TESIS DEFENDIDA POR

José Salvador Portugal Luna

Y aprobada por el siguiente comité:

Dr. José Antonio García Macías
Director del Comité

M.C. Christian Paúl García Martínez
Miembro del Comité

Dr. Luis Armando Villaseñor González
Miembro del Comité

Dr. Pedro Gilberto López Mariscal
Miembro del Comité

Dr. Pedro Gilberto López Mariscal
Coordinador del Programa de Posgrado en Ciencias de la Computación

Dr. David Hilario Covarrubias Rosales
Director de Estudios de Posgrado

31 de Octubre de 2008

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE
EDUCACIÓN SUPERIOR DE ENSENADA

PROGRAMA DE POSGRADO EN CIENCIAS
EN CIENCIAS DE LA COMPUTACIÓN

Protocolo de Red Ligero para Redes Inalámbricas de Monitoreo y Control que Utilizan el Estándar IEEE 802.15.4

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN CIENCIAS

Presenta:

José Salvador Portugal Luna

Ensenada, Baja California, México. Octubre de 2008.

RESUMEN de la tesis que presenta **José Salvador Portugal Luna**, como requisito parcial para obtener el grado de MAESTRO EN CIENCIAS en CIENCIAS DE LA COMPUTACIÓN. Ensenada, Baja California. Octubre de 2008.

Protocolo de Red Ligero para Redes Inalámbricas de Monitoreo y Control que Utilizan el Estándar IEEE 802.15.4

Resumen aprobado por:

Dr. José Antonio García Macías
Director de Tesis

Las redes inalámbricas de monitoreo y control son utilizadas en aplicaciones donde se requiere comunicar dispositivos con bajos requerimientos de transmisión y procesamiento de datos. Algunas de estas aplicaciones son monitoreo de temperatura, humedad, control de riego, luces, entre otros.

Para la comunicación en éste tipo de redes, el IEEE ha creado el estándar 802.15.4 el cual define los niveles físico y de enlace de datos (según el modelo de referencia OSI), haciendo posible la comunicación con dispositivos que se encuentran a un salto. Debido a las limitaciones de los transmisores, se requieren protocolos de red que permitan la comunicación de dispositivos que se encuentran mas allá de ésta distancia.

Con el objetivo de desarrollar una especificación abierta de un protocolo de red que pueda ser utilizado para fines académicos o en el área comercial con bajo costo, se propone un protocolo con tamaño de código pequeño que permite el envío de mensajes por múltiples saltos sin necesidad de mensajes de inundación para descubrimiento de rutas, también posee la capacidad de enviar mensajes *broadcast* sin número de secuencia reduciendo el tamaño de la información de control enviada y la administración de ésta sin el problema de rebote de mensajes *broadcast*.

Como parte de este trabajo se realiza una evaluación del protocolo propuesto para que investigadores o desarrolladores del área de redes inalámbricas de monitoreo y control puedan determinar la viabilidad de su utilización en aplicaciones específicas.

Palabras clave: Redes Inalámbricas de Monitoreo y Control, Protocolos de Red, IEEE 802.15.4, Redes de Sensores.

ABSTRACT of the thesis presented by **José Salvador Portugal Luna**, as a partial requirement to obtain the MASTER SCIENCE degree in COMPUTER SCIENCE. Ensenada, Baja California. October 2008.

Lightweight Network Protocol for Wireless Control and Monitoring Networks Using the IEEE 802.15.4 Standard

Wireless control and monitoring networks are used in applications that require low data rate communication. Some of these applications are temperature and humidity monitoring, irrigation and light controlling, and so on.

For communication in this kind of networks, IEEE has created the standard 802.15.4 which defines the physical and data link layer (according to the OSI reference model). This standard provides single hop communications and requires network protocols to allow communication between devices more than one hop away.

This work proposes the open specification of a network protocol that may be used for academic purposes or at the commercial level at low cost. This protocol is lightweight in code size and allows sending messages through multiple hops without the need for flooding messages for routes discovery; another feature is that it can send broadcast messages without sequence number, reducing the size of control information, without incurring in message bouncing found in typical broadcast.

As part of this work, an evaluation of the proposed protocol was performed. This allows researchers and developers in the wireless control and monitoring networks area to determine the viability of using the protocol in specific applications.

Keywords: Wireless control and monitoring networks, Network protocol, IEEE 802.15.4, Wireless sensor networks.

Dedicatoria

A mis padres y a Daniela

Agradecimientos

A Daniela, por acompañarme en esta aventura
y en las que faltan.

A mis padres, por su apoyo incondicional.

A mis hermanos, por que a pesar de la distancia,
me hacen sentir cerca.

A mi asesor, Dr. J. Antonio García Macías,
por su tiempo, atención y consejos.

A los miembros de mi comité, M.C. Christian Paúl García Martínez,
Dr. Luis Armando Villaseñor González y Dr. Pedro Gilberto
López Mariscal, por sus invaluables observaciones.

A M.C. Christian Paúl García Martínez,
por facilitarme las herramientas necesarias
para realizar este trabajo.

A mis compañeros de la generación 2006, por
la amistad otorgada durante estos dos años.

Al Centro de Investigación Científica y de Educación
Superior de la ciudad de Ensenada, Baja California.

Al Consejo Nacional de Ciencia y Tecnología,
por su apoyo económico otorgado para
la realización de este trabajo de investigación.

Ensenada, México
31 de Octubre de 2008.

José Salvador Portugal Luna

Tabla de Contenido

Tabla de Contenido (Continuación)

Tabla de Contenido (Continuación)

Capítulo Página

Lista de Figuras

Lista de Tablas

Capítulo I

Introducción

Las redes inalámbricas de monitoreo y control son un tipo de tecnología que permiten alcanzar ambientes de cómputo ubicuo como el visionado por Weiser, donde se cuenta con dispositivos embebidos en el entorno de los usuarios, mismos que se comunican de forma automática para ayudar en tareas cotidianas (Weiser, 1991).

Los dispositivos utilizados cuentan con pequeños procesadores capaces de realizar operaciones tan complejas que pueden tomar decisiones como seleccionar y transmitir sólo los datos requeridos, por ejemplo, enviar lecturas de temperatura cada 10 minutos.

Los protocolos y algoritmos de las redes inalámbricas de monitoreo y control tienen capacidades de autoorganización, que permiten cumplir con las tareas para las que fueron programadas a través de un esfuerzo cooperativo. Por estas características, las redes de sensores tienen una gran variedad de aplicaciones, algunas de estás son en entornos de salud, militares y de seguridad. Por ejemplo, los datos fisiológicos de pacientes pueden ser monitoreados remotamente por su médico, lo cual resulta conveniente para los pacientes, porque pueden estar cómodamente instalados en su casa y permite al médico un mejor entendimiento de la situación.

Ante la necesidad de establecer mecanismos de comunicación para este tipo de aplicaciones, el IEEE (*Institute of Electrical and Electronics Engineers, Inc.*) ha creado la especificación 802.15.4, en la cual se definen los niveles: físico y de enlace de datos (según el modelo de referencia OSI (ISO/IEC7498-1, 1994)), para redes inalámbricas de área personal[1] con bajas velocidades de transmisión, pero vidas útiles muy altas, con alimentación limitada (baterías) y muy baja complejidad. Actualmente la especificación IEEE 802.15.4 es un estándar ampliamente utilizado, según firma de investigación de mercado In-Stat (http://www.in-stat.com), se espera tener un mercado anual de alrededor de 150 millones de unidades para el año 2009.

I.1. Planteamiento del problema

El estándar IEEE 802.15.4 por ser una especificación de nivel físico y de enlace de datos, proporciona comunicación entre nodos que se encuentran dentro del alcance de los radiotransmisores. El estándar requiere de mecanismos de comunicación que permitan la interconexión de los dispositivos que se encuentran más allá de ese alcance, los encargados de realizar esta tarea son los protocolos de red.

Una solución como protocolo de red utilizada desde hace varios años y que cuenta con amplio soporte por parte de la industria es ZigBee, la cual es un conjunto de protocolos de alto nivel sobre el estándar IEEE 802.15.4 contemplando un extenso mercado que va desde entretenimiento hasta control industrial. Como se ha visto por varias empresas (e.g. Synapse, San Juan Software, Jennic), por esta gran diversidad de aplicaciones para las que esta enfocado, ZigBee cuenta con funcionalidades que no son aprovechadas en su totalidad cuando se implementan aplicaciones específicas. Por lo que se requiere de protocolos que satisfagan necesidades de aplicaciones con características homogéneas, permitiendo

[1] Red de área personal se concibe como la interconexión de dispositivos separados por una distancia de hasta 10 metros.

eliminar funcionalidades que encarecen los requerimientos de procesamiento y almacenamiento para los nodos y para el usuario final, que en ocasiones debe pagar por hacer uso de protocolos bajo licencia que brindan características que no son utilizadas por las aplicaciones. Es por esto que es importante contar con protocolos de red, que al igual que los nodos de las redes inalámbricas de monitoreo y control, cuenten con baja complejidad, pero brinden una funcionalidad completa para grupos de aplicaciones con características similares.

Existen protocolos de red propietarios cuyas especificaciones no son del todo accesibles a la comunidad de desarrolladores e investigadores para permitir mejorar su comportamiento en aplicaciones con características definidas. De aquí surge la necesidad de especificaciones abiertas, que de cierta manera, permitan estandarizar las comunicaciones de redes inalámbricas de monitoreo y control.

Así mismo, existe una insuficiencia de implementaciones de protocolos de comunicación que establezcan una base para futuros desarrollos con bajo costo, eliminando gastos en licencias de utilización de protocolos propietarios. Esto es importante para permitir investigaciones en el ámbito académico que en ocasiones no cuenta con los recursos suficientes. Estas implementaciones deben proveer una interfaz para programación de aplicaciones en lenguajes que no requieran tiempo de aprendizaje considerable.

I.2. Objetivo general

Diseñar e implementar un protocolo de red, ligero para redes inalámbricas de monitoreo y control que utilizan el estándar IEEE 802.15.4.

I.3. Objetivos específicos

- Identificar los servicios provistos por la capa de enlace de datos.
- Establecer un modelo de direccionamiento y enrutamiento.
- Identificar y establecer los servicios que debe proveer la capa de red.
- Desarrollar un prototipo funcional con las siguientes características:
  - El tamaño del código que requiera poco espacio en memoria.
  - Sistema de direccionamiento con soporte para hasta 1024 nodos.
  - Enrutamiento multisalto.
  - Autoformación de redes.
  - Comunicación *unicast* y *broadcast*.
  - Proveer una interfaz para desarrollar aplicaciones sobre él.

I.4. Metodología

Las actividades realizadas en el desarrollo del protocolo ligero de red para redes de monitoreo y control que utilizan el estándar IEEE 802.15.4, son las siguientes:

- Revisión de la literatura: se analizaron los requerimientos de las aplicaciones actuales, así como las soluciones dadas por diferentes protocolos de red existentes.
- Definición de los requerimientos de protocolos de red: se analizaron los servicios provistos por las capas inferiores a la de red, para establecer la manera de interactuar con ellas y cuáles son las funcionalidades que le corresponden a la capa de red.
- Familiarización con herramientas de desarrollo: esta actividad se refiere a aprender a utilizar el lenguaje de programación y el ambiente de desarrollo para la implementación del prototipo.

- Análisis de los requerimientos para el desarrollo del prototipo: en esta etapa se establecerán los requerimientos que deberán ser cubiertos por el prototipo del protocolo de red.
- Diseño de un prototipo de protocolo: se planea y establece la funcionalidad del prototipo.
- Implementación del prototipo: desarrollo del código del prototipo.
- Evaluación del prototipo: implementación de una aplicación de redes inalámbricas de monitoreo y control, haciendo uso del prototipo de protocolo de red y validación de la viabilidad de la utilización de este protocolo.

I.5. Contenido de la tesis

El contenido de esta tesis se divide en cinco capítulos. En el primer capítulo se muestra el planteamiento del problema, así como la metodología empleada para proponer una solución.

El segundo capítulo presenta un panorama de las redes inalámbricas de monitoreo y control y de los protocolos de comunicación para este tipo de dispositivos, entre estos protocolos se encuentra el estándar IEEE 802.15.4 utilizado como base para el desarrollo de este protocolo de red.

En el capítulo tres se describe la especificación del protocolo de red desarrollado. Se muestra la arquitectura de direccionamiento, los formatos de mensajes utilizados, las operaciones realizadas por el protocolo y la interfaz de programación de aplicaciones, entre otras particularidades.

El capítulo cuatro trata acerca de la evaluación del protocolo de red, en el se exponen las funcionalidadades que cumple la especificación del protocolo de red. También, se muestra

las pruebas de desempeño realizadas al prototipo desarrollado, estas pruebas arrojan medidas, como ancho de banda para datos, que permiten elegir la utilización del protocolo en distintas aplicaciones.

El quinto y último capítulo, muestra las conclusiones finales de este trabajo, así como las principales aportaciones y propuestas para trabajo futuro que mejoren el protocolo aquí descrito.

Capítulo II

Redes Inalámbricas de Monitoreo y Control

II.1. Introducción

Las redes inalámbricas de monitoreo y control son utilizadas en aplicaciones donde el cablear un alto número de sensores y dispositivos controladores resultaría impráctico y costoso.

Las redes están formadas por nodos que son puestos dentro o muy cerca de un fenómeno a monitorear, y por medio de los efectos o cambios que causa el mismo fenómeno, se puede identificar qué está sucediendo. La colocación de los nodos no necesariamente es prediseñada, esto permite utilizarlos en una posición aleatoria dentro de terrenos inaccesibles o en zonas de desastres. Los nodos son capaces de observar el mundo físico por medio de sensores, procesar los datos obtenidos o enviarlos a otro nodo para su procesamiento, tomar decisiones basadas en las observaciones y ejecutar acciones adecuadas por medio de actuadores que pueden informar lo que está sucediendo o activar algún dispositivo que modifique el fenómeno monitoreado (Akyildiz y Kasimoglu, 2004).

Las redes inalámbricas de monitoreo y control, en las que está enfocado este trabajo, son aquellas constituidas por pequeños dispositivos de bajas prestaciones y bajo costo, los cuales en la mayoría de los casos son del tamaño de su fuente de energía (un par de baterías AA), y consisten de una unidad de sensado, una unidad procesadora y un componente de comunicación. Estos dispositivos están limitados en poder de cómputo y capacidades de almacenamiento por el espacio fisico y mayormente por la capacidad de su fuente de energía. La comunicación entre los dispositivos se realiza por medios inalámbricos a través de radiofrecuencias, pero al contar con restricciones como la vida de las baterías, que son la fuente principal de energía, el rango de transmisión es corto y para comunicarse con todos los nodos de la red se requiere de un esfuerzo cooperativo.

Dependiendo de la aplicación, los nodos pueden contar con diferentes tipos de sensores para medir distintos tipos de variables (luz, humedad, sonido, etc.). Los nodos obtienen lecturas de los sensores, procesan la información y son capaces de diseminar la información utilizando distintos nodos para enrutar los paquetes y entregarlos a un nodo central llamado *sink*. Una red de sensores es un tipo especial de una red *ad hoc* con algunas diferencias (Akyildiz et al., 2002):

- El número de nodos en este tipo de redes puede ser mucho mayor que en una red *ad hoc*.
- Los nodos son densamente emplazados.
- Los nodos son más propensos a fallas.
- La topología de la red puede variar con mayor frecuencia.
- La comunicación está basada en mensajes *broadcast*, en la mayoría de las redes *ad hoc* es comunicación punto a punto.
- A diferencia de las redes *ad hoc*, los nodos cuentan con más restricciones en cuanto a energía, capacidades computacionales y memoria.

- Los nodos pueden no tener un identificador global, debido al alto *overhead* y a la cantidad de nodos en la red.

II.1.1. Plataformas de *hardware*

Existen varias compañías fabricantes de los dispositivos utilizados en las redes inalámbricas de monitoreo y control, pero la principal particularidad que comparten estas es el bajo costo, restringiendo de cierta manera las demás características (memoria RAM, memoria no volátil, capacidades de procesamiento, etc.), que comparadas con las capacidades de una computadora para uso doméstico son muy pequeñas. En la Figura 1 se muestran los componentes de estos dispositivos.

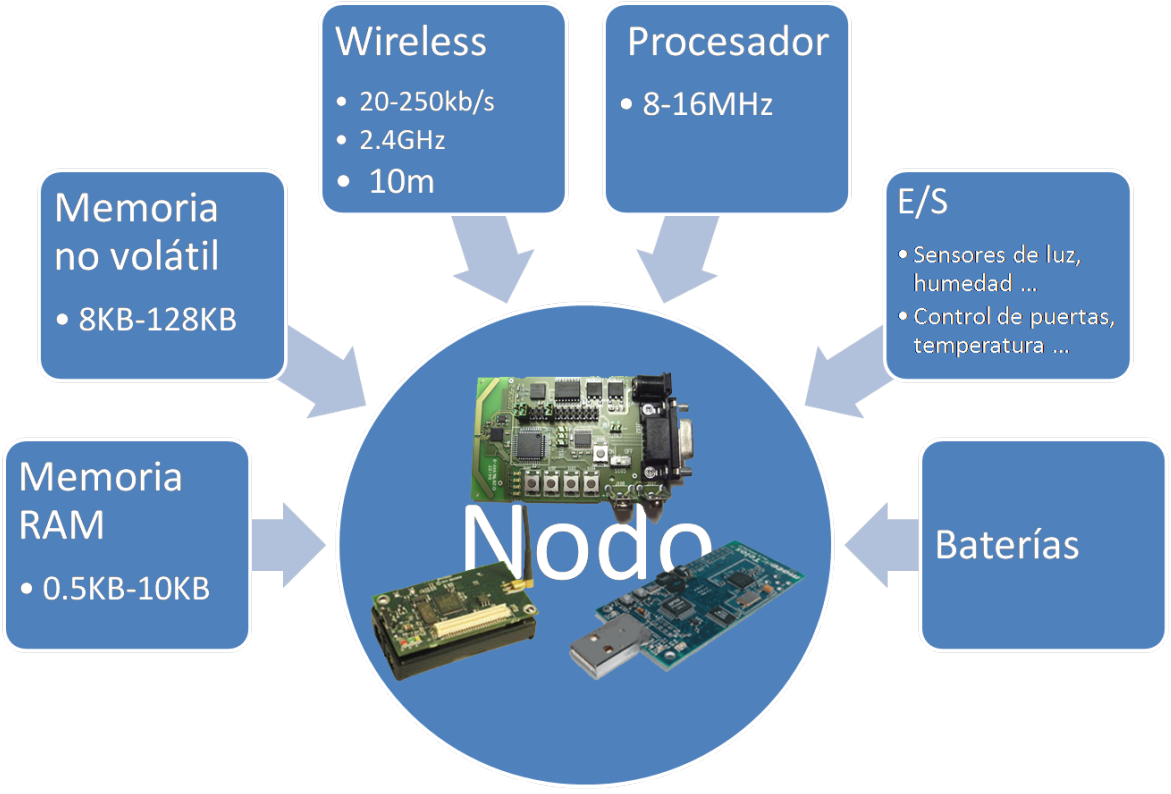


Figura 1: **Componentes principales de los dispositivos de redes inalámbricas de monitoreo y control.**

La memoria *RAM* (*Random Access Memory*), utilizada para almacenar los valores que

toman las variables durante la ejecución de programas es volátil, así que cuando el dispositivo se apaga la información almacenada en ella se pierde. Esta puede ir de los 0.5 kBytes a los 10 kBytes.

La Unidad Central de Procesamiento (*CPU*, *Central Processor Unit*) de este tipo de dispositivos son en su gran mayoría de 8 bits, aunque también hay de 16 y 32 bits, implicando un costo mayor. Las velocidades del *CPU* varían desde los 8 MHz hasta 20 MHz.

La memoria no volátil es utilizada principalmente para almacenar el programa que será ejecutado por el *CPU*; también se utiliza para almacenar información que debe estar disponible después de una pérdida de energía o reinicio del sistema. Los tamaños de este tipo de memoria se encuentran entre 8 kBytes y 128kBytes.

Para capturar valores de distintas variables, según sea la aplicación, se cuenta con diferentes tipos de sensores como de luz, humedad, temperatura, sonido, presión, etcétera. Estos convierten datos analógicos a una representación digital que puede ser procesada por la red para informar lo que está sucediendo y tomar las acciones pertinentes.

Las acciones que debe realizar la red son ejecutadas por actuadores. Estos pueden realizar tareas como prender un foco, subir o bajar la temperatura de una habitación o de todo un edificio, encender dispersores de agua en jardines o invernaderos, etcétera.

Todo esto, memoria *RAM*, *CPU*, memoria no volátil, sensores y actuadores, son provistas de energías por baterías, la utilización de estas permiten implantarlos en diferentes lugares sin necesidad de tener una red eléctrica para su operación.

II.1.2. Retos de las redes inalámbricas de monitoreo y control

Debido a las limitaciones características de los nodos, las cuales van desde la velocidad del procesador hasta el tiempo de vida de las baterías, las redes inalámbricas de sensores cuentan con retos particulares. En la literatura se mencionan los siguientes retos de (Stankovic, 2004).

