El nodo malicioso recibe el tráfico de todos los nodos de la red menos del 0x0000 y 0x0001. Por lo tanto, el atacante debería descartar los paquetes que provienen de los nodos: 0x0002, 0x0004, 0x0006, 0x0008, 0x000A y 0x000C, dejando pasar solamente el tráfico generado por los nodos: 0x0003, 0x0005, 0x0007, 0x0009 y 0x000B.
En la captura 8.13 el primero paquete es creado por el nodo 0x000B (y reenviado por el nodo 0x0008), por lo que a continuación se puede observar que el atacante (0x000D) le reenvia el paquete a la estación base (0x0001). El tercer paquete tiene como origen al nodo 0x000C y mirando los paquetes posteriores podemos observar que el atacante nunca reenvía este paquete. A continuación el atacante recibe un paquete que originó el nodo 0x0005, por lo que decide reenviarlo. Notar que en este caso el atacante debe reenviarlo dos veces para que llegue satisfactoriamente al nodo 0x0001. Puede ser que el primer paquete enviado haya colisionado con otro paquete o que el ACK correspondiente se haya perdido. Continuando con la misma lógica se reenvían los paquetes creados por los nodos 0x0009 y 0x0003. Luego se descartan los paquetes de los nodos 0x0002 y 0x000A.

8.5 Ataque ACK Spoofing

En esta sección se analizan el funcionamiento del ataque *ACK spoofing*. En este caso también se muestra sólo el desempeño funcional ya que este ataque normalmente se utiliza para convencer a un nodo víctima de que un enlace débil es fuerte (afectando la estimación que se realiza de la calidad del enlace) o que un nodo que está caído aún permanece vivo. En el ambiente de pruebas con que se trabajó no se pueden recrear estas situaciones, por ejemplo, no se pueden agregar o quitar enlaces, por lo que fue imposible realizar estas pruebas. Por lo tanto, a pesar de que se lograron envíar los ACKs falsos, no se pudo comprobar el efecto que estos tienen en la topología de la red o realizar un análisis del impacto de este ataque.

8.5.1 Análisis del funcionamiento del ataque

En la figura 8.15 se muestra una captura de paquetes realizada durante este ataque. Los paquetes número 129 y 130 son ACKs idénticos, que como se explica a continuación, fueron enviados en respuesta al paquete número 128. En la columna 'Orig Sim'se observa que dichos ACKs fueron enviados por los nodos 0x0001 y 0x000D respectivamente. Donde el nodo 0x0001 es la estación base, nodo legítimo que debía enviar el ACK ya que es el destino del mismo y el nodo 0x000D es el atacante que envía un ACK falso. Observar que ambos ACKs, el original y el falso, son idénticos.
En las figuras 8.16 y 8.17 se puede apreciar en más detalle los campos de las tramas que contienen respectivamente al paquete de datos y el ACK. Observar que en la trama de ACK no se envía la dirección de origen o de destino, sólo tiene el número de secuencia del paquetes de datos que está reconociendo (marcado en color rojo en las figuras). Por lo tanto, el atacante no tiene que hacerse pasar por el nodo víctima, alcanza con que envíe un ACK con el número de secuencia del paquete que acaba de escuchar.
A continuación, en la captura de la figura 8.15, los nodos 0x0004 y 0x000A le envían también una trama de datos a la estación base. Sin embargo, en estos dos casos el atacante no envía los reconocimientos falsos ya que ambos nodos se encuentran fuera del rango de alcance de su radio (esto es conocido por como se dispusieron los nodos en el archivo de configuración utilizado en la simulación Avrora), por lo

Num	Destino	Origen	Largo datos	id Grupo	Tipo AM	Payload	Orig Sim	Tiempo	Topologia
127	0001	0000	19	0022	71	00 00 00 0B 00 00 01 EE 00 00 00 01	0000	0:00:05.004474	Ver
128	0001	000C	19	0022	71	00 00 00 0B 00 0C 01 EE 00 0C 00 01	000C	0:00:05.006053	Ver
129						00 00 00 0F A7 05 02 00 09 49 FF	0001	0:00:05.006861	Ver
130						00 00 00 0F A7 05 02 00 09 49 FF	000D	0:00:05.006858	Ver
131	0001	000A	19	0022	71	00 00 00 0B 00 0A 01 EE 00 0A 00 01	000A	0:00:05.008876	Ver
132						00 00 00 0F A7 05 02 00 06 A6 0E	0001	0:00:05.009685	Ver
133	0001	0004	19	0022	71	00 00 00 0B 00 04 01 EE 00 04 00 01	0004	0:00:05.011908	Ver
134						00 00 00 0F A7 05 02 00 09 49 FF	0001	0:00:05.012713	Ver
135	0001	0002	19	0022	71	00 00 00 0B 00 02 01 EE 00 02 00 01	0002	0:00:05.014451	Ver
136	0001	0005	19	0022	71	00 00 00 0B 00 05 01 EE 00 05 00 01	0005	0:00:05.014436	Ver
138	0001	0008	19	0022	71	00 00 00 0B 00 08 01 EE 00 08 00 01	0008	0:00:05.024268	Ver
139						00 00 00 0F A7 05 02 00 08 68 EF	0001	0:00:05.025076	Ver
140						00 00 00 0F A7 05 02 00 08 68 EF	000D	0:00:05.025074	Ver
141	0001	0003	19	0022	71	00 00 00 0B 00 03 01 EE 00 03 00 01	0003	0:00:05.027473	Ver
142						00 00 00 0F A7 05 02 00 09 49 FF	000D	0:00:05.028280	Ver
143						00 00 00 0F A7 05 02 00 09 49 FF	0001	0:00:05.028282	Ver
144	0001	000B	19	0022	71	00 00 00 0B 00 0B 01 EE 00 0B 00 01	000B	0:00:05.030575	Ver
145						00 00 00 0F A7 05 02 00 08 68 EF	000D	0:00:05.031379	Ver
146						00 00 00 0F A7 05 02 00 08 68 EF	0001	0:00:05.031381	Ver
147	0001	0000	19	0022	71	00 00 00 0B 00 00 01 EE 00 00 00 01	0000	0:00:05.034195	Ver
148						00 00 00 0F A7 05 02 00 08 68 EF	0001	0:00:05.035004	Ver

Figura 8.15: ACK SPOOFING - Captura de paquetes.

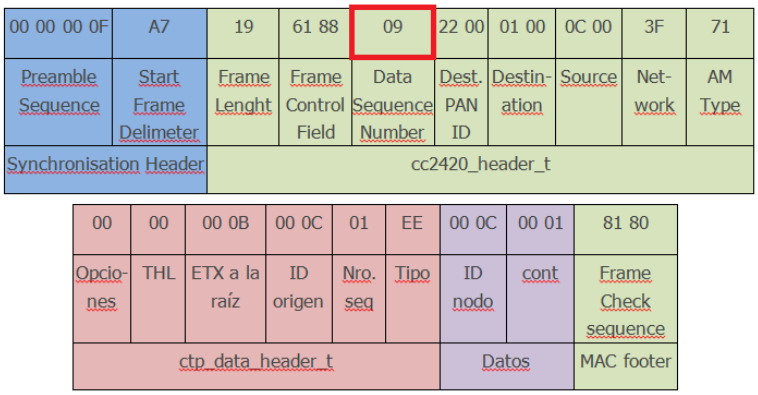


Figura 8.16: ACK SPOOFING - Paquete de datos

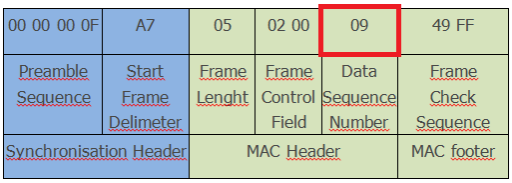


Figura 8.17: ACK SPOOFING - ACK

que no recibe estas tramas de datos. Luego, el atacante vuelve a enviar ACKs para los paquetes número 138, 141 y 144 de la figura 8.15.

En la figura 8.18 se muestra la topología de la red durante el ataque. Observar que el atacante (nodo con identificador 0x000D) no se encuentra presente ya que este solo envía ACKs haciendose pasar por otros nodos de la red pero no se anuncia a través de *beacons* o envía datos propios. Por lo que, a los ojos de los demás nodos de la red, este nodo no existe.

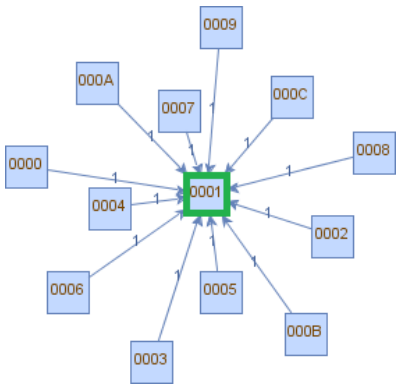


Figura 8.18: ACK SPOOFING - Topología de la red.

En la mayoría de los casos se reciben dos reconocimientos. Sin embargo, para corregirlo no basta simplemente con escuchar el canal para determinar si el nodo verdadero envió el ACK ya que no siempre se puede saber si el nodo verdadero envió el ACK. Por ejemplo, debido al problema de la estación oculta, donde puede suceder que el nodo víctima este en el rango de alcance del atacante pero el nodo destino del paquete de datos no. En tal situación, el atacante escucha el paquete de datos pero no el ACK de respuesta. Tampoco se puede saber si el otro nodo aún no envió el ACK pero lo va a enviar en un futuro cercado y antes de que el otro nodo de por perdido el paquete. Por ejemplo, para los paquetes de datos número 141 y 144, el nodo atacante envía el ACK antes de que lo envíe el nodo legítimo.

8.6 Ataque jellyfish

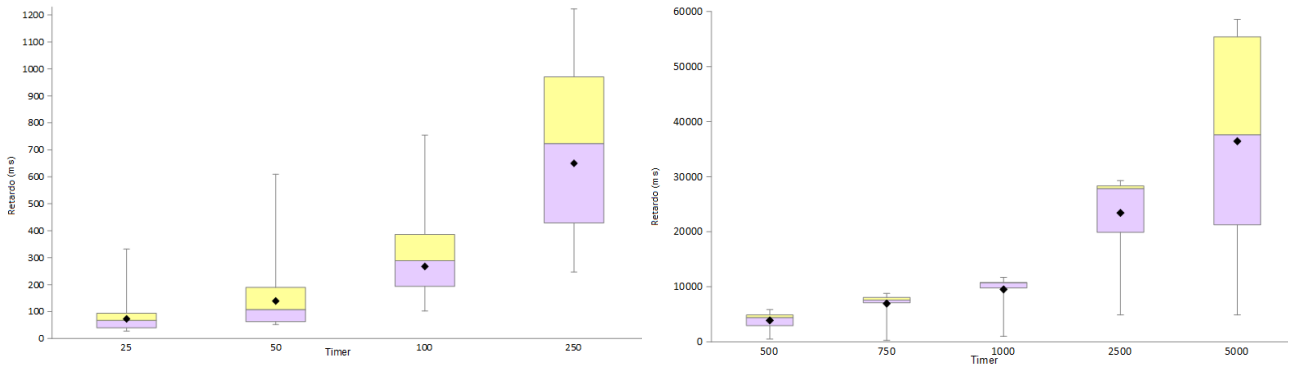
En esta sección se analizan los efectos del ataque *jellyfish*. Se presenta un estudio del impacto de este ataque considerando la pérdida de paquetes y el retardo efectivo generado en los paquetes que pasan a través del nodo atacante. Cuando se menciona el retardo efectivo en el nodo atacante se hace referencia al tiempo transcurrido entre que el paquete es enviado por el nodo hijo del nodo atacante y que este lo vuelve a reenviar hacia el próximo nodo en el árbol. Es decir, esto incluye el retardo introducido intencionalmente por el nodo malicioso, el retardo de transmisión y propagación a través de un enlace, retardo de procesamiento y de cola en el nodo atacante.

Uno de los parametros configurables en este ataque es el tiempo que el atacante decide retardar un paquete (tiempo que será utilizando por el temporizador para retener el paquete antes de volver a enviarlo). Por lo que, se establecieron distintos valores para este parámetros para analizar el retardo efectivo experimentado por los paquetes en el nodo atacante y las pérdidas de paquetes generadas en la red. Los resultados aquí presentados fueron ejecutados para el escenario 4 que representa la red de área corporal.

8.6.1 Impacto del ataque

Retardo de paquetes

Para analizar el tiempo que un paquete permanece en el nodo atacante no se consideraron los paquetes que tuvieron que ser reenviados más de una vez por el nodo hijo para ser recibidos correctamente por el nodo malicioso (ya que no se puede determinar cual de todos los paquetes enviados fue el recibido correctamente).



(a) Retardo para temporizador menor a 500 ms. **(b)** Retardo para temporizador mayor o igual a 500 ms.

Figura 8.19: Jellyfish: Retardo de paquetes (mínimo, 1er. cuartil, mediana, media, 3er. cuartil, máximo).

Los valores considerados por el temporizador del atacante fueron los siguientes: 25, 50, 100, 250, 500, 750, 1000, 2500 y 5000 ms. Los resultados obtenidos para cada uno de estos valores pueden verse en las figuras 8.19a y 8.19b donde se observa el mínimo y máximo así como los cuartiles y la media obtenidos para el retardo efectivo obtenido con los distintos valores del parámetro. La gráfica fue separada en dos: para valores del temporizador menores a 500ms y para valores mayores o igual a 500ms. Esto se debe a que los resultados obtenidos para los primeros son valores mucho menores en magnitud que los del segundo gráfico por lo que en la misma figura no se pueden apreciar bien.

Parámetro [ms]	25	50	100	250	500	750	1000	2500	5000
Mediana [ms]	67,21	107,66	288,70	723,36	4403,19	7582,47	10750,30	27844,00	37608,91
Incremento [%]	168,82	115,32	188,70	189,35	780,64	911,00	975,03	1013,76	652,18

Cuadro 8.1: Mediana según el parámetro del temporizador.

Como era de esperarse, a mayor valor del temporizador mayor es el retardo que sufre el paquete. En la tabla 8.1 se observa el valor de la mediana para los distintos valores del temporizador. Así como el porcentaje de incremento del retardo efectivo sufrido en el nodo atacante con respecto al valor del temporizador configurado. Para valores del temporizador más grandes el procentaje de incremento es significativamente mayor que para valores pequeños, esto se debe a que el retardo de cola comienza a hacerse mayor. Por otro lado, los recursos del nodo malicioso comienzan a agotarse y las pérdidas de paquetes son más frecuentes. En la Figura 8.20 se puede apreciar el porcetaje de pérdidas de paquetes ocurrido en la red para las distintas configuraciones del temporizador. Para valores menores o iguales a 500 ms se obtienen porcentajes de pérdidas menores a 20 %, llegándose a obtener un retardo (mediana) de 4403 ms.
Por otro lado, en la figura 8.19a para los valores del temporizador menores a 500 ms se observa que la distribución de probabilidad es simetrica para los cuartiles 1, 2 y 3. Esto se corresponde con lo observado en las figuras 8.21a, 8.21b, 8.21c y 8.21d donde se muestra el retardo de cada uno de los paquetes que pasaron por el nodo atacante durante una simulación. En los casos de las figuras 8.21a

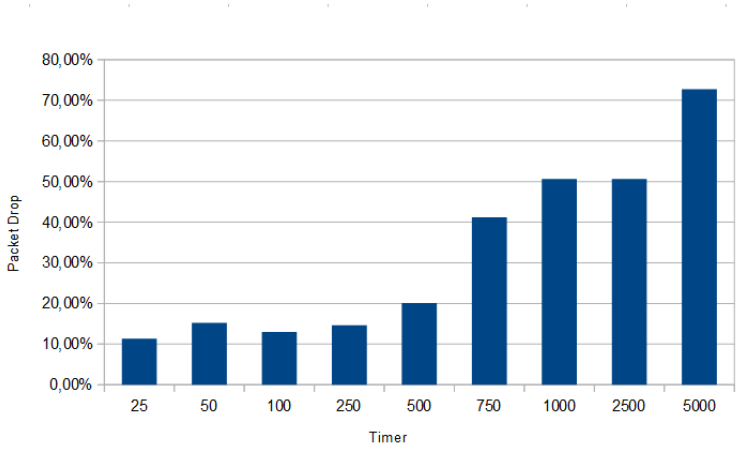
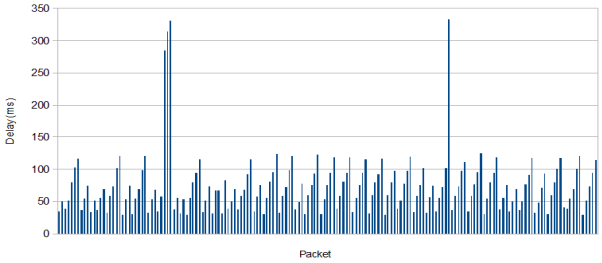


Figura 8.20: Jellyfish - Pérdida de paquetes para los distintos valores del temporizador

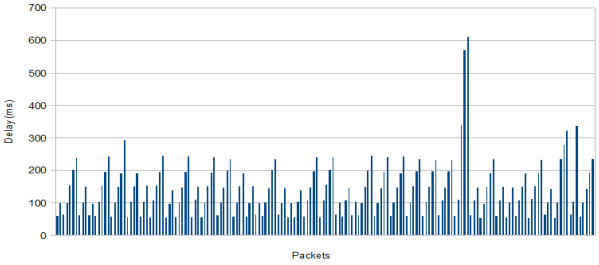
y 8.21b se observa mayormente un comportamiento cíclico en el retardo de los paquetes, con algunas instancias aisladas donde el retardo de pocos paquetes tiene un retardo significativamente mayor que el resto. Para el caso de 100 ms, en la figura 8.21c, los picos observados son de menor magnitud con respecto al resto de los retardos normalmente experimentados. En la figura 8.21d se observa un comportamiento ciclico de los retardos sin valores apartados de lo normal. Esto se corresponde con la distribución de probabilidad simétrica observada para este caso en la figura 8.19a.
Para todos los casos de la figura 8.19b se observa una distribución de probabilidad sesgada a la izquierda, que podemos observar que concuerda con las gráficas de las figuras 8.21e, 8.21f, 8.21g, 8.21h y 8.21i donde se comienza con un retardo igual al tiempo configurado para el temporizador, que luego se incrementa y perdura para el resto de los paquetes.
Para este ataque se debe considerar el retardo que se desea introducir en la red considerando el costo en recursos que se va a generar para el nodo atacante y en consecuencia las pérdidas de paquetes que se van a generar. Para temporizadores entre 100 y 500 ms los resultados obtenidos son bastante buenos. Para valores mayores se generan demasiadas pérdidas mientras que para valores menores si bien se genera un retardo este no es muy significativo.

Pérdida de paquetes

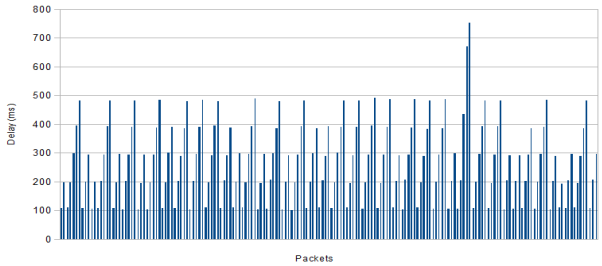
Otra de las pruebas realizadas consistió en analizar la pérdida de paquetes según distintas proximidades del atacante a la estación base. En la figura 8.22 se muestran los resultados obtenidos para este ataque considerando los paquetes perdidos (es decir, los que no llegan a la estación base). Se puede observar que los resultados obtenidos son muy variables. En la mayoría de los casos donde hay un valor pequeño de pérdida de paquetes se verificó que esto se debía a que pocos nodos víctima decidían enrutar sus paquetes a través del nodo atacante. En el 9 % de los casos que se ejecutaron no se perdió ningún paquete (0 % de paquetes perdidos), mientras que en el 25 % de los casos se perdió menos del 1 % de los paquetes. En este ataque, a diferencia de los demás, la proximidad a la estación base no parece tener un claro efecto sobre los resultados obtenidos.



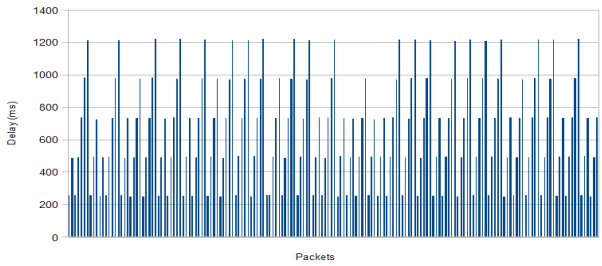
(a) Temporizador de 25 ms.