Protocolos del mundo real

Los nuevos protocolos de redes toman en cuenta los requerimientos de las realidades en las comunicaciones inalámbricas. Las nuevas investigaciones están enfocadas en:

- Medida y evaluación de las propiedades teóricas de las comunicaciones inalámbricas.
- Establecimiento de mejores modelos de comunicación reales para retroalimentar a las herramientas de simulación.
- Invención de protocolos de red que tomen en cuenta las características de las comunicaciones en entornos del mundo real.
- Probar soluciones individuales en plataformas del mundo real.
- Sintetizar las nuevas soluciones en pilas de protocolos para aplicaciones reales.

Tiempo real

Las redes de sensores funcionan en ambientes del mundo real, en muchas ocasiones, los datos del sensor deben enviarse con restricciones de tiempo a fin de que puedan hacerse observaciones adecuadamente, y a partir de éstas tomar las medidas pertinentes.

Hasta la fecha existen muy pocos resultados sobre los requisitos de los sistemas en tiempo real con redes de sensores.

Administración de energía

Las limitantes en cuanto a procesador y memoria son dos restricciones que desaparecerán con el desarrollo de las técnicas de fabricación. Pero, las limitaciones de energía no tienen las mismas posibilidades, debido a que el desarrollo de las capacidades de las baterías está progresando de manera muy lenta.

Abstracción de programación

Un factor clave para el crecimiento de las redes de sensores es el proveer un nivel de abstracción para programadores. Actualmente, los programadores interactúan con los sensores a un nivel muy bajo para sensado y comunicación.

La actual investigación puede ser clasificada en siete áreas: ambientes, APIs de *middleware*, bases de datos céntricas, basadas en eventos, máquinas virtuales, scripts y basados en componentes.

Seguridad y privacidad

En contraste con las redes tradicionales, los nodos de las redes de sensores, en ocasiones están localizados en lugares extremadamente accesibles, por lo tanto se encuentran expuestas a riesgos de ataques físicos. Esto se debe a la interacción tan estrecha que tienen con el ambiente físico y con las personas, lo que conlleva problemas adicionales.

Análisis

Debido a que las redes de sensores están en una etapa relativamente inicial, no es de sorprenderse que existan muy pocos análisis de resultados. Los investigadores están tan ocupados en la creación de nuevos protocolos y aplicaciones para las redes de sensores,

que construyen una solución; la prueban y la evalúan, y toda esa información queda almacenada, de cierta manera, empíricamente. Es por esto, que se requiere utilizar el método científico para diseñar y analizar los sistemas antes de ponerlos en marcha.

II.1.3. Aplicaciones

Las aplicaciones que utilizan redes de monitoreo y control se pueden clasificar de diferentes maneras:

- Una de ellas es por la extensión donde se encuentran emplazados los nodos de la red, en varios kilómetros como en monitoreo de hábitats o ambientes extensos; en decenas de metros como monitoreo y control industrial o sistemas de seguridad en hogares; o pocos metros como monitoreo de las funciones del cuerpo humano en pacientes médicos.
- Otra clasificación puede ser por sector, entre ellos industrial, logística, construcción, agricultura, medicina y cuidado de la salud, militar o arte.

Aunque este tipo de redes fueron inicialmente pensadas para funcionar en escenarios bélicos, con miles de nodos autónomos que se autoconfiguran y establecen redes sin intervención humana, en la actualidad los proyectos que han sido desarrollados y la mayoría de las aplicaciones realistas, únicamente necesitan unos cientos de nodos. Por ejemplo, Great Duck Island (Cerpa et al., 2001), creado en el año 2002, es un sistema desarrollado por la Universidad de Berkeley/Intel Research Laboratory para el monitoreo del ambiente, en el cual se utilizaron alrededor de 150 nodos que cubrían un área de 25 hectáreas.

Otro proyecto desarrollado es el llamado CORIE (CORIE, 2007), el cual monitorea el río Columbia e integra una red de sensores en tiempo real, haciendo uso de alrededor de 25 nodos.

LOFAR_Agro (Langendoen et al., 2006) es un proyecto en el que sus creadores dicen haber utilizado un gran número de nodos sensores, pero solo fueron alrededor de 150 y se utilizaron en el área de la agricultura para controlar plagas.

Algunos otros ejemplos de aplicación incluyen los siguientes:

- Control de luces.
- Detectores de humo y monóxido de carbono (CO).
- Control de calefacción.
- Control de ambientes artificiales de temperatura adaptable.
- Controles de persianas y cortinas.
- Juguetes y dispositivos de entretenimiento.
- Monitoreo médico.
- Monitoreo ambiental en exteriores.

Estas aplicaciones requieren de bajas tasas de transmisión, de algunos cuantos kilobits por segundo. Por la extensión de la red, pueden requerir de múltiples saltos para entregar un paquete de datos a su destino. La ubicación de los nodos para monitoreo y control utilizan baterías y requieren estar por largo tiempo desatendidos sin necesidad de cambiarlas o recargarlas si fuera el caso.

II.2. Protocolos

Las redes de monitoreo y control, al igual que otros tipos de redes de comunicación, utilizan diferentes protocolos para entenderse. Los dispositivos que conectamos en red hoy en día son muy heterogéneos, impidiendo que puedan ser interconectados haciendo uso

solo de *hardware*. Por ello se requiere el uso de software y *hardware* un tanto genérico.

II.2.1. Modelo de capas

Con el objetivo de hacer esta interconexión por medio de software, se han realizado diferentes trabajos para organizarlo de manera tal que el diseño de nuevo software y *hardware* se haga de una forma más fácil. Para lograr esto, los sistemas de red están organizadas en capas, donde cada una está arriba de otra, formando una pila de capas, dependiendo del tipo de red que se esté utilizando será el conjunto de capas que se utilicen.

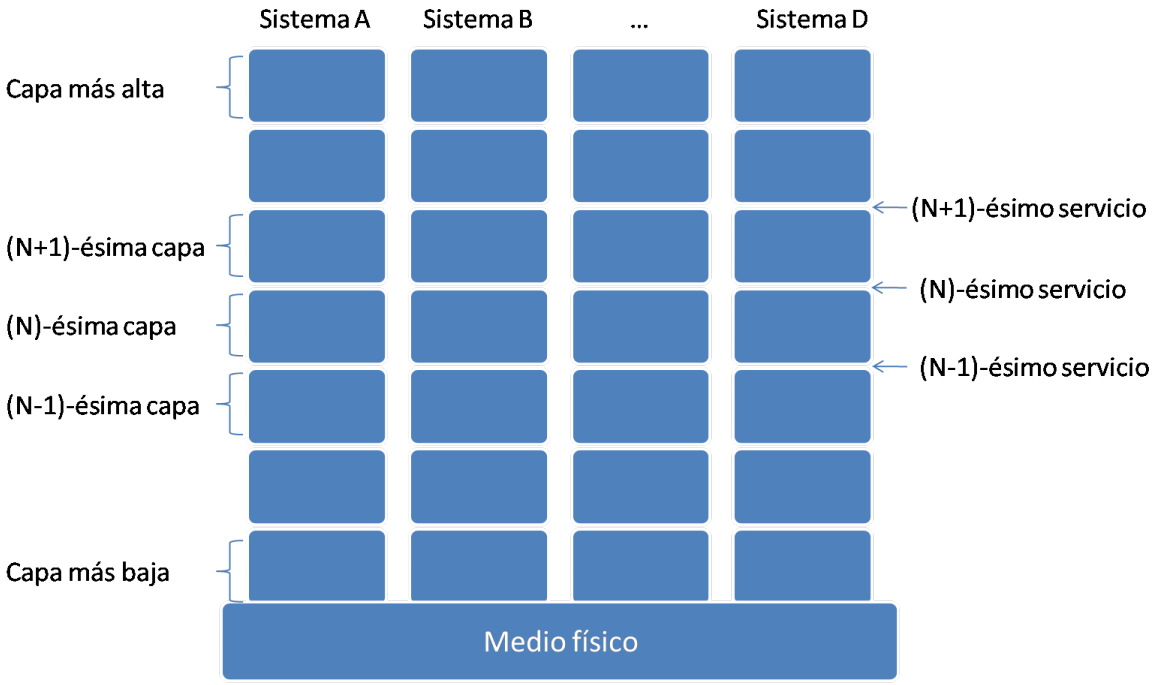


Figura 2: **Sistemas, capas y servicios**

El propósito de cada capa es brindar servicios a la capa que se encuentra directamente arriba de ella, de esta manera se encapsula la complejidad de cada uno de los servicios, la capa superior hace una petición de un servicio y la capa inferior lo satisface. En la Figura 2 se muestra cómo están organizadas conceptualmente las capas.

La capa *N* de un dispositivo se comunica con la capa *N* en otro haciendo uso de un lenguaje conocido como *protocolo*. Un protocolo es un acuerdo entre las partes que se estén comunicando que especifica cómo debe ser el proceso de comunicación.

A un conjunto de capas y protocolos se le conoce como arquitectura de red, la especificación de una arquitectura debe contener la suficiente información para permitir que alguien sea capaz de hacer un software o hardware que trabaje en alguna capa con el protocolo apropiado. La arquitectura de la red no especifica cómo es la implementación ni las interfaces, ya que no es necesario que las interfaces de todos los dispositivos de la red sean iguales, sólo es importante que cada capa haga un uso adecuado del protocolo establecido. A la lista de protocolos usados en cierto sistema, donde cada capa utiliza un protocolo se le conoce como *pila de protocolos*.

Cuando una capa de un dispositivo se comunica con la capa del mismo nivel en otro no lo hace directamente, sino a través de la capa que se encuentra debajo de ella y tampoco lo hace porque ella quiera comunicarse sino porque la capa superior se lo ha pedido. A excepción de la capa de aplicación, la cual es la única que inicia la comunicación por ser la más alta del modelo, por otro lado, la capa más baja que es la capa física si se comunican directamente. La Figura 3 muestra la forma en que se comunican capas del mismo nivel.

Debido a que cada capa le solicita servicios a la capa inferior, la capa superior manda información que la capa inferior toma como datos y agrega un encabezado o la encapsula para poder transmitirla con el protocolo adecuado de esa capa, a su vez la capa inferior se la pasa a la capa que está debajo de ella y se repite el mismo procedimiento. Por ejemplo si se utiliza un modelo de 5 capas, la capa de aplicación en la capa 5 produce un mensaje y se lo pasa a la capa 4 para que lo transmita, la capa 4 le agrega un encabezado al inicio del mensaje para identificarlo y permitir que la capa 4 de la máquina remota lleve la secuencia de

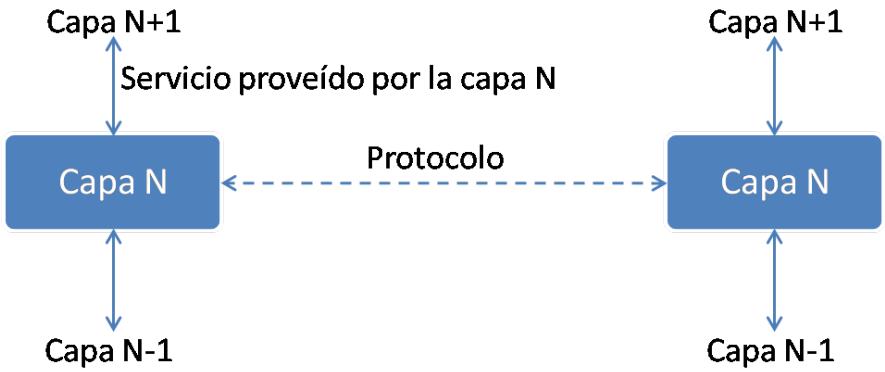


Figura 3: **Relacion entre un servicio y un protocolo.**

los paquetes recibidos y si le falta alguno entonces haga la petición, posteriormente la capa 4 se lo pasa a la capa 3 la cual divide el mensaje en un tamaño preestablecido según sea el protocolo que se esté utilizando en la capa 3 y además agrega a cada parte un encabezado. Después la capa 3 envía cada uno de los segmentos a la capa 2 que agrega información al inicio y al final del paquete y se lo envía a la capa 1 para la transmisión física. Del otro lado, en la máquina receptora, cada capa va desempaquetando los mensajes quitándole la información que le agregó su contraparte en la máquina transmisora y únicamente pasa la información que recibió la capa de su mismo nivel. Entre la información que cada capa le agrega al mensaje se encuentra el direccionamiento, el control de errores (ya que los medios físicos no garantizan que lleguen al otro lado) y el control de flujo.

Las capas pueden ofrecer dos tipos de servicios: orientados a conexión y no orientados a conexión. Los orientados a conexión primeramente establecen un canal por donde se hará la comunicación, después se transmiten los mensajes de la información y por último se libera el canal, mientras que en los no orientados a conexión cada mensaje es enviado por diferentes canales o caminos y puede ser que no lleguen en el orden que fueron transmitidos, entonces aquí es donde interviene el protocolo que se esté usando en cada capa para garantizar que todos los datos que fueron enviados sean los que recibió la máquina remota.

II.2.2. Tareas de las capas

Existen diferentes caracterizaciones de las capas de los sistemas de comunicación, una de las más utilizadas es la de TCP/IP, que cuenta con 5 capas, aplicación, transporte, red, enlace de datos y física, pero el modelo de referencia es el de Interconexión de Sistemas Abiertos (*OSI, Open System Interconnection*) lanzado en 1984 por Organización Internacional para la Estandarización (*ISO, International Organization for Standardization*). El modelo OSI define 7 capas, aplicación, presentación, sesión, transporte, red, enlace de datos y nivel físico (ISO/IEC7498-1, 1994). Aunque este modelo no es muy reciente, las tareas que indica que realiza cada capa se siguen manteniendo como referencia hasta la actualidad.

Capa de aplicación

Ofrece a las aplicaciones (de usuario o no) la posibilidad de acceder a los servicios de las demás capas y define los protocolos que utilizan las aplicaciones para intercambiar datos, como correo electrónico (POP y SMTP), gestores de bases de datos y servidor de ficheros (FTP). Hay tantos protocolos como aplicaciones distintas y puesto que continuamente se desarrollan nuevas aplicaciones el número de protocolos crece sin parar.

Cabe aclarar que el usuario normalmente no interactúa directamente con el nivel de aplicación. Suele interactuar con programas que a su vez interactúan con el nivel de aplicación pero ocultando la complejidad subyacente. Así por ejemplo un usuario no manda una petición ”HTTP/1.0 GET index.html”para conseguir una página en html, ni lee directamente el código html/xml.

Capa de presentación

El objetivo de la capa de presentación es encargarse de la representación de la información, de manera que aunque distintos equipos puedan tener diferentes representaciones

internas de caracteres (ASCII, Unicode, EBCDIC), números (*little-endian* tipo Intel, *big-endian* tipo Motorola), sonido o imágenes, los datos lleguen de manera reconocible.

Esta capa es la primera en trabajar más el contenido de la comunicación que en cómo se establece la misma. En ella se tratan aspectos tales como la semántica y la sintaxis de los datos transmitidos, ya que distintas computadoras pueden tener diferentes formas de manejarlas. La capa de presentación también permite cifrar los datos y comprimirlos para ocupar menos ancho de banda. En pocas palabras es un traductor.

Capa de sesión

Esta capa establece, gestiona y finaliza las conexiones entre usuarios (procesos o aplicaciones) finales. Ofrece varios servicios que son cruciales para la comunicación, como son:

- Control de la sesión a establecer entre el emisor y el receptor.
- Control de la concurrencia.
- Mantener puntos de verificación (*checkpoints*), que sirven para que, ante una interrupción de transmisión por cualquier causa, la misma se pueda reanudar desde el último punto de verificación en lugar de repetirla desde el principio.

El servicio provisto por esta capa es la capacidad de asegurar que, dada una sesión establecida entre dos máquinas, la misma se pueda efectuar para las operaciones definidas de principio a fin, reanudándolas en caso de interrupción. En muchos casos, los servicios de la capa de sesión son parcialmente, o incluso, totalmente prescindibles.

Capa de transporte

Su función básica es aceptar los datos enviados por las capas superiores, dividirlos en pequeñas partes si es necesario, y pasarlos a la capa de red. En el caso del modelo OSI, también se asegura que lleguen correctamente al otro lado de la comunicación. Otra característica a destacar es que debe aislar a las capas superiores de las distintas posibles implementaciones de tecnologías de red en las capas inferiores, lo que la convierte en el corazón de la comunicación. En esta capa se proveen servicios de conexión para la capa de sesión que serán utilizados finalmente por los usuarios de la red al enviar y recibir paquetes. Estos servicios estarán asociados al tipo de comunicación empleada, la cual puede ser diferente según el requerimiento que se le haga a la capa de transporte. Por ejemplo, la comunicación puede ser manejada para que los paquetes sean entregados en el orden exacto en que se enviaron, asegurando una comunicación punto a punto libre de errores, o sin tener en cuenta el orden de envío. Una de las dos modalidades debe establecerse antes de comenzar la comunicación para que una sesión determinada envíe paquetes, y ése será el tipo de servicio brindado por la capa de transporte hasta que la sesión finalice.

Capa de red

La capa de red, según la normalización OSI, es una capa que proporciona conectividad y selección de ruta en la red para interconectar dos dispositivos. Es el tercer nivel del modelo OSI y su misión es conseguir que los datos lleguen desde el origen al destino aunque no tengan conexión directa. Ofrece servicios al nivel superior (nivel de transporte) y se apoya en el nivel de enlace, es decir, utiliza sus funciones. Para realizar su tarea, puede asignar direcciones de red únicas, interconectar subredes distintas, encaminar paquetes y utilizar un control de congestión.

Capa de enlace de datos

El nivel de enlace es el segundo del modelo OSI. Recibe peticiones del nivel de red y utiliza los servicios del nivel físico. El objetivo del nivel de enlace es conseguir que la información fluya, libre de errores, entre dos máquinas que estén conectadas directamente (servicio orientado a conexión). Para lograr este objetivo tiene que montar bloques de información (llamados tramas en este nivel), dotarles de una dirección de nivel de enlace, gestionar la detección o corrección de errores, y ocuparse del control de flujo entre equipos (para evitar que un equipo más rápido desborde a uno más lento).

Capa de física

La capa física del modelo de referencia OSI es la que se encarga de las conexiones físicas de la computadora hacia la red, tanto en lo que se refiere al medio físico (medios guiados: cable coaxial, cable de par trenzado, fibra óptica y otros tipos de cables; medios no guiados: radio, infrarrojos, microondas, láser y otras redes inalámbricas); características del medio (por ejemplo, tipo de cable o calidad del mismo; tipo de conectores normalizados o en su caso tipo de antena; etc.) y la forma en la que se transmite la información (codificación de señal, niveles de tensión/intensidad de corriente eléctrica, modulación, tasa binaria, etc.).

Es la encargada de transmitir los bits de información a través del medio utilizado para la transmisión. Se ocupa de las propiedades físicas y características eléctricas de los diversos componentes; de la velocidad de transmisión, si ésta es uni o bidireccional (*símplex*, *dúplex* o *full-dúplex*). También de aspectos mecánicos de las conexiones y terminales, incluyendo la interpretación de las señales eléctricas/electromagnéticas.

Se encarga de transformar una trama de datos proveniente del nivel de enlace en una señal adecuada al medio físico utilizado en la transmisión. Estos impulsos pueden ser

eléctricos (transmisión por cable) o electromagnéticos (transmisión sin cables). Estos últimos, dependiendo de la frecuencia/longitud de onda de la señal pueden ser ópticos, de micro-ondas o de radio. Cuando actúa en modo recepción el trabajo es inverso; se encarga de transformar la señal transmitida en tramas de datos binarios que serán entregados al nivel de enlace.

II.2.3. Conclusión

Las redes de monitoreo y control cuentan con características peculiares, debido a que los dispositivos de estas cuentan con capacidades más limitadas que las computadoras comunes, como memoria, procesador y principalmente energía.

Por ello se requiere desarrollar protocolos de comunicación que se adapten y utilicen los recursos de una manera adecuada. En el presente trabajo se describe un protocolo de capa de red para este tipo de sistemas, para ello se toma como base otros protocolos de capa física y de enlace de datos, estos son los definidos por el estándar IEEE 802.15.4.

II.3. El estándar IEEE 802.15.4

Por las aplicaciones potenciales a las que está dirigido el protocolo que se presenta, como sensores, juguetes interactivos, controles remotos, domótica o que comparten las características como baja transmisión de datos (pocos *kb/s*), distancia entre los dispositivos no muy extensas y sobretodo *hardware* de bajo costo, se optó por utilizar el estándar IEEE 802.15.4, el cual está diseñado para dispositivos con esas características y además proporciona las ventajas de ser un estándar.

El estándar IEEE 802.15.4 define el protocolo e interconexión de dispositivos con comunicación vía radio en redes de área personal (*Personal Area Network*, *PAN*). Este estándar

utiliza *CSMA-CA* (*Carrier Sense Multiple Access with Collission Avoidance*), el cual es un protocolo de control de acceso al medio de redes, utilizado para evitar colisiones entre los paquetes de datos soportando topologías de tipo estrella, así como de tipo punto a punto.

Los principales problemas a resolver de las redes de monitoreo y control, se encuentran en los protocolos de comunicación, porque es en la transmisión de la información donde se consume gran parte de la energía, limitando en cierta medida la posibilidad de tener algún tipo de procesamiento adicional en los nodos.

II.3.1. Descripción general

El propósito del estándar es definir los niveles de red básicos para dar servicio a un tipo específico de red inalámbrica de área personal (*Wireless Personal Area Network* o WPAN) centrada en la habilitación de comunicación entre dispositivos ubicuos con bajo costo y velocidad (en contraste con esfuerzos orientados directamente a los usuarios medios, como WiFi). Se enfatiza el bajo costo de comunicación con nodos cercanos y sin infraestructura o con muy poca, para favorecer aún más el bajo consumo.

En su forma básica se concibe un área de comunicación de 10 metros con una tasa de transferencia de 250 kbps. Se puede reducir el consumo de energía para sistemas empotrados que así lo requieran. Para ello, se definen no uno, sino varios niveles físicos. Se definieron inicialmente tasas alternativas de 20 y 40 kbps; la versión actual añade una tasa adicional de 100 kbps. Como se ha indicado, la característica fundamental de 802.15.4 entre las WPAN's, es la obtención de costos de fabricación excepcionalmente bajos, sin perjuicio de la generalidad o la adaptabilidad.

Entre los aspectos más importantes se encuentra la adecuación de su uso para tiempo real por medio de *slots* de tiempo garantizados, prevención de colisiones por CSMA/CA

(*Carrier Sense Multiple Access/Collision Avoidance*) y soporte integrado a las comunicaciones seguras. También se incluyen funciones de control del consumo de energía como calidad del enlace y detección de energía Un dispositivo que implementa el 802.15.4 puede transmitir en una de tres posibles bandas de frecuencia, 868-868.8 MHz para su uso en Europa, 902-928 MHz para uso en Norte América y 2400-2483.5 MHz para todo el mundo.

Topología

El estándar define dos tipos de nodo en la red. El primero, es el dispositivo de funcionalidad completa (*Full-Function Device, FFD*). Puede funcionar como coordinador de una red de área personal (PAN) o como un nodo normal. Implementa un modelo general de comunicación que le permite establecer un intercambio con cualquier otro dispositivo. Además puede encaminar mensajes, en cuyo caso se le denomina coordinador (coordinador de la PAN si es el responsable de toda la red y no sólo de su entorno). El segundo tipo son los dispositivos de funcionalidad reducida (*Reduced-Function device*, *RFD*), se plantean como dispositivos muy sencillos con recursos y necesidades de comunicación limitadas, por ello sólo pueden comunicarse con FFDs y nunca pueden ser coordinadores.

Las redes de nodos pueden construirse como redes punto a punto o en estrella. En cualquier caso, toda red necesita al menos un FFD que actúe como su coordinador. Las redes están compuestas por grupos de dispositivos separados por distancias suficientemente reducidas; cada dispositivo posee un identificador único de 64 bits, aunque si se dan ciertas condiciones de entorno en éste pueden usarse identificadores cortos de 16 bits. Probablemente éstos se utilizarán dentro del dominio de cada PAN separada.

Las redes punto a punto pueden formar patrones arbitrarios de conexionado y forman la base de redes *ad hoc* autoorganizadas, su extensión está limitada únicamente por la distancia existente entre cada par de nodos. El estándar no define un nivel de red, por lo que

no contiene funciones de ruteo que permitian comunicaciones a través de múltiples saltos, esto lo definiremos en nuestro protocolo de red. Pueden imponerse otras restricciones topológicas; en concreto, el estándar menciona el árbol de *clusters* como una estructura que aprovecha que los RFD's sólo pueden conectarse con un FFD al tiempo para formar redes en las que los RFD's son siempre hojas del árbol, y donde la mayoría de los nodos son FFD's. Puede relajarse la estructura para formar redes en malla genéricas, cuyos nodos sean árboles de *clusters* con un coordinador local para cada *cluster*, junto con un coordinador global.

También pueden formarse redes en estrella, en las que el coordinador va a ser siempre el nodo central. Una red así se forma cuando un FFD decide crear su PAN y se nombra a sí mismo coordinador, después de elegir un identificador de PAN único. Una vez seleccionado, otros dispositivos pueden unirse a la red sin necesidad de comunicarse con los otros nodos de la red a la que se quiere unir u otras existentes en el área. En la figura 4 podemos ver la topología punto a punto y estrella.

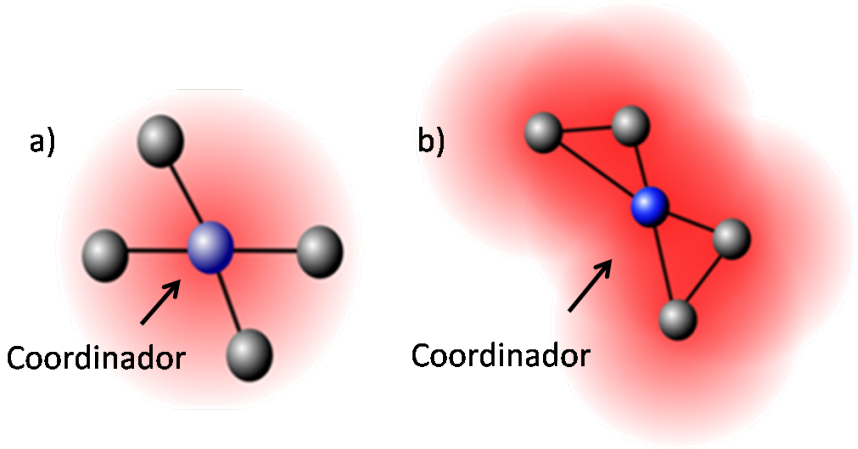


Figura 4: **Topologías de red a) red en estrella, b) red punto a punto.**

Modelo de transferencia de datos

El estándar define principalmente dos modelos de transferencia de datos, con *beacons* habilitados y sin ellos. *Beacon* o faro son un tipo de mensaje que envía el coordinador para sincronizar a los nodos que están asociados a su red.

En una red con *beacons* habilitados, los dispositivos asociados a la red esperan por el mensaje *beacon* para enviar datos al coordinador. Con este tipo de transferencia, se pueden destinar espacios de tiempo para que cierto nodo transmita, evitando competir por el medio de transmisión. También se puede destinar un espacio para que los nodos compitan para transmitir. De manera general, la comunicación con *beacons* habilitados se esquematiza en la Figura 5.

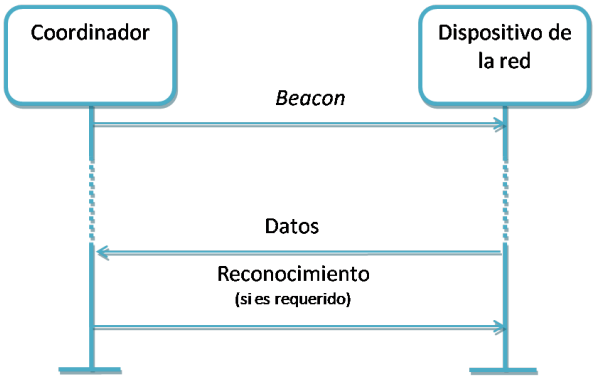


Figura 5: **Comunicación hacia el coordinador en una red con *beacons* habilitada.**

Mientras que en una red donde no hay *beacons*, la transmisión se puede dar en cualquier momento, sin necesidad de esperar por un mensaje *beacon*, pero haciendo imposible el destinar un espacio de tiempo garantizado para que un nodo transmita datos. En la Figura 6, se esquematiza este tipo de comunicación.

Cuando un coordinador requiere enviar datos a un dispositivo asociado a su red, debe

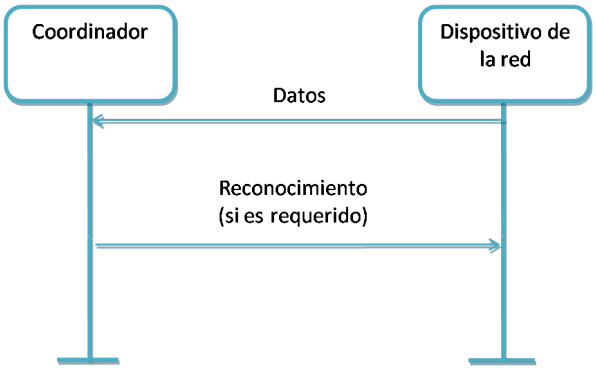


Figura 6: **Comunicación hacia el coordinador en una red sin *beacons* habilitada.**

guardar la información y esperar por un mensaje de petición de datos. Esto es porque algunos dispositivos apagan su unidad receptora y es imposible establecer comunicación con ellos. En la Figura 7 se ilustra el envío de datos desde un coordinador hacia un dispositivo asociado a él en redes con y sin *beacons* habilitados.

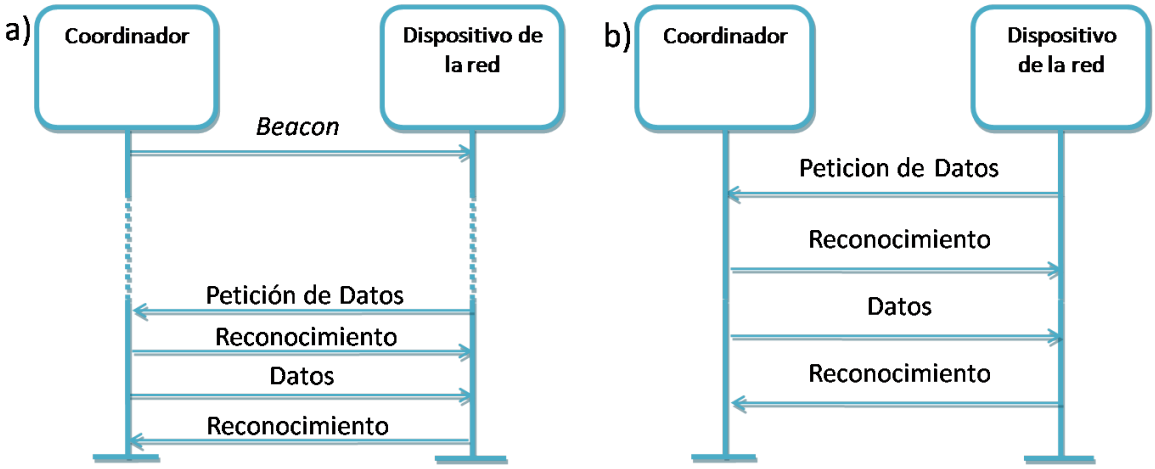


Figura 7: **a) Comunicación hacia un dispositivo en una red con *beacons* habilitada. b) Comunicación hacia un dispositivo en una red sin *beacons* habilitada.**

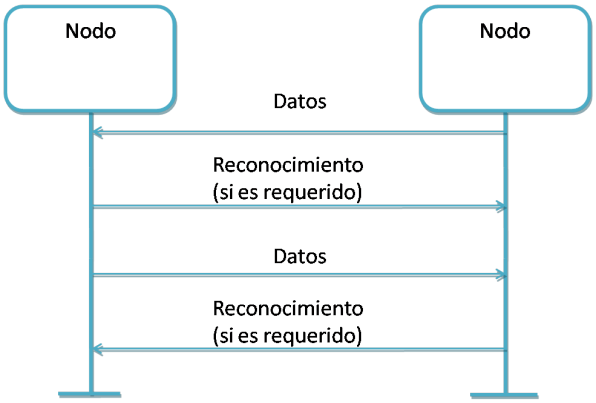


Figura 8: **Comunicación entre pares.**

Otro modelo de transferencia de datos *beaconless* es el utilizado en redes punto a punto, donde cualquier nodo en la red puede comunicarse con cualquier otro nodo dentro del alcance del radio, en la Figura 8 se representa el proceso. Para hacerlo eficientemente, los nodos que desean comunicarse deben recibir constantemente mensajes de sincronización o simplemente transmitir los datos utilizando CSMA/CA. Para este último caso, se deben utilizar mecanismos que se encuentran fuera del alcance del estándar IEEE 802.15.4.

En esta sección se ha mostrado de manera general los componentes y el funcionamiento del estándar IEEE 802.15.4 utilizado como base para el desarrollo del protocolo de red de este trabajo. Una de las características del estándar IEEE 802.15.4 es su flexibilidad para ser utilizado en distinto tipos de aplicaciones. Es por ello que se requiere establecer un marco de trabajo para delimitar una configuración del estándar IEEE 802.15.4 a utilizar. La configuración seleccionada es la siguiente:

- Se utiliza una configuración *beaconless* o sin *beacons* habilitados, donde un nodo central crea la red y los demás nodos se van agregando a ella.
- La comunicación entre los nodos es punto a punto para poder enviar mensajes por

múltiples saltos. El protocolo supone que todos los nodos son FFD y permiten la asociación de otros nodos a la red a través de ellos.

- Como el estándar IEEE 802.15.4 marca, se utilizan direcciones extendidas de 64 bits para la asociación de nodos a la red y después se hace uso de direcciones cortas en la capa de enlace de datos, las cuales son de 2 bytes y también son utilizadas en la capa de red para direccionamiento.
- Únicamente el nodo central o *sink* puede crear una red, seleccionando el canal para trabajar y el identificador único de esta. Los nodos dentro del alcance del radio del *sink* se unen a la red por medio de esté, si cuenta con direcciones para asignar, y permiten que otros nodos que se unan a través de ellos formando a la red.

II.4. Protocolos de red

En esta sección se mencionan algunos de los protocolos y sistemas de red existentes para redes inalámbricas de monitoreo y control y las características de estos.

II.4.1. SNAP

SNAP (*Synapse Network Application Protocol*) es un protocolo de red propiedad de Synapse, el cual utiliza únicamente la capa física del estándar IEEE 802.15.4 para redes de monitoreo y control (Synapse, 2007). Este protocolo forma redes automáticamente, puede ser controlado y modificado por medio de la herramienta de PC para la administración de redes Synapse Portal®, además puede configurar el comportamiento de los dispositivos sin necesidad de reprogramarlos. SNAP se comunica vía un puerto serial embebido, permitiendo fácilmente comunicar los dispositivos embebidos y las redes inalámbricas de monitoreo y control.

Entre las principales características de SNAP, se encuentra que utiliza solamente 16 kBytes de memoria para almacenar el código. Ademas puede ser configurado en el aire por medio de Synapse Portal, este soporta *scripts* que definen el comportamiento de los dispositivos y con esto la capa de aplicación no es requerida.

El tamaño pequeño de SNAP hace que pueda ser implementado en microcontroladores de bajo costo, aún utilizando encriptación AES128. Con la utilización de SNAP se requieren menos de 50μA en ciclos de bajo consumo. Soporta poco más de 65,000 dispositivos en una red y más de 65,000 redes en total. Puede emplear *beacons* y provee la capacidad de modificar la tasa de transmisión de estos, también se puede configurar como *beaconless* y crear redes con baja latencia.

SNAP es capaz de soportar repetidores, a través de una configuración especial con hasta cuatro dispositivos, colocados a 3 millas de distancia (la cual es el máximo alcance de los transmisores), se puede extender una red de 15 millas. Ademas cuenta con la capacidad de utilizar repetidores de respaldo, lo que permite obtener mayor grado de tolerancia a fallos.

La comunicación con los dispositivos que cuentan con SNAP se hace a través de comandos seriales AT, proporcionando así un nivel de abstracción más elevado para permitir a los usuarios enfocarse en las aplicaciones que estén desarrollando.

Las principales desventajas asociadas a las soluciones basadas en SNAP son que no se trata de un estándar abierto y tiene un costo el adquirirlo, el precio de cada modulo Synapse RF Engine con la pila SNAP precargada inicia de los 24 USD, además de necesitar comprar por separado el Synapse Portal en un precio de 249 USD[1].

[1] Obtenido de www.synapse-wireless.com en diciembre de 2007

II.4.2. PopNet

PopNet^{TM} es un protocolo de red con soporte a topologías de malla y también es un entorno operativo para trabajar con radios IEEE 802.15.4 y microcontroladores de bajo costo para aplicaciones de monitoreo y control.

El principal objetivo en el desarrollo de PopNet es hacerlo fácil de usar. PopNet incluye un pequeño conjunto de funciones que responden con códigos de estados. Todos los datos son manipulados a través de un único administrador de mensajes (San Juan Software, 2007).

Cualquier dispositivo que utilice PopNet puede iniciar una red, hacer funciones de enrutador para reenviar mensajes a otros dispositivos o "dormirse" para conservar energía y tener una vida más prolongada. No hay necesidad de utilizar dispositivos con capacidades diferentes como coordinadores, enrutadores y dispositivos finales, ya que todos pueden hacer cualquiera de las tareas de la capa de red.

No hay un punto crítico que fallé e inhabilite la red por completo, si un nodo que era utilizado para comunicarse desaparece de la red, PopNet encuentran nuevos caminos para establecer comunicación. Además de las funciones de protocolo de red, PopNet incluye funciones de acceso a los periféricos como *timers*, UART, LEDs, unidades de almacenamiento y otras interfaces.

PopNet puede ser ejecutado en dispositivos de Freescale con soporte a 802.15.4, incluyendo EVB (*Evaluation Board*), SARD (*Sensor Applications Reference Design*), SRB (*Sensor Remote Board*) y NCB (*Network Control Board*).

Gracias al tamaño de su código, una aplicación con PopNet puede ejecutarse con una pequeña memoria flash de 16K, 1K en RAM con un procesador de 8 bits. Pero también se

puede tomar ventaja de procesadores más poderosos y con recursos adicionales para utilizar encriptación AES de la capa de enlace de datos.

PopNet esta basado en la capa física que define el estándar IEEE 802.15.4, el cuál se en carga de enviar los datos por el aire asegurándose que no salieron corruptos por el radiotransmisor. PopNet provee un sistema de reintentos y reconocimientos a nivel de red para asegurar que el mensaje sea entregado a su destino. La topología en malla de PopNet permite tener caminos redundantes y asegurar que el mensaje llegue a su destino, incluso si uno de los caminos ya no se encuentra disponible.

En lo referente al consumo de energía, PopNet incluye un administrador de energía que le da la habilidad de dormir hasta que una interrupción o un *timer* despierte la unidad. Un nodo que funcione con baterias AAA o AA y no funcione como ruteador puede durar mas de un años funcionando.

PopNet no esta diseñado para interoperar con ZigBee, IEEE 802.15.4 o WiFi, pero puede coexistir en entornos de múltiples protocolos. A través de gateways, PopNet puede interactuar con otros estándares como ZigBee o IP. Esto permite que dispositivos de bajo costo sean integrados a redes más extensas y con dispositivos de precio más elevado.

El kit para desarrollo de PopNet tiene un costo de alrededor de 399 USD[2]. Incluye tres módulos RF con baterías, un cable para interactuar con la PC y el software de evaluación de PopNet. Los costos por licencias varían.

[2]Obtenido de www.sanjuansw.com/?p=15 en diciembre de 2007

II.4.3. 6LoWPAN

Este proyecto se enfoca en cómo poner los beneficios de IP, especialmente IPv6, en redes de dispositivos de bajas prestaciones, como las redes inalámbricas de monitoreo y control. 6LoWPAN es un estándar en desarrollo bajo el grupo de trabajo de la IETF (Internet Engineering Task Force) y esta diseñado para ser usado en pequeñas y micro redes de sensores.

IP siempre ha sido considerado el protocolo para redes de área local(*Local Area Network* o LAN) o redes de área amplia(*Wide Area Network* o WAN), donde se interconectan PCs y servidores. No se había pensado que fuera apropiado para redes de sensores o redes de área personal(*Personal Area Network* o PANs), debido a la percepción de ser muy pesado para las aplicaciones de este tipo de redes (Mulligan, 2007).

Recientemente se ha comenzado a replantear la posibilidad de utilizarlo para redes de bajas prestaciones y de esta manera se ha iniciado el desarrollo y estandarización de 6LoWPAN. La primicia de este grupo de desarrollo es "¿Porqué inventar un protocolo cuando tenemos IP?", 6LoWPAN define cómo montar IPv6 sobre radios con bajas tasas de transmisión, baja potencia y poco espacio de almacenamiento.

El grupo de trabajo 6LoWPAN escribió el documento que lleva por título *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals* (Kushalnagar et al., 2007) y recientemente publicado como el RFC 4919. En este RFC se describen las suposiciones que se tomarán en cuenta para montar IP sobre estas redes de bajas prestaciones, así como algunos de los problemas que conlleva el hacer esto y sobretodo delimita el alcance que tiene 6LoWPAN.

Entre las principales restricciones se encuentra la poca capacidad de los paquetes de la capa de enlace de datos que se utilizan por el estándar IEEE 802.15.4. Mientras el tamaño

máximo de un paquete IPv6 es de 1280 bytes, el utilizado por IEEE 802.15.4 es de 127 bytes menos los utilizados por la capa física y de enlace de datos.

Otro documento creado por este grupo de trabajo es el RFC 4944 y lleva por título *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*; en el se describe el formato de las tramas IPv6 y el método para formar las direcciones de enlace local que son utilizadas por IPv6 (Montenegro et al., 2007).

El escoger o desarrollar un protocolo de enrutamiento en malla no era uno de las metas originales de este grupo de trabajo, en lugar de esto se desarrolló un mecanismo de codificación en la capa 2 y 3 de la pila. De esto resulto un concepto donde se "paga" sólo por lo que se usa, es decir por el *overhead* de la cabecera y procesamiento extra. En lugar de tener una cabecera monolítica como en IPv4, se utiliza la misma técnica de IPv6 con encabezados apilados. De esta manera si un dispositivo esta enviando paquetes pequeños a otro nodo, no necesita enviar o "pagar" por los campos para encaminamiento en malla o para fragmentación de paquetes (Mulligan, 2007).