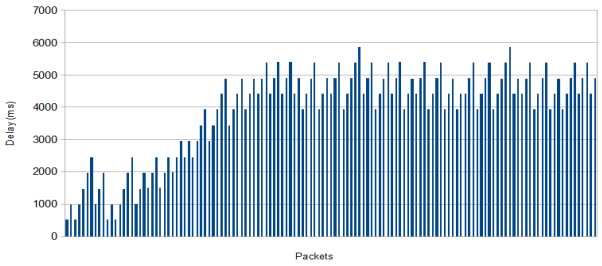
(b) Temporizador de 50 ms.



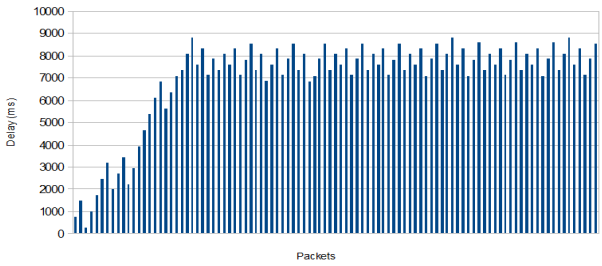
(c) Temporizador de 100 ms.



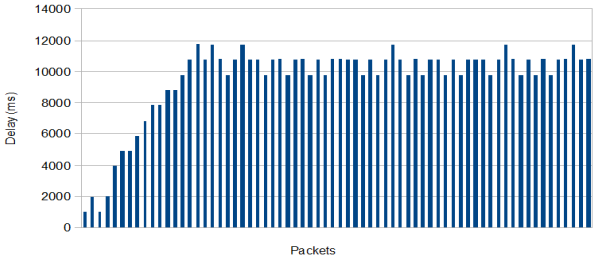
(d) Temporizador de 250 ms.



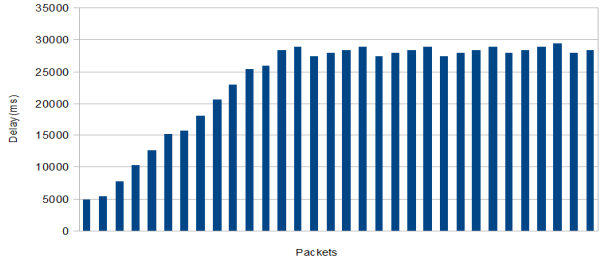
(e) Temporizador de 500 ms.



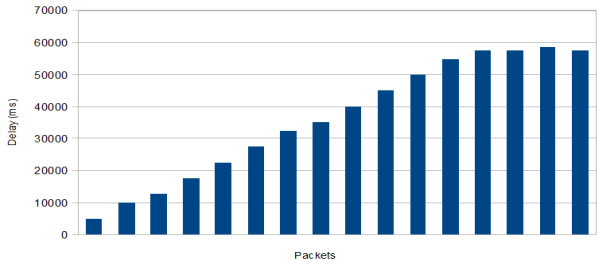
(f) Temporizador de 750 ms.



(g) Temporizador de 1000 ms.

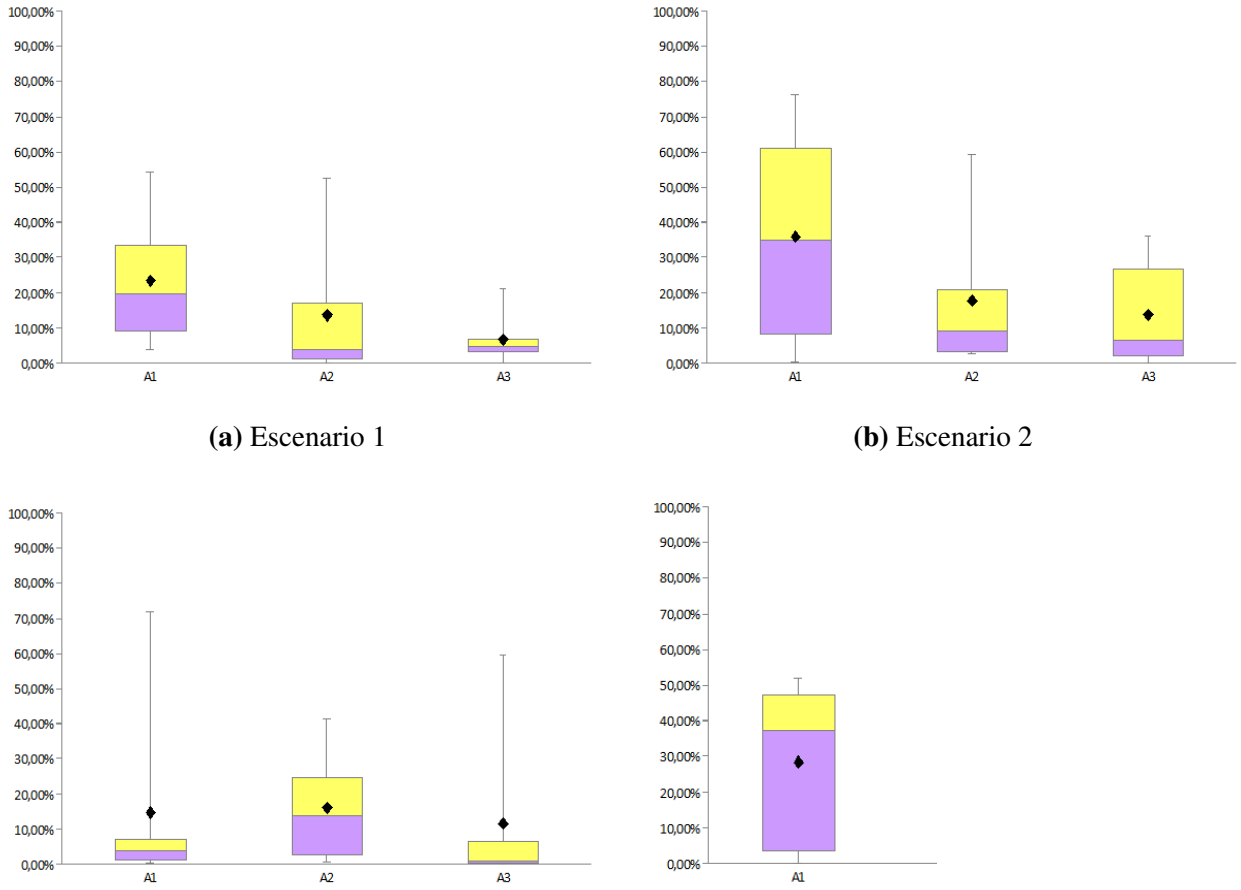


(h) Temporizador de 2500 ms.



(i) Temporizador de 5000 ms.

Figura 8.21: Jellyfish: Retardo de paquetes.



(a) Escenario 1

(b) Escenario 2

(c) Escenario 3

(d) Escenario 4

Figura 8.22: Jellyfish: Pérdida de paquetes (mínimo, 1er. cuartil, mediana, 3er. cuartil, máximo).

8.7 Ataque sybil

En esta sección se analiza el funcionamiento del ataque *sybil* presentando un estudio funcional. La implementación realizada de este ataque envía *beacons* falsos haciéndose pasar por los nodos vecinos del atacante. Sin embargo, se podría haber implementado cualquiera de las otras aplicaciones mencionadas en el estado del arte, ya que el objetivo principal era lograr enviar paquetes utilizando un identificador de nodo falso sin afectar el correcto funcionamiento de la pila de protocolos.
Los mecanismos de detección de este tipo de ataque no se basan en los mensajes particulares enviados (ya que pueden ser de distinto tipo, por ejemplo se pueden enviar paquetes falsos de datos, de enrutamiento o de control), sino que se basan en testear los recursos que el nodo tiene disponible, verificar/registrar su identidad o verificar la posición física del nodo como se menciona en el Anexo A.6.10.

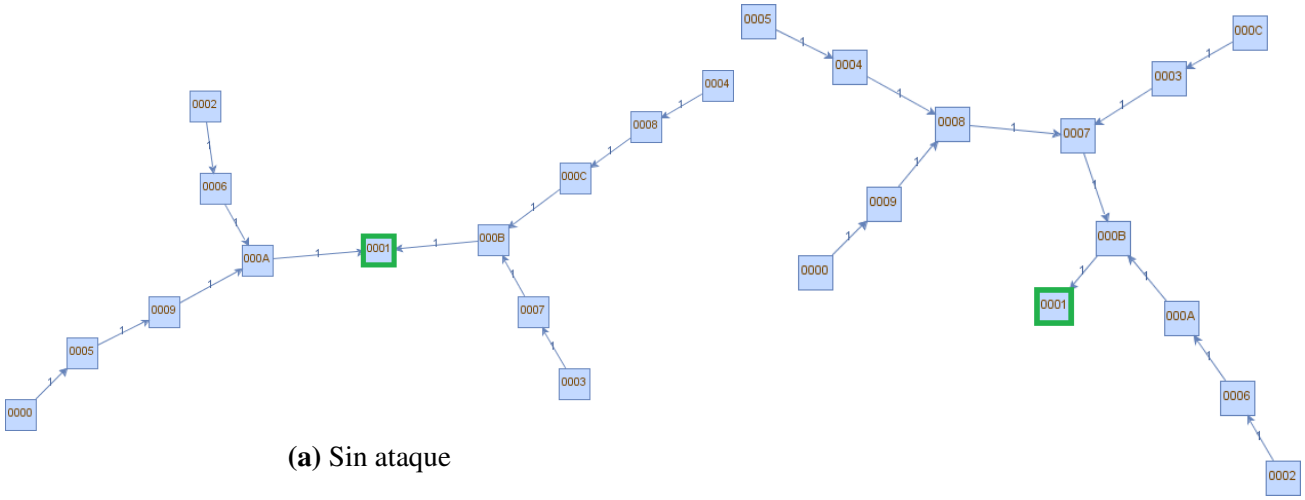
8.7.1 Análisis del funcionamiento del ataque

En la figura 8.23 se muestra una de las capturas realizadas con la herramienta para este ataque. En este caso las dos columnas de interés son 'Origen'y 'Orig Sim'que contienen la información del origen de un paquete y el origen dado por el simulador. Es decir, el primero es extraído de la estructura del paquete, es el identificador del nodo que se envía en el paquete. El segundo origen es un dato brindado por el simulador que indica que nodo envío el paquete. Observar que el primer origen puede ser falso si el nodo esta mintiendo sobre su identidad mientras que el segundo es el identificador real del nodo y no puede ser falsificado. De esta forma podemos identificar los paquetes enviados por el nodo atacante.

Num	Destino	Origen	Orig Sim	Largo datos	id Grupo	Tipo AM	Payload	Tiempo	Topologia
626	FFFF	0004	000D	1A	0022	70	02 3E 00 00 04 00 41 00 04 31 00 0C 31	0:00:11.869659	Ver
627	FFFF	000C	000D	1A	0022	70	02 3F 00 00 04 00 41 00 04 31 00 0C 31	0:00:11.978295	Ver
629	FFFF	0004	000D	1A	0022	70	02 40 00 00 04 00 41 00 04 31 00 0C 31	0:00:12.126721	Ver
630	FFFF	000C	000D	1A	0022	70	02 41 00 00 04 00 41 00 04 31 00 0C 31	0:00:12.232996	Ver
631	FFFF	0004	000D	1A	0022	70	02 42 00 00 04 00 41 00 04 31 00 0C 31	0:00:12.383194	Ver
632	FFFF	000C	000D	1A	0022	70	02 43 00 00 04 00 41 00 04 31 00 0C 31	0:00:12.458358	Ver
633	FFFF	0004	000D	1A	0022	70	02 44 00 00 04 00 41 00 04 31 00 0C 31	0:00:12.586022	Ver
635	FFFF	000C	000D	1A	0022	70	02 45 00 00 04 00 41 00 04 31 00 0C 31	0:00:12.704172	Ver
653	FFFF	0004	000D	1A	0022	70	02 46 00 00 04 00 41 00 04 31 00 0C 31	0:00:12.829675	Ver
706	FFFF	000C	000D	1A	0022	70	02 47 00 00 04 00 41 00 04 31 00 0C 31	0:00:12.992717	Ver
713	FFFF	0004	000D	1A	0022	70	02 48 00 00 04 00 41 00 04 31 00 0C 31	0:00:13.072569	Ver
715	FFFF	000C	000D	1A	0022	70	02 49 00 00 04 00 41 00 04 31 00 0C 31	0:00:13.238392	Ver
718	FFFF	0004	000D	1A	0022	70	02 4A 00 00 04 00 41 00 04 31 00 0C 31	0:00:13.344251	Ver
719	FFFF	000C	000D	1A	0022	70	02 4B 00 00 04 00 41 00 04 31 00 0C 31	0:00:13.465144	Ver
722	FFFF	0004	000D	1A	0022	70	02 4C 00 00 04 00 41 00 04 31 00 0C 31	0:00:13.603085	Ver
726	FFFF	000C	000D	1A	0022	70	02 4D 00 00 04 00 41 00 04 31 00 0C 31	0:00:13.723805	Ver
730	FFFF	0004	000D	1A	0022	70	02 4E 00 00 04 00 41 00 04 31 00 0C 31	0:00:13.852857	Ver
732	FFFF	000C	000D	1A	0022	70	02 4F 00 00 04 00 41 00 04 31 00 0C 31	0:00:14.000903	Ver
733	FFFF	0004	000D	1A	0022	70	02 50 00 00 04 00 41 00 04 31 00 0C 31	0:00:14.125477	Ver
734	FFFF	000C	000D	1A	0022	70	02 51 00 00 04 00 41 00 04 31 00 0C 31	0:00:14.241856	Ver
735	FFFF	0004	000D	1A	0022	70	02 52 00 00 04 00 41 00 04 31 00 0C 31	0:00:14.386568	Ver

Figura 8.23: Sybil: Captura de paquetes.

En la captura mostrada se filtran los paquetes que fueron enviados por el nodo 0x000D (columna 'Orig Sim'), que es el nodo atacante. Observar que este nodo envía *beacons* (Tipo AM igual a 0x70) cuyo destino es la dirección de *broadcast* (0xFFFF). Como nodo origen de los paquetes tenemos a los nodos 0x0004 y 0x000C. Observar que el nodo físico 0x000D envía paquetes como si fuera los nodos 0x0004 y 0x000C.
En las figuras 8.24a y 8.24b se muestra la topología de la misma red sin el ataque y con el ataque. Observar en la red con el ataque que el nodo 0x000D no existe, sino que se logra esconder utilizando los identificadores de los nodos víctimas. Por otro lado, las topologías de ambos casos son significativamente distintas. En la presencia del ataque se logra afectar la estructura de la red obteniendo un árbol mas desbalanceado que sin el ataque.



(a) Sin ataque

(b) Con ataque

Figura 8.24: Ataque sybil

CAPÍTULO

9

Conclusiones

Este capítulo resume los principales resultados, logros y contribuciones de este trabajo. Luego se presenta el trabajo futuro con la tendencia futura de las redes de sensores inalámbricos así como posibles líneas de trabajo para extender esta investigación.

9.1 Resultados y contribuciones

TinyOS y el estándar IEEE 802.15.4 son los componentes dominantes en las WSN de la actualidad. En este trabajo se logró estudiar la vulnerabilidades de estos y demostrar los efectos graves que pueden tener ciertos ataques en el funcionamiento de la red. Por otro lado herramientas como la que se desarrolló en este trabajo se pueden utilizar para buscar soluciones mejores para la defensa de las WSN. En particular, la herramienta aquí desarrollada permite la inspección de la funcionalidad de una red de sensores mediante el análisis de los mensajes de radio escuchados, así como la ejecución de varios ataques que se enfocan en los protocolos de la capa de enlace y de red.
En este trabajo se logró estudiar exhaustivamente distintos tipos de ataques que pueden ocurrir en las WSN, identificar la posibilidad de aplicarlos en protocolos utilizados por las WSN e implementarlos en el marco de la herramienta de *ethical hacking*. Por otro lado se logró estudiar el impacto y comportamiento de cada uno de los ataques implementados que resalta la necesidad de mejores mecanismos de seguridad y deja al descubierto las graves consecuencias que pueden tener en este tipo de redes. A través del estudio de estas consecuancias se quiere motivar a un mejor diseño de los protocolos de seguridad.
Como se mencionó al inicio de este proyecto los mecanismos convencionales de seguridad pueden no ser adecuados para este tipo de redes. Por otro lado, debido a las propiedades de las WSN, donde los nodos mismos deben enrutar sus paquetes, es que los mecanismos típicos que proporcionana confidencialidad, integridad y autenticación no son suficientes. Por tal motivo, existe una tendencia a intentar proteger este tipo de redes con otros mecanismos de seguridad, como por ejemplo, sistemas de detección de intrusos (IDS) donde los nodos colaboran entre sí para detectar situaciones anormales en el comportamiento de otros nodos y así poder tomar medidas correctivas. Para el desarrollo de este tipo de mecanismos de seguridad se necesitan detectar patrones y comportamientos típicos de ataque. Por otro lado, una vez implementado el mecanismo de defensa también se necesita contar con nodos maliciosos que realicen el ataque para poder probar si el mecanismo realmente funciona y detecta su presencia. Actualmente para testear la implementación del mecanismo de seguridad se debe primero implementar el ataque. Por lo que contar con herramientas que brinden un conjunto de ataques ya implementados, facilitaría es desarrollo y *testing* de este tipo de mecanismos.

Por otro lado, esta herramienta permite realizar escuchas y análisis del tráfico, lo cual a parte de permite estudiar la seguridad de la red, también sirve como una herramienta de *debug* y *testing* de las aplicaciones comunmente implementadas.

9.2 Trabajo futuro

Como se ha mencionado a lo largo de este estudio es importante seguir investigando nuevos modelos de amenazas y vulnerabilidades de las WSN que pueden ser explotadas por un adversario con el fin de dañar la red. Por otro lado, es vital contar con un protocolo de red más seguro que CTP. También se deben considerar las vulnerabilidades aquí presentadas para los protocolos de acceso del medio, tratando de mitigar debilidades inherentes del medio físico como puede ocurrir con las colisiones en el canal. A lo largo de este trabajo se han planteado ciertas cuestiones respecto al impacto de los ataques que quedaron pendientes de resolver y que se podrían continuar estudiando.
Por otro lado, es escencial contar con más y mejores herramientas para *testing* de seguridad como actualmente se tienen con las redes tradicionales. Por tal motivo, nuevos ataques pueden ser estudiados, implementados e incorporados a la herramienta.