II.4.4. ZigBee

ZigBee es un protocolo de comunicaciones inalámbrico, similar al Bluetooth, pero para redes de bajo consumo y utiliza como base el estándar IEEE 802.15.4 para redes inalámbricas de área personal (*Wireless Personal Area Networks*, WPANs) . Surge del fruto de una alianza de más de 200 empresas, la mayoría de ellas fabricantes de semiconductores, con el objetivo de conseguir el desarrollo e implantación de una tecnología inalámbrica de bajo costo.

Destacan empresas como Invensys, Mitsubishi, Honeywell, Philips y Motorola, que trabajan para crear un sistema estándar de comunicaciones, vía radio y bidireccional. Los

miembros de esta alianza justifican el desarrollo de este estándar para cubrir el vacío que se produce por debajo del Bluetooth, refiriéndose a consumo de energía y tasa de transmisión (ZigBee Alliance, 2007). El ámbito principal de uso es en aplicaciones que requieren bajas tasas de transmisión de datos y bajo consumo de energía, por ejemplo, en control industrial, colección de datos médicos, en domótica, automatización de edificios, etc..

II.4.5. TinyOS

TinyOS es un sistema operativo *open source* basado en componentes para redes inalámbricas de sensores . Está escrito en el lenguaje de programación nesC como un conjunto de tareas y procesos que colaboran entre sí. Está diseñado para incorporar nuevas innovaciones rápidamente y para funcionar bajo las importantes restricciones de memoria que se dan en las redes de sensores. TinyOS está desarrollado por un consorcio liderado por la Universidad de California en Berkley en cooperación con Intel Research (TinyOS, 2007).

Las aplicaciones para TinyOS se escriben en nesC, un dialecto del lenguaje de programación C optimizado para las limitaciones de memoria de las redes de sensores. Existen varias herramientas que completan y facilitan su uso, escritas en su mayoría en Java y en Bash. Otras herramientas y librerías asociadas están escritas principalmente en C.

TinyOS proporciona interfaces, módulos y configuraciones específicas, que permiten a los programadores construir programas como una serie de módulos que hacen tareas específicas. Los módulos de TinyOS proporcionan interfaces para los tipos estándar de entradas y salidas de hardware y sensores.

II.4.6. JenNet

JenNet es una pila de protocolos diseñado para trabajar con la familia de microcontroladores y modulos inálambricos. JenNet permite construir fácilmente redes de sensores inálambricas, desde una red *punto a punto* para reemplazo de cables hasta complejas redes en árbol (Jennic, 2008).

JenNet soporta topologías de red tipo estrella, árbol y lineal. También provee funcionalidad de autoreparación para ofrecer comunicación robusta. Está desarrollado sobre el estándar IEEE802.15.4, por lo tanto puede coexistir con WiFi y Bluetooth.

La interacción con JenNet es lograda a través de Jenie y AT-Jenie: Jenie es una interfaz de programacion en C, mientras que AT-Jenie es una API(*Application Programing Interface*) que puede ser embebida, accesada a través de la unidad UART (*Universal Asynchronous Receiver-Transmitter*) para aplicaciones de seguridad y pasado a través de la capa de red entre los dispositivos. Estas interfaces proveen acceso simplificado a toda la red y a las funciones de los microcontroladores.

II.4.7. Synkro

El protocolo de red Synkro, desarrollado por Freescale Semiconductor, ha sido ofrecido como una especificación abierta para liderar la industria de los electrónicos. Freescale planea hacerlo disponible a la industria a través de un foro abierto y global de diseñadores de estándares para permitir una amplia adopción de esta tecnología (Freescale, 2008).

Synko es un software protocolo de red escrito sobre el estándar IEEE 802.15.4 para ser usado en el diseño de productos de entretenimiento para el hogar, como televisiones digitales, reproductores de DVD, receptores de audio y video, estaciones de acoplamiento y controles remoto.

El protocolo Synkro provee un camino para migración de *software* y *hardware* a futuros productos diseñados para revolucionar la manera en la que los consumidores controlan los dispositivos de entretenimiento para el hogar. La capa de red para control de dispositivos de entretenimiento fue desarrollada con los consumidores en mente. Esta tecnología se ocupa de una preocupación creciente entre fabricantes de electrónica de hoy en día: las limitaciones tecnológicas y las interrupciones causadas por soluciones basadas en infrarrojo que no son compatibles con los planes de trabajo de los productos. Esta plataforma permite opciones avanzadas de control, como comunicación en dos vías entre dispositivos de entretenimiento. Elimina la necesidad de tener línea de vista entre los dispositivo de control y provee la bases para un control remoto universal para cualquier componente electrónico.

II.4.8. MiWi

MiWi es un protocolo inalámbrico, para redes de área personal basado en el estándar IEEE 802.15.4 que engloba las redes de área personal con bajas tasas de transmisión de datos. Está dirigido a dispositivos y redes de bajo coste, que no necesitan una alta transferencia de datos (250Kbit/s), a corta distancia (100 metros sin obstáculos), y con un consumo energético mínimo.

Desarrollado por Microchip Technology (Microchip Technology, 2008), MiWi se trata de un proyecto de código abierto, con la única limitación de utilizar microcontroladores de Microchip Technology y su transceptor MRF24J4. Al ser de código abierto, no necesita licencias de pago, ofreciendo así un coste menor y facilidad de implementación. Está dentro del espectro de los 2,4Ghz a través de su transceptor MRF4J4 y es compatible con todos los dispositivos IEEE 802.15.4.

Es capaz de formar redes punto a punto, de estrella o malla, según IEEE 802.15.4. Tiene una capacidad de infraestructura limitado en número de nodos; puede tener 8 coordinadores como máximo por red, y cada uno de éstos 127 hijos, en total 1024 nodos como máximo por red, frente a más de 65,000 de ZigBee.

II.4.9. Discusión

Aunque ZigBee es un estándar altamente utilizado, cuenta con características que agregan complejidad no necesaria para un gran número de las aplicaciones de redes de sensores; entre estas se encuentran las capas de abstracción y la sobrecarga de comandos de administración de la red.

El proyecto 6LowPAN, toma como primicia que se requiere comunicar desde cualquier computadora en Internet a un nodo específico por medio del protocolo IPv6. Se utiliza un *gateway* que proporciona la conectividad de una red de sensores inalámbrica a Internet. Este *gateway* trabaja desde la capa física hasta la de red, porque no todos los mensajes de red de 6LoWPAN pueden ser entendidos por un dispositivo que cumpla con el protocolo IPv6, esto es debido a la compresión de ciertos mensajes para la utilización en redes de bajas prestaciones. Por otro lado, se requiere de un dispositivo que limite el acceso a los nodos en la red, porque no es muy conveniente que se haga una cantidad de preguntas sin restricciones a un nodo y esto provoque que el tiempo de vida de la red disminuya.

En cuanto a los protocolos JenNet, SNAP y PopNet, tienen la limitante de no ser un estándar abierto y al igual que ZigBee se debe pagar una cantidad por licencias. Por otro lado, MiWi es abierto pero su uso se restringe a *hardware* de Microchip Technology Inc.

TinyOS es una solución para redes inalámbicas de monitoreo y control ampliamente utilizada en el ámbito académico. Una de las principales limitantes para la utilización de

TinyOS es el uso del lenguaje de programación nesC, un lenguaje orientado a componentes que requiere especial atención para su aprendizaje.

Es por esto que se plantea el desarrollo de un protocolo de red para redes inalámbricas de monitoreo y control, que utilicen como base el estándar IEEE 802.15.4 por ser un estándar altamente aceptado. El protocolo debe incluir capacidad de comunicación a gran alcance a través de múltiples saltos. También debe proveer una interfaz de desarrollo que permita su utilización. Este protocolo debe utilizar un pequeño espacio de memoria para almacenamiento de programas y procesadores de bajo desempeño, estas características disminuirán sin duda los costos. Además, se enfoca en redes personales y de uso doméstico que conecte alrededor de 1000 nodos.

Capítulo III

Protocolo de red

El protocolo de capa de red como lo específica el modelo de referencia *OSI*, es el encargado de proveer comunicación entre cualquiera de los dispositivos conectados a la red, por ello debe establecer un esquema de direccionamiento para identificar a los nodos y encaminar los paquetes a través de múltiples saltos si es necesario para entregar los datos al nodo correcto.

III.1. Descripción general

El protocolo de red soporta una topología tipo árbol, en la cual el nodo central es el responsable de iniciar la red. Los demás dispositivos unidos a la red deben tener la capacidad de aceptar que otros dispositivos se unan a la red a través de ellos, lo que significa que en algún momento tendrán la necesidad de enviar datos y estos deben ser capaces de encaminarlos para hacerlos llegar a su destino final. La red emplea una comunicación *beaconless* por lo que no es necesario esperar para poder enviar mensajes, todos los nodos de la red son FFD y permiten comunicarse en cualquier momento con los nodos dentro del alcance de su radio, proporcionando una comunicación punto a punto. Esta especificación describe sólo comunicaciones en redes intra-PAN, esto es, comunicación que se inician y

terminan dentro de la misma red.

III.2. Arquitectura

Se utiliza el modelo de arquitectura de capas, donde la capa superior (capa de aplicación o capa de transporte) obtiene servicios de envío y recepción de datos de la capa de red. Es decir, esta última se encarga de obtener los mensajes de la red destinados para el nodo y entregar a la capa superior los datos recibidos, también es responsable de enviar los datos por la red y entregarlos a la capa superior de otro nodo cuando este los requiera. La Figura 9 muestra la pila del protocolo ligero de red con los puntos de acceso.

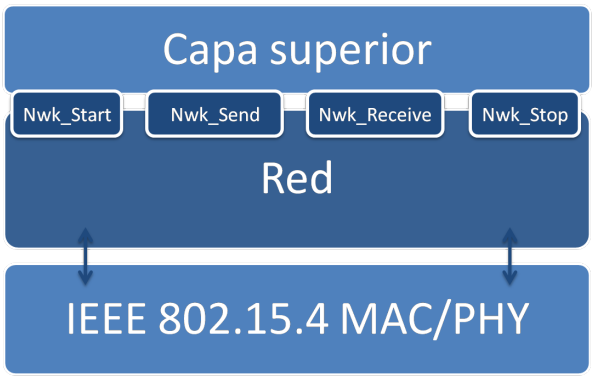


Figura 9: **Arquitectura del protocolo y puntos de acceso.**

La interacción con la capa de red se lleva a cabo a través de los puntos de acceso proporcionados por un conjunto de primitivas que conforman el API (Application Programing Interface). Las primitivas son:

- **Nwk_Start** Indica a la capa de red que inicie la búsqueda de dispositivos para unirse a ellos y permitir la posterior utilización de la capa de red.
- **Nwk_Send** Envía a la capa de red los datos que se requieren transportar por la red.

- **Nwk_Receive** Obtiene los datos recibidos de la red que llevan como destino el nodo.
- **Nwk_Stop** Detiene el motor de la capa de red.

III.3. Topología de direccionamiento

III.3.1. Selección de topología

En la literatura podemos encontrar distintos tipos de topologías, en árboles, mallas o *clusters* como se muestra en (YC.˜Chang, 2006). En (Cuomo et al., 2007)se menciona que una topología tipo árbol reduce considerablemente el costo de descubrimiento de rutas que tiene la topología en malla, pero sus estudios y análisis se reducen a aplicaciones donde se cuenta con un nodo central *sink* y los nodos siempre requieren enviar mensajes al *sink*. Este protocolo no es útil para las aplicaciones que hemos definido, ya que se requiere que el *sink* pueda enviar datos a un nodo específico en la red, pero proporciona una buena base para desarrollar nuestro protocolo de red.

La utilización del estándar IEEE 802.15.4 (2003), permite partir de una base para el desarrollo del protocolo de red. Las asociaciones realizadas por el estándar IEEE 802.15.4 en la capa MAC son utilizadas para formar un árbol a nivel de capa de red para direccionamiento de los dispositivos y enrutamiento de datos.

Si bien una topología en malla proporciona caminos redundantes, para recuperar la conexión cuando por alguna razón se elimine un enlace en la ruta, en una topología en árbol se puede solucionar permitiendo que los nodos desconectados se asocien a otra parte del árbol. Lo que puede llevar tiempo extra, pero se reduce el tiempo de búsqueda de rutas y el espacio en memoria para almacenar rutas activas y demás estructuras de datos necesarias para ello.

III.3.2. Modelo de direccionamiento

Por las características de las aplicaciones mencionadas en secciones anteriores, se utiliza una topología en árbol donde las direcciones se reparten como sigue:

- Las direcciones están separadas lógicamente en bloques de 4 bits.
- Cada nodo asociado a la red obtiene su dirección y un bloque de direcciones que puede asignar a los nodos que se asocien a través de él a la red.
- Las direcciones se dividen en 14 subredes y cada nodo reparte las direcciones de esas subredes. Las cuales a su vez se reparten en 14 subredes.

Las direcciones tienen una longitud de 16 bits, donde cada nodo puede tener hasta 14 hijos asociados a la red a través de él. La separación a nivel de bits de la dirección de red se muestra en la Figura 10. El nodo *sink* tiene la dirección 0000, 0000, 0000, 0000 y la dirección de broadcast es la dirección 1111, 1111, 1111, 1111.

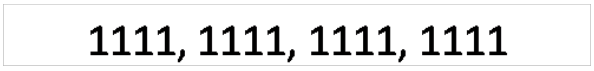


Figura 10: **Dirección de red en binario.**

A un nodo se le asigna una dirección y este puede utilizar direcciones en las que únicamente cambian los siguientes 4 bits (después del grupo diferente a cero) para direcciones de los dispositivos asociados a él. Por ejemplo, si a un nodo se le asignó la siguiente dirección 0001, 1100, 0000, 0000, este nodo puede asignar las direcciones que empiecen con 0001,1100 como 0001, 1100, 0001,0000 y este a su vez puede asignar direcciones que empiecen con 0001, 1100, 0001 como 0001, 1100, 0001, 0111.

En la siguiente figura se muestra el árbol de asociaciones lógicas, donde el *sink* tiene la dirección 0000, 0000, 0000, 0000.

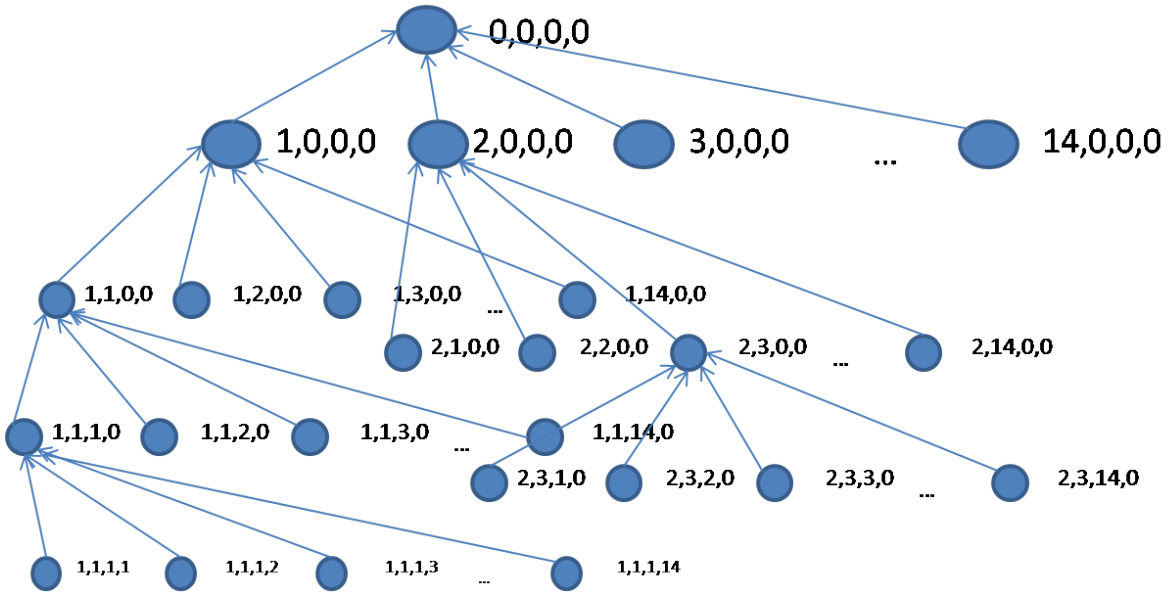


Figura 11: **Árbol de asociaciones lógicas y direcciones.**

Con este tipo de asignación de direccionamiento se cuenta con las siguientes características:

- Máximo 5 niveles en el árbol.
- Máximo 8 saltos para alcanzar cualquier nodo dentro de la red.
- Máximo 14 hijos por nodo.
- Máximo 41,356 nodos en la red.

Aunque el espacio de direccionamiento es de más de 41,000 direcciones, esto no quiere decir que sea factible conectar al mismo tiempo ese número de dispositivos, debido a la sobre carga que lleva el número de mensajes de control y el cuello de botella que se genera por la naturaleza de la topología.

III.4. Manejo de las funcionalidades de IEEE 802.15.4

La capa de red utiliza ciertos mecanismos provistos por el estándar IEEE 802.15.4 (2003) para poder realizar su tarea de intercomunicación de nodos que no están en el enlace directamente.

Una de estas funcionalidades es la detección de energía. Este provee un mapeo de los canales con la interferencia de otras redes u otros dispositivos en el área. Este es utilizado para seleccionar canales con poca interferencia, reduciendo la posibilidad de errores de comunicación.

Otra funcionalidad provista por el estándar IEEE 802.15.4 y utilizada por este protocolo de red es el escaneo activo. Este tipo de escaneo se utiliza para detectar a los nodos que aceptan asociaciones de otros dispositivos a la red. Con este tipo de escaneo se obtienen los nodos cercanos y la calidad de cada la señal de los enlaces. Una vez seleccionado el nodo con la mejor calidad de señal, se le envía una petición de asociación para que se le asigne al nuevo nodo una dirección corta de 16 bits.

Para el envío de los paquetes de red, el protocolo de red utiliza comunicaciones punto a punto provistas por IEEE 802.15.4. Esto se lleva a cabo encapsulando los paquetes de red en tramas de datos en la capa de enlace de datos.

El estándar IEEE 802.15.4 (2003) no define cómo son repartidas las direcciones cortas a los dispositivos de la red, sólo define los mensajes que contienen las direcciones para asignar a los dispositivos y que la capa superior a la capa MAC debe generar estos mensajes en respuesta a un mensaje de solicitud de asociación. En nuestro caso la capa de red es la que manipula a la capa MAC para su correcto funcionamiento.

III.5. Operaciones del protocolo

La operación de este protocolo se realiza por medio de tres tareas principales, asociación a la red, envío de mensajes de datos (*unicast* y *broadcast*) y mantenimiento de la conexión, mismos que serán descritos a continuación:

III.5.1. Asociación a la red

Únicamente un nodo inicia la red, el *sink*, y con esto todos los nodos que se agreguen tendrán el mismo identificador de red. Se sigue el procedimiento establecido en el estándar IEEE 802.15.4, un nodo busca redes dentro del rango de alcance de su radio a las que se pueda asociar y selecciona aquella con mejor calidad de señal. Le envía un mensaje MAC de asociación como se establece en IEEE 802.15.4, el nodo que recibe un mensaje de solicitud de asociación busca en su tabla de direcciones asignadas una libre y responde con un mensaje MAC de confirmación de asociación donde se especifica que se utilizará la dirección corta enviada en este mensaje para comunicación a nivel de enlace de datos. Esta dirección corta también es utilizada para direccionamiento y enrutamiento en la capa de red.

III.5.2. Envío y ruteo de mensajes de datos *unicast*

Una vez asociados los nodos, se puede enviar mensajes de datos entre cualquier par de nodos. Para esto se crean mensajes de datos con la dirección de red a la que se desea enviar los datos como la dirección destino y la dirección del nodo que genera el mensaje de datos como la dirección fuente. Con el mensaje creado se procede a enrutarlo por la red. Para esto, se compara la dirección destino con la dirección del nodo para verificar si la dirección pertenece a alguno de los subárboles por debajo del nodo fuente en el árbol de asociaciones. Si pertenece a alguno, se determina a qué nodo asociado al nodo fuente se debe enviar el paquete. Si el paquete no pertenece a algún nodo del subárbol con raíz en

el nodo fuente, el paquete es enviado al nodo que se utilizó para asociarse a la red, o nodo padre.

Cuando llega un mensaje de datos a un nodo, verifica si la dirección destino es la de él y si lo es pasa el mensaje al *buffer* de datos de entrada para que posteriormente sea recuperado por la capa de aplicación u otra capa superior. Si la dirección destino no es la del nodo al que llegó el paquete de datos, se determina si la dirección destino se encuentra en algún nodo del subárbol con raíz en el nodo que recibió el mensaje, si es así se envía el mensaje al nodo que es la raíz del subárbol que contiene el nodo destino. Si la dirección destino no está en el subárbol con raíz en el nodo que recibió el mensaje, este se envía al nodo que se utilizó para asociarse a la red. Este procedimiento se repite hasta alcanzar el nodo con la dirección igual a la del destino del paquete. Si no se encuentra el nodo con la dirección destino, el mensaje es eliminado.

III.5.3. Envío y ruteo de mensajes de datos *broadcast*

Los mensajes *broadcast* son enviados a través de los enlaces establecidos durante la asociación a la red hasta llegar a todos los nodos. Para esto, si un nodo desea enviar un mensaje *broadcast*, debe enviar un paquete de datos con su dirección como la dirección fuente y la dirección 0xFFFF como la dirección destino y los datos que desea enviar. La trama de enlace de datos enviada lleva la dirección destino 0xFFFF y la dirección fuente como la suya.