Bibliografía

[1] Kashif Kifayat, Madjid Merabti, Qi Shi, and David Llewellyn-Jones. Security in wireless sensor networks. In Peter Stavroulakis and Mark Stamp, editors, *Handbook of Information and Communication Security*, chapter 26, pages 513–552. Springer Berlin Heidelberg, 2010. vii, viii, 6, 17, 18, 20, 22, 24, 114, 115, 120, 121, 127

[2] Arquitectura de una aplicación militar para vigilancia utilizando una WSN. En línea: http://www.tomas-sanchez.com/technicalblog/post2/military_prof.png. Último acceso: 20 Mayo 2013. vii, 7, 23

[3] Aplicación de una red de sensores para monitorizar la estructura de una construcción. En línea: http://www.prismaelectronics.eu/site/images/stories/prisma_sense/shm2.jpg. Último acceso: 20 Mayo 2013. vii, 8, 23, 24

[4] Aplicación de una red de sensores para monitorizar la estructura de una construcción. En línea: http://www.prismaelectronics.eu/site/images/stories/prisma_sense /shm1.jpg. Último acceso: 20 Mayo 2013. vii, 8

[5] Y. Hovakeemian, K. Naik, and A. Nayak. A survey on dependability in body area networks. In *Medical Information Communication Technology (ISMICT), 2011 5th International Symposium on*, pages 10–14, 2011. vii, 10

[6] A.K. Pathan, Hyung-Woo Lee, and Choong seon Hong. Security in wireless sensor networks: issues and challenges. In *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, volume 2, pages 6 pp.–1048, 2006. vii, 18, 20, 22, 23

[7] García C. Impacto de la seguridad en redes inalámbricas de sensores ieee 802.15.4. Master's thesis, Universidad Complutense de Madrid, 2010. vii, 21, 22, 23

[8] Plataforma Shimmer SPAN: Especificaciones. En línea: http://www.shimmersensing.com/images/uploads/docs/span_wireless_platform_spec_-sheet.pdf. vii, 26

[9] Shimmer Dock. En línea: http://www.shimmersensing.com/shop/shimmer-dock. vii, 26

[10] James Kurose and Keith Ross. *Computer Networking: A Top-Down Approach*. Addison-Wesley Publishing Company, 5th edition, 2009. Computer Networking. vii, 32, 33

[11] SangShin Jung, Marco Valero, Anu Bourgeois, and Raheem Beyah. Attacking beacon-enabled 802.15.4 networks. In Sushil Jajodia and Jianying Zhou, editors, *Security and Privacy in Communication Networks*, volume 50 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 253–271. Springer Berlin Heidelberg, 2010. vii, 36, 38

[12] Interface Zigbee.
En línea: http://researcher.most.gov.tw/public/8905780/attachment/791017365871.pdf. vii, 39

[13] Sangsoon Lim, Suchul Lee, Joon Yoo, and Chong-Kwon Kim. Nbp: light-weight narrow band protection for zigbee and wi-fi coexistence. *EURASIP Journal on Wireless Communications and Networking*, 2013(1):1–13, 2013. vii, 44

[14] Islam Hegazy, Reihaneh Safavi-Naini, and Carey Williamson. Exploiting routing tree construction in ctp. In Souhwan Jung and Moti Yung, editors, *Information Security Applications*, volume 7115 of *Lecture Notes in Computer Science*, pages 256–270. Springer Berlin Heidelberg, 2012. vii, 56, 61, 83, 84, 85

[15] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, and Philip Levis. Four-bit wireless link estimation. En línea: http://sing.stanford.edu/pubs/sing-07-00.pdf, 2007. vii, 58, 59

[16] Ugo Colesanti and Silvia Santini. The collection tree protocol for the castalia wireless sensor networks simulator. (729), 2011. vii, viii, 60, 131, 132

[17] Mauri Kuorilehto, Marko Hännikäinen, and Timo D. Hämäläinen. A survey of application distribution in wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.*, 2005(5):774–788, October 2005. viii, 114

[18] V. Madan and SRN. Reddy. Review of wireless sensor mote platforms. *VSRD International Journal of Electrical, Electronics & Communication Engineering*, 2(2):50–55, 2012. xi, 12

[19] Mica2 DataSheet.
En línea: http://www.eol.ucar.edu/isf/facilities/isa/internal/crossbow/datasheets/mica2.pdf. Último acceso: 2 Febrero 2014. xi, 12

[20] M. Johnson, M. Healy, P. van de Ven, M.J. Hayes, J. Nelson, T. Newe, and E. Lewis. A comparative review of wireless sensor network mote technologies. In *Sensors, 2009 IEEE*, pages 1439–1442, 2009. xi, 12

[21] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 162–175, New York, NY, USA, 2004. ACM. 2

[22] A. Nayak and I. Stojmenovic. *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*. Wiley, 2010. 6

[23] T Arampatzis, J. Lygeros, and S. Manesis. A survey of applications of wireless sensors and wireless sensor networks. In *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, pages 719–724, 2005. 6

[24] S.M. Diamond and M.G. Ceruti. Application of wireless sensor network to military information integration. In *Industrial Informatics, 2007 5th IEEE International Conference on*, volume 1, pages 317–322, 2007. 6

[25] Michael Winkler, Klaus dieter Tuchs, Kester Hughes, and Graeme Barclay. Paper theoretical and practical aspects of military wireless sensor networks. Journal of Telecommunications and information technology, February 2008. 6

[26] Youngsoo Kim, Jonggu Kang, Daeyoung Kim, Eunjo Kim, Poh Kit Chong, and Suckbin Seo. Design of a fence surveillance system based on wireless sensor networks. In *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, Autonomics '08, pages 4:1–4:7, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 6

[27] Kewei Sha, Weisong Shi, and O. Watkins. Using wireless sensor networks for fire rescue applications: Requirements and challenges. In *Electro/information Technology, 2006 IEEE International Conference on*, pages 239–244, 2006. 7

[28] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 13–24, New York, NY, USA, 2004. ACM. 7

[29] Aravind Iyer, SunilS. Kulkarni, Vivek Mhatre, and CatherineP. Rosenberg. A taxonomy-based approach to design of large-scale sensor networks. In Yingshu Li, MyT. Thai, and Weili Wu, editors, *Wireless Sensor Networks and Applications*, Signals and Communication Technology, pages 3–33. Springer US, 2008. 8

[30] Aplicación para estudiar el comportamiento de los animales. En línea: http://images.fastcompany.com/upload/glab2.jpg. Último acceso: 20 Junio 2013. 9

[31] Aplicación de una WSN para monitorización de cultivos. En línea: http://inmtn.com/images/csi/wireless_sensors.png. Último acceso: 19 Junio 2013. 9

[32] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a wireless sensor network on an active volcano. *Internet Computing, IEEE*, 10(2):18–25, 2006. 8, 9

[33] Kshitij Shinghal, Arti Noor, Neelam Srivastava, and Raghuvir Singh. Wireless sensor networks in agriculture: For potato farming. *International Journal of Engineering Science and Technology*, 2(8):3955–3963, 2010. 9

[34] DD Chaudhary, SP Nayse, and LM Waghmare. Application of wireless sensor networks for greenhouse parameter control in precision agriculture. *International Journal of Wireless & Mobile Networks (IJWMN) Vol*, 3(1):140–149, 2011. 9

[35] Laurynas Riliskis, Evgeny Osipov, and Miklós Maróti. Tos-ns3: a framework for emulating wireless sensor networks in the ns3 network simulator. In *Proceedings of the 3rd International Workshop on NS3, in conjunction with SimuTOOLS, Malaga, Spain*, 2010. 10

[36] TinyOS. En línea: http://www.tinyos.net/. Último acceso: 26 Mayo 2013. 10

[37] Contiki. En línea: http://www.contiki-os.org/start.html. Último acceso: 26 Mayo 2013. 10, 27

[38] Mantis. En línea: http://mantisos.org/index/tiki-index.php.html. Último acceso: 26 Mayo 2013. 10

[39] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462, 2004. 11

[40] Shah Bhatti, James Carlson, Hui Dai, Jing Deng, Jeff Rose, Anmol Sheth, Brian Shucker, Charles Gruenwald, Adam Torgerson, and Richard Han. Mantis os: an embedded multithreaded operating system for wireless micro sensor platforms. *Mob. Netw. Appl.*, 10(4):563–579, August 2005. 11

[41] MuhammadOmer Farooq, Sadia Aziz, and AbdulBasit Dogar. State of the art in wireless sensor networks operating systems: A survey. In *Future Generation Information Technology*, volume 6485 of *Lecture Notes in Computer Science*, pages 616–631. Springer Berlin Heidelberg, 2010. 11

[42] NesC. En línea: http://nescc.sourceforge.net/. Último acceso: 26 Julio 2013. 11

[43] Tmote Sky DataSheet. En línea: http://www.snm.ethz.ch/projects/tmotesky. Último acceso: 11 Agosto 2013. 12, 13

[44] Shimmer DataSheet.
En línea: http://www.shimmersensing.com/images/uploads/docs/shimmer_wireless_sensor_platform_spec_sheet.pdf. Último acceso: 20 Agosto 2013. 13

[45] IRIS Datasheet. En línea: http://www.dinesgroup.org/projects/images/pdf_files/iris_datasheet.pdf. Último acceso: 20 Agosto 2013. 13

[46] Sun SPOT DataSheet. En línea: http://www.sunspotworld.com/products/index.html. Último acceso: 20 Agosto 2013. 13

[47] TelosB datasheet. En línea: http://www.willow.co.uk/telosb_datasheet.pdf. Último acceso: 26 Julio 2013. 12

[48] TmoteSky datasheet.
En línea: http://www.eecs.harvard.edu/ konrad/projects/shimmer/references/tmote-sky-datasheet.pdf. Último acceso: 26 Julio 2013. 12

[49] MicaZ datasheet. En línea: http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf. Último acceso: 26 Julio 2013. 12

[50] Mica2 DataSheet.
En línea: http://www.eol.ucar.edu/isf/facilities/isa/internal/crossbow/datasheets/mica2.pdf. Último acceso: 20 Agosto 2013. 12

[51] Sriram Sanka and Gaurav Konchady. Communication between wireless sensor devices and gnu radio, 2009. 13, 138

[52] En línea: http://www.snm.ethz.ch/snmwiki/Projects/ComparisonOfCC2420CC100CC1021 CommunicationRanges. Comparison of cc2420 cc1000 and cc1021 communication ranges. 13

[53] Jaydip Sen. A survey on wireless sensor network security. *International Journal of Communication Networks and Information Security (IJCNIS)*, abs/1011.1529(2), 2010. 15, 17, 19

[54] HC Chaudhari and LU Kadam. Wireless sensor networks: Security, attacks and challenges. *International Journal of Networking*, 1(1):04–16, 2011. 16, 18, 19, 23, 24

[55] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *In First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, 2002. 16, 20, 21

[56] Mayank Saraogi. Security in wireless sensor networks. *Department of Computer Science, University of Tennessee, Knoxville*, 2004. 17, 18, 21, 22, 120, 126, 127

[57] Jing Deng, R. Han, and S. Mishra. Countermeasures against traffic analysis attacks in wireless sensor networks. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 113–126, 2005. 17

[58] J.R. Ward and M. Younis. A physical layer metric for measuring base station anonymity in wireless sensor networks. In *Communications (ICC), 2012 IEEE International Conference on*, pages 6689–6693, 2012. 17

[59] Uday Acharya and Mohamed Younis. An approach for increasing base-station anonymity in sensor networks. In *Proceedings of the 2009 IEEE international conference on Communications*, ICC'09, pages 609–613, Piscataway, NJ, USA, 2009. IEEE Press. 17

[60] Y. Ebrahimi and M. Younis. Increasing transmission power for higher base-station anonymity in wireless sensor network. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5, 2011. 17

[61] Dijiang Huang. On measuring anonymity for wireless mobile ad-hoc networks. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 779–786, 2006. 17

[62] Tanya Roosta, Shiuhpyng Shieh, and Shankar Sastry. taxonomy of security attacks in sensor networks and countermeasures. In *In The First IEEE International Conference on System Integration and Reliability Improvements. Hanoi*, pages 13–15, 2006. 17, 24, 122

[63] John Paul Walters, Zhengqiang Liang, Weisong Shi, and Vipin Chaudhary. Wireless sensor network security: A survey," in book chapter of security. In Yang Xiao, editor, *Security in Distributed, Grid, and Pervasive Computing*, volume 17. Citeseer, 2007. 18, 22

[64] Yee Wei Law, Marimuthu Palaniswami, Lodewijk Van Hoesel, Jeroen Doumen, Pieter Hartel, and Paul Havinga. Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols. *ACM Trans. Sen. Netw.*, 5(1):6:1–6:38, February 2009. 18

[65] Teodor-Grigore Lupu. Main types of attacks in wireless sensor networks. In *Proceedings of the 9th WSEAS international conference on signal, speech and image processing, and 9th WSEAS international conference on Multimedia, internet & video technologies*, SSIP '09/MIV'09, pages 180–185, Stevens Point, Wisconsin, USA, 2009. World Scientific and Engineering Academy and Society (WSEAS). 18

[66] V Manju. Analysis of denial of sleep attack in wsn. In *International Conference on Recent Development in Engineering and Technology*. 19

[67] Doddapaneni C. and Ghosh A. Analysis of denial-of-service attacks on wireless sensor networks using simulation. In *Middlesex University*. 19

[68] David R. Raymond, R.C. Marchany, M.I. Brownfield, and S.F. Midkiff. Effects of denial-of-sleep attacks on wireless sensor network mac protocols. *Vehicular Technology, IEEE Transactions on*, 58(1):367–380, 2009. 19

[69] T. Giannetsos, T. Dimitriou, and N.R. Prasad. State of the art on defenses against wormhole attacks in wireless sensor networks. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pages 313–318, 2009. 21, 125

[70] Wen-Tao Zhu. Node replication attacks in wireless sensor networks: Bypassing the neighbor-based detection scheme. In *Network Computing and Information Security (NCIS), 2011 International Conference on*, volume 2, pages 156–160, 2011. 22

[71] Jun-Won Ho, M. Wright, and S.K. Das. Fast detection of mobile replica node attacks in wireless sensor networks using sequential hypothesis testing. *Mobile Computing, IEEE Transactions on*, 10(6):767–782, June 2011. 22, 126

[72] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, IPSN '04, pages 259–268, New York, NY, USA, 2004. ACM. 22, 126

[73] Driver para nodo Shimmer. En línea: http://www.ftdichip.com/drivers/vcp.htm. 26

[74] Wireshark. En línea: http://www.wireshark.org/. 27

[75] Simulador AVRORA. En línea: http://compilers.cs.ucla.edu/avrora/. 27

[76] Institute of Electrical and Inc. Electronics Engineers. *IEEE Std. 802.15.4-2006, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specic requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*. New York, September 2006. 31

[77] David Moss, Jonathan Hui, Philip Levis, and Jung Il Choi. Tep 126 - cc2420 radio stack. 2007. 33

[78] Che Woo Na. Ieee 802.15.4 wireless sensor networks: Gts scheduling and service differentiation. *Dissertation for the degree Doctor of Philosophy in Electrical and Comptuer Engineering*, 2011. 35, 36

[79] KTH. En línea: https://eeweb01.ee.kth.se/upload/publications/reports/2011/trita-ee_2011_-002.pdf. 39

[80] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, Sukun Kim, Philip Levis, , and Alec Woo. Tep 123 - collection tree protocol (ctp). 2007. 53

[81] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14, New York, NY, USA, 2009. ACM. 54

[82] Omprakash Gnawali. Tep 124 - the link estimation exchange protocol (leep). 2007. 56

[83] M. Sukor, S. Ariffin, N. Fisal, S.K.S. Yusof, and A. Abdallah. Performance study of wireless body area network in medical environment. In *Modeling Simulation, 2008. AICMS 08. Second Asia International Conference on*, pages 202–206, 2008. 117

[84] Kyung Sup Kwak, Sana Ullah, and Niamat Ullah. An overview of ieee 802.15. 6 standard. In *Applied Sciences in Biomedical and Communication Technologies (ISABEL), 2010 3rd International Symposium on*, pages 1–6. IEEE, 2010. 117

[85] Katsuyuki Okeya and Tetsu Iwata. Side channel attacks on message authentication codes. In Refik Molva, Gene Tsudik, and Dirk Westhoff, editors, *Security and Privacy in Ad-hoc and Sensor Networks*, volume 3813 of *Lecture Notes in Computer Science*, pages 205–217. Springer Berlin Heidelberg, 2005. 120

[86] S. H. Jokhio, I. A. Jokhio, and A. H. Kemp. Node capture attack detection and defence in wireless sensor networks. *Wireless Sensor Systems, IET*, 2(3):161–169, 2012. 121

[87] Chi Zhang, Yanchao Zhang, and Yuguang Fang. Defending against physical destruction attacks on wireless sensor networks. In *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pages 1–7, 2006. 121

[88] V.C. Manju and M.S. Kumar. Detection of jamming style dos attack in wireless sensor network. In *Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on*, pages 563–567, 2012. 122

[89] P. Reindl, K. Nygard, and Xiaojiang Du. Defending malicious collision attacks in wireless sensor networks. In *Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on*, pages 771–776, 2010. 122

[90] Tianzhen Cheng, Ping Li, and Sencun Zhu. An algorithm for jammer localization in wireless sensor networks. In *Proceedings of the 2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, AINA '12, pages 724–731, Washington, DC, USA, 2012. IEEE Computer Society. 122

[91] Mario Cagalj, Srdjan Capkun, and Jean pierre Hubaux. Wormhole-based anti-jamming techniques in sensor networks. *IEEE Transactions on Mobile Computing*, 6:100–114, 2007. 122

[92] S.K. Jain and K. Garg. A hybrid model of defense techniques against base station jamming attack in wireless sensor networks. In *Computational Intelligence, Communication Systems and Networks, 2009. CICSYN '09. First International Conference on*, pages 102–107, 2009. 122

[93] S.K. Dhurandher, S. Misra, D. Agrawal, and A. Rayankula. Using honeynodes along with channel surfing for defense against jamming attacks in wireless networks. In *Systems and Networks Communications, 2008. ICSNC '08. 3rd International Conference on*, pages 197–201, 2008. 123

[94] Hung-Min Sun, Shih-Pu Hsu, and Chien-Ming Chen. Mobile jamming attack and its countermeasure in wireless sensor networks. In *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*, volume 1, pages 457–462, 2007. 123

[95] Jian Yin and S.K Madria. A hierarchical secure routing protocol against black hole attacks in sensor networks. In *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, volume 1, pages 8 pp.–, 2006. 123

[96] Hongmei Deng, W. Li, and D.P. Agrawal. Routing security in wireless ad hoc networks. *Communications Magazine, IEEE*, 40(10):70–75, 2002. 123

[97] Bo Yu and Bin Xiao. Detecting selective forwarding attacks in wireless sensor networks. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 8 pp.–, 2006. 124

[98] J. Brown and Xiaojiang Du. Detection of selective forwarding attacks in heterogeneous sensor networks. In *Communications, 2008. ICC '08. IEEE International Conference on*, pages 1583–1587, 2008. 124

[99] Wang Xin-sheng, Zhan Yong-zhao, Xiong Shu-ming, and Wang Liang-min. Lightweight defense scheme against selective forwarding attacks in wireless sensor networks. In *Cyber-Enabled Distributed Computing and Knowledge Discovery, 2009. CyberC '09. International Conference on*, pages 226–232, 2009. 124

[100] S. Kaplantzis, A. Shilton, Nallasamy Mani, and Y.Ahmet Sekercioglu. Detecting selective forwarding attacks in wireless sensor networks using support vector machines. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pages 335–340, 2007. 124

[101] Tran Hoang Hai and Eui-Nam Huh. Detecting selective forwarding attacks in wireless sensor networks using two-hops neighbor knowledge. In *Network Computing and Applications, 2008. NCA '08. Seventh IEEE International Symposium on*, pages 325–331, 2008. 124

[102] Hung-Min Sun, Chien-Ming Chen, and Ying-Chu Hsiao. An efficient countermeasure to the selective forwarding attack in wireless sensor networks. In *TENCON 2007 - 2007 IEEE Region 10 Conference*, pages 1–4, 2007. 124

[103] C. Tumrongwittayapak and R. Varakulsiripunth. Detecting sinkhole attacks in wireless sensor networks. In *ICCAS-SICE, 2009*, pages 1966–1971, 2009. 124

[104] D. Sheela, K.C. Naveen, and G. Mahadevan. A non cryptographic method of sink hole attack detection in wireless sensor networks. In *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, pages 527–532, 2011. 124

[105] S. Sharmila and G. Umamaheswari. Detection of sinkhole attack in wireless sensor networks using message digest algorithms. In *Process Automation, Control and Computing (PACC), 2011 International Conference on*, pages 1–6, 2011. 125

[106] E.C.-H. Ngai, Jiangchuan Liu, and M.R. Lyu. On the intruder detection for sinkhole attack in wireless sensor networks. In *Communications, 2006. ICC '06. IEEE International Conference on*, volume 8, pages 3383–3389, 2006. 125

[107] Changlong Chen, Min Song, and G. Hsieh. Intrusion detection of sinkhole attacks in large-scale wireless sensor networks. In *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on*, pages 711–716, 2010. 125

[108] J.S. Terence. Secure route discovery against wormhole attacks in sensor networks using mobile agents. In *Trendz in Information Sciences and Computing (TISC), 2011 3rd International Conference on*, pages 110–115, 2011. 125

[109] Yih-Chun Hu, Adrian Perrig, and David B Johnson. Wormhole attacks in wireless networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):370–380, 2006. 125

[110] Yih-Chun Hu, A. Perrig, and D.B. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1976–1986 vol.3, 2003. 125

[111] Ali Modirkhazeni, Saeedeh Aghamahmoodi, and N Niknejad. Distributed approach to mitigate wormhole attack in wireless sensor networks. In *Networked Computing (INC), 2011 The 7th International Conference on*, pages 122–128. IEEE, 2011. 125

[112] T. Korkmaz. Verifying physical presence of neighbors against replay-based attacks in wireless ad hoc networks. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 2, pages 704–709 Vol. 2, 2005. 125

[113] Dhara Buch and D. Jinwala. Detection of wormhole attacks in wireless sensor network. In *Advances in Recent Technologies in Communication and Computing (ARTCom 2011), 3rd International Conference on*, pages 7–14, 2011. 125

[114] Md. Abdul Hamid, Md. Mamun or rashid, and Choong Seon Hong. Defense against lap-top class attacker in wireless sensor network. ISBN 89-5519-129-4, Feb 2006. 126

[115] Wassim Znaidi, Marine Minier, and Stéphane Ubéda. Hierarchical node replication attacks detection in wireless sensors networks. In *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, pages 82–86. IEEE, 2009. 126

[116] Kai Xing, Fang Liu, Xiuzhen Cheng, and David Hung-Chang Du. Real-time detection of clone attacks in wireless sensor networks. In *Distributed Computing Systems, 2008. ICDCS'08. The 28th International Conference on*, pages 3–10. IEEE, 2008. 126

[117] Mauro Conti, Roberto Di Pietro, Luigi Vincenzo Mancini, and Alessandro Mei. Distributed detection of clone attacks in wireless sensor networks. *Dependable and Secure Computing, IEEE Transactions on*, 8(5):685–698, 2011. 126

[118] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *Security and Privacy, 2005 IEEE Symposium on*, pages 49–63. IEEE, 2005. 126

[119] Bo Zhu, Venkata Gopala Krishna Addada, Sanjeev Setia, Sushil Jajodia, and Sankardas Roy. Efficient distributed detection of node replication attacks in sensor networks. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 257–267. IEEE, 2007. 126

[120] Chia-Mu Yu, Chun-Shien Lu, and Sy-Yen Kuo. Efficient and distributed detection of node replication attacks in mobile sensor networks. In *Vehicular Technology Conference Fall (VTC 2009-Fall), 2009 IEEE 70th*, pages 1–5, 2009. 126

[121] Shaohe Lv, Xiaodong Wang, Xin Zhao, and Xingming Zhou. Detecting the sybil attack cooperatively in wireless sensor networks. In *Computational Intelligence and Security, 2008. CIS '08. International Conference on*, volume 1, pages 442–446, 2008. 127

[122] Ren Xiu-li and Yang Wei. Method of detecting the sybil attack based on ranging in wireless sensor network. In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, pages 1–4, 2009. 127

[123] Murat Demirbas and Youngwhan Song. An rssi-based scheme for sybil attack detection in wireless sensor networks. In *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, WOWMOM '06, pages 564–570, Washington, DC, USA, 2006. IEEE Computer Society. 127

[124] Shanshan Chen, Geng Yang, and Shengshou Chen. A security routing mechanism against sybil attack for wireless sensor networks. In *Communications and Mobile Computing (CMC), 2010 International Conference on*, volume 1, pages 142–146, 2010. 127

[125] En línea: https://eva.fing.edu.uy/mod/page/view.php?id=33568. Configurar sniffer de atmel + wireshak. 129

APÉNDICE

A

Apéndice

A.1 Arquitectura de un nodo

Un nodo sensor es un pequeño dispositivo alimentado por batería que es capaz de procesar la información obtenida de los sensores y de comunicarse con otros nodos en la red. La arquitectura típica de un nodo sensor se muestra en la Figura A.1 . Esta se compone de los elementos mencionados a continuación [17].

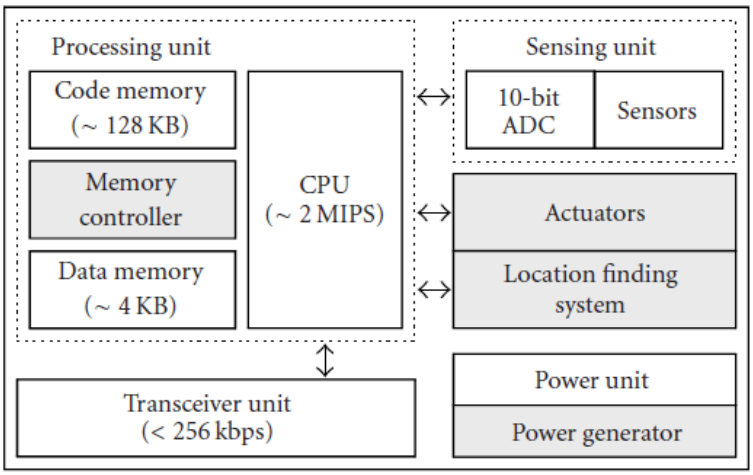


Figura A.1: Arquitectura hardware de un nodo[17]

Una unidad de procesamiento que consta de CPU, dispositivos de almacenamiento, y un controlador de memoria (opcional) para el acceso a la memoria de instrucciones de la CPU principal.
Una unidad de detección que consta de sensores y un convertidor analógico a digital (ADC - *Analogic to Digital Converter*). Los sensores son dispositivos físicos que producen señales eléctricas en respuesta a los cambios físicos que ocurren en el medio ambiente que los rodea. Permiten midir parámetros ambientales tales como la temperatura, la humedad, la concentración de gas, la intensidad de la luz, entre otros. Las señales analógicas producidas por los sensores son enviadas al ADC y luego los valores digitales obtenidos se envían a la unidad de procesamiento. Dentro de la unidad de procesamiento, los valores digitales son calibrados utilizando una ecuación de calibración ya definida. La mayoría de los microcontroladores tienen la unidad de ADC incorporada, por lo que no se necesita un módulo ADC externo.
Una unidad de transmisión que permita la comunicación con otros nodos. Los transmisores pueden operar en tres modos: transmisión, recepción y dormido. El consumo de energía de un nodo es del orden de mW cuando está activo y del orden de μW cuando está en modo dormido. La mayor parte del consumo de energía en el nodo sensor se da durante la transmisión de datos.
Una fuente de energía que se puede extender con un generador de energía. En un nodo sensor, la energía se consume para la detección, procesamiento de datos y la comunicación, siendo la mayor parte de la energía consumida en la comunicación de datos. Las fuentes de energía más populares utilizados en los nodos sensores son las baterías recargables. Recientemente, se comenzaron a utilizar paneles solares en conjunto con las baterías para recargarlas.
Otros dispositivos periféricos, como actuadores (por ejemplo un sistema motor para mover al nodo) o sistemas de localización, pueden ser agregados dependiendo de los requerimientos de la aplicación.

A.2 Pila de protocolos

La pila de protocolos de una WSN se muestra en la figura A.2 y en una combinación de las siguientes capas [1]: física, enlace, red, transporte y aplicación. Además se cuenta con los planos de adminis-

tración de la energía, administración de la movilidad y administración de las tareas. Cada capa/plano cuenta con un conjunto de protocolos con diferentes operaciones.

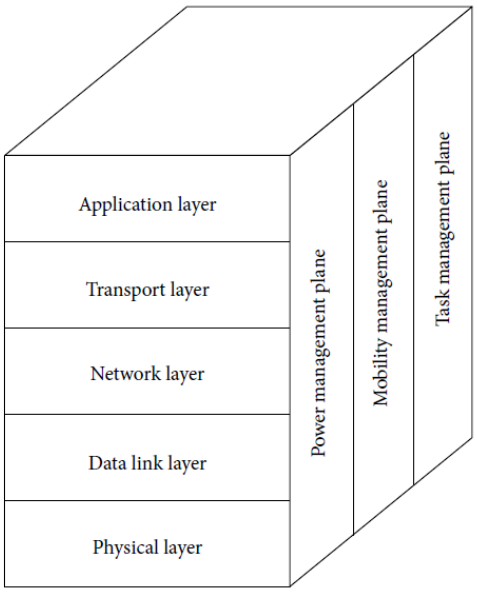


Figura A.2: Stack de Protocolos de una WSN [1]

Capa física: La capa física es la responsable de seleccionar la frecuencia, detectar señales, modulación y cifrado de los datos.
Capa enlace: La capa de enlace es responsable de multiplexar los flujos de datos, detección de las tramas de datos, acceso al medio y control de errores. También asegura las conexiones punto-a-punto o punto-a-multipunto confiables. El protocolo de acceso al medio (MAC-*Medium Access Protocol*) debe tratar de minimizar las colisiones con los vecinos teniendo en cuenta que estas redes se despliegan en ambientes ruidosos y los nodos pueden ser móviles, también se debe tomar en cuenta la energía con la que cuenta el dispositivo.
Capa red: La capa de red se encarga de encontrar rutas para encaminar los paquetes de extremo a extremo de la red. Los principios básicos a tener en cuenta cuando se diseña una capa de red para WSN son manejar eficiente la energía disponible, proveer direccionamiento basado en atributos, proveer conocimiento de la ubicación y agregación de datos.
Capa transporte: La capa de transporte ayuda a mantener el flujo de datos si la aplicación de la WSN lo necesita.
Capa aplicación: La capa de aplicación se encarga de la gestión y asignación de tareas así como de anunciar datos tales como la activación y desactivación de los nodos, sincronización de reloj, consultas sobre la configuración de la red de sensores, autenticación, distribución de claves, etc.
Administración de energía: tiene como tarea principal gestionar el consumo de energía de un nodo. Por ejemplo, un nodo puede dormirse luego de recibir un mensaje de su vecino. También puede avisar cuando le queda poca energía a sus vecinos para no participar en los mensajes de enrutamiento.
Administración de movilidad: detecta y registra el movimiento de los nodos de modo que un nodo siempre conozca quienes son sus nodos vecinos y mantener una ruta hacia la estación base. Conociendo quienes son sus vecinos un nodo puede balancear el uso de energía y las tareas a realizar.
Administración de las tareas: balancea y organiza las tareas de captura de datos de una región específica, por ejemplo, a veces no es necesario que todos los nodos en una región capturen datos al mismo tiempo u otras veces, algunos nodos realizan más tareas que otros dependiendo del nivel de energía.

Estos planes de administración nos necesarios para que los nodos puedan trabajar juntos de forma colaborativa, haciendo un uso eficiente de la energía, enrutando datos es una red de sensores móvil y compartiendo recursos entre los nodos.

A.3 Tecnologías de radio

Actualmente, la tecnología de comunicación utilizada en las WSN se basa en tecnologías de red inalámbricas de radio frecuencia entre las cuales se encuentran Bluetooth, Bluetooth Low Energy, ZigBee, WiFi y UWB. En la tabla A.1 se presenta un resumen de las principales características de cada una de esas tecnologías.

	Bluetooth	Bluetooth Low Energy	ZigBee	WiFi	UWB
Banda de Frecuencia	2.4 GHz ISM	2.4 GHz ISM	2.4 GHz ISM	2.4/3.6/5Gb	3.1-10.6 GHz
Tasa de transmisión	24 Mbps	1 Mb/s	250 Kb/s	54 Mb/s	480 Mb/s
Área de cubrimiento	10 mts	10 mts	30-100 mts	30 mts	menor 10 mts
Topología de red	Estrella	Estrella	Estrella / Malla	—	Estrella
Costo	Barato	—	Barato	Barato	—
Confiabilidad	No Confiable	—	Confiable	No confiable	—
Escalabilidad	Pobre		Excelente	Razonable	—
Consumo de energía	Promedio	—	Bajo	Alto	—
Interoperabilidad	Excelente	—	Buena	Excelente	—

Cuadro A.1: Tecnologías de radio

Bluetooth - IEEE 802.15.1

Bluetooth es una tecnología inalámbrica diseñada para comunicar una variedad de dispositivos como teléfonos, PDAs, computadoras portátiles, etc. Se basa en el estándar IEEE 802.15.1 el cual especifica tres tipos de dispositivos con diferentes poderes de transmisión y cubrimiento.
Bluetooth tiene problemas de escalabilidad, dado que solamente soporta un máximo de siete dispositivos en una sola picored (*Piconet*) organizada en una topología estrella. Tampoco soporta de forma adecuada las comunicaciones multi salto. La pila de protocolos es compleja y el consumo de energía alto en relación a IEEE 802.15.4.

Bluetooth low energy

Bluetooth low energy presenta un consumo de energía bajo brindando una tasa de transmisión de datos mayor a 1 Mbps. La sincronización de los dispositivos puede ser realizada en unos pocos milisegundos comparado a los segundos que requiere Bluetooth. Esto beneficia el ahorro de energía y la sincronización en nodos móviles. Utiliza una pila de protocolos más simple y se enfoca en comunicaciones de corto alcance sobre una topología estrella.

ZigBee - IEEE 802.15.4

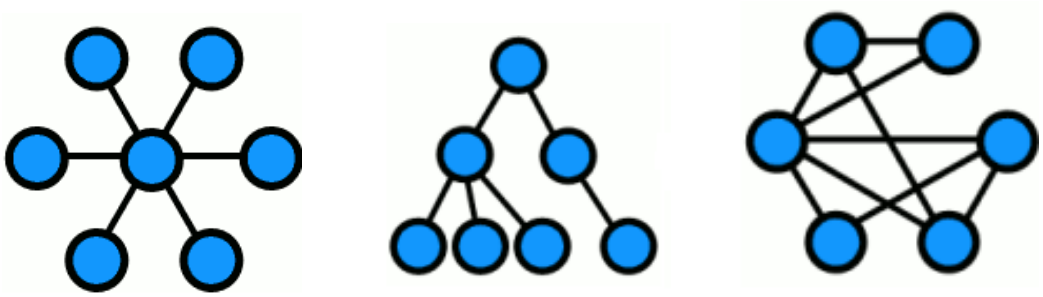
El estándar IEEE 802.15.4 define la capa física y la capa MAC. La característica más importante de este estándar es su flexibilidad de red, bajo coste y bajo consumo de energía. ZigBee es un conjunto de protocolos de alto nivel de comunicación inalámbrica que utiliza el estándar IEEE 802.15.4 para la capa física y la capa MAC. Esta es la tecnología más utilizada por los nodos de las WSN ya que provee bajo consumo de energía y bajo costo. Sin embargo, también provee una baja tasa de transmisión debido al bajo consumo de energía.
IEEE 802.15.4 define cuatro estructuras de tramas: tramas de datos, tramas de reconocimiento, tramas de comandos MAC y tramas *beacon*. Para la transferencia de datos existen tres tipos de transacciones en ZigBee: desde un coordinador a un dispositivo, de un dispositivo a un coordinador y entre dos dispositivos. La transferencia de datos está completamente controlada por los dispositivos más que por el coordinador. Un dispositivo trasfiere datos al coordinador o le pregunta al coordinador si desea enviar datos. Esta propiedad es la que permite la conservación de energía característica de ZigBee/IEEE 802.15.4, dado que un dispositivo puede dormirse siempre que le es posible sin mantener su receptor continuamente activo. ZigBee ofrece las topologías en estrella, árbol y malla (ver Anexo A.4). También provee enrutamiento multi-salto por lo que el cubrimiento de la red puede expandirse.
IEEE 802.15.4 permite el acceso al canal a través de CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*).
La capa física define tres tasas de transmisión dependiendo del medio inalámbrico y utilizando tres bandas de frecuencia distintas. Las tasas de transmisión son de 20, 40 y 250 kbit/s usando las bandas de frecuencia 868-868.8, 902-928 y 2400-2483.5 MHz. En estas bandas de frecuencia hay uno, diez y dieciséis canales respectivamente [83].
Una de las mayores desventajas de los sistemas IEEE 802.15.4 es la alta vulnerabilidad a interferencia con las WLAN IEEE 802.11 y los transmisores Bluetooth IEEE 802.15.1. En consecuencia, técnicas para mitigar las interferencias deben ser implementadas antes de desplegar una red de sensores inalámbricos. La tasa de transmisión máxima soportada es de 250 kbps lo que puede ser inadecuado para soportar WSN de gran escala o que soporten aplicaciones de tiempo real.

UWB - IEEE 802.15.6

UWB (*Ultra Wide Band*) se refiere a cualquier tecnología de radio con una transmisión mayor a 500 MHz. Esta tecnología es adecuada para comunicaciones de corto alcance, provee altas tasas de trasmisión de datos y es ideal para realizar localizaciones precisas.
UWB es difícil de detectar y es robusto frente a interferencias debido a su baja densidad espectral. Por lo tanto UWB ayuda a proteger la transmisión de datos sensibles reduciendo la probabilidad de detección. Por otro lado, las señales de UWB no causan una significativa interferencia con otros sistemas operando en la vecindad. El actual estándar IEEE 802.15.6 define tres capas físicas llamadas: *Narrowband* (NB), *Ultra Wide Band* (UWB), y *Human Body Communications* (HBC). Una descripción detallada de ellas se da en [84].

A.4 Topología de la red

Los nodos que forman la red se pueden interconectar de distintas formas, siendo la elección de esta forma un factor fundamental que determina el rendimiento y la funcionalidad de la red. La forma adecuada de organizar los componentes depende entre otras cosas de la cantidad de nodos, si estos son estáticos o dinámicos, etc.
Dentro de las WSN generalmente se utiliza alguna de las topologías que se presentan a continuación:



(a) Topología en estrella **(b)** Topología en árbol **(c)** Topología en malla

Figura A.3: Topologías de redes de sensores

Topología en estrella: Consiste de un coordinador y varios nodos que son sistemas terminales (ver figura A.3a). Estos nodos solo se pueden comunicar con el coordinador, es decir, si dos nodos se quieren comunicar entre sí, entonces deben pasar a través del coordinador. Esta es la topología que se utiliza por ejemplo, en las BAN (*Body Area Networks*).
Topología en árbol: consiste de un nodo central o raíz que realiza las tareas del coordinador, varios routers que extienden el cubrimiento de la red y sistemas terminales (ver figura A.3b). Los routers y el coordinador tienen hijos, mientras que los sistemas terminales son hojas que solo se comunican con sus padres. Por esta razón, si un padre queda deshabilitado, los hijos no se pueden comunicar con otros dispositivos en la red. Esta topología es la que se utiliza, por ejemplo, con el protocolo CTP (*Collection Tree Protocol*).
Topología en malla: consiste de un coordinador, routers, y sistemas terminales (ver figura A.3c). Este es un tipo de red multi-salto donde el área de cubrimiento puede ser incrementada agregando más dispositivos a la red. Todo nodo de la red puede comunicarse con cualquier otro nodo ya que los datos son enrutados a través de otros nodos hasta llegas al destino.

A.5 Análisis de Seguridad en TinyOS

La implementación actual de TinyOS no proporciona ningún control del acceso a memoria. En los sistemas operativos tradicionales el control de acceso se logra autenticando a los procesos para permitir o denegar el acceso a los diferentes recursos del sistema. Otro mecanismo de control de acceso es el método de protección en anillos el cual consiste de un número de niveles hardware de privilegio. Estos niveles se organizan en una jerarquía comenzando desde los procesos más confiables hasta los procesos menos confiables. El hardware que utiliza protección en anillos restringe cuidadosamente el modo en que un anillo le pasa el control a otro anillo. El hardware también impone restricciones en el tipo de acceso a memoria que se puede hacer en cada anillo. Para implementar una arquitectura en anillos el hardware y el software deben cooperar. Una posible solución a futuro es diseñar una plataforma hardware para nodos sensores que soporte una arquitectura en anillo.
Por otro lado, para proveer control de acceso a memoria, Regehr et al. [37] propone el concepto de dibujar una "línea roja", la cual se refiere a tener un límite entre código confiable y no confiable, utilizando un concepto similar a las técnicas de *sandboxing*. Su solución es conocida como *Untrusted TinyOS* (UTOS) y provee un ambiente en el cual el código no confiable y posiblemente malicioso corra sin afectar al *kernel*. UTOS crea el *sandbox* usando extensiones que son interfaces entre código no confiable y los componentes de TinyOS. UTOS asegura que una aplicación no pueda acceder directamente al hardware, no pueda utilizar los recursos de la red inapropiadamente y no pueda invalidar un nodo deshabilitándole las interrupciones. Regehr también propone otro entorno para aplicación en TinyOS llamado Safe TinyOS, el cual provee un *kernel* mínimo para poder ejecutar de forma segura aplicaciones confiables. Esta ejecución segura identifica y atrapa mediante manejadores de errores dinámicos los *bugs* que de otro modo corromperían silenciosamente la RAM, como por ejemplo, desreferenciar un puntero nulo, acceder a una posición fuera de los límites de un array, etc.
Por otro lado, Roosta et al. [36] señala una vulnerabilidad en las operaciones sobre los puertos en TinyOS ya que es posible abrir un puerto remotamente en un sensor utilizando un puerto USB, una PC y el componente *serial forwarder* de TinyOS el cual permite abrir un puerto en un nodo. Cuando un usuario intenta abrir un puerto no existe un chequeo para autenticarlo. Pudiendo realizarse un ataque de software donde el atacante abre un puerto para subir software al nodo o descargar información del mismo. Por otro lado, TinyOS omite explícitamente protección contra *replay* es su arquitectura. La razón principal es para poder evitar el requerimiento extra de memoria que se necesitaría ya que se tendrían que almacenar el número de secuencia para cada nodo con el que el receptor se esté comunicando. Otra razón es porque los desarrolladores creen que la protección contra *replay* es mejor y más eficiente si se maneja desde la aplicación ya que esta mantiene una vista de la topología completa y de los patrones de comunicación dentro de la red.