Cuando se recibe un mensaje de *broadcast* los nodos deben hacer una de las siguientes acciones:

- Si tiene dirección MAC de un nodo que se encuentra en el subárbol con raíz en el nodo que recibe el mensaje y la fuente del paquete de red se encuentra también en ese subárbol pero no es la dirección del nodo que analiza el paquete, entonces el

paquete es pasado a la capa superior y reenviado.

- Si tiene dirección MAC del padre y la fuente del paquete de red no se encuentra en el subárbol con raíz en el nodo que recibe el mensaje, el paquete es enviado a la capa superior y reenviado.
- En cualquier otro caso el paquete es descartado.

En otras palabras, cuando le llega un mensaje de *broadcast* a un nodo, este compara la dirección de la capa de enlace de datos para ver si proviene de uno de sus hijos o de su padre. Si es así, el paquete es procesado para pasarlo a la capa superior de la capa de red y reenviado con la dirección *broadcast* en la capa de enlace de datos y la información del paquete de la capa de red queda intacta.

III.5.4. Mantenimiento de conexión

Por la topología y el esquema de asignación de direcciones, donde un nodo tiene un bloque de direcciones para asignar a nuevos nodos que se unen a la red, se requiere una constante verificación de los nodos activos en la red, para que al momento que otro nodo se quiera asociar a ésta, se conozcan las direcciones disponibles y pueda asignarle una al nuevo nodo si la hay.

Para informar que un nodo se mantiene activo envía mensajes de control al nodo que utilizó para asociarse a la red, es decir, a su nodo padre, y este a su vez responde con un mensaje de respuesta. Estos mensajes son conocidos como *mensaje de eco* y *mensaje respuesta de eco* respectivamente.

Los mensajes de eco son enviados después de un tiempo determinados por *PeriodicidadEco*, este tiempo empieza su cuenta al establecer la conexión a nivel MAC con el padre o después de recibir un mensaje respuesta de eco. Si el nodo padre no recibe mensajes

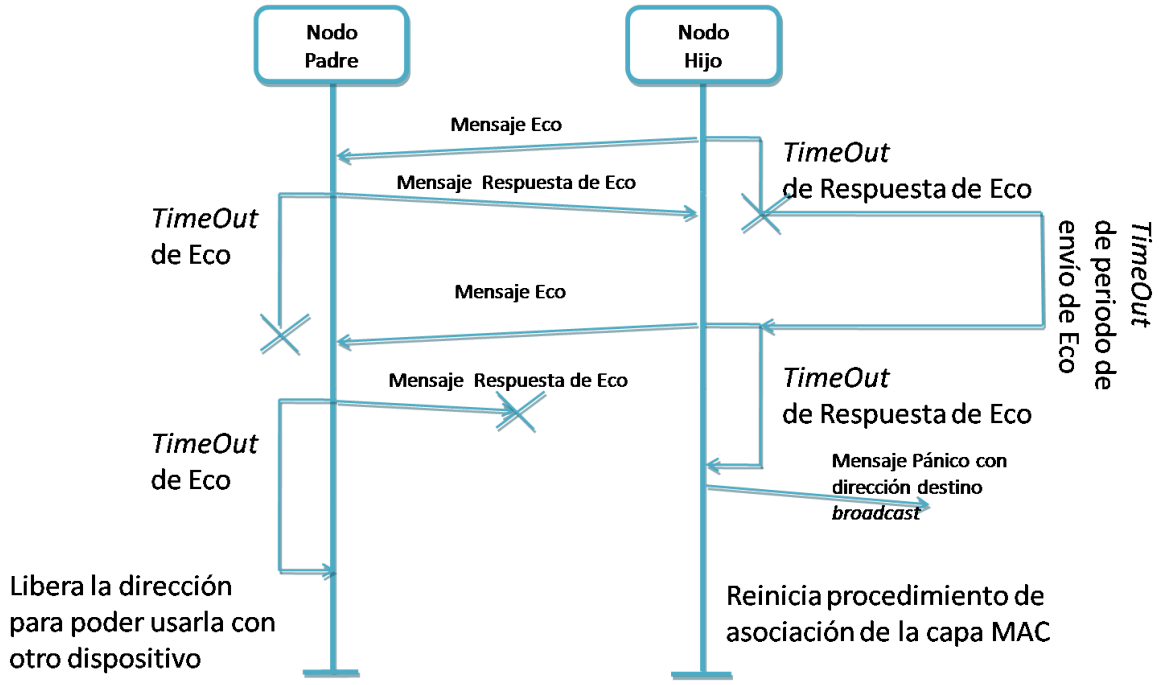


Figura 12: **Comunicación entre un nodo padre y uno de sus hijos para mantener la conexión entre ellos.**

de eco en este tiempo después de consumido el tiempo *PeriodicidadEco*, se asume que el nodo ya no está conectado, este tiempo inicia su cuenta a partir del establecimiento de la conexión a nivel MAC y después de recibir un mensaje de eco. Si el nodo que envió un mensaje de eco no recibe un mensaje de respuesta de eco en un tiempo determinado por *EsperaEcoRespuesta*, se asume que el nodo padre no se encuentra en línea, se envía un mensaje de pánico con dirección destino *broadcast* en la capa de red y en la de enlace de datos, después se reinicia el procedimiento de asociación a la red provisto por el estándar IEEE 802.15.4 para encontrar nodos cercanos que permiten asociarse. El motivo de este mensaje es advertir a los nodos hijos que un nodo antecesor que los conectaba a la red a desaparecido del enlace.

Estos dos valores, *PeriodicidadEco* y *EsperaEcoRespuesta* son iguales para todos los nodos de la red. Es decir, si un nodo se está reportando cada *PeriodicidadEco* tiempo con su padre, significa que los hijos conectados a él deben estarse reportando cada ese mismo

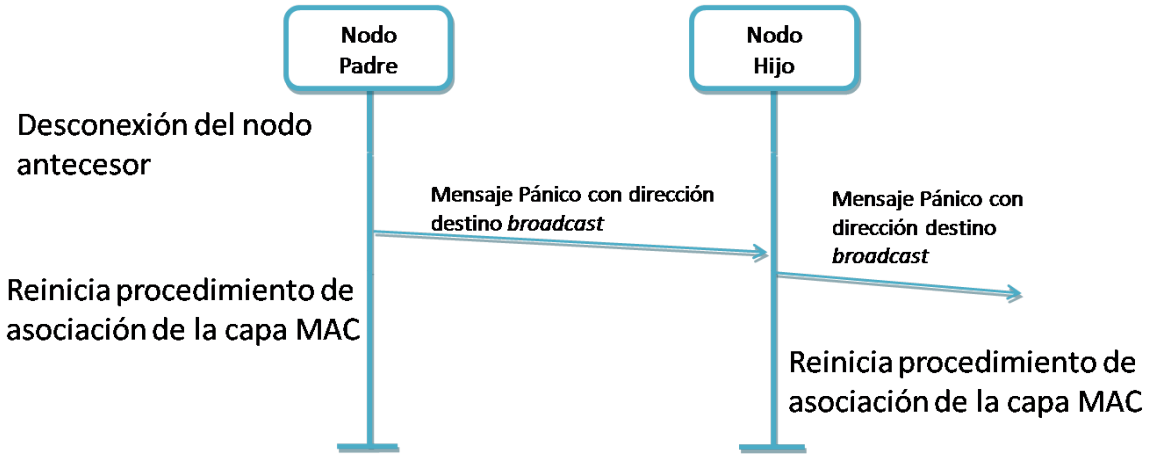


Figura 13: **Envío de mensajes de pánico para informar la desconexión del nodo padre.**

tiempo.

Cuando un nodo recibe un mensaje de pánico verifica que provenga del nodo padre, si es así lo retransmite con la dirección *broadcast* como destino en la capa de enlace de datos, posteriormente reinicia el procedimiento de asociación de la capa de enlace de datos para obtener una nueva dirección que permita comunicarse en la red. Si el mensaje no tiene como origen el nodo padre a nivel de enlace de datos, entonces es descartado.

En la Figura 12, se muestra cómo los nodos se encuentran constantemente enviando mensajes de control de eco y respuesta de eco para mantener la conexión. El envío del mensaje de pánico en *broadcast* a los hijos es ilustrado en la Figura 13.

Debe tomarse en cuenta que los dispositivos utilizados para el estándar IEEE 802.15.4, son en su gran mayoría de bajo costo y los cristales o relojes utilizados no son de gran precisión, lo que puede repercutir en tener variaciones en la toma del tiempo de hasta $\pm 40\,\%$. Esta variación es la esperada por los dispositivos utilizados en el desarrollo del prototipo de este protocolo (HCS08).

III.6. Formato de mensajes

Los mensajes del protocolo están encapsulados en mensajes de datos de la capa MAC de IEEE 802.15.4. El primer octeto indica el tipo de mensaje, el segundo y tercer octeto indican la dirección destino en formato *little-endian*, el cuarto y quinto octeto es la dirección fuente en formato *little-endian* y los últimos octetos son para los datos de la capa superior cuando se trata de mensajes de datos.

III.6.1. Mensaje Eco

Bytes: 1	2	2
Tipo	Dirección destino	Dirección fuente

Figura 14: **Formato de mensaje de Eco.**

El mensaje de eco mostrado en la Figura 14 contiene los siguientes campos:

Tipo: 1 para indicar que es un mensaje de eco.

Dirección destino: 16 bits correspondientes a la dirección a la que se le solicita hacer eco.

Dirección fuente: 16 bits correspondientes a la dirección que solicita hacer eco.

Descripción: Los nodos envían este tipo de mensaje al nodo que utilizaron para asociarse a la red, pidiendo con esto un mensaje de respuesta de eco que indica que el enlace continúa.

III.6.2. Mensaje Respuesta de Eco

Bytes: 1	2	2
Tipo	Dirección destino	Dirección fuente

Figura 15: **Formato de mensaje de respuesta de eco.**

El mensaje de respuesta de eco mostrado en la Figura 15 contiene los siguientes campos:

Tipo: 2 para indicar que es un mensaje de eco.

Dirección destino: 16 bits correspondientes a la dirección que solicitó hacer eco.

Dirección fuente: 16 bits correspondientes a la dirección que responde al eco.

Descripción: Los nodos envían este tipo de mensaje en respuesta a un mensaje de eco, indicando con este que el enlace continúa establecido.

III.6.3. Mensaje de Pánico

Bytes: 1	2	2
Tipo	Dirección destino	Dirección fuente

Figura 16: **Formato de mensaje de pánico.**

El mensaje de pánico en la Figura 16 contiene los siguientes campos:

Tipo: 3 para indicar que es un mensaje de pánico.

Dirección destino: 16 bits correspondientes a la dirección *broadcast* 0xFFFF.

Dirección fuente: 16 bits correspondientes a la dirección que originó el mensaje de pánico al detectar la ausencia de su padre.

Descripción: Los nodos envían este tipo de mensaje para informar a los hijos la desaparición del enlace de un nodo que los conectaba a la red.

III.6.4. Mensaje de Datos

El mensaje de datos mostrado en la Figura 17 contiene los siguientes campos:

Tipo: 0 para indicar que es un mensaje de datos.

Bytes: 1	2	2	1-97
Tipo	Dirección destino	Dirección fuente	Datos

Figura 17: **Formato de mensaje de datos.**

Dirección destino: 16 bits correspondientes a la dirección destino del paquete de datos.

Dirección fuente: 16 bits correspondientes a la dirección fuente del paquete de datos.

Datos: De 1 a 97 bytes utilizados para almacenar información que se pasará a la capa superior.

Descripción: Los nodos envían este tipo de mensaje cuando la capa superior a la de red requiere enviar información.

III.7. Estructuras de datos

III.7.1. Tabla de direcciones asignadas

La tabla de direcciones asignadas, es una estructura de datos que almacena las direcciones que el nodo ha establecido a otros nodos que se asociaron a la red a través de él.

Conceptualmente, la tabla de direcciones asignadas cuenta con el siguiente campo:

Dirección del nodo: dirección de nodo actualmente asociado.

Adicionalmente cuenta con *timers* asociados al tiempo del último mensaje de mantenimiento de la conexión.

III.8. Parámetros de configuración

Los parámetros de configuración permiten modificar el comportamiento del protocolo para tener detección casi inmediata de eventos que ocurran en la red, como la perdida de

la conexión con el nodo padre, o ahorrar energía y liberar ancho de banda al utilizar menos mensajes de control.

Algunos parámetros de configuración pueden ser diferentes en los nodos como el tamaño del *buffer* para mensajes de entrada y salida, otros deben ser iguales como la frecuencia de envío de mensajes de solicitud de eco y el tiempo de respuesta de eco y otros deben ser diferentes para todos los nodos como la dirección extendida que identifica al nodo. Los parámetros de configuración son los siguientes:

mDefaultValueOfChannel_c: Específica el o los canales donde puede establecer la red, este parámetro debe contener el canal seleccionado por el nodo *sink* para poder encontrar el coordinado en la capa de enlace de datos.

mDefaultValueOfExtendedAddress_c: Establece la dirección extendida utilizada por el nodo, esta dirección debe ser diferente para cada uno de ellos, en caso contrario puede presentarse una inconsistencia y asignar la misma dirección de red a dos nodos.

nMaxPendingDataPackets_c: Es el número máximo de paquetes en espera para ser transmitidos por el enlace de datos. Si este valor es alcanzado no se podrán enviar más datos hasta que sea recibida una notificación de envío satisfactorio o no de un mensaje y se producirá un error al invocar a la función `Nwk_Send()`.

nMaxInputDataPackets_c: Es el número máximo de paquetes en el *buffer* de entrada. Si este valor es alcanzado los siguientes paquetes de datos serán descartados.

nSendREQECHOTimerTickInMiliSec: Es el tiempo en milisegundos que espera un nodo para enviar un mensaje de control eco a su nodo padre. Este valor debe ser igual para todos los nodos en la red.

nWaitREPECHOTimerTickInMiliSec: Es el tiempo en milisegundos que espera un nodo para recibir la respuesta a un mensaje de control eco con un mensaje de control respuesta de eco. Si después de este tiempo no se recibe un mensaje de control respuesta de eco, el nodo reinicia el procedimiento de asociación para encontrar otro nodo

para conectarse a la red.

nRecieveREQECHOTimerTickInMiliSec: Es el tiempo en milisegundos que espera un nodo para recibir un mensaje de control eco de un nodo hijo. Si no se recibe un mensaje de control eco después de este tiempo se asume que el nodo hijo ya no se encuentra en el enlace.

III.9. Implementación

III.9.1. Interfaz de programación de aplicaciones (API)

La interfaz de programación que proporciona la implementación de este protocolo consta de una estructura de datos donde se almacena la información recibida y de varias funciones que sirven para interactuar con la capa de red implementada.

Estructuras de datos

La estructura de datos utilizada para recibir los datos de la red es `nAppData_t`, consta de los siguientes campos:

`networkAddress[2]` Son 2 bytes en *little-endian* utilizados para almacenar la dirección de red fuente del paquete de datos.

`length` Es un byte que indica la longitud de los datos recibidos.

`data[nSizeDataInPackets_c]` Es un arreglo de tamaño `nSizeDataInPackets_c` que puede ser de hasta 97 bytes. Sólo los primeros bytes indicados por `length` son los datos recibidos de la red.

Funciones y procedimientos

El procedimiento para enviar y recibir datos por la red es el siguiente:

- Primero se deben inicializar la pila por medio de la instrucción `Nwk_Init()`.
- Después se inicializa el motor del protocolo por medio de la instrucción `Nwk_Start()`.
- Por último se puede enviar mensajes de datos por la red con la función `Nwk_Send()` o recibirlo con la función `Nwk_Receive()`.

A continuación se describen las funciones y parámetros utilizados para interactuar con la capa de red:

void Nwk_Init(void) Es el procedimiento de inicialización de la pila, entre sus funciones esta inicializar la capa de enlace de datos y los *buffer* de mensajes.

No recibe ni devuelve datos.

void Nwk_Start(void) Este procedimiento inicia el motor del protocolo de red. Se encarga de identificar las redes y dispositivos dentro del alcance, para asociarse a ellas y con esto obtener una dirección de enlace de datos, que también es utilizada en la capa de red.

No recibe ni devuelve datos.

void Nwk_Stop(void) Este procedimiento detiene el motor del protocolo de red. Inhabilita los temporizadores utilizados en el mantenimiento de la conexión. Al detener los *timers* permite a la capa de aplicación poner al nodo en ciclos de bajo consumo.

No recibe ni devuelve datos.

uchar Nwk_Send(uchar *netAddrDest, uchar * data, uchar length) La función `Nwk_Send()` permite enviar datos a través de la red hasta llegar a la dirección `netAddrDest`.

Los parámetros que se envían son:

- netAddrDest: es un puntero a la dirección de memoria donde se encuentra almacenada la dirección destino, en *little-endian*.
- data: es un puntero a la dirección de memoria donde inician los datos a enviar a través de la red.
- length: es la cantidad de bytes que se desean enviar por la red. Con este parámetro se puede saber donde terminan los datos a partir de la dirección de data.

Valores de retorno:

- Sin error: 0x00
- Datos nulos: 0x01
- Longitud de datos cero: 0x02
- Sin memoria para alojar mensaje: 0x03
- Conexión no establecida: 0x04

void Nwk_Receive(nAppData_t * data) La función `Nwk_Receive()` permite recuperar los datos de la capa de red. Para poder ser utilizado se debe reservar espacio en memoria para almacenar esos datos y enviar donde empieza esa memoria reservada a la función.

Los parámetros que se envían son:

- `data`: es la posición de inicio de memoria reservada para almacenar los datos recibidos de la red. data es de tipo `nAppData_t` definido anteriormente.

void Nwk_getExtendedAddress(uint8_t * address) La función `Nwk_getExtendedAddress()` permite obtener la dirección extendida del nodo, esta consta de 64 bits que identifican al nodo.

Los parámetros que se envían son:

- `address`: es la posición de inicio de memoria reservada para almacenar la dirección extendida del dispositivo en *little-endian*, esta debe ser de 64 bits.

void Nwk_getShortAddress(uint8_t * address) La función `Nwk_getShortAddress()` permite obtener la dirección corta del nodo, esta consta de 16 bits que identifican al nodo en la red.

Los parámetros que se envían son:

- `address`: es la posición de inicio de memoria reservada para almacenar la dirección corta del dispositivo en *little-endian*, esta debe ser de 16 bits.

III.9.2. Software y Hardware utilizado

Para el desarrollo del prototipo se utiliza la implementación del estándar IEEE 802.15.4 versión 2003 desarrollada por Freescale.

Se utilizó Freescale BeeKit Wireless Connectivity Toolkit para configurar y tener una base de código de esta implementación. Aunque para tener el código no se necesita licencia, para compilarlo sí. Se utiliza CodeWarrior como IDE (*Integrated Development Environment*).

BeeKit provee código base para utilizar la implementacion del estándar IEEE 802.15.4 de Freescale. Se puede obtener en ensamblador, C y C++, el utilizado para este prototipo es C.

El hardware usado es MC13192 *Sensor Applications Reference Design* (SARD) el cual contiene 4kBytes de memoria RAM, 60kBytes de memoria no volátil, la velocidad del procesador puede configurarse hasta 20MHz, cuenta con 4 LEDs y 4 botones, acelerómetros

para tres dimensiones, puerto de depuración de programas, puerto serie DB-9 para comunicación con el nodo, interfaz de radio frecuencia, puede ser alimentado de energía por baterías de 9 V (6LR61) o por un adaptador de 9 V de corriente directa, si los acelerómetros no son utilizados se pueden utilizar baterías con 2-3.4 V.

Capítulo IV

Evaluación

La evaluación realizada en este trabajo se enfoca en analizar el protocolo de red en dos rubros: funcionalidades y comportamiento de la implementación. La evaluación de funcionalidad, se refiere a identificar qué características y tareas de un protocolo de capa de red de propósito general cumple el protocolo propuesto, así como las particularidades de un protocolo de capa de red para redes inalámbricas de monitoreo y control. También se analiza cómo son cumplidos los requerimientos establecidos inicialmente para el protocolo de red.

Por otro lado, la evaluación del comportamiento está enfocada en observar el desempeño de la implementación del protocolo bajo diferentes circunstancias; esto se lleva a cabo con la realización de diferentes pruebas que obtienen valores prácticos que caracterizan la implementación del protocolo.

IV.1. Funcionalidad

La evaluación de funcionalidad se divide en dos grupos, el primero se centra en identificar las características de la especificación del protocolo comparadas con las tareas que

debe desarrollar un protocolo de red de propósito general (ISO/IEC7498-1, 1994). En una segunda etapa se comparan las funcionalidades que debe proveer un protocolo de red para redes inalámbricas de monitoreo y control con lo establecido por el protocolo ligero de red propuesto. Por último, se coteja como los requerimientos identificados inicialmente son cumplidos por el protocolo.