A.5.1 TinySec

Las primeras versiones de TinyOS no tenían implementados mecanismos de seguridad. Luego, en el 2004 se introduce TinySec que es la primera arquitectura segura de capa de enlace para WSN completamente implementada que se enfoca en garantizar autenticidad, integridad y confidencialidad de los mensajes. No se atacan problemas como ataques que consumen los recursos, resistencia a la manipulación física ni ataques de captura de nodos.
TinySec es un paquete que los desarrolladores pueden integrar fácilmente a las aplicaciones TinyOS. Desafortunadamente, TinySec solamente funciona en el simulador TOSSIM y en los motes Mica y Mica2.
TinySec provee las propiedades de seguridad básicas autenticación e integridad de los mensajes a través del uso de MAC, confidencialidad de los mensajes usando cifrado, seguridad semántica [1] utilizando

[1] Se llama seguridad semántica a la propiedad que cumplen los esquemas de cifrado que además de impedir recuperar un mensaje también evitan que un atacante obtenga información parcial del mensaje cifrado. La seguridad semántica implica que un adversario tenga menos del 50 % de probabilidad de responder correctamente cualquier pregunta del tipo "si o no" acerca de un mensaje cifrado.'

un vector de inicialización y finalmente, protección contra *replay*.

Modos de seguridad: TinySec soporta dos modos de seguridad diferentes: *authenticated encryption* (TinySec-AE) y *authentication only* (TinySec-Auth). Con *authenticated encryption*, TinySec cifra la carga útil del mensaje y autentica el paquete con un MAC, el cual se computa sobre los datos cifrados y el cabezal del paquete. En el modo *authentication only*, TinySec autentica el paquete complete con un código MAC pero la carga útil no está cifrada. El modo de TinySec utilizado se indica en el campo MSB de 2 bits de largo dentro de la cabecera del paquete.
Cifrado: Para cifrar los mensajes, TinySec utiliza un vector de inicialización (VI) de 8 bytes y la técnica de cifrado en bloques (*CBC- Cipher Block Chaining*).
Integridad de los mensajes: TinySec usa cifrado en bloques CBC-MAC para computar y verificar las MACs de modo de minimizar el número de primitivas criptográficas que deben ser implementadas en la limitada memoria disponible. Sin embargo, el CBC-MAC estándar no es seguro para mensajes de tamaño variable ya que el atacante puede forzar MACs para ciertos mensajes. La variante que utiliza TinySec utiliza un XOR entre el largo del mensaje cifrado y el primer bloque del texto plano [56]. El código MAC tiene un largo de 32 bits y la clave de autenticación de 64 bits.
Sin embargo, los ataques de canal lateral son posibles en WSN. Okeya et al [85] muestra dos ataques de canal lateral utilizando análisis simple de potencia y análisis de potencial diferencial sobre message authentication codes (MACs), pudiendo extraer varios bits de la clave. En conclusión, el cifrado en bloques es vulnerable a los ataques de canal lateral. TinySec usa un esquema de cifrado con cifrado en bloques por lo que presenta cierta debilidad ante este tipo de ataque [1].
Manejo de claves: TinySec utiliza distintos métodos para el manejo de claves. Uno de ellos es utilizar claves separadas para cada par de nodos que se comunican. Esto provee mejor resistencia a los ataques de captura de nodos ya que un nodo comprometido solo puede descifrar tráfico dirigido a él. Otro método es utilizar una clave por enlace entre nodos vecinos, esto tiene como desventaja que se necesita un protocolo de distribución de claves y se prohíbe la participación pasiva así como el broadcast local. Un enfoque menos restrictivo apunta a compartir claves TinySec entre grupos de nodos vecinos. Esto provee un nivel intermedio de resistencia a los ataques de captura de nodos, pudiendo descifrar e inyectar tráfico solo a nivel de grupo en caso de un ataque.

El formato de los paquetes utilizados por TinySec está basado en el formato de paquetes de TinyOS. Un dispositivo TinyOS está identificado por una dirección de 16 bits, la cual es aumentada por un identificador de grupo de 8 bits. El identificador de grupo es análogo a la dirección de red de un grupo de nodos que cooperan para realizar una tarea y permite que varios grupos de nodos distintos compartan el mismo canal de radio. TinyOS implementa el sistema de *Active Messag*e (AM). Los tipos AM ocupan un campo de 8 bits en el cabezal y son utilizados para especificar que manejador se debe utilizar para extraer e interpretar los paquetes recibidos del lado del receptor.

Tanto los paquetes TinyOS como los TinySec-Auth no tienen un campo de dirección origen por lo que el receptor no puede conocer de quien envió el mensaje. La dirección origen solamente se incluye en los paquetes TinySec-AE.
Para detector errors de transmission, TinyOS computa un código de redundancia cíclica (CRC- *Cycle Redundancy Check*) sobre el paquete del lado del origen de 16 bits. El destino luego vuelve a computar el CRC para verificar que este coincida con el CRC recibido. Si son iguales, el receptor acepta el paquete, de otro modo lo descarta. Sin embargo, los CRCs no proveen seguridad contra modificación maliciosa. Por lo que TinySec remplaza los CRC por MAC. Un código MAC protege el paquete completo incluyendo la dirección destino, el tipo AM, el largo, la dirección origen, el contador (si este existe) y los datos (cifrados o no). Esto protege los datos contra manipulación y los ataques de re dirigir un paquete que va a un nodo hacia otro distinto. El identificador de grupo que utiliza TinyOS tampoco es necesario en TinySec ya que este fuerza un control de acceso por medio de los códigos MAC, donde diferentes redes utilizan diferentes claves.

A.6 Contramedidas de los ataques

A.6.1 Manipulación - ataques invasivos

En [1] se identifican tres factores importantes que ayudan a los atacantes durante un ataque de captura de un nodo por lo que tenerlos en cuenta puede mejorar la resistencia contra este tipo de ataques:

- Si los nodos sensores dentro de la red comparten las mismas claves de cifrado con los nodos vecinos este tipo de ataque puede ser una gran amenaza. Siendo mayor la amenaza a la privacidad cuanto más se comparten las claves entre los nodos vecinos.

- La topología de la WSN afecta al impacto de los ataques de captura de nodo. En general, a menor cantidad de enlaces de comunicación entre los nodos sensores, mayor será la posibilidad de que un atacante pueda bloquear la comunicación entre un origen y un destino. Por ejemplo, ataques de este tipo son generalmente más eficaces en una topología de árbol que en una topología en malla, ya que en la primera topología sólo hay una ruta desde un hijo a su padre. Si el nodo primario se ve comprometido, toda la comunicación desde sus nodos hijos hacia abajo potencialmente se verá comprometida.

- La densidad de la WSN tiene una influencia directa en los ataques de captura de nodo. Por lo tanto, la cantidad óptima de nodos vecinos debe ser identificados para cada aplicación específica después de realizado un análisis.

Jokhio et al. [86] propone un protocolo para detectar y defenderse frente a ataques de captura de nodos, llamado SCADD (*Sensor node Capture Attack Detection and Defence protocol*). Este protocolo consta de dos bloques:
1. NAD (*Node Attack Detection*): Bloque de detección de ataques.
2. DAM (*Defense Advocating Measure*): Bloque de defensa.
El bloque NAD identifica el estado o severidad del ataque realizado a un sensor y clasificándolo en distintas categorías. Mientras que el bloque DAM involucra tomar medidas defensivas contra el ataque utilizando un protocolo inteligente de autodestrucción borrando información vital de la memoria del nodo. Cada nodo tiene en memoria una tabla VMAT (*Vital Memory Address Table*) que contiene la ubicación de la información vital que deberá ser borrada en caso de un ataque de captura de nodo, como por ejemplo: claves criptográficas, localización de los nodos vecinos, información de ruteo, identificador del nodo, etc. El algoritmo de autodestrucción utilizado por DAM es iniciado por un procedimiento que lee la tabla VMAT y sobre escribe con ceros todas las posiciones de memoria especificadas en la lista. Luego de identificar la severidad del ataque se envía una alerta de forma *broadcast*. Además, en cualquier momento, si el resto de los nodos no pueden comunicarse con ese sensor envían una alerta *broadcast* indicando un posible nodo comprometido. Los nodos que reciben esa alerta marcan al nodo en cuestión como sospechoso y realizan chequeos de seguridad cuando este nodo intenta re establecer la comunicación con la red.
Con este mecanismo de autodestrucción los servicios de radio y la capacidad de tomar datos del nodo no son dañados en caso de un falso positivo por lo que con una restauración de los valores apropiados en la memoria el nodo puede quedar operable nuevamente.
Zang et al. [87] propone un método para detectar la destrucción física de los nodos, para ello la estación base debe monitorizar continuamente cambios en el cubrimiento de la red. Donde el cubrimiento de la red incluye todas las posiciones dentro del rango donde al menos un nodo conectado a la estación base puede capturar datos. Proponen un protocolo seguro de inferencia del cubrimiento (SCIP) el cual le provee a la estación base una medida del cubrimiento de la red. También proponen un protocolo de inferencia del cubrimiento básico (CIP) el cual solo necesita la información de los vecinos a un salto de distancia y se integra con un protocolo de administración de claves simétricas basadas en la ubicación (LBSK).

A.6.2 Manipulación - ataques no invasivos

De acuerdo a [62] algunas medidas contra este tipo de ataque utilizadas en los sistemas tradicionales y embebidos son:

- Consumo de energía aleatorio
- Ejecución aleatoria del conjunto de instrucciones
- Uso aleatorio de los registros de memoria
- Reloj de la CPU aleatorio
- Uso de instrucciones falsas

A.6.3 Interferencias

Manju y Kumar en [88] proponen un método llamado identificación de interferencias en la capa física. Este método se basa en marcar nodos como monitores en base a la energía residual (RE, *Residual Energy*). Estos nodos monitorizan si está ocurriendo un ataque de interferencia chequeando el indicador de la intensidad de la señal del receptor (RSSI, *Receiver Signal Strength Indicator*) y la tasa de envío de paquetes (PDR, *Packet Delivery Ratio*) de los otros nodos. En este método los nodos calculan su RSSI, PDR y RE. Los nodos con mayor RE son seleccionados como los nodos monitores. Estos nodos serán los encargados de monitorizar si existe un ataque de interferencia recolectando periódicamente los valores de RSSI y PDR de sus vecinos para computar una ponderación para cada nodo. Esta ponderación es comparada con cierto valor de umbral. Cuando el valor ponderado es menor que el umbral, el nodo correspondiente es marcado como el nodo malicioso que esta originando las interferencias y es aislado de la transmisión de datos. Este mecanismo funciona en base a que durante la transmisión de datos, un atacante que hace uso de las interferencias corrompe sus paquetes de datos y por lo tanto se reduce su PDR. Esta técnica hace uso del PDR como una métrica para identificar el atacante. Además otra característica de los ataques de interferencia es que degradan la calidad del canal que rodea a ese nodo interrumpiendo las señales de radio. Es por esta razón, que también se considera el RSSI como una métrica.
Por otro lado, Reindl et al. [89] también propone un esquema distribuido basado en el análisis del RSSI que puede identificar el atacante que produce las colisiones. Modelando la red como un conjunto heterogéneo de sensores, donde existe un número pequeño de sensores más potentes y un gran número de sensores con menos capacidad en cuanto a comunicación, cómputo, energía, almacenamiento y otros aspectos. Además muestran que el adversario puede ser correctamente identificado con una precisión mayor a 85 %.
Encontrar la ubicación del dispositivo que está realizando este tipo de ataque es muy importante para tomar acciones de seguridad y restaurar las comunicaciones dentro de la red. En [90] se propone un algoritmo para realizar esta tarea llamado *Double Circle Localization* (DCL).
Cagalj et al. [91] muestra que los *Wormholes* (ver ataques de capa de red) pueden ser utilizados como un mecanismo de defensa reactivo frente a ataques de este tipo. En su solución, se hace uso de la diversidad de canales para crear una ruta de comunicación que permita a la información recolectada en la zona de interferencia llegar a destino. La creación del *wormhole* es disparada debido a la ausencia de reconocimientos (ACK) luego de varias transmisiones.
Jain y Garg et al. [92] proponen un modelo híbrido de defensa que intenta mitigar el hecho de que la estación base es un único punto de fallo de la red. Este es la combinación de tres técnicas. La primera se llama replicación de la estación base, en la cual ante un ataque de interferencia existen otras estaciones base para proveer servicios a la red. La segunda se llama evasión de la estación base desde una ubicación bajo interferencia a otra ubicación libre de interferencias. Finalmente, la última técnica se llama enrutamiento *multipath* para mitigar los efectos de los ataques de interferencia a la estación base.

Sanjay et al. [93] propone una estrategia de detección usando *«HoneyNodes»* y un mecanismo de respuesta basado en el algoritmo existente de *Channel Surfing* para proteger a los nodos de las interferencias. Los HoneyNodes generan comunicaciones falsas a una frecuencia muy similar a la frecuencia de operación actual por lo que los nodos verdaderos pueden cambiarse a otra frecuencia incluso antes de que el atacante comience a escanear esa frecuencia. En el algoritmo de *Channel Surfing* cuando un nodo detecta un ataque de interferencia se cambia a otra frecuencia de operación siguiendo una secuencia predefinida pseudo- aleatoria. La estación base envía mensajes a una tasa regular a todos sus nodos asociados para chequear si estos están presentes en la nueva frecuencia. Si algún nodo no le responde a la estación base esta envía un comando de cambio de frecuencia a todos los nodos para cambiar a la próxima frecuencia definida en la secuencia. Del mismo modo, si un nodo no recibe señales de la estación base entonces también se cambia a la próxima frecuencia definida en la secuencia pseudo-aleatoria.
En [94] se presenta un poderoso ataque de este tipo llamado *mobile jamming* donde el atacante puede moverse por las distintas áreas críticas de la red. Para ello primero debe descubrir un camino crítico de la red realizando escuchas del tráfico o la dirección del flujo de datos. De este modo, si la carga de tráfico no alcanza cierto umbral, el *mobile jammer* se mueve al próximo enlace siguiendo el flujo de datos o volviendo a escuchar el tráfico de nuevo. Cuando la carga del tráfico alcanza el umbral, el atacante comienza a interferir en la red. En general esto deja inutilizable un camino crítico de la red afectando a los nodos que están dentro del área de interferencia y todos los nodos hijos. Ante este ataque la red puede tomar medidas de defensa activa o pasiva. Las medidas activas implican determinar el área que sufre la interferencia y enviar el flujo de datos por un camino que evite el área afectada. Sin embargo, el atacante puede detectar que la carga de tráfico desciende rápidamente. Parando la interferencia y volviendo a escuchar nuevamente puede descubrir un nuevo camino crítico en poco tiempo. De este modo el atacante puede cambiar de camino crítico constantemente haciendo que los nodos estén ocupados intentando re enrutar por otro camino. Las medidas defensivas pasivas intentar preservar las reservas energéticas y sobrevivir mientras el ataque no termina. Este mecanismo tiene como inconveniente que si los nodos afectados (incluyendo los hijos de los que están en el área de interferencia) pasan a modo de reposo o a estar apagados los eventos de importancia no serán reportados. Por ejemplo, un intruso con acceso físico a los nodos podría atacar el lugar que está siendo monitorizados sin ser detectado.

A.6.4 Ataque Blackhole

En [95] se propone un protocolo de enrutamiento jerárquico seguro, llamado HSRBH, el cual sirve para detectar y defenderse frente a ataques de tipo *blackhole*. Este protocolo solo utiliza claves criptográficas simétricas y divide la red en distintos grupos, donde cada grupo tiene su líder y se organiza en una estructura de árbol siendo el líder la raíz del mismo. Se tiene una clave interna al grupo compartida por los nodos vecinos al líder y una clave entre grupos que es manejada por los líderes de dos grupos vecinos.
De acuerdo a la arquitectura de red que ellos proponen pueden ocurrir cuatro tipos de ataque *blackhole*: un solo nodo distinto del líder realiza el ataque, varios nodos entre dos líderes de grupos vecinos realizan el ataque, un líder de grupo realiza el ataque o el líder de grupo y sus vecinos realizan el ataque. Los primeros tres tipos de ataques pueden ser detectados localmente.

Deng et al. [96] propone AODV, que es un protocolo de enrutamiento que crea rutas solamente cuando es pedido por un nodo emisor, es decir, cuando un nodo solicita una ruta hacia un destino se inicia un proceso de descubrimiento de rutas dentro de la red. Para ello, el emisor envía en *broadcast* una solicitud a sus vecinos, que a su vez lo vuelven a enviar a sus vecinos y así sucesivamente. El proceso finaliza cuando esta solicitud llega al nodo destino o a un nodo intermedio con una ruta "fresca" hasta el destino. Luego de seleccionar y establecer la ruta, comienza un procedimiento de mantenimiento de ruta. La forma de detectar un ataque de *blackhole* es confirmando que cuando un nodo intermedio dice conocer una ruta hacia un destino esta exista verdaderamente. Si esta ruta existe podemos confiar en ese nodo y enviarle los paquetes. Si no existe se descarta ese mensaje de respuesta recibido y se lanza

una alarma para aislar ese nodo de la red.

A.6.5 Ataque Selective Forwarding

Varios mecanismos de detección y defensa han sido propuestos en los últimos tiempos para este ataque. Yu y Xiao [97] proponen un esquema de detección de ataques de reenvío selectivo utilizando una técnica de reconocimientos *multi-hop* para enviar alarmas. Cada nodo en el camino de reenvío puede detectar un nodo malicioso, en cuyo caso enviará un paquete de alarma al nodo o a la estación base dependiendo de quién sea el emisor del paquete.
Brown y Du et al. [98] proponen un esquema de detección de ataques de reenvío selectivo en redes de sensores heterogéneas donde distintos nodos tienen diferentes capacidades.
Xing-Sheng et al. [99] propone un esquema de defensa contra reenvío selectivo donde los nodos vecinos actúan como nodos monitores. El ataque es detectado monitorizando el reenvío de paquetes por dos nodos vecinos en el camino de transmisión y reenviando los paquetes que fueron descartados por el atacante.

Kaplantzis et al. [100] proponen un esquema de detección de intrusos centralizado, el cual se basa en clasificar patrones de datos utilizando un clasificador SVM (*Support Vector Machines*), que es una clase de algoritmo de aprendizaje automático que permite detectar ataques de tipo *blackhole* y de reenvío selectivo. El sistema de detección de intrusos usa información de enrutamiento local a la estación base y dispara alarmas basado en un vector de dos dimensiones que contiene información del ancho de banda y de la cantidad de saltos.
En [101] proponen un algoritmo de detección de ataques de reenvío selectivo el cual se basa en información del vecindario. Cada nodo tiene un módulo de detección en la capa de aplicación y utiliza información de los vecinos a dos *hops* de distancia (cada nodo tiene una tabla que contiene estos vecinos). El mecanismo de detección se basa en las siguientes dos reglas. Primero, el nodo monitoriza que el siguiente nodo reenvíe el paquete según el camino hasta la estación base. Si no lo hace se lanza una alerta. Segundo, el nodo espera y detecta el paquete que ha sido reenviado. Chequea su vecino a dos *hops* de distancia para comprobar que este también reenvíe el paquete por el camino correcto.

En [102] proponen otro mecanismo de defensa contra este tipo de ataques que utiliza varios flujos de datos. La red es dividida en diferentes topologías, lo que hace que un nodo que pertenece a cierta topología solo pueda comunicarse y enviar información a través de los nodos en la misma topología. La condición que deben cumplir estas topologías es que deben cubrir completamente el área de monitorización donde la red está desplegada. Así, cuando un evento ocurre, nodos en ambas topologías captan el evento y se lo envían a la estación base. De forma que si un nodo en una topología descarta el paquete este igual llega a la estación base a través de otra topología.

A.6.6 Ataque sinkhole

En [103] presentan un método para detectar ataque de *sinkhole* basados en el RSSI (*Received Signal Strength Indicatior*) y en la ayuda de algunos nodos monitores que cuentan con una gran antena receptora. Al desplegarse la red inicialmente se crea un mapa geográfico (VGM, *Visual Geographic Map*) de la red utilizando los valores de RSSI de los nodos monitores. Cuando un nodo en la red envía un mensaje, los nodos monitores recibirán el mensaje y el valor de RSSI debido a su poderosa antena receptora. Si el destino del mensaje es la estación base, todos los nodos monitores le envían el RSSI al detector de *sinkhole* basado en RSSI que se encuentra en la estación base para localizar la posición del emisor. Si el flujo de mensajes recibidos no corresponde con un flujo normal de acuerdo al mapa VGM entonces se detectó un ataque *sinkhole*.
Sheela et al. [104] propone un esquema para defenderse frente a ataques *sinkhole* utilizando agentes móviles. Estos agentes móviles van de nodo en nodo transmitiendo datos, realizando cálculos y con-

trolando la red. Los nodos de la red reciben datos de los agentes, sin tener que recibir información de nodos maliciosos o comprometidos. Este mecanismo no necesita que los mensajes estén cifrados. También se proponen dos algoritmos, uno indica como los agentes deben dar información de la red a los nodos y como deben recorrerlos. El otro algoritmo le indica a un nodo como usar la información global de la red dada por un agente para enrutar los paquetes de datos.
Otras propuestas para detectar ataques de este tipo se encuentran en [105] [106] [107].