IV.1.1. Evaluación de protocolo de red de propósito general

Esta evaluación se enfoca en identificar qué tareas de un protocolo de red según el modelo OSI de ISO (ISO/IEC7498-1, 1994) se encuentran contempladas en la especificación del protocolo ligero de red. Estas tareas son las siguientes:

- Servicios proveídos a la capa de transporte
    - Transferencia de datos entre entidades de transporte
    - Interfaz para la capa de transporte consistente e independiente
    - Calidad de servicio negociable
    - En modo orientado a conexión la capa de red debe proveer:
        - Direcciones de red
        - Conexiones de red
        - Identificadores de conexiones de red
        - Transferencia de unidades de datos del servicio de red
        - Parámetros de calidad de servicio
        - Notificación de errores
        - Transferencia expedita de unidades de datos del servicio de red
        - Reinicio
        - Liberación
        - Recibo de confirmación
    - En modo no orientado a conexión la capa de red debe proveer los servicios de:

    - Transferencia de unidades de datos del servicio de red del tamaño máximo definido
    - Parámetros de calidad de servicio
    - Notificación de errores locales

- Funciones dentro de la capa de red
    - Ruteo de mensajes
    - Conexiones de red
    - Multiplexado de conexiones de red
    - Segmentación y bloqueo
    - Detección de errores
    - Recuperación de errores
    - Secuenciación
    - Control de flujo
    - Transferencia expedita de datos
    - Reinicio
    - Selección de servicio
    - Mapeo entre direcciones de red y direcciones de enlace de datos
    - Mapeo de transmisiones de red en modo no orientado a conexión a transmisiones de enlace de datos no orientada a conexión
    - Conversión de servicio de enlace de datos orientado a conexión a un servicio de red no orientado a conexión
    - Aumentar un servicio de enlace de datos no orientado a conexión a proveer un servicio de red orientado a conexión
    - Gestión de la capa de red

A continuación describiremos cada una de las características mencionadas para identificar de qué manera son cubiertas por el protocolo ligero de red y cuáles no son contempladas, después se resumen estas en una tabla para mejor apreciación.

Servicios proveídos a la capa de transporte

La primer característica es la transferencia de datos entre entidades de transporte, esta es cumplida ya que con la utilización de éste protocolo se pueden comunicar las capas de transporte de diferentes nodos por medio del uso de las funciones `Nwk_Send()` y `Nwk_Receive()` que permiten enviar y recibir datos.

La segunda característica es proveer una interfaz para la capa de transporte consistente e independiente. Esta es cumplida por medio de las funciones `Nwk_Start()`, `Nwk_Stop()`, `Nwk_Send()` y `Nwk_Receive()` utilizadas para interactuar con la capa de red.

La calidad de servicio negociable no es posible para este protocolo. La calidad de servicio de la capa de red siempre es *el mejor esfuerzo* para reducir complejidad en la implementación del protocolo de capa de red; esto repercute en el tamaño del código utilizado para ello y el número y tamaño de mensajes necesarios para comunicarse.

El modelo OSI define dos modos de conexión utilizados para establecer comunicación entre diferentes nodos. El primero es orientado a conexión, en el cual para comunicarse debe establecer una conexión hacia el nodo que se desea enviar información, después transmitir los mensajes y por último liberar la conexión. El segundo modo es el no orientado a conexión, en el cual se transmite una unidad de datos simple desde una entidad (en este caso la capa de red), a otra entidad en un nodo diferente. El protocolo de red especificado define un modo de transmisión no orientado a conexión, para reducir el tamaño del código, memoria y procesamiento requerido para establecer las conexiones, mantenerlas,

informar de errores y liberarlas.

El protocolo es no orientado a conexión, pero define direcciones de red para identificar a cada nodo, también permite enviar unidades de datos de diferentes tamaños y provee informe de errores locales cuando se envía información a través de la red. No proporciona parámetros para configuración de la calidad del servicio, ésta siempre es *el mejor esfuerzo*, estableciendo prioridad a quien solicita el servicio primero.

Funciones dentro de la capa de red

Dentro de las funciones de la capa de red, el protocolo proporciona ruteo de mensajes de datos, permitiendo entregarlos a su destino a través de múltiples saltos.

Por no ser orientado a conexión, no provee conexiones de red y multiplexado de estas. Tampoco incorpora segmentación y bloqueo de los mensajes a transmitir, estos deben ser divididos por la capa superior en un tamaño máximo de 97 bytes y mínimo de 1 byte si se requiere enviar mensajes de tamaño mayor.

El protocolo suministra detección de errores locales cuando se requiere enviar datos por la red y los informa, pero no tiene la capacidad de recuperarse de ellos. Esta funcionalidad debe ser implementada por la capa superior si es requerida, así como la detección de errores a través de toda la comunicación.

La funcionalidad de secuenciación de los mensajes es cumplida debido a la topología de la red, donde solo se cuenta con un único camino para llegar desde un nodo a otro, permitiendo que los paquetes de red sean entregados en el orden en el que fueron enviados, pero sin garantizar la entrega de todos los paquetes enviados.

El control de flujo es implementado restringiendo la cantidad de mensajes enviados a través del punto de acceso `Nwk_Send()` de la capa de red a un máximo de `mMaxPendingDataPackets_c`, cuando se envía un mensaje se aumenta un contador y es disminuido en el momento en que se recibe la confirmación de la entrega del mensaje de la capa de enlace de datos o la notificación de un error ocurrido durante la transmisión. En caso de superar este límite se produce un error al no poder tener en procesamiento más mensajes y es informado a la capa superior.

Todos los mensajes son transmitidos con la misma prioridad inmediatamente después de ser solicitado su envío, lo cual no permite transmitir un mensaje expeditamente si se cuenta con otros en espera. La funcionalidad de transferencia expedita de datos no está contemplada.

El protocolo considera la funcionalidad de reinicio, en la cual se desconecta de la red para volver a conectarse al nodo que se encuentre más cercano. Se descartan los paquetes en cola de transmisión y también se reinicia la capa MAC.

No se proporciona selección de servicio, porque la selección está orientada a redes donde los enlaces son de calidades diferentes. En este protocolo se contempla una red homogénea en la que todos los nodos tienen la misma calidad de servicio (mejor esfuerzo) y se encuentran dentro de la misma red.

El protocolo de red provee un mapeo directo de direcciones de red y direcciones de enlace de datos, es decir, si la dirección de red es 0x1000, la dirección de enlace de datos también es 0x1000. Con esto se elimina la necesidad de tener tablas de las relaciones de estas dos direcciones y de realizar el procedimiento de resolución de direcciones.

No se cuenta con mapeo de transmisiones de red en modo no orientado a conexión

a transmisiones de enlace de datos no orientada a conexión, por que el enlace de datos utilizado es orientado a conexión, por lo tanto se convierte un servicio de enlace de datos orientado a conexión a un servicio de red no orientado a conexión. También queda descartado el aumento del servicio de enlace de datos no orientado a conexión a un servicio de red orientado a conexión.

Los protocolos de capa de red se ocupan con algunas de las actividades de gestión como activación y control de errores. En este caso el protocolo de red incorpora administración de memoria para uso de la capa de red, reconfiguración y autoconfiguración para conectarse a la red y reinicio cuando no se puede establecer comunicación por la conexión actual.

Conclusiones

Debido a las limitaciones de los dispositivos para los que este protocolo está enfocado, no cumple con varias de las características contempladas por el modelo de referencia OSI, cubriendo aquellas que hacen posible que los datos sean enviados desde su origen hasta su destino aunque no tengan una conexión directa. Para una mejor apreciación en la siguiente tabla se resumen las características que contempla el protocolo de red.

Característica	Soportada		
	Si	No	Parcial
• Servicios provistos a la capa de transporte			
○ Transferencia de datos entre entidades de transporte	✓		
○ Interfaz para la capa de transporte consistente e independiente	✓		
○ Calidad de servicio negociable		X	
○ En modo orientado a conexión la capa de red debe proveer:		X	
▪ Direcciones de red	✓		
▪ Conexiones de red		X	
▪ Identificadores de conexiones de red		X	
▪ Transferencia de unidades de datos del servicio de red	✓		
▪ Parámetros de calidad de servicio		X	
▪ Notificación de errores en conexiones de red		X	
▪ Transferencia expedita de unidades de datos del servicio de red		X	
▪ Reinicio de conexiones		X	
▪ Liberación de conexiones		X	
▪ Recibo de confirmación		X	
○ En modo no orientado a conexión la capa de red debe a través puntos de acceso a los servicios de red debe proveer:	✓		
▪ Transferencia de unidades de datos del servicio de red del tamaño máximo definido	✓		
▪ Parámetros de calidad de servicio		X	
▪ Notificación de errores locales	✓		

• Funciones dentro de la capa de red			
○ Ruteo de mensajes	✓		
○ Conexiones de red		X	
○ Multiplexado de conexiones de red		X	
○ Segmentación y bloqueo		X	
○ Detección de errores			O
○ Recuperación de errores		X	
○ Secuenciación		X	
○ Control de flujo	✓		
○ Transferencia expedita de datos		X	
○ Reinicio	✓		
○ Selección de servicio		X	
○ Mapeo entre direcciones de red y direcciones de enlace de datos	✓		
○ Mapeo de transmisiones de red en modo no orientado a conexión a transmisiones de enlace de datos no orientada a conexión		X	
○ Conversión de servicio de enlace de datos orientado a conexión a un servicio de red no orientado a conexión	✓		
○ Aumentar un servicio de enlace de datos no orientado a conexión a proveer un servicio de red orientado a conexión		X	
○ Gestión de la capa de red			O

IV.1.2. Evaluación de protocolo de red para redes inalámbricas de monitoreo y control

Esta evaluación se enfoca en identificar que característica de un protocolo de red para redes inalámbricas de monitoreo y control cubre el protocolo ligero propuesto. Aunque

las características de las redes de monitoreo y control están extremadamente ligadas a las aplicaciones, entre estas destacan las siguientes (Akkaya y Younis, 2005) (Akyildiz et al., 2002):

- Consideraciones en consumo de energía
- Sistema auto-organizados
- Movilidad de los nodos
- Centrado en datos
- Agregación de datos
- Orientado en direccionamiento
- Consciencia de localización de los nodos
- Heterogeneidad de los dispositivos

Para el envío y ruteo de mensajes de datos no se busca la ruta que requiera menor cantidad de energía o la que cuente con mayor reserva de esta, aunque el consumo de energía es tomado en cuenta por el protocolo de red ya que es posible que algunos nodos vayan a dormir sin preocuparse por si están siendo utilizados para el envío de mensajes de datos. Los nodos restantes son capaces de obtener nuevas rutas si es necesario, para comunicarse con los nodos activos unidos a la red. Por otro lado, el protocolo no especifica la utilización de funciones *low power*, debido a esto la capa de aplicación o una capa superior debe encargarse de la administración de la energía, poniendo a dormir al nodo, los sensores, dispositivos periféricos y almacenado la información que crea conveniente antes de hacerlo.

El protocolo de red es un sistema auto-organizado que permite a los nodos conectarse a la red y mantener siempre esta conexión, incluso cuando los nodos utilizados para conectarse a la red desaparecen del enlace. En este caso, los nodos buscan a otros dentro del alcance de su radio para conectarse a la red y poder comunicarse en el momento que lo

requieran.

La topología utilizada por el protocolo es en árbol, pero la ubicación de los nodos no está preestablecida. Es posible que los nodos cambien de ubicación cambiando con esto la topología de la red y aún así pueden comunicarse con los nodos restantes.

El protocolo de red no está centrado en datos y tampoco contempla agregación de datos para el ruteo de mensajes. Este protocolo está enfocado para ser utilizado en aplicaciones de monitoreo y control donde no solo hay aplicaciones en las que los datos obtenidos de los sensores son importantes, también existen aplicaciones donde el control de los dispositivos actuadores es primordial.

Se utiliza un sistema de direccionamiento donde cada nodo obtiene una dirección al conectarse a la red. Esta dirección puede cambiar si el nodo se mueve de lugar y su radio ya no puede comunicarse con el nodo que utilizaba para conectarse a la red. En este caso, se reinicia el procedimiento de asociación a la red para encontrar otro nodo conectado a la red y obtener una nueva dirección.

No se contempla la consciencia de localización, debido a los dispositivos especializados que se requieren para ésta función. Otra manera de tener consciencia de localización es indicar a cada nodo su ubicación, lo cual limitaría su movilidad.

El protocolo de red está basado en el estándar IEEE 802.15.4, mismo que define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal. Esto puede permitir utilizar dispositivos heterogéneos que implementen el estándar IEEE 802.15.4 de diferentes fabricantes y distintas capacidades.

Conclusiones

Las características básicas un protocolo de red para redes inalámbricas de monitoreo y control son parcialmente cumplidas, faltando incluir el enfoque centrado en datos que es utilizado en aplicaciones donde la red inalámbrica es para uso de monitoreo. Se opto por un enfoque centrado en direcciones y no en datos debido a que no solamente se quiere recolectar datos, sino también controlar dispositivos específicos.

Las características no contempladas por el protocolo como consideraciones en consumo de energía, centrado en datos, agregación de datos y conciencia de localización, en realidad son módulos independientes que pueden ser agregados por otras capas dependiendo de la aplicación en la que sea utilizado el protocolo. Esto permite que aquellas aplicaciones que no requieran de esa funcionalidades requieran de dispositivos con recursos adicionales.

Característica	Soportada		
	Si	No	Parcial
• Consideraciones en consumo de energía		✗	
• Sistema auto-organizados	✓		
• Movilidad de los nodos	✓		
• Centrado en datos		✗	
• Agregación de datos		✗	
• Orientado en direccionamiento	✓		
• Consciencia de localización de los nodos		✗	
• Heterogeneidad de los dispositivos	✓		

IV.1.3. Evaluación de fase de obtención de requerimientos

En la fase de obtención de requerimientos se detectaron las siguientes características que debe cubrir el protocolo ligero de red:

- Tamaño del código pequeño comparado con otros existentes.
- Sistema de direccionamiento con soporte para hasta 1024 nodos.
- No necesitar de un nodo coordinador.
- Enrutamiento multisalto.
- Autoformación de redes.
- Comunicación *unicast* y *broadcast*.
- Proveer una interfaz para desarrollar aplicaciones sobre él.

El tamaño del código del prototipo únicamente para la capa de red es de 4.7 kBytes, adicionalmente se requiere un espacio de entre 24.2 y 35.4 kBytes para la implementación de la capa de enlace de datos y la capa física, la cual puede o no contar con utilización de *beacons*. Además se necesita memoria para almacenar el planificador de tareas, *drivers* y la aplicación. En total, para la pila de la capa MAC y la de red se requieren alrededor de 40 kBytes utilizando la implementación completa del estándar IEEE 802.15.4 o 28.9 kBytes empleando la capa MAC del estándar sin la funcionalidad de *beacons* e incluyendo la capa de aplicación el espacio total requerido puede ir desde los 36 kBytes.

Comparando la memoria total requerida para el almacenamiento del código del prototipo con implementaciones de ZigBee como la de Freescale, la cual necesita un espacio de 60 kBytes (Freescale, 2008), el espacio requerido por el protocolo propuesto es menor en 14 kBytes, aunque ZigBee tiene bastante más funcionalidad. Por otro lado, la implementación NanoStack (2008) de 6LoWPAN utiliza un espacio total de 36 kBytes que es muy cercano a la implementación del protocolo ligero, pero NanoStack incluye la implementación del estándar IEEE 802.15.4 y el protocolo UDP utilizado para capa de transporte y envío de mensajes de control para ruteo de datos (Arch Rock, 2008). De la misma manera ZigBee también incorpora un protocolo de capa de transporte entre otras funcionalidades.

Otros protocolos como PopNet, SNAP y MiWi requieren un menor tamaño de memoria, de alrededor de 16 kBytes para almacenamiento del código, esto es posible porque utilizan una capa MAC reducida sólo con las funciones indispensables para el funcionamiento de cada uno de estos protocolos. Por otro lado, estos protocolos de red restringen su uso a hardware fabricado por compañías específicas.

El diseño del protocolo de red propuesto le permite ser montado sobre alguna capa MAC mínima que proporcione únicamente los servicios indispensables requeridos por el protocolo de red, estos servicios son: envío de mensajes *unicast* y *broadcast* a través de un salto; método de búsqueda de dispositivos cercanos y de asociación a uno seleccionado. Con esto se elimina el espacio necesario para almacenar la capa MAC completa de IEEE 802.15.4, aunque ello significa perder la robustez proporcionada por el estándar, por ejemplo tener menor entrega efectiva por no contar con algoritmos de control de acceso al medio como CSMA-CA. Uno de estos protocolos de capa MAC reducidos es SMAC (*Simple Media Access Controller*) de Freescale, que proporciona comunicación bidireccional para dispositivos MC1319x, MC1320x, y MC1321x de Freescale (SMAC, 2006); requiere de un espacio de 4 kBytes para almacenamiento de programa y un par de kBytes extras para cubrir los servicios que el protocolo de red propuesto necesita de la capa MAC. Por lo que el espacio aproximado total requerido es de 11.7 kBytes, de los cuales 6 kBytes son para capa MAC y 4.7 kBytes para la capa de red.

Las aplicaciones objetivo son aquellas en cuyo número de dispositivos no excediera a 1000, razón por la cual se propuso 1024 al tratarse de un número potencia de 2 con más de 1000 nodos (direcciones de 10 bits). Este número fue superado hasta obtener 41,356 direcciones de red (direcciones de 16 bits separadas en 4 grupos de 4 bits), cifra muy ambiciosa como para contar con ese número de dispositivos conectados al mismo tiempo, pero permite encaminar los paquetes de una forma directa y sin necesidad de grandes tablas de rutas.

Originalmente se contemplaba una red sin un nodo coordinador que llevara el control, pero al observar las características de las aplicaciones para las que este protocolo será usado, en las cuales la información es enviada a un punto central en la red donde es procesada y almacenada o en las que se cuenta con un centro de mando que controla los diferentes dispositivos en la red, se ha decidido utilizar un nodo central o *sink* del que depende la red. Al contar con este nodo central la utilización de una topología en árbol se hace apropiada por necesitar menos mensajes de control para descubrimiento de rutas y llegar al nodo *sink*.

Una de las principales limitantes de los dispositivos utilizados para este protocolo es la distancia de alcance de sus radios, requiriendo un esfuerzo cooperativo para encaminar los mensajes y entregarlos a su destino. El protocolo contempla un máximo de 8 saltos para llegar a un nodo conectado a la red, con 4 saltos como máximo desde el nodo *sink* hasta cualquier otro dispositivo.

Para la formación de la red se requiere tener siempre activo el nodo *sink*, después de esto los demás dispositivos se autoconfiguran para conectarse a la red. Si un dispositivo queda inhabilitado para recibir y enviar mensajes, los dispositivos enlazados a él buscarán otro nodo para conectarse a la red y tener comunicación con el resto de los nodos. Cuando es el *sink* el que falla, ningún dispositivo puede comunicarse y se mantienen en búsqueda de redes establecidas dentro del alcance de su radio a las que puedan asociarse. Este caso ocurre debido a que las aplicaciones objetivo contemplan un dispositivo central al que los nodos envían datos relevantes para su procesamiento y almacenamiento, también a partir de él se controla al resto de los dispositivos de la red.

El protocolo contempla dos tipos de direcciones *unicast* y *broadcast* las cuales son utilizadas para envío de mensajes de datos desde 1 byte hasta 97 bytes. La dirección de *broadcast* es 0xFFFF. Las direcciones *unicast* son utilizadas para identificar a cada nodo conectado y son asignadas al momento de conectarse a la red.

La implementación del prototipo provee una interfaz para desarrollar aplicaciones sobre el protocolo, esta interfaz consta de las funciones Nwk_Start(), Nwk_Stop(), Nwk_Reset(), Nwk_Send(), Nwk_Receive(), Nwk_getExtendedAddress() y Nwk_getShortAddress(), las cuales permiten iniciar la capa de red, enviar y recibir datos, obtener las direcciones corta y extendida y reiniciar la capa de red.

Conclusiones

Los requerimientos iniciales cambiaron para adecuarse a las aplicaciones objetivo, aún así se mantuvo la idea principal en el desarrollo de este protocolo de red ligero y su implemenetación de código libre que permita futuras investigaciones y desarrollos para incorporar funcionalidades como agregación de datos, *low-power*, entre otros. Para una mejor apreciación en la siguiente tabla se resumen las características que cumple el protocolo de red propuesto, mismas que fueron detectadas en la fase de obtención de requerimientos.

Característica	Soportada		
	Si	No	Parcial
• Tamaño del código pequeño comparado con otros existentes	✓		
• Sistema de direccionamiento con soporte para hasta 1024 nodos	✓		
• No necesitar de un nodo coordinador		✗	
• Enrutamiento multisalto	✓		
• Autoformación de redes	✓		
• Comunicación *unicast* y *broadcast*	✓		
• Proveer una interfaz para desarrollar aplicaciones sobre él	✓		

IV.2. Comportamiento

Esta evaluación está centrada en determinar cómo se desempeña el protocolo en la práctica durante situaciones de uso común. Para ello se pone a prueba la implementación que se desarrolló del protocolo. Las pruebas realizadas fueron:

- Integridad de los datos en un salto.
- Integridad de los datos en múltiple saltos.
- Tiempo de respuesta en un salto.
- Tiempo de respuesta en múltiple saltos.
- Tiempo de recuperación de ruta.
- Tiempo de formación de red con dos nodos.
- Tiempo de formación de red con tres nodos.
- Entrega exitosa de datos para un salto.
- Entrega exitosa de datos para estrella.

Con la evaluación se obtienen valores como número promedio de pérdida de paquetes, tiempo de recuperación del enlace, tiempo de formación de red, tasa de transferencia para datos, entre otros parámetros, que caracterizan a la implementación del protocolo a diferentes intervalos de transmisión de mensajes de control. Estos valores permitirán en algún momento a desarrolladores de sistemas de monitoreo y control decidir utilizar este protocolo para aplicaciones específicas.

Para la definición de las pruebas se utiliza el formato propuesto en (Pinedo˜Frausto, 2008), el cual está basado en el formato utilizado para pruebas de interoperabilidad de ZigBee. En este formato se establece una descripción general de la prueba, la topología de la red, las condiciones iniciales, el procedimiento a desarrollar durante la prueba, las salidas que se obtienen de estos procedimientos y las características relacionadas que se desean medir durante la prueba.

El escenario para la ejecución de las pruebas cuenta con las siguientes características:

- El área física es en un ambiente de oficina con computadoras y redes inalámbricas.
- Los nodos se encuentran preconfigurados para utilizar un mismo canal.
- El envío de mensajes de datos es de 20 bytes, si no se indica lo contrario.
- Se utilizaron distintos intervalos para el envío de mensajes de control: cada 10mSeg, 20mSeg, 2Seg y 4Seg.

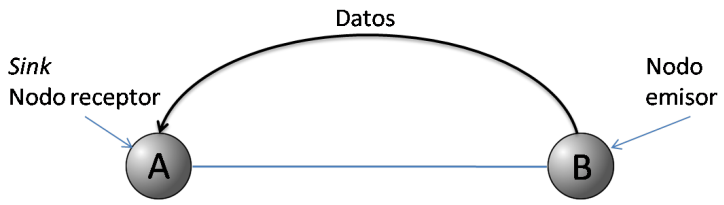
IV.2.1. Integridad de los datos en un salto

Descripción general: Esta prueba se enfoca en medir el número de mensajes correctos entregados en la transmisión a través de un salto en la red. Se verifica también el efecto que tiene cambiar el tiempo de envío de mensajes de control eco y aumentar la cantidad de datos enviados por segundo. La prueba se realizará para cuatro diferentes intervalos de transmisión de mensajes de control eco, 10mSeg, 20mSeg, 2Seg

y 4Seg y se enviarán mensajes de datos de 20 bytes cada 5, 10, 20, 40 y 50mSeg hasta completar 1000 mensajes. Por último, observar el número de mensajes recibidos correctamente. Un mensaje correcto contiene el número de secuencisia del mensaje repetido 10 veces utilizando 2 bytes, si un mensaje no tiene diez veces el mismo número de secuencia se descarta, también se descartan los mensajes con un número de secuencia previamente recibido.

Las tareas involucradas en esta prueba son:

- Envío de mensajes de datos *unicast*.
- Recepción de mensajes de datos *unicast*.

Topología	
	

Condiciones iníciales	
1	El nodo *sink* inicia la red con el PanID 0xFEED
2	Se encuentra conectado un nodo en la red y el nodo *sink.*
3	El nodo A esta programado para llevar un control de la cantidad de mensajes correctos recibidos.
4	El nodo B está programado para enviar mensajes a el nodo *sink*

Procedimiento		
Paso	Descripción	Salida
1	El nodo B envía un mensaje de 20 bytes al nodo A. Cada byte es el número de mensaje enviado.	
2	El nodo A recibe el mensaje y verifica que todos los bytes sean iguales. Si es así, incrementa la cantidad de mensajes recibidos correctamente.	1
3	Después de 5, 10, 20, 40 o 50mSeg el nodo B envía el siguiente mensaje	
4	Una vez terminada la transmisión de los 1000 mensajes cambiar el intervalo de transmisión de mensajes a 5, 10, 20, 40 o 50mSeg según sea el caso.	
5	Cambiar intervalo de envío de mensajes de control a 10mSeg, 20mSeg, 2Seg y 4Seg según sea el caso.	

Salida	
	Cantidad de mensajes recibidos correctamente

Características relacionadas con las pruebas
Se verifica como afecta la integridad de los datos de acuerdo a las variaciones en la frecuencia de envío de mensajes de datos y de control. También que tanto afecta en la transferencia efectiva permitida para datos.

Resultado

Las gráficas mostradas en la Figura 18 corresponden a los resultados obtenidos de las pruebas realizadas, en ellas se puede apreciar el número de mensajes recibidos. El número total de mensajes es 1000, se puede observar que solo existe diferencia en la prueba donde

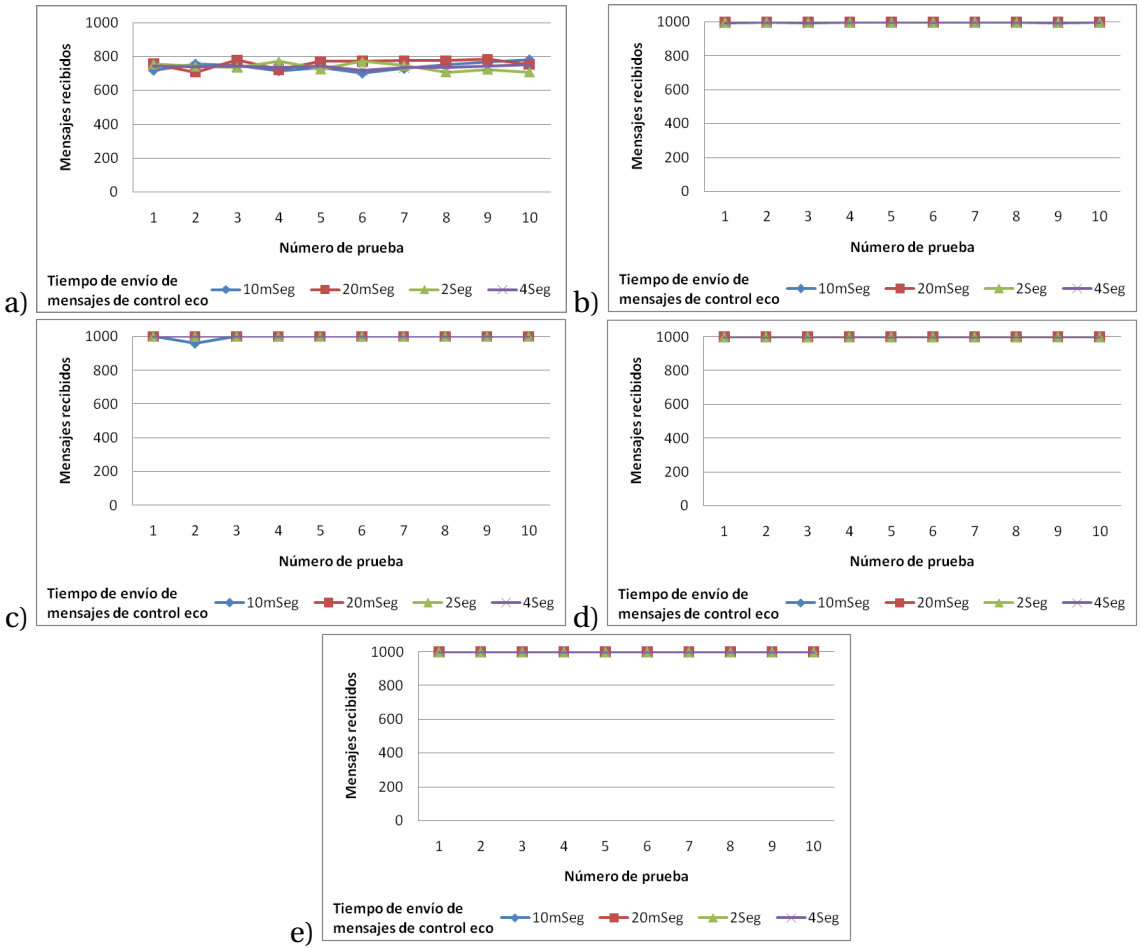


Figura 18: **Resultados de prueba de integridad de los datos en un salto. Las gráficas corresponden a las diferentes intervalos de envío de datos: a) 5mSeg, b) 10mSeg, c) 20mSeg, d) 40mSeg y e) 50mSeg.**

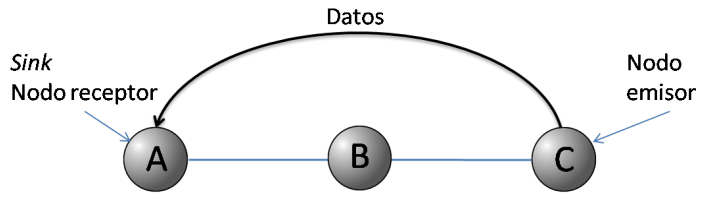
el intervalo de envío de mensajes de datos es cada 5mSeg, en la cual el porcentaje de mensajes recibidos es de alrededor de 75 %. La pérdida del resto de los mensajes ocurrió en el nodo emisor debido a la espera que se hace por el reconocimiento enviado por la capa de enlace de datos y la existencia de un límite de mensajes enviados sin reconocimiento. El especificado por la implementacion de Freescale del estándar 802.15.4 es de 4 mensajes para el tipo de dispositivos utilizados, *Full-Function Device* y *beaconless* (802154MACPHY-Software).

IV.2.2. Integridad de los datos en múltiple saltos

Descripción general: Esta prueba se centra en la medición del número de mensajes correctos entregados en la transmisión a través de dos saltos en la red y el efecto que tiene cambiar el tiempo de envío de mensajes de control eco y aumentar la cantidad de datos enviados por segundo. La prueba se realizará para cuatro diferentes intervalos de transmisión de mensajes de control eco y se enviarán mensajes de datos de 20 bytes cada 5, 10, 20, 40 y 50mSeg hasta completar 1000 mensajes. Por último, observar el número de mensajes recibidos correctamente, para esto, cada mensaje contiene en cada 2 bytes el número de mensaje y la prueba se realizará para cuatro diferentes intervalos de transmisión de mensajes de control eco, 10mSeg, 20mSeg, 2Seg y 4Seg. Un mensaje correcto contiene el número de secuencia del mensaje repetido diez veces utilizando 2 bytes, si un mensaje no tiene diez veces el mismo número de secuencia se descarta, también se descartan los mensajes con un número de secuencia previamente recibido.

Las tareas involucradas en esta prueba son:

- Envío de mensajes de datos *unicast*
- Recepción de mensajes de datos *unicast*
- Ruteo de mensajes de datos *unicast*

Topología	
	

Condiciones iníciales	
1	El nodo *sink* inicia la red con el PanID 0xFEED
2	Se encuentran conectados dos nodos en la red y el nodo *sink.*
3	El nodo A esta programado para llevar un control de la cantidad de mensajes correctos recibidos.
4	El nodo B está programado solo para actuar como enrutador
5	El nodo C está programado para enviar mensajes a el nodo *sink*

Procedimiento		
Paso	Descripción	Salida
1	El nodo C envía un mensaje de 20 bytes al nodo A. Cada byte es el número de mensaje enviado.	
2	El nodo A recibe el mensaje y verifica que todos los bytes sean iguales. Si es así, incrementa la cantidad de mensajes recibidos correctamente.	1
3	Después de 5, 10, 20, 40 o 50mSeg el nodo B envía el siguiente mensaje	
4	Una vez terminada la transmisión de los 1000 mensajes cambiar el intervalo de transmisión de mensajes a 5, 10, 20, 40 o 50mSeg según sea el caso.	
5	Cambiar el intervalo de envío de mensajes de control a 10mSeg, 20mSeg, 2Seg y 4Seg según sea el caso.	

Salida	
1	Cantidad de mensajes recibidos correctamente

Características relacionadas con las pruebas
Se verifica como afecta la integridad de los datos de acuerdo a las variaciones en la frecuencia de envío de mensajes de datos y de control. También que tanto afecta en el la transferencia efectiva para envío de datos.

Resultado

Las gráficas que mostrada en la Figura 19 corresponden a los resultados obtenidos en las pruebas realizadas, siendo posible la apreciación del número de mensajes recibidos por A. En las primeras dos gráficas, correspondientes a los intervalos de envío de datos de 5mSeg y 10mSeg, se puede observar una pérdida significativa de los mensajes enviados de hasta el 98 % de un total de 1000 mensajes. Se observó que ocurrió pérdida de mensajes en el nodo emisor debido a la espera del reconocimiento enviado por la capa de enlace de datos y la existencia de un límite de mensajes enviados sin reconocimiento, establecido el cual es de 4 mensajes. No se registró la perdida ocurrida en el nodo B, si la hubo, por que el protocolo sólo porporciona deteccion de errores local.

Al igual que en la primer prueba, no se observa cambio significativo en las diferentes frecuencia de envío de mensajes de control. Esto nos indica que la pérdida de datos es ocasionada principalmente por la frecuencia de envío de mensajes de datos que saturan el *buffer* de salida.

IV.2.3. Tiempo de respuesta en un salto

Descripción general: Esta prueba se enfoca en medir el tiempo de respuesta a través de un salto en la red. Para esto se enviarán mensajes y se medirá el tiempo que tarda en

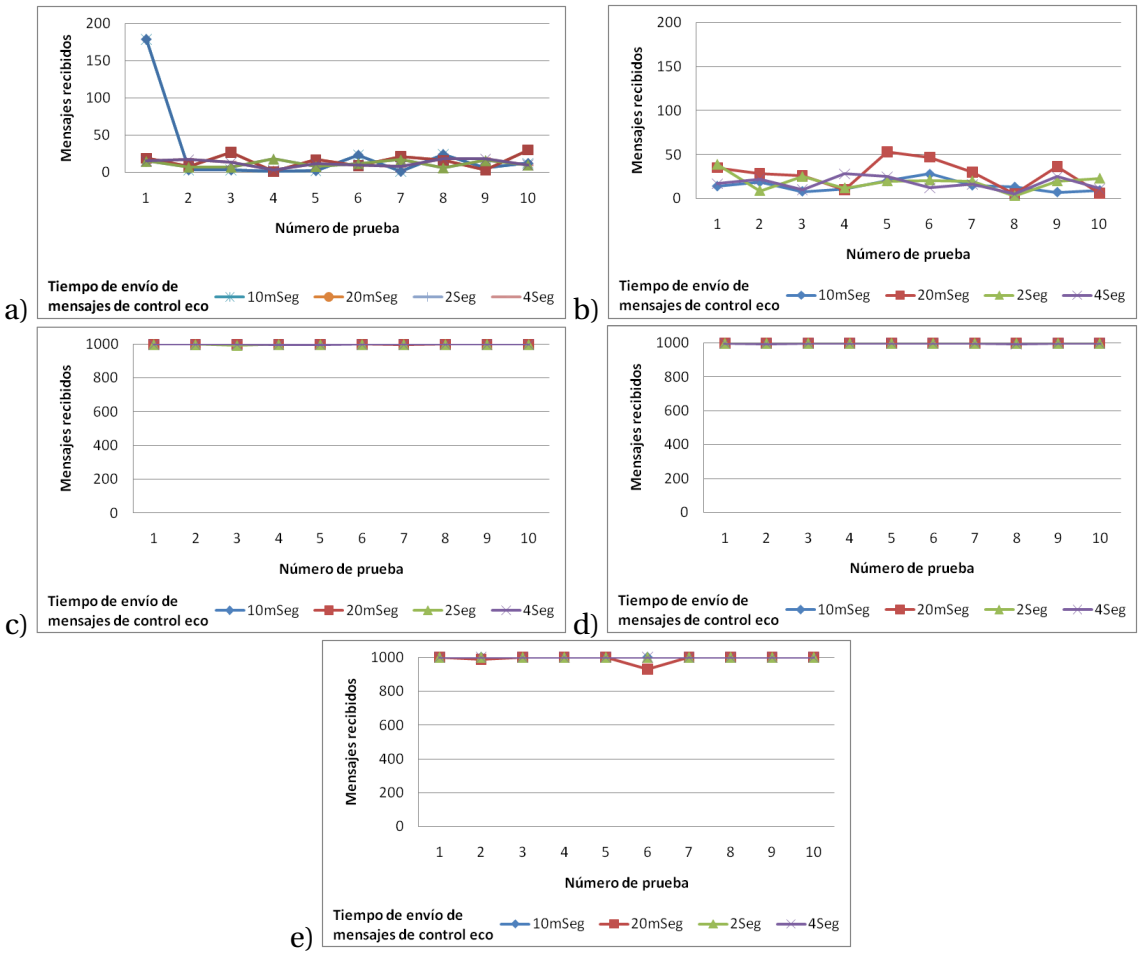


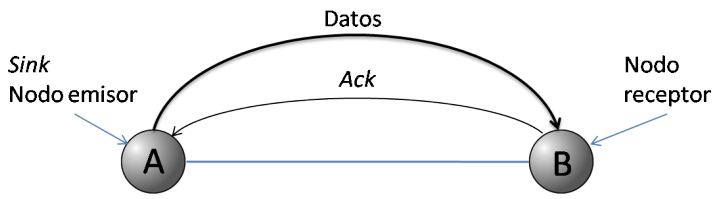
Figura 19: **Resultados de prueba de integridad de los datos en múltiple saltos. Las gráficas corresponden a las diferentes intervalo de envío de mensajes de datos: a) 5mSeg, b) 10mSeg, c) 20mSeg, d) 40mSeg y e) 50mSeg.**

responder el nodo al que se le envío el mensaje. La prueba se realizará para cuatro diferentes intervalos de transmisión de mensajes de control eco, 10mSeg, 20mSeg, 2Seg y 4Seg y se enviarán mensajes de datos de 20 bytes hasta completar 1000 mensajes. Cada mensaje contendrá el número de mensaje en cada byte y el nodo destino debe contestar con un mensaje de reconocimiento con la suma de los 20 bytes recibidos.

Las tareas involucradas en esta prueba son:

- Envío de mensajes de datos *unicast*

- Recepción de mensajes de datos *unicast*

Topología	
	

Condiciones iníciales	
1	El nodo *sink* inicia la red con el PanID 0xFEED
2	Se encuentra conectado un nodo en la red y el nodo *sink.*
3	El nodo A esta programado para llevar un control del tiempo promedio entre el envío de los datos y la recepción del reconocimiento. Si no se recibe reconocimiento después de 5 segundos, se repite el envío de datos.
4	El nodo B está programado para recibir mensajes de datos y enviar un mensaje de reconocimiento de cada uno de estos.

Procedimiento		
Paso	Descripción	Salida
1	El nodo A envía un mensaje de 20 bytes al nodo B. Cada byte es el número de mensaje enviado.	
2	El nodo B recibe el mensaje y verifica que todos los bytes sean iguales. Si es así, realiza la suma de los 20 bytes y lo envía al nodo A.	
3	El nodo A recibe el mensaje de reconocimiento y calcula el tiempo promedio de respuesta.	1
4	Una vez terminada la transmisión de los 1000 mensajes cambiar el intervalo de transmisión de mensajes de control a 10mSeg, 20mSeg, 2Seg y 4Seg según sea el caso.	

Salida	
1	Tiempo promedio de respuesta.

Características relacionadas con las pruebas
Verifica como afecta el tiempo de envío de mensajes de control en el tiempo promedio de respuesta.

Resultado

La gráfica presentada en la Figura 20 corresponde a los tiempos promedios de respuesta obtenidos de las 1000 pruebas realizadas en diferentes intervalos de tiempo de envío de mensajes de control. En la gráfica se puede observar una disminución del tiempo promedio de respuesta a mayor tiempo de envío de mensajes de control eco.

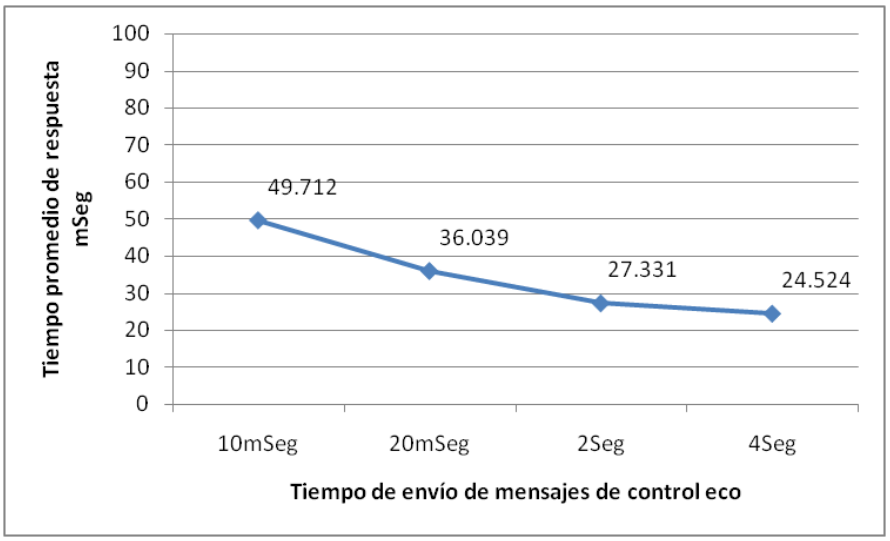


Figura 20: **Tiempos de respuesta promedios para un salto.**

De acuerdo al análisis de varianza (apéndice B) realizado se rechaza la afirmación de tener medias de tiempos de respuesta iguales para diferentes intervalos de envío de mensajes de control. Acorde con esto, a mayor intervalo de envío de mensajes de control el tiempo de respuesta es menor.