A.6.7 Ataque wormhole

Para detectar los ataques de tipo *wormhole* existen varios enfoques. Por ejemplo, algunos métodos hacen uso de que una mayor distancia de propagación es síntoma de *wormhole*. También se pueden chequear ciertos parámetros como el rango de transmisión, o el nodo previo dentro de una ruta y compararlo respecto de la condición normal. Algunas de las clasificaciones presentadas para estos distintos enfoques se pueden encontrar en [108] [69] y son las que se mencionan a continuación.
Packet leashes: Hu et al. [109] [110] propone una técnica para detectar *wormholes* en WSN móviles llamada "*packet leashes*". La idea básica consiste en agregar restricciones de seguridad (como por ejemplo el tiempo o la ubicación) en cada paquete. Esta información luego es utilizada para calcular si el paquete a viajado una distancia mayor a la físicamente posible. Dentro de esta técnica hay dos enfoques: "*Geographical packet leashes*" y "*Temporal packet leashes*". El primero requiere que los nodos conozcan su ubicación (por ejemplo accediendo a información de GPS) y que mantengan correctamente sincronizados sus relojes. Cuando un nodo envía un paquete este concatena al cabezal el tiempo y el lugar de transmisión. Cuando el paquete es recibido, el destino utiliza su propia ubicación y el tiempo de recepción para calcular la máxima distancia posible entre los nodos, tomando en consideración distintos factores de error que pueden ocurrir en el camino. El segundo enfoque no requiere que los nodos conozcan su ubicación pero si es necesario que los relojes estén perfectamente sincronizados (del orden de nanosegundos). Cuando un emisor envía un paquete este agrega un *time stamp* autenticado. El nodo receptor calcula con el *time stamp* el tiempo de recepción. Luego puede calcular la distancia que viajó el paquete basado en el tiempo de transmisión. Si la distancia estimada es muy grande, estamos ante la presencia de un *wormhole*. Esto se basa en que si un paquete viaja a través de un *wormhole*, este va a seguir una ruta más larga que la real entre dos nodos vecinos.
Enfoque basado en la teoría de grafos
En [111] proponen un método distribuido para detectar *wormholes* basados en que un nodo receptor pueda determinar si los datos que le llegan provienen o no de un nodo que es vecino. El protocolo propuesto cifra los mensajes utilizando claves compartidas entre cada par de nodos vecinos. Adicionalmente también envían información como el identificador del emisor, un número aleatorio utilizado como *nonce* y un *message digest* computado usando algoritmos de *hash* como por ejemplo MD5 y SHA1.
Enfoque basado en la distancia o en el tiempo
Este tipo de soluciones implica basarse en el tiempo necesario para enviar paquetes individuales. Algunas soluciones utilizan el *Round Trip Time* (RTT) de los paquetes, otros el RSSI o la diferencia de tiempos de arribo. Kormaz [112] propone un protocolo que utiliza el RTT de un mensaje y su ACK. No requiere información de ubicación de los nodos o sincronización de reloj ya que la distancia entre dos nodos participantes se calcula con el RTT. Luego se determina si la distancia estimada está o no dentro de un máximo rango posible de comunicación.
Enfoque estadístico
Dentro de este enfoque se utilizan métodos estadísticos donde mediante cambios en ciertos patrones estadísticos de puede decidir si existe o no un *wormhole* en la red. Buch et al. [113] proponen un método utilizando este enfoque que principalmente se enfoca en el número de paquetes enviados y recibidos de cada nodo en la red. Las estadísticas del número de paquetes enviados y recibidos son mantenidas por cada nodo en dos tablas. Cada tabla tiene una entrada para cada vecino a un salto de distancia donde se tiene un contador inicializado en cero y que luego es incrementado. Idealmente, el total de paquetes enviados a un nodo X debería ser igual al número total de paquetes recibidos por

sus vecinos a un salto de distancia. Cuando esta ecuación no se satisface significa que algún paquete fue descartado o reenviado a través de un canal fuera de banda a algún nodo remoto. Ese nodo X, probablemente sea uno de los extremos del *wormhole*.

A.6.8 Ataque hello flood

Para este ataque Hamid et al. [114] propone un protocolo probabilístico para compartir claves entre dos nodos, de modo que estos puedan autenticarse entre sí. También propone un algoritmo de ruteo multi-camino que tiene como objetivo llegar a las múltiples estaciones bases para defenderse de los ataques de *hello flood*.

A.6.9 Ataque de replicación de nodos

Znaidi et al. [115] propone un algoritmo distribuido y jerárquico para detectar ataques de replicación de nodos utilizando un mecanismo de filtrado y un protocolo de selección de la cabeza de cada cluster. Cada cabeza de cluster intercambia los identificadores de sus nodos miembros a través del filtro con las otras cabezas de cluster para detectar los eventuales nodos replicados. También proponen un mecanismo de reacción frente a la detección de un ataque de este tipo.
Xing et al. [116] propone un esquema para detectar nodos replicados en WSN estáticas y homogéneas. Para ello utiliza una huella digital basada en información «social» extraída de las características del vecindario donde se encuentra el nodo. La legitimidad del originador de un mensaje puede ser chequeada utilizando esta huella. La idea subyacente a este método tiene en cuenta que una vez que los nodos son desplegados estos residen dentro de un vecindario fijo. De este modo, un sensor y sus vecinos forman una especie de "comunidad" o "red social". Cada sensor puede ser caracterizado por esta comunidad y un nodo replicado puede tener las credenciales legítimas (identificadores, claves, etc) del nodo original pero no los mismos vecinos. Un nodo replicado puede ser fácilmente identificador por sus vecinos si tiene una "firma social" que pertenece a otra comunidad. El mecanismo anterior (y muchos más que se han propuesto [117] [118] [119]) son útiles solo para redes estáticas ya que se basan en identificar la posición de los nodos e intentan detectar reportes que indiquen un nodo en múltiples ubicaciones. Sin embargo, Ho et al. [71] propone un esquema de detección de nodos replicados para WSN móviles. Se basa en el principio de que un nodo legítimo no debería moverse a una velocidad mayor a cierto umbral predefinido. Mientras que un nodo replicado que está en dos o más lugares al mismo tiempo parece moverse a una velocidad mucho mayor que la de un nodo legítimo por lo que tenderá a moverse a una velocidad mayor que la del umbral. Yu et al. [120] también propone un método para WSN móviles motivado por las siguientes observaciones. En una red sin nodos replicados, la cantidad de veces que un nodo X encuentra a un nodo específico Y está limitada dentro de cierto intervalo de tiempo con alta probabilidad. En una red con dos réplicas del nodo Y, la cantidad de veces que X encuentra réplicas de Y debería ser mayor que cierto umbral dentro de un intervalo de tiempo dado.

A.6.10 Ataque Sybil

Este ataque puede afectar a distintos protocolos tales como los que se presentan a continuación [72].

- Protocolos de enrutamiento: Este tipo de ataque representa una amenaza significativa para los protocolos de enrutamiento geográfico, el cual depende de la ubicación física y requiere que los nodos intercambien información de coordinación con sus vecinos para rutear eficientemente paquetes direccionados geográficamente [56]. Los mecanismos de ruteo multi-camino también son vulnerables ya que caminos aparentemente disjuntos puede pasar por un único nodo malicioso que presenta distintas identidades. Por otro lado, los ataques *sybil* pueden ser utilizados para evadir los mecanismos de detección de ataques de ruteo como por ejemplo los *blackholes*.

- Agregación de datos: La agregación de datos se utiliza para reducir el tráfico dentro de la red y para retornar información más elaborada que los simples datos obtenidos individualmente por los sensores. Un número pequeño de nodos maliciosos reportando datos erróneos pueden afectar de forma significativa la agregación de los datos. Con un ataque *sybil*, un solo nodo puede contribuir varias veces con datos falsos. Por lo que con suficientes nodos *sybil* un atacante puede alterar completamente las agregaciones.

- Votaciones: Las WSN utilizan votaciones para ciertas tareas como por ejemplo para detectar anomalías, mal uso del sistema y nodos que se esten comportando de forma inadecuada. Con un ataque *sybil* se pueden alterar estas votaciones. Por ejemplo, el atacante podría acusar a un nodo legítimo de incorrecto funcionamiento o cuando un nodo malicioso es acusado de mal comportamiento, poder influenciar la votación para que se sentencie como un nodo legítimo.

- Protocolos de asignación de recursos: algunos recursos de la red son asignados limitando un máximo por nodo, por ejemplo nodos compartiendo un canal de radio pueden tener asignada un intervalo de tiempo para transmitir. Con un ataque *sybil* un nodo puede obtener mayor cantidad de recursos. En consecuencia, nodos legítimos obtienen una menor cantidad de recursos mientras que un atacante tiene más recursos para poder realizar sus ataques.

Algunas soluciones propuestas para los ataques *sybil* son [1]:

1. Testeo de los recursos de radio, lo cual asume que cada nodo físico tiene un solo radio.

2. Pre-distribución de claves aleatoria, la cual intenta asociar la identidad de cada nodo con la clave asignada y valida la clave para confirmar que el nodo es realmente quien dice ser.

3. Registrar las identidades de los nodos en una estación base central.

4. Verificar la posición del nodo, esto es posible siempre que la topología de la WSN sea estática.

En general, técnicas de autenticación y cifrado pueden prevenir los ataques *sybil* externos. Sin embargo, no se puede prevenir que un atacante dentro de la red participe en ella pero ese nodo comprometido solo debería poder utilizar su identidad legítima. Utilizar claves compartidas globalmente permite a un atacante interno enmascararse como cualquier nodo, incluso uno no existente en la red. Utilizar criptografía de clave pública podría prevenir este ataque interno pero es muy costosa para ser usada con los recursos de este tipo de redes. Una solución es que cada nodo tenga una clave criptográfica simétrica única con una estación base confiable. Por otro lado, dos nodos podrían usar un protocolo como *Needham-Schroeder* para verificar la identidad del otro y para establecer la clave compartida [56].
Muchas de las técnicas propuestas para detectar este tipo de ataques se basan en tratar de localizar donde se encuentran los nodos [121] [122] [123]. En [121] se propone un mecanismo para detectar ataques de tipo *sybil* en redes de sensores estáticas. Utiliza el valor de RSSI para inferir la distancia entre dos identidades ya que el valor de RSSI de paquetes que provienen de identidades *sybil* es muy similar. Este mecanismo clasifica las identidades con valor RSSI similar como un grupo que corresponde a una sola identidad, es decir, un ataque *sybil* es detectado cuando dos o más identidades diferentes están casi en el mismo lugar. Este mecanismo tiene más precisión cuando múltiples nodos cooperan para establecer la posición de las identidades.
Chen et al. [124] también presenta un mecanismo de detección de ataques *sybil* utilizando el indicador RSSI, dividiendo la red en clusters y que solamente se dispara cuando el número de cabezas de los clusters sobrepasa cierto umbral.

A.7 Formato de las tramas GTS

Los nodos pueden solicitar la asignación y la desasignación de las ranuras de tiempo dedicadas mediante las tramas *GTS allocation request* y *GTS deallocation request*. La estructura de estas tramas se puede observar en la figura A.4.

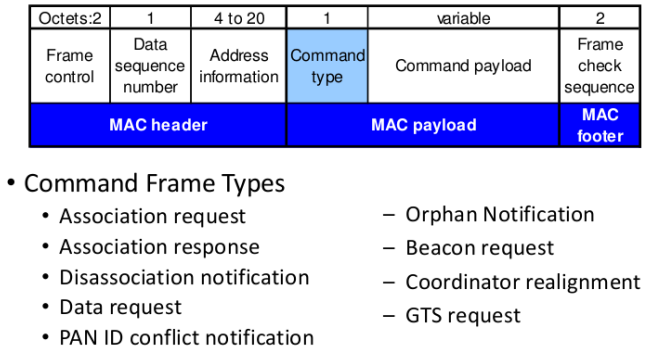


Figura A.4: Formato de los comandos MAC. Los GTS allocation/deallocation request se envían según esta estructura, indicando en el command type que se trata de un GTS request y en el campo command payload se deberá indicar las GTS characteristics (detalladas en la figura A.5).

bits: 0—3	4	5	6—7
GTS length	GTS direction	Characteristics type	Reserved

Figura A.5: GTS characteristics: cantidad de ranuras solicitadas (length), tx/rx (direction), allocation/deallocation (type).

En la Figura A.6 se presenta la estructura de un *beacon*.

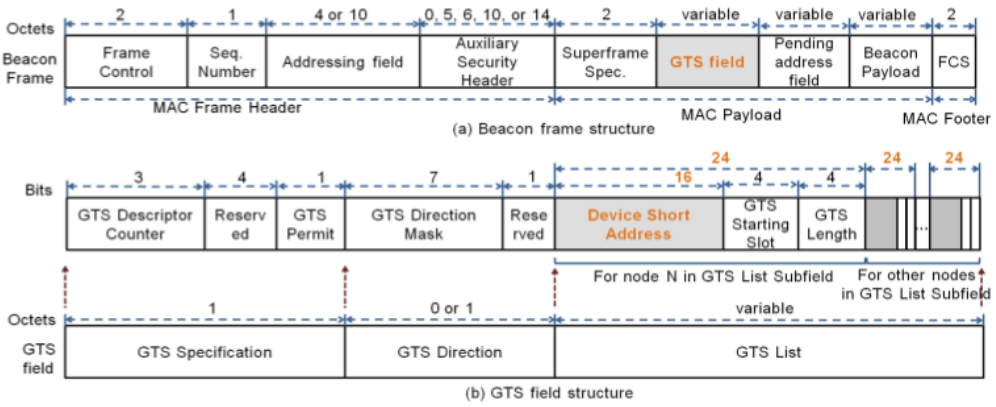


Figura A.6: Estructura del Beacon y del campo GTS.

A.8 Verificación del funcionamiento y análisis de tramas GTS

El objetivo de esta parte es comprobar que la estructura de los paquetes intercambiados coincide con lo mencionado en la sección A.7 e identificar los distintos campos de los paquetes intercambiados entre los nodos.

El protocolo GTS trae algunas aplicaciones de prueba para poder utilizarlo. Para realizar las pruebas se utilizaron estas aplicaciones por defecto. Se instaló la aplicación *coordinator* en un nodo y en otro la aplicación *device* de la aplicación TestGTS[1].

Para realizar el análisis se realiza una captura de dicho tráfico utilizando el sniffer AVR de Atmel[2] conjuntamente con Wireshark. Para configurar el sniffer se siguen los pasos provistos en [125] y además se debe configurar el canal pues la aplicación está definida para que por defecto se comuniquen usando el canal 16.

En las capturas realizadas se observó que el coordinador comienza enviando *beacons* de forma periódica. Estos tienen una estructura como la que se muestra en la figura A.7. Los campos tienen los valores esperados de acuerdo a los archivos «.h» de configuración.
Dentro del campo Super *frame specifications*, se envían los siguientes datos (ordenados según aparecen en el *beacon*): *beacon order*(4 bits), *superframe order*(4 bits), *Final CAP slot* (4), *Battery life extension* (1), *Reserved* (1), *PAN coordinator*(1) y *Association Permit* (1).

Beacon

Nro. Seq	PAN id	Origen	Superframe specifications	GTS Specifications		FCS
94	34 12	00 00	66 4F	80	00	77 CD

Figura A.7: Estructura de los beacons GTS.

El nodo una vez que se sincroniza solicita una ranura de tiempo dedicada, para ello manda un *GTS request allocation*. La estructura de lo capturado con Wireshark se observa en la figura A.8. El campo *Characteristic type* en uno indica que se realiza una petición de *allocation*, mientras el campo *direction* en uno indica que las ranuras solicitadas son para recepción.

Una vez que le llega el *GTS request* al coordinador, éste envía un ACK (la estructura no se detalla porque es muy simple y no brinda información relevante). Luego envía un nuevo *beacon* indicando que le fue asignado una ranura al nodo con identificador 0x0002, de largo uno (1) y que comienza a partir de la ranura número 15 (F). Esto se puede ver en la figura A.9.
Para la implementación, el nodo malicioso deberá ser capaz de identificar, a partir de un *beacon*, cuáles nodos tienen ranuras asignadas. Debe identificar además, cuales y cuántas son esas ranuras, que largo tienen y si son de recepción o de transmisión. Con esta información debe enviar pedidos de *GTS deallocations*. Por ejemplo, si se quisiera mandar un trama de *deallocation* para la solicitud de *allocation* de la figura A.8 se debería enviar la trama de la figura A.10. Esta es análoga a la de *allocation* pero cambia el bit **type**.

[1] El simulador Cooja no permite visualizar correctamente las tramas intercambiadas al utilizar el modo GTS.
[2] No se pudieron sniffiar paquetes ni con el BaseStation ni con el sniffer Z-Monitor, es por ello que se tuvo que recurrir al AVR.

GTS Request Allocation

Nro. Seq	PAN id	Origen	GTS request (Command type)	GTS characteristics (Command payload)	FCS
58	34 12	02 00	09	31	2F FD

bits: 0–3	4	5	6–7
GTS length	GTS direction	Characteristics type	Reserved
1000	1	1	00
	Recepcion	Allocation	

Figura A.8: Estructura de los GTS allocation request.

Beacon

						GTS descriptor			
Nro. Seq	PAN id	Origen	Superframe specifications	GTS Specifications	GTS directions	Id nodo	largo, comienzo		FCS
97	34 12	00 00	66 4E	81	01	02 00	1F	00	E3 49
					Recepcion en el primer GTS descriptor				

Figura A.9: Estructura del beacon GTS.

GTS Request Dellocation

Nro. Seq	PAN id	Origen	GTS request (Command type)	GTS characteristics (Command payload)	FCS
B2	34 12	02 00	09	11	47 41

bits: 0–3	4	5	6–7
GTS length	GTS direction	Characteristics type	Reserved
1000	1	0	00
	Recepción	Deallocation	

Figura A.10: Estructura de los GTS deallocation request.

A.9 Formato de los paquetes CTP

La figura A.11 muestra las estructuras de las tramas que contienen los paquetes de datos y enrutamiento CTP, mas los datos agregados por la capa física y MAC (para un transmisor de radio siguiendo el estándar IEEE 802.15.4). Los reconocimientos son manejados a nivel de capa de enlace.

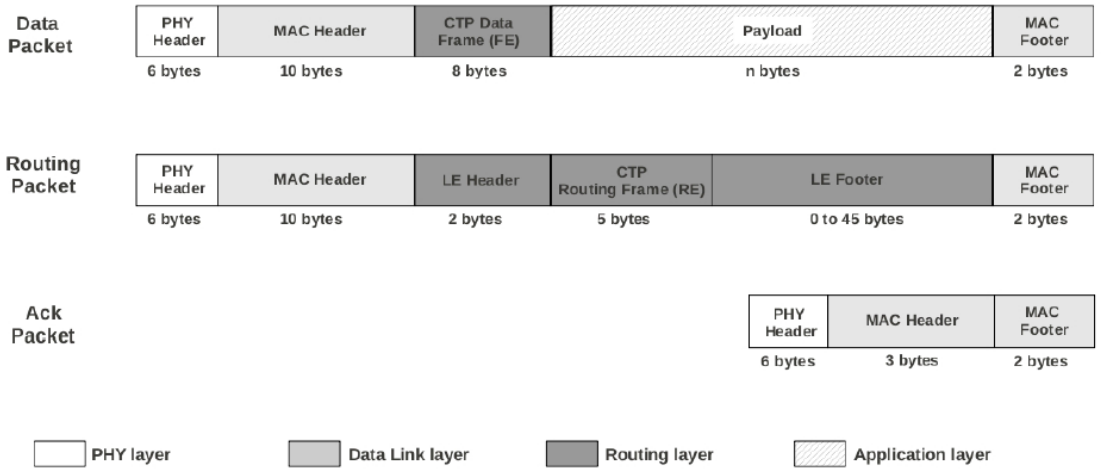


Figura A.11: Estructura de las tramas de reconocimientos, datos e información de enrutamiento del protocolo CTP[16].

La figura A.12 muestra en mayor detalle la estructura de los paquetes CTP de datos, CTP de enrutamiento y LEEP.

A.9.0.1 Paquetes de datos CTP

Los campos que contiene un paquete de datos CTP son los siguientes (ver figura A.12):

- El bit P (*Pull*) le permite a un nodo solicitar información de ruteo desde otros nodos. Si un nodo con una ruta válida escucha un paquete con el bit P prendido este debería transmitir un *beacon* en el futuro cercano.
- El bit C (*Congestion*) es una notificación de congestión. Si un nodo descarta un paquete de datos CTP, este debe prender el bit C de el próximo paquete de datos que transmita.
- El campo THL (*Time Has Lived*) es inicializado en cero cuando el paquete es creado. Cada vez que un nodo recibe ese paquete de datos CTP incrementa en uno el valor del campo THL. Si un nodo recibe un paquete con THL igual a 255, este resetea el valor a cero.
- Cuando un nodo trasmite un paquete de datos CTP incluye el valor ETX de su ruta a través del próximo salto elegido, aquí se envía el $ETX_{multi-HOP}$ hasta la raíz del arbol. Si un nodo recibe una trama con ETX menor al suyo debe enviar una trama de ruteo en el futuro cercano.
- El campo origen contiene la dirección del nodo que originó la trama. Un nodo que reenvía la trama no debe modificar este campo.
- El campo *seqno* contiene el número de secuencia que el nodo origen le asignó a la trama. Un nodo que reenvía la trama no debe modificar este campo.

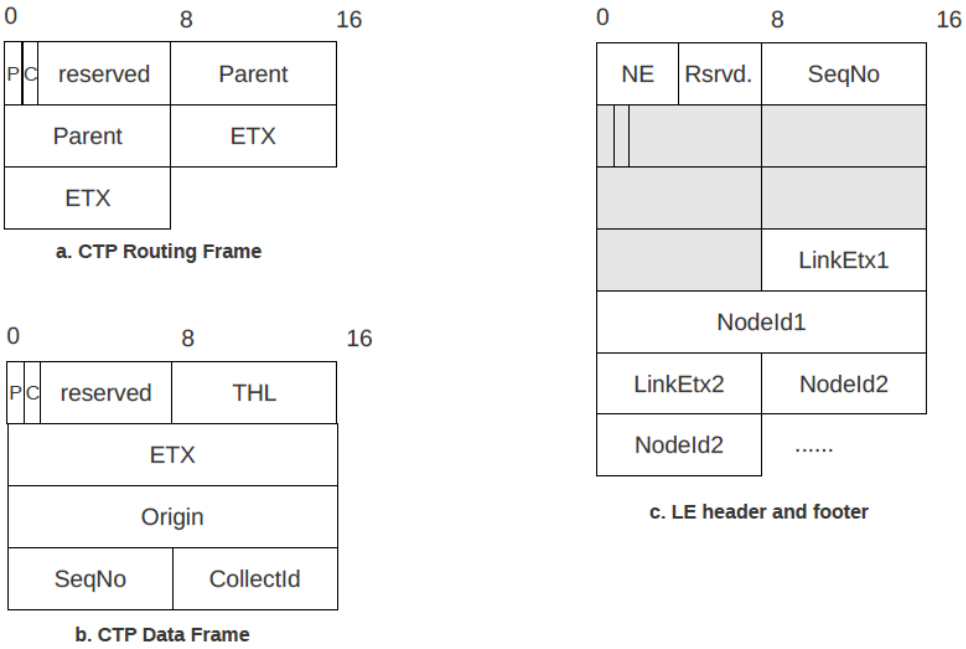


Figura A.12: Estructura de los paquetes: CTP de datos, CTP de enrutamiento y LEEP (encabezado y pie)[16].

- El campo *collect_id* es el identificador del protocolo que se encuentra en el nivel superior. Este valor es asignado por el nodo origen y no debe ser modificado en un reenvío.

- Finalmente, viene el campo que contiene los datos, que puede ser de cero o más bytes.

Un "paquete de origen" queda identificado por los campos: origen, *seqno* y *collect_id*. Mientras que una "instancia de paquete" queda identificado por los campos origen, *seqno*, *collect_id* y THL. Esta distinción es importante para identificar la presencia de bucles de ruteo. Si un nodo recibe la misma instancia de paquete dos veces es muy probable que este se trate de una retrasmisión. En cambio, si se reciben dos paquetes de origen con distinto THL es muy probable que sea el mismo paquete que ha regresado al mismo nodo debido a un bucle en la ruta.
Este tipo de paquetes se indentifican con el tipo AM número 71 y su estructura se encuentra definida en la estructura *ctp_data_header_t* (ver cuadro 26) del archivo Ctp.h.

Cuadro 26 Estructura ctp_data_header_t

```
typedef nx_struct {
  nx_ctp_options_t    options;
  nx_uint8_t          thl;
  nx_uint16_t         etx;
  nx_am_addr_t        origin;
  nx_uint8_t          originSeqNo;
  nx_collection_id_t  type;
  nx_uint8_t          (COUNT(0) data)[0];
} ctp_data_header_t;
```

A.9.0.2 **Paquetes de enrutamiento CTP**

Los campos que contiene un paquete de enrutemiento CTP o *beacon* son los siguientes (ver figura A.12):

- Los bits P y C son análogos a los paquetes de datos CTP.
- El campo *parent* contiene el identificador del actual nodo padre.
- El campo ETX contiene el valor de la métrica actual de la ruta.

Este tipo de paquetes se indentifican con el tipo AM número 70 y su estructura se encuentra definida en la estructura *ctp_routing_header_t* (ver cuadro 27) del archivo Ctp.h.

Cuadro 27 Estructura ctp_routing_header_t

```
typedef nx_struct {
  nx_ctp_options_t    options;
  nx_am_addr_t        parent;
  nx_uint16_t         etx;
  nx_uint8_t          (COUNT(0) data)[0];
} ctp_routing_header_t;
```

A.9.0.3 **Paquetes LEEP**

Un paquete LEEP se compone de un cabezal, una sección de datos y un pie como puede verse en la figura A.12.
El cabezal se encuentra definido en la estructura *linkest_header_t* (ver cuadro 28) del archivo LinkEstimator.h y sus campos son los siguientes:

- *NE* indica el número de *Link Information entries* en el pie LEEP.
- *Rsrvd* es un campo reservado y debe ser cero.
- *SeqNo* es el número de secuencia utilizado por LEEP.

Al contar el número de *beacons* recibidas de cada vecino y compararlo con el *SeqNo* el estimador de enlaces puede estimar el número de *beacons* que faltan sobre el total de *beacons* enviados por un vecino.

Cuadro 28 Estructura linkest_header_t

```
typedef nx_struct linkest_header {
  nx_uint8_t    flags;
  nx_uint8_t    seq;
} linkest_header_t;
```

El pie contiene un número variable de *Link Information entries*, pudiendo ser eventualmente cero. El formato de cada *Link Information entry* es el siguiente:

- *LinkEtx* es la estimación de la calidad *in-bound* del enlace que comunica el nodo cuyo identificador está en el siguiente campo con el nodo que emite el *beacon*.
- *NodeId* es el identificador del nodo vecino para el cual se envía el ETX.

El pie se encuentra definido en la estructura *linkest_footer_t* y cada entrada en *neighbor_stat_entry_t* (ver cuadro 29) del archivo LinkEstimator.h.

Cuadro 29 Estructuras linkest_footer_t y neighbor_stat_entry_t

```
typedef nx_struct neighbor_stat_entry {
  nx_am_addr_t      ll_addr;
  nx_uint8_t        inquality;
} neighbor_stat_entry_t;

typedef nx_struct linkest_footer {
  neighbor_stat_entry_t      neighborList[1];
} linkest_footer_t;
```

A.10 Verificación del funcionamiento y análisis de tramas CTP

En estas sección se analizan las tramas obtenidas mediantes capturas y simulaciones Avrora. Para las capturas se utilizó el transmisor de radio CC2420. Mientras que para las simulaciones se consideraron los transmisores CC1000 y CC2420.

A.10.1 Capturas de nodos

Paquete de enrutamiento CTP

Para comprobar el funcionamiento del protocolo CTP en dispositivos reales se utilizaron dos nodos sensores Shimmer y un adaptador de red IEEE 802.15.4 SPAN. En los dos nodos se instaló la aplicación TestNetwork, que es una aplicación que viene por defecto en TinyOS 2.x y cuyo objetivo es probar el protocolo CTP. Uno de los nodos actúa como la raíz del arbol CTP mientras que el otro es un nodo simple que envía sus datos a la raíz. En el SPAN se instaló la aplicación *base station* y utilizando la aplicación *Listen* que viene con TinyOS se obtuvieron los paquetes que el módulo SPAN envia por el puerto USB. Esto se puede realizar ejecutando el comando *java net.tinyos.tools.Listen -comm serial@/dev/ttyUSBx:shimmer* donde en /dev/ttyUSBx debe ponerse el puerto donde está conectado el SPAN, por ejemplo /dev/ttyUSB0. Como fue explicado anteriormente la aplicación *base station* recibe paquetes a través de su transmisor de radio, los transforma en un paquete de tipo serial y los envía a través del puerto UBS. Por lo tanto, los paquetes capturados en este caso difieren un poco de las tramas observadas en simulación que contenían también los bytes agregados por la capa física y la capa de enlace.
Los paquetes de tipo serial tiene la estructura mostrada en el cuadro 31 y definida en la estructura *serial_packet_t*, se compone de un cabezal y de sus datos. El cabezal se encuentra definido en la estructura *serial_header_t* que se muestra en el cuadro 30.

Cuadro 30 Estructura serial_header_t

```
typedef nx_struct serial_header {
  nx_am_addr_t      dest;
  nx_am_addr_t      src;
  nx_uint8_t        length;
  nx_am_group_t     group;
  nx_am_id_t        type;
} serial_header_t;
```

Cuadro 31 Estructura serial_packet_t

```
typedef nx_struct serial_packet {
  serial_header_t     header;
  nx_uint8_t          data[];
} serial_packet_t;
```

En la figura A.13 se encuentra uno de los *beacons* CTP obtenidos a través de la captura. Observar que los bytes capturados se corresponden con la sucesión de las siguientes estructuras: *serial_header_t – linkest_header_t – ctp_routing_header_t – linkest_footer_t*.

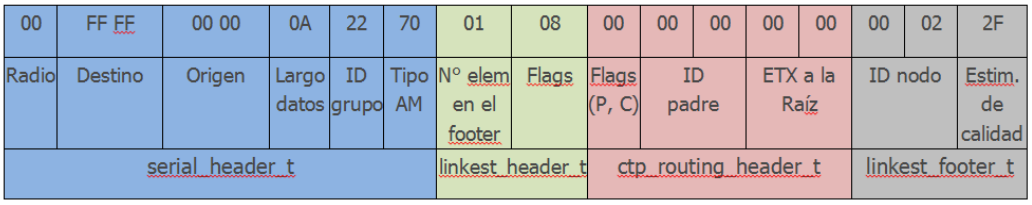


Figura A.13: Paquete de enrutamiento CTP obtenido realizando sniffing.

El primer byte es agregado al formar un paquete serial con los bytes recibido a través de la radio por lo que puede ser ignorado, para todos los paquetes capturados, se observa el byte 00 que representa un mensaje AM (*Active Message*). En este caso el Tipo AM es 70, lo que indica que se trata de un paquete de enrutamiento CTP. El primer byte del *linkest_header_t* indica la cantidad de elementos que hay en el pie *linkest_footer_t*. En este caso, solo contiene un elemento, donde cada elemento es una estructura de tipo *neighbor_stat_entry_t* 29.

Paquete de datos CTP

En la figura A.14 se muestran los bytes correspondientes a un paquete de datos obtenidos luego de realizer *sniffing* de la aplicación TestNetwork en el escenario explicado anteriormente.

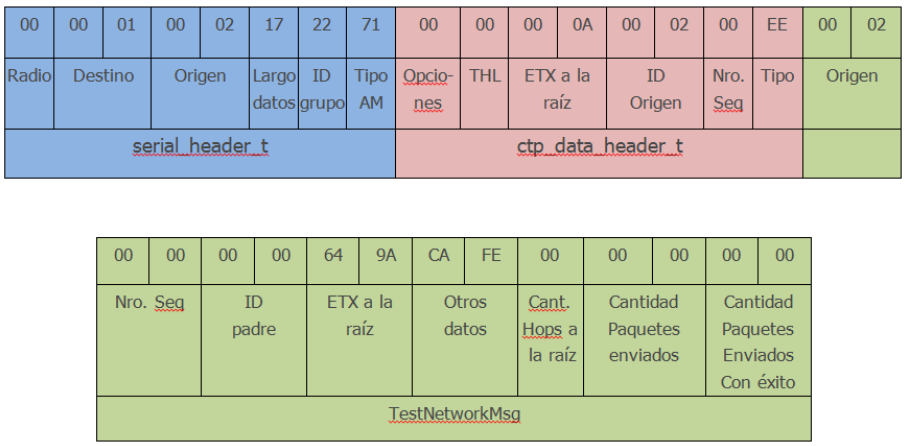


Figura A.14: Paquete de datos CTP obtenido realizando sniffing.

La parte verde representa los datos del paquete que son enviados por la aplicación TestNetwork que hace uso de CTP como su capa de red. Esta estructura se encontra definida en la estructura *TestNetworkMsg* que se muestra en el cuadro 32 archivo TestNetwork.h.

A.10.2 Simulador AVRORA

A continuación se presentan las tramas enviadas por los transmisores de radio CC2420 y CC1000 conteniendo paquetes del protocolo CTP.

Cuadro 32 Estructura TestNetworkMsg

```
typedef nx_struct TestNetworkMsg {
        nx_am_addr_t     source;
         nx_uint16_t     seqno;
         nx_am_addr_t    parent;
         nx_uint16_t     metric;
         nx_uint16_t     data;
         nx_uint8_t      hopcount;
         nx_uint16_t     sendCount;
         nx_uint16_t     sendSuccessCount;
} TestNetworkMsg;
```

A.10.2.1 **Transmisor de radio CC2420**

Paquete de enrutamiento CTP

En la figura A.15 se muestran los bytes correspondientes a un paquete de enrutamiento obtenido a través de una simulación con AVRORA para la plataforma MicaZ que utiliza el transmisor de radio CC2420.

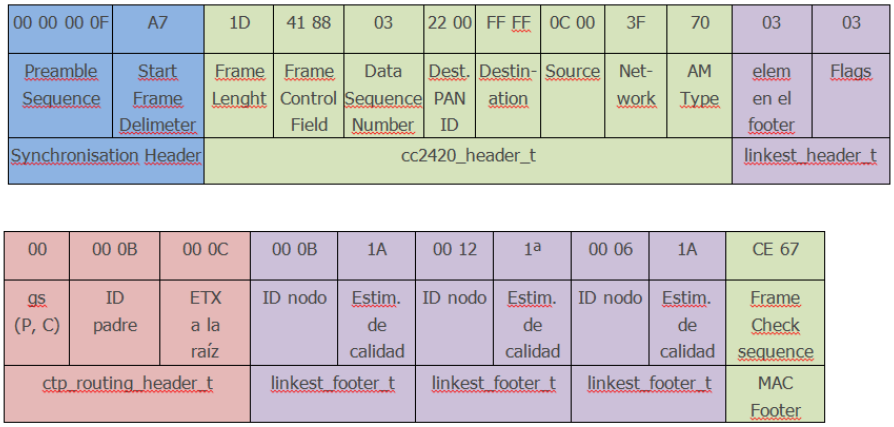


Figura A.15: Paquete de enrutamiento CTP obtenido con el simulador AVRORA (para el transmisor de radio CC2420).

Cuadro 33 Estructura cc2420_header_t

```
typedef nx_struct cc2420_header_t {
  nxle_uint8_t      length;
  nxle_uint16_t     fcf;
  nxle_uint8_t      dsn;
  nxle_uint16_t     destpan;
  nxle_uint16_t     dest;
  nxle_uint16_t     src;
  nxle_uint8_t      network;
  nxle_uint8_t      type;
} cc2420_header_t;
```

En este caso se observan todos los datos enviados a través del canal, incluyendo los bits de sincronización (en azul), los datos agregados por el controlador de radio CC2420 (en verde) que se corresponden con la estructura *cc2420_header_t* (ver cuadro 33) que se encuentra en la especificación del transmisor de radio y definida en el archivo CC2420.h y luego las estructuras enviadas por los protocolos CTP y

LEEP que fueron explicadas anteriomente. Al final de la trama se envían dos bytes que son un código de corrección de errores para chequear si los bits recibidos no sufrieron modificaciones mientras se transmitían en el canal.

Paquete de datos CTP

En la figura A.16 se muestra un paquete de datos utilizando el protocolo CTP obtenido a través de una simulación con AVRORA para la plataforma MicaZ que utiliza el transmisor de radio CC2420.

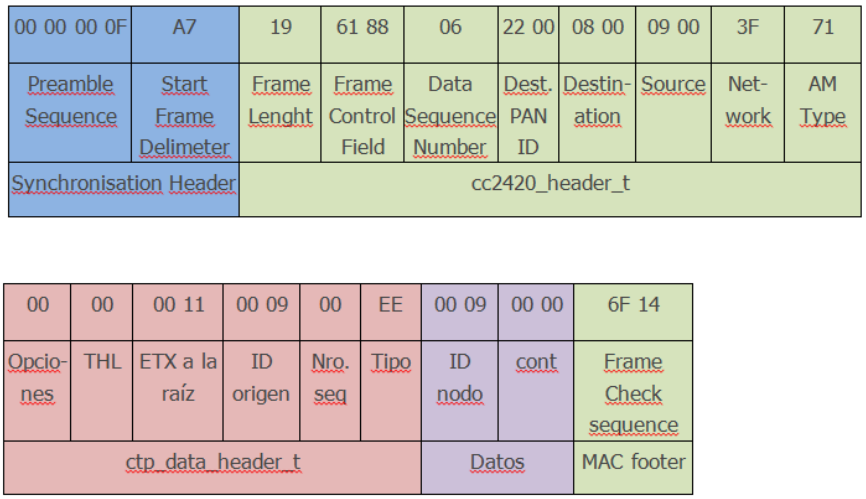


Figura A.16: Paquete de datos CTP obtenido con el simulador AVRORA (para el transmisor de radio CC2420).

Los datos enviados de la aplicación que se envían en esta trama se muestran en color violeta y se corresponden con la estructrura *EasyCollectionMsg* que se muestra en el cuadro 34. Esta estructura de datos es la que se utilizará en las pruebas realizadas para los ataques. Contiene dos campos que almacenan el identificador del nodo y un contador que es incrementado cada vez que se envía un paquete nuevo. De esta forma, en cualquier salto podremos identificar claramente quien es el origen del paquete (esta información es enviada en los datos por la aplicación pero también está en el cabezal CTP) y si el paquete enviado es una retransmisión de la capa de enlace o es un paquete nuevo.

Cuadro 34 Estructura EasyCollectionMsg

```
typedef nx_struct EasyCollectionMsg {
        nx_uint16_t     id;
        nx_uint16_t     cont;
} EasyCollectionMsg;
```

El resto de los datos son análogos a los explicados en las secciones anteriores.

Reconocimientos CTP

Los reconocimientos utilizados por el protocolo CTP son los reconocimientos de la capa de enlace, por lo que el protocolo no debe encargarse de enviarlos expresamente. En la figura A.17 puede apreciar la estructura de una trama de reconocimiento obtenida en una simulación con Avrora para la plataforma MicaZ con el transmisor de radio CC2420.

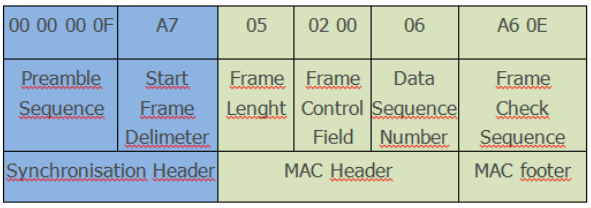


Figura A.17: Trama de reconocimiento obtenida con el simulador AVRORA (para el transmisor de radio CC2420).

A.10.2.2 **Transmisor de radio CC1000**

Paquete de enrutamiento CTP

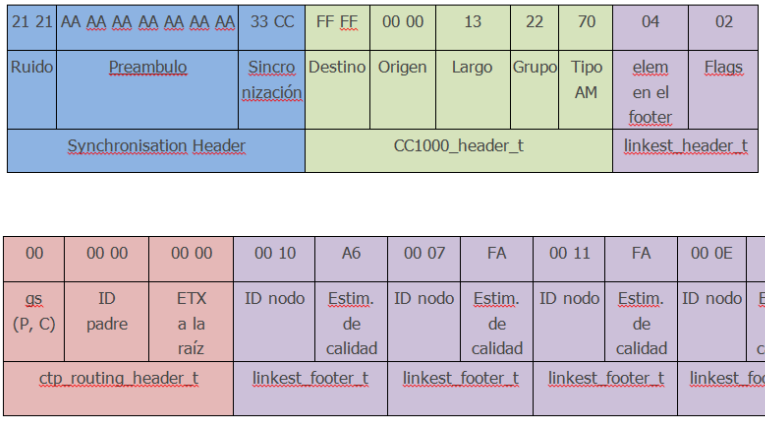


Figura A.18: Paquete de enrutamiento CTP obtenido con el simulador Avrora (para el transmisor de radio CC1000).

En la figura A.18 se muestran los bytes de un paquete de enrutamiento obtenido en una simulación con Avrora para la plataforma Mica2 que utiliza el transmisor de radio CC1000 [51].
En azul se muestran los bits de sincronización, luego se muestran los datos agregados por el controlador de radio CC1000 (en verde) que se corresponden con la estructura *cc1000_header_t* que se muestra en el cuadro 35 definida en el archivo CC1000Msg.h. También se pueden observar las estructuras enviadas por los protocolos CTP y LEEP que fueron explicadas anteriomente y al final de la trama se envían tres bytes que son un código de corrección de errores que se corresponde con la estructura *cc1000_footer_t* que se muestra en el cuadro 36 también definida en el archivo CC1000Msg.h. El largo del campo ruido puede variar entre uno y dos bytes.

Cuadro 35 Estructura cc1000_header_t

```
typedef nx_struct cc1000_header_t {
    nx_am_addr_t    addr;
    nx_uint8_t      length;
    nx_am_group_t   group;
    nx_am_id_t      type;
} cc1000_header_t;
```

Cuadro 36 Estructura cc1000_footer_t

```
typedef nx_struct cc1000_footer_t {
    nxle_uint16_t     crc;
} cc1000_footer_t;
```

Paquete de datos CTP

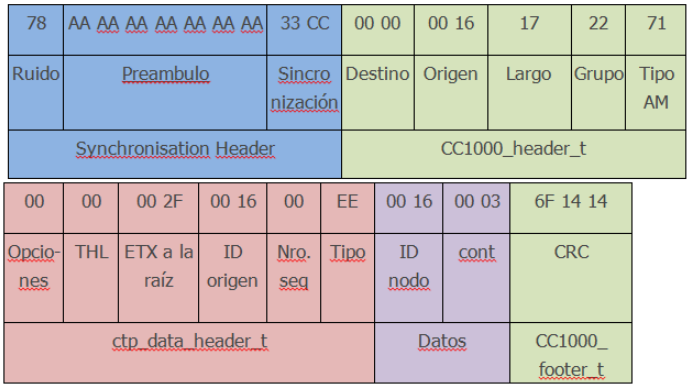


Figura A.19: Paquete de datos CTP obtenido con el simulador Avrora (para el transmisor de radio CC1000).

En la figura A.19 se muestran un paquete de datos del protocolo CTP obtenido de una simulación con Avrora para la plataforma Mica2 con el transmisor de radio CC1000.
Los bytes enviados en esta trama son análogos a los explicados para los casos anteriores. Los datos, al igual que en el caso del transmisor de radio CC2420, son enviados en la estructrura EasyCollectionMsg.

A.11 Capturas para el ataque al protocolo GTS

A continación, en las figuras A.20, A.21 y A.22, se muestran las tramas de solicitud de ranuras GTS.

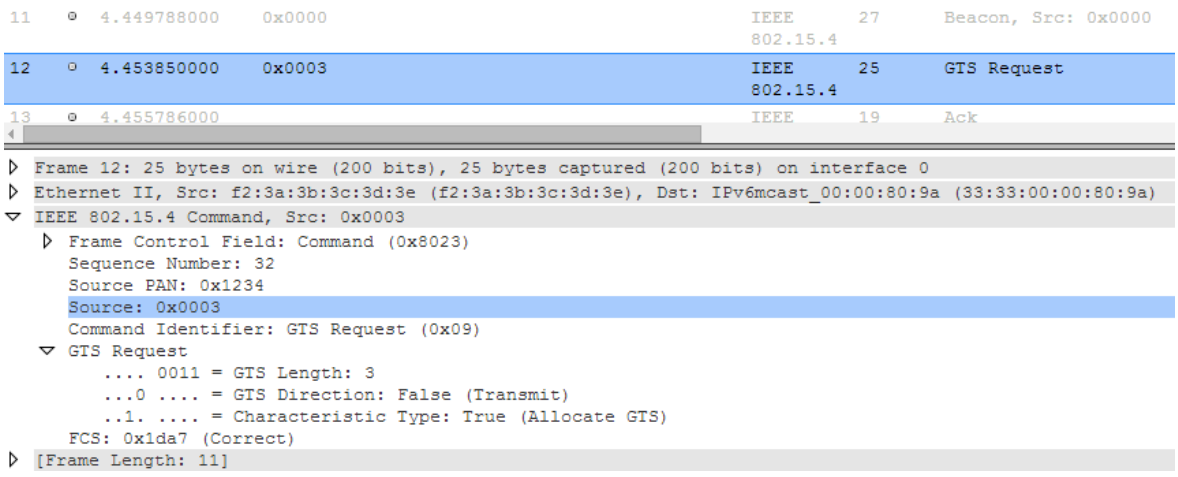


Figura A.20: Escenario 2 - ataque GTS - allocation request - trama 12

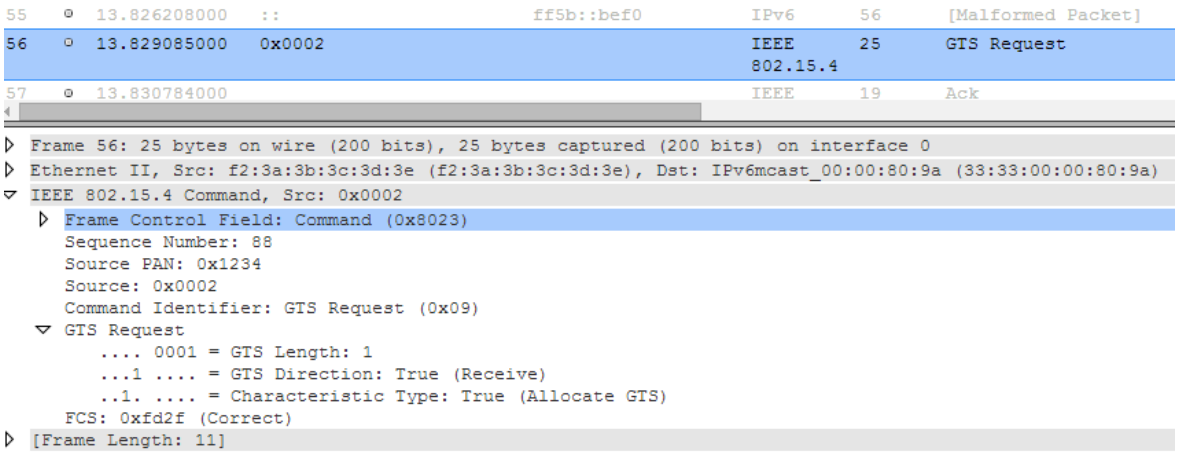


Figura A.21: Escenario 2 - ataque GTS - allocation request - trama 56

Luego estas ranuras son asignadas por el coordinador de la red. Esto se puede ver en la figura A.23.
A continuación el atacante comienza a mandar las solicitudes de *deallocation*. Estas tramas pueden observarse en las figuras A.24, A.25 y A.26.
Luego el coordinador de la red le quita las ranuras a los nodos víctimas. Por lo que el siguiente *beacon* no contiene *GTS descriptors* como puede verse en la figura A.27.

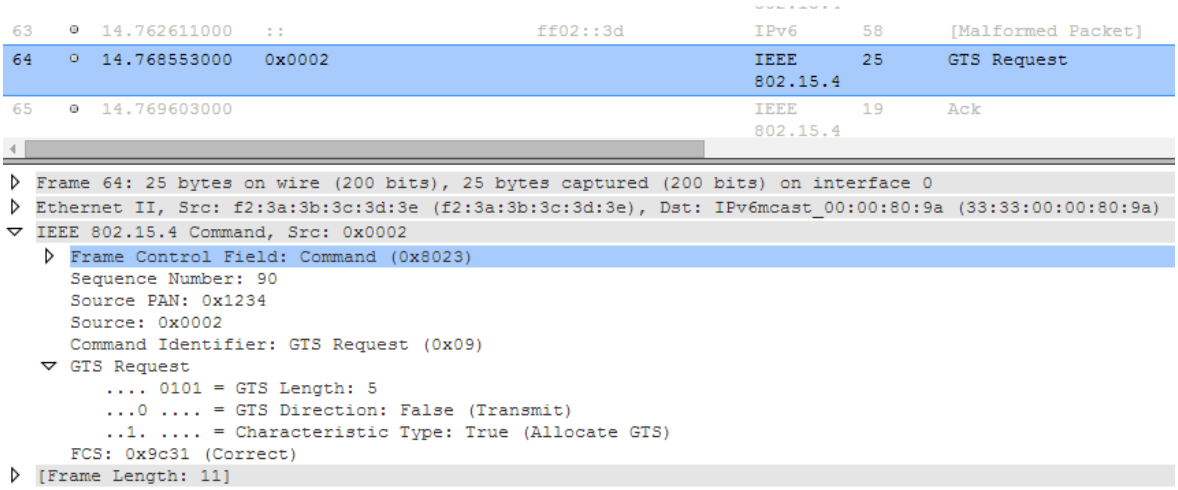


Figura A.22: Escenario 2 - ataque GTS - allocation request - trama 64

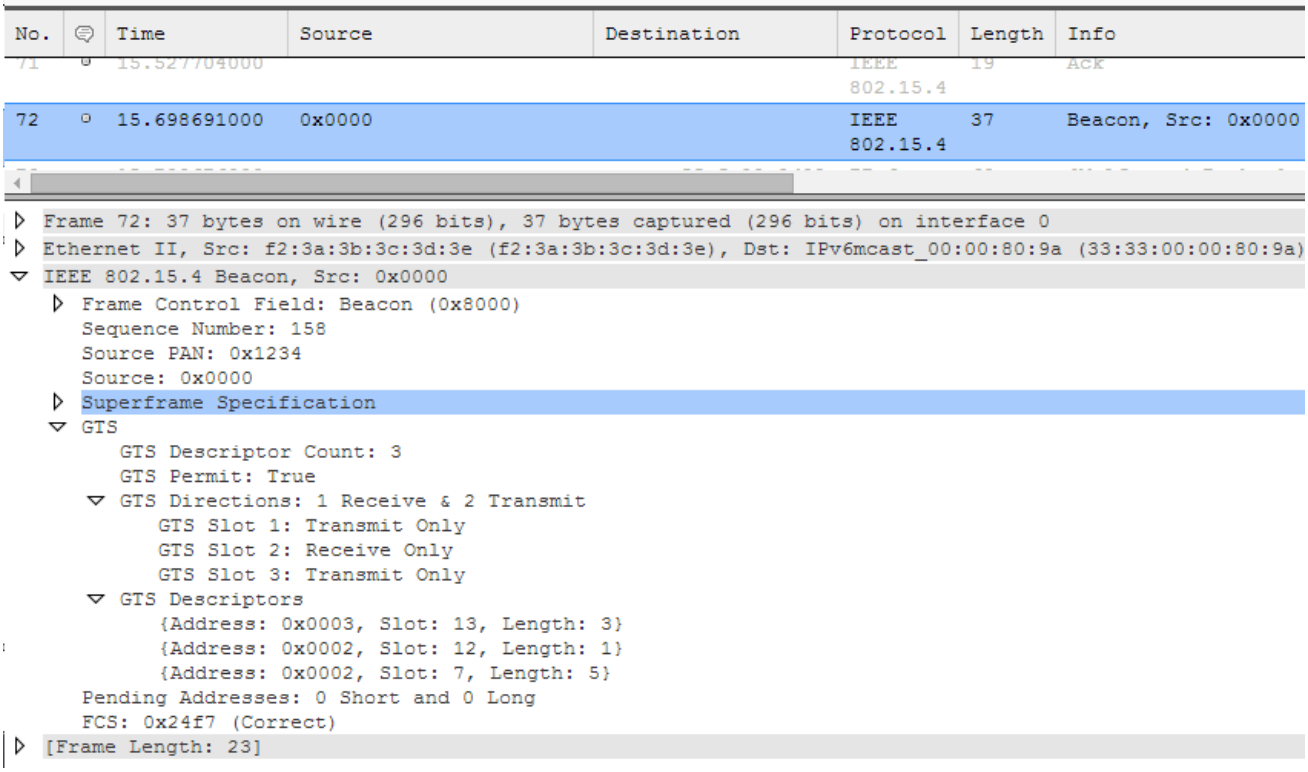


Figura A.23: Escenario 2 - ataque GTS - beacon

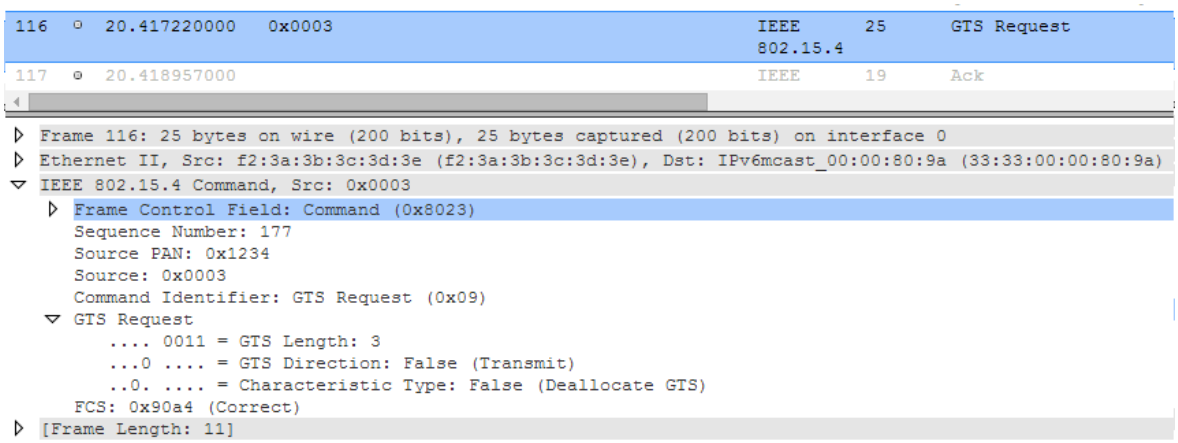


Figura A.24: Escenario 2 - ataque GTS - deallocation request - trama 116

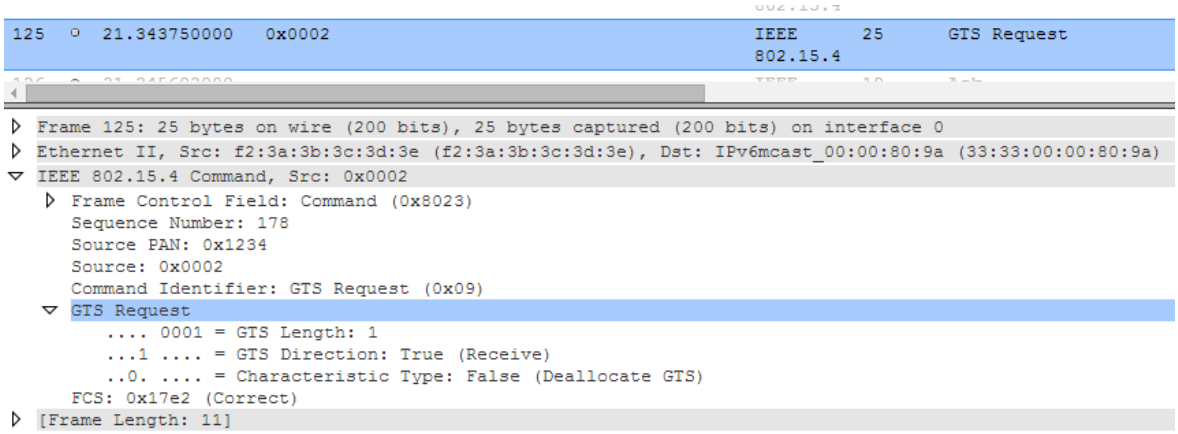


Figura A.25: Escenario 2 - ataque GTS - deallocation request - trama 125

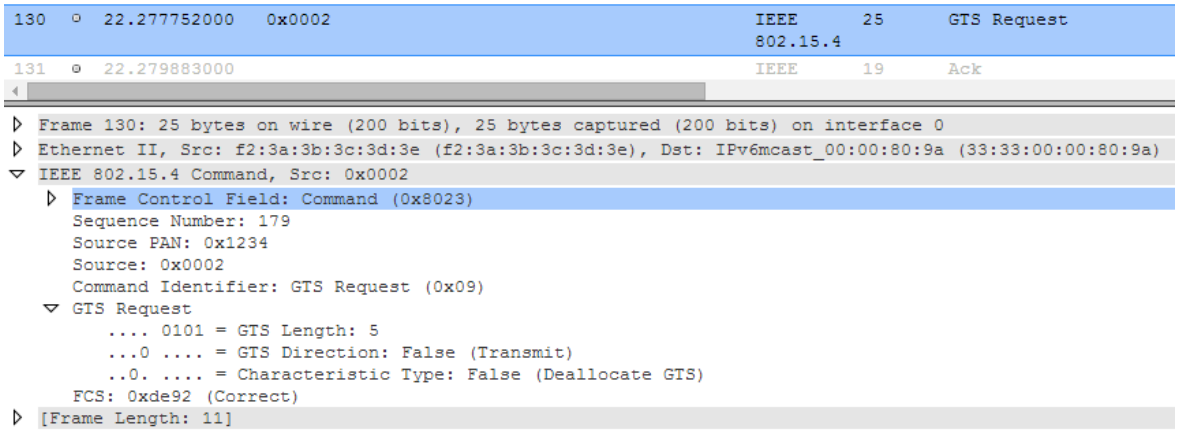


Figura A.26: Escenario 2 - ataque GTS - deallocation request - trama 130

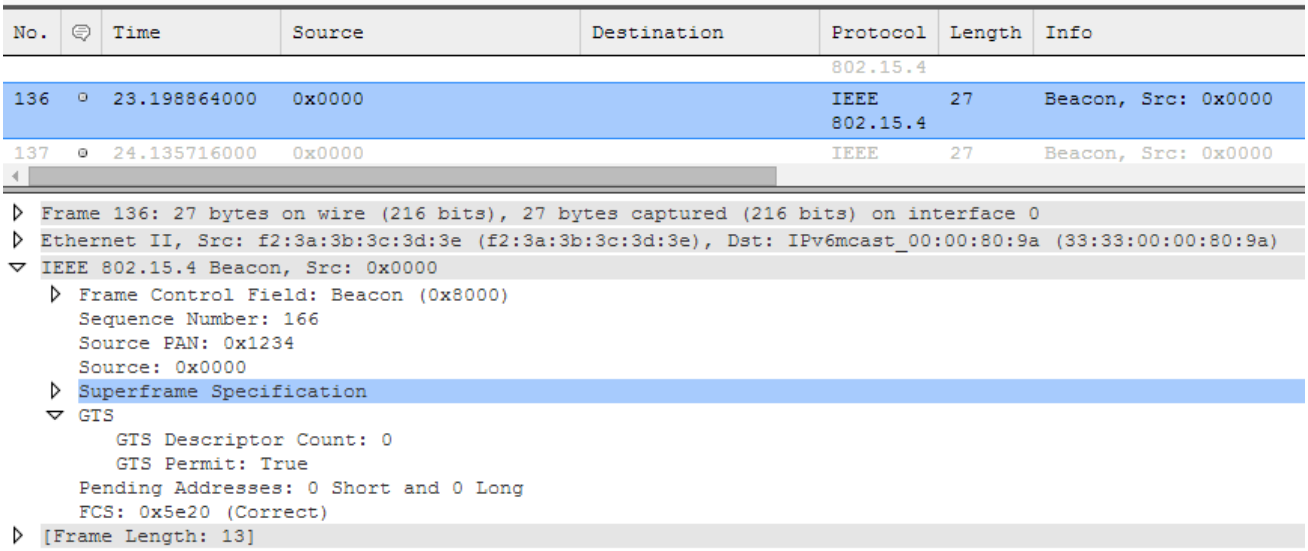


Figura A.27: Escenario 2 - ataque GTS - beacon