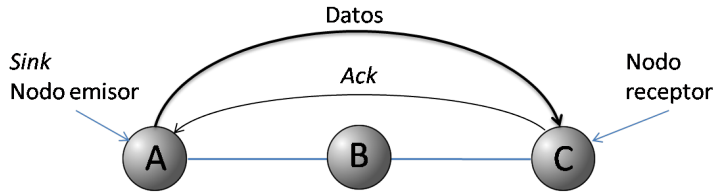
Los resultados son comparables a los obtenidos por (Pinedo˜Frausto, 2008) acerca de la implementación de ZigBee versión 2006 de Freescale, en la cual se obtiene un tiempo de respuesta de 48mSeg para un salto. Aunque para la implementación del protocolo ligero propuesto se obtienen promedios de respuesta menores de 50mSeg el tiempo de respuesta mayor fue de hasta 1.094Seg, el cual es mucho mayor a los 67mSeg obtenidos de la implementación de ZigBee. Por otro lado, el tiempo mínimo de respuesta de la implementación de ZigBee de 44mSeg es más de cinco veces el mínimo obtenido por la implementación del protocolo ligero de 8.224mSeg (vea apéndice B).

IV.2.4. Tiempo de respuesta en múltiple saltos

Descripción general: Esta prueba se enfoca en medir el tiempo de respuesta a través de dos saltos en la red. Para esto se enviarán mensajes y se medirá el tiempo que tarda en responder el nodo al que se le envió el mensaje. La prueba se realizará para cuatro diferentes intervalos de transmisión de mensajes de control eco, 10mSeg, 20mSeg, 2Seg y 4Seg y se enviarán mensajes de datos de 20 bytes hasta completar 1000 mensajes. Cada mensaje contendrá el número de mensaje en cada byte y el nodo destino debe contestar con un mensaje de reconocimiento con la suma de los 20 bytes recibidos.

Las tareas involucradas en esta prueba son:

- Envío de mensajes de datos *unicast*.
- Recepción de mensajes de datos *unicast*.
- Ruteo de mensajes de datos *unicast*.

Topología	
	

Condiciones iníciales	
1	El nodo *sink* inicia la red con el PanID 0xFEED.
2	Se encuentran conectados dos nodos en la red y el nodo *sink.*
3	El nodo A esta programado para llevar un control del tiempo promedio entre el envío de los datos y la recepción del reconocimiento. Si no se recibe reconocimiento después de 5 segundos, se repite el envío de datos.
4	El nodo C está programado para recibir mensajes de datos y enviar un mensaje de reconocimiento de cada uno de estos.

Procedimiento		
Paso	Descripción	Salida
1	El nodo A envía un mensaje de 20 bytes al nodo C. Cada byte es el número de mensaje enviado.	
2	El nodo C recibe el mensaje y verifica que todos los bytes sean iguales. Si es así, realiza la suma de los 20 bytes y lo envía al nodo A.	
3	El nodo A recibe el mensaje de reconocimiento y calcula el tiempo promedio de respuesta.	1
4	Una vez terminada la transmisión de los 1000 mensajes cambiar el intervalo de transmisión de mensajes de control a 10mSeg, 20mSeg, 2Seg y 4Seg según sea el caso.	

Salida	
1	Tiempo promedio de respuesta.

Características relacionadas con las pruebas
Verifica como afecta el tiempo de envío de mensajes de control en el tiempo promedio de respuesta.

Resultado

La gráfica presentada en Figura 21 corresponde a los tiempos promedios de respuesta obtenidos de las 1000 pruebas realizadas en diferentes intervalos de tiempo de envío de mensajes de control. En la gráfica puede observarse una disminución del tiempo promedio de respuesta a mayor tiempo de envío de mensajes de control eco.

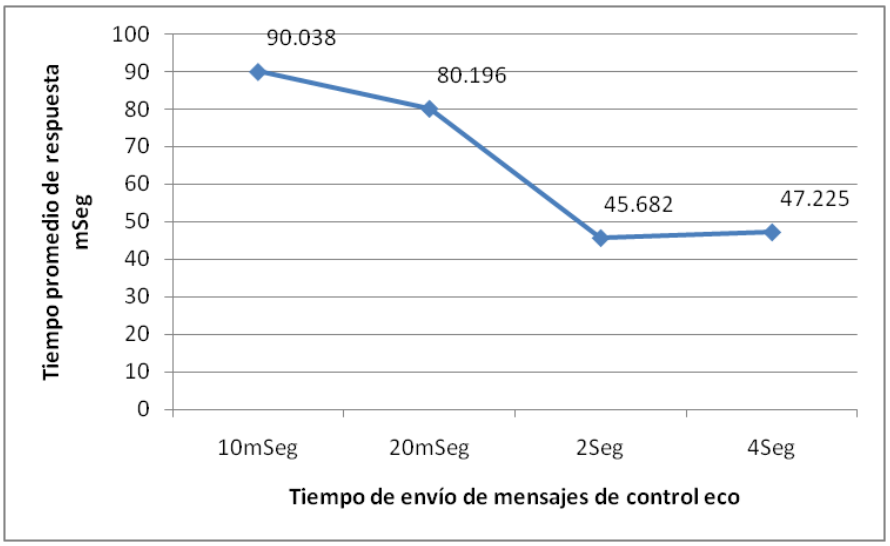


Figura 21: **Tiempos de respuesta promedios para dos saltos.**

De la misma manera que en la prueba de tiempo de respuesta en un salto, con el análisis de varianza realizado (apéndice B) se rechaza la afirmación de tener medias de tiempos de respuesta iguales para diferentes intervalos de envío de mensajes de control. Acorde con esto, a mayor intervalo de envío de mensajes de control el tiempo de respuesta es menor

hasta llegar a estabilizarse alrededor de los 46mSeg que se alcanzan a partir de los 2Seg en el intervalo de tiempo de envío de mensajes de control en las pruebas realizadas.

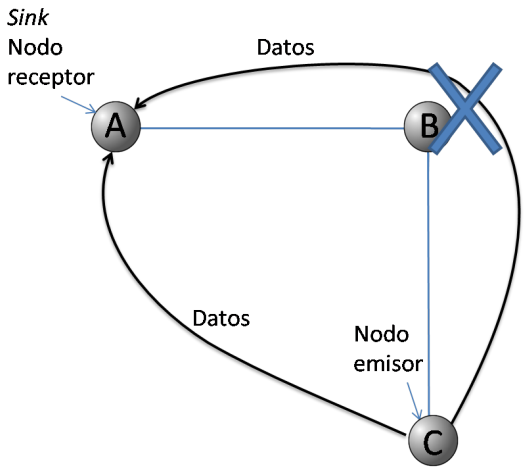
El tiempo de respuesta para dos saltos con ZigBee es mayor a 50mSeg (Pinedo˜Frausto, 2008), el cual se mejora utilizando intervalos de envío de mensajes de control mayor a 2Seg. Sin embargo, este tiempo llegó hasta 90.038mSeg en promedio, utilizando un intervalo de envío de mensajes de control de 10mSeg.

IV.2.5. Tiempo de recuperación de ruta

Descripción general: Esta prueba se centra en medir el tiempo que tarda en recuperar la conexión a la red. Para esto se formará la red con tres nodos, se estarán enviando mensajes a través del nodo intermedio, se eliminará ese nodo y se observará el tiempo consumido entre cada mensaje recibido. La prueba se realizará para cuatro diferentes intervalos de transmisión de mensajes de control eco, 10mSeg, 20mSeg, 2Seg y 4Seg y se enviarán mensajes de datos de 20 bytes con el número de mensaje en cada 2 bytes.

Las tareas relacionadas con esta prueba son:

- Asociación a la red.
- Asignación de direcciones.
- Envío de mensajes de datos *unicast*.
- Recepción de mensajes de datos *unicast*.
- Ruteo de mensajes de datos *unicast*.
- Envío de mensajes de control eco.
- Envío de mensajes de control respuesta de eco.
- Envío de mensajes de pánico.

Topología	
	

Condiciones iníciales	
1	El nodo *sink* inicia la red con el PanID 0xFEED
2	Se encuentran conectados dos nodos en la red y el nodo *sink*. Primero se conecta el nodo B y a él se conecta el nodo C.
3	El nodo A esta programado para llevar registro del tiempo entre el último mensaje de la primera dirección asignada al nodo C y el primer mensaje con la dirección final asignada a C.
4	El nodo C está programado para enviar mensajes de datos cada 20mSeg.

Procedimiento		
Paso	Descripción	Salida
1	El nodo C envía mensajes cada 20mSeg al nodo A.	
2	El nodo A recibe el mensaje de datos y registra el tiempo en el que lo recibió.	
3	El nodo B es apagado.	
4	El nodo C se desconecta de la red y se reconecta automáticamente.	
5	El nodo A recibe un mensaje de datos del nodo C desde una nueva dirección y registra el tiempo transcurrido entre el último mensaje desde la dirección antigua de C y el primer mensaje desde la nueva dirección.	1
6	Repetir la prueba 10 veces para 10mSeg, 20mSeg, 2Seg y 4Seg como intervalo de envío de mensajes de control.	

Salida	
1	Tiempo de recuperación de conexión.

Características relacionadas con las pruebas
Esta prueba permite medir el tiempo que tarda un nodo en recuperar la conexión a la red en diferentes intervalos de envío de mensajes de control.

Resultado

La gráfica de la Figura 22 muestra los tiempos de respuesta promedios con diferentes intervalos de envío de mensajes de control eco. Se observa que existe una relación entre el tiempo de envío de mensajes de control eco y el tiempo de recuperación de la conexión. Esto es debido a la espera producida para de detectar la ausencia del nodo padre utilizado para conectarse a la red.

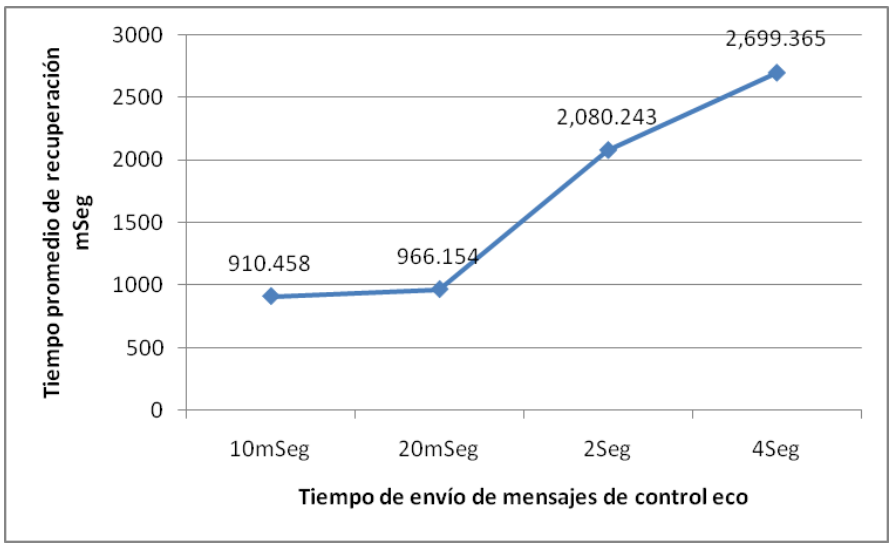


Figura 22: **Tiempos promedios de recuperación de conexión.**

Por medio del análisis de varianza de los resultados obtenidos de la realización de esta prueba (apéndice B), se rechaza la afirmación de tener tiempos de recuperación iguales para diferentes intervalos de transmisión de mensajes de control eco.

El tiempo de recuperación de ruta es mayor que el observado en la implementación de ZigBee, el cual fue en promedio de 120mSeg comparado con los 910.458Seg obtenidos de la implementación del protocolo ligero propuesto, con un intervalo de envío de mensajes de control de 10mSeg. Esto es debido a la espera para la reconexión de la capa de enlace de datos, la cual es necesaria para obtener una nueva dirección que identifique al nodo.

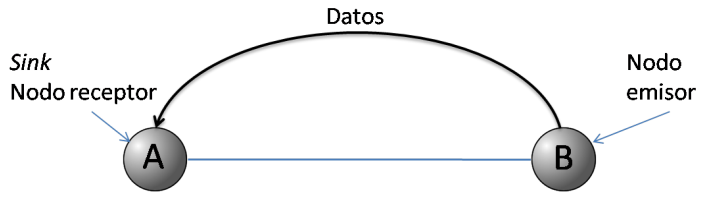
IV.2.6. Tiempo de formación de red con dos nodos

Descripción general: Esta prueba se centra en medir el tiempo empleado para formar la red, para esto se tendrá un nodo intentando enviar datos al nodo *sink*, se tomará el tiempo desde que se inicia la capa de red hasta que se recibe un mensaje con dirección origen 0x1000. Los mensajes enviados son de 20 bytes con el número de mensaje

en cada 2 bytes. La prueba se realizará para cuatro diferentes intervalos de transmisión de mensajes de control eco, 10mSeg, 20mSeg, 2Seg y 4Seg.

Las tareas relacionadas con esta prueba son:

- Asociación a la red.
- Asignación de direcciones.
- Envío de mensajes de datos *unicast*.
- Recepción de mensajes de datos *unicast*.

Topología	
	

Condiciones iníciales	
1	El nodo B está programado para iniciar la capa de red e intentar enviar mensajes al nodo *sink*.
2	El nodo A es el nodo *sink* y está programado para llevar registro del tiempo entre el inicio de la capa de red y la recepción del primer mensaje del nodo B.

Procedimiento		
Paso	Descripción	Salida
1	El nodo B inicia la capa de red y comienza a enviar mensajes al nodo A.	
2	El nodo A comienza la capa de red y registra el momento en el que lo hizo.	
3	El nodo A recibe un mensaje del nodo B y registra el tiempo transcurrido desde el inicio de la capa de red.	1
4	Repetir la prueba 10 veces para obtener un tiempo promedio para 10mSeg, 20mSeg, 2Seg y 4Seg como intervalo de envío de mensajes de control.	

Salida	
	Tiempo transcurrido desde el inicio de la capa de red hasta el primer mensaje recibido.

Características relacionadas con las pruebas
Esta prueba permite medir el tiempo empleado para formar la red con un nodo y el nodo *sink*.

Resultado

En la Figura 23 se muestra la gráfica de los tiempos promedios de conexión entre dos nodos con diferentes intervalos de envío de mensajes de control eco. No se aprecia variación significativa entre estos tiempos, obteniendo el mayor promedio en tiempo de conexión para el intervalo de tiempo de envío de mensajes de control de 20mSeg y el menor para el intervalo de envío de mensajes de control de 4mSeg.

Si bien el promedio de los datos indica que se tiene mayor tiempo de conexión utilizando un intervalo de envío de mensajes de control de 20mSeg, a través del análisis de

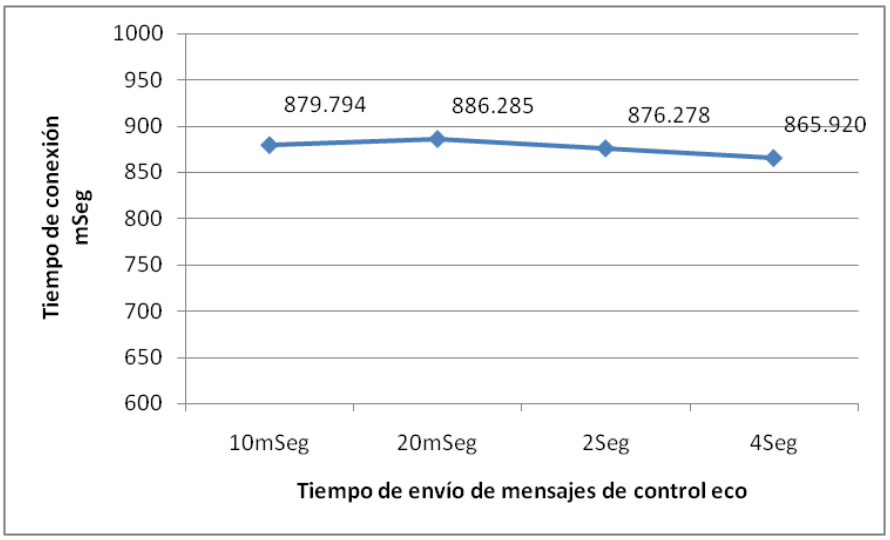


Figura 23: **Tiempos promedios de conexión con dos nodos.**

varianza de los datos (apéndice B), se puede concluir que los tiempos de conexión con los diferentes intervalos de envío de mensajes de control son iguales.

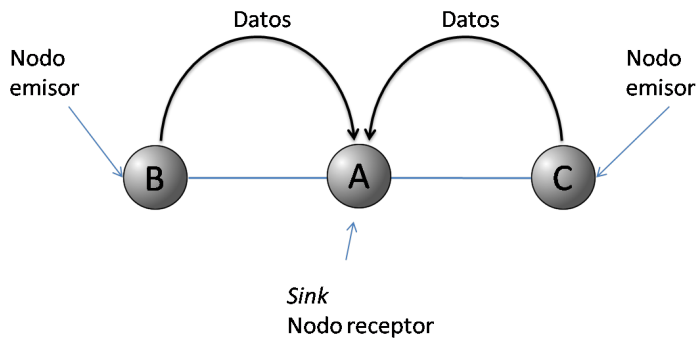
El tiempo de conexión con un nodo hijo es 2.5 veces menor a los obtenidos con ZigBee, donde solo existe un dispositivo coordinador, con una topología en estrella y en el cual se obtuvieron tiempos de alrededor de 2.54Seg comparado con el tiempo promedio de conexión de 886.285mSeg utilizando un intervalo de tiempo de envío de mensajes de control de 20mSeg. El tiempo máximo de conexión observado fue de 941.648mSeg.

IV.2.7. Tiempo de formación de red con tres nodos

Descripción general: Esta prueba se centra en medir el tiempo empleado para formar la red, para esto se tendrán dos nodos intentando enviar datos al nodo sink, se tomará el tiempo desde que se inicia la capa de red hasta que se reciben los primeros mensajes del nodo B y del nodo C. Los mensajes enviados son de 20 bytes. La prueba se realizará para cuatro diferentes intervalos de transmisión de mensajes de control eco, 10mSeg, 20mSeg, 2Seg y 4Seg.

Las tareas relacionadas con esta prueba son:

- Asociación a la red.
- Asignación de direcciones.
- Envío de mensajes de datos *unicast*.
- Recepción de mensajes de datos *unicast*.

Topología	
	

Condiciones iníciales	
1	El nodo B y C están programados para iniciar la capa de red e intentar enviar mensajes al nodo *sink*.
2	El nodo A es el nodo *sink* y está programado para llevar registro del tiempo entre el inicio de la capa de red y la recepción del primer mensaje del nodo B o el nodo C.

Procedimiento		
Paso	Descripción	Salida
1	El nodo B y C inician la capa de red y comienzan a enviar mensajes al nodo A.	
2	El nodo A comienza la capa de red y registra el momento en el que lo hizo.	
3	El nodo A recibe un mensaje del nodo B y registra el tiempo transcurrido desde el inicio de la capa de red.	1
4	Repetir la prueba 10 veces para obtener un tiempo promedio para 10mSeg, 20mSeg, 2Seg y 4Seg como intervalo de envío de mensajes de control.	

Salida	
1	Tiempo transcurrido desde el inicio de la capa de red hasta el primer mensaje recibido.

Características relacionadas con las pruebas
Esta prueba permite medir el tiempo empleado para formar la red con dos nodos y el nodo *sink*.

Resultado

En la Figura 24 se muestra la gráfica de los tiempos promedios de conexión entre tres nodos con diferentes intervalos de envío de mensajes de control eco. Al igual que en la prueba seis en la que se utilizan dos nodos, no se aprecia una variación significantiva entre estos tiempos, obteniendo el mayor promedio en tiempo de conexión para el intervalo de tiempo de envío de mensajes de control de 20mSeg y el menor para el intervalo de envío de mensajes de control de 4mSeg.

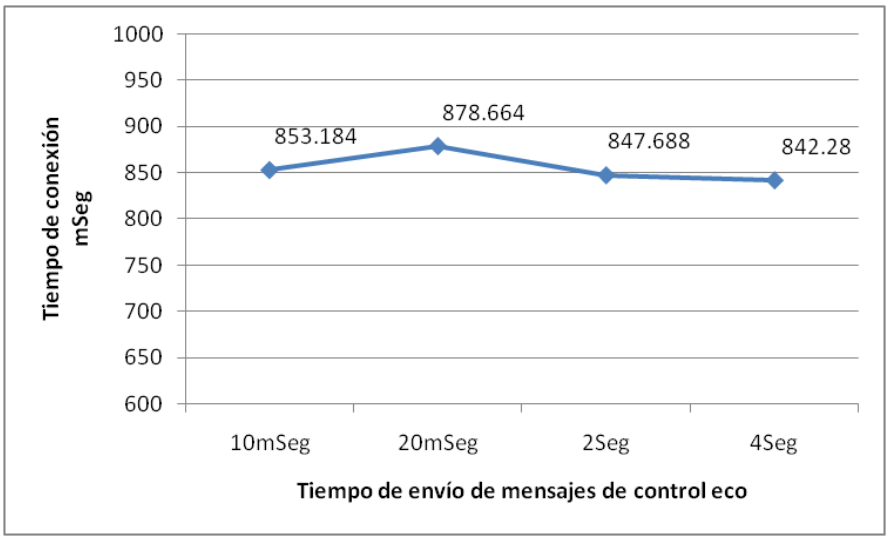


Figura 24: **Tiempos promedios de conexión con tres nodos.**

Al igual que en la prueba seis, realizando un análisis de varianza de los datos (apéndice B), se concluye que los tiempos de conexión con los diferentes intervalos de envío de mensajes de control son iguales, aunque el tiempo promedio de conexión utilizando un intervalo de envío de mensajes de control de 20mSeg sea mayor.

El tiempo de conexión para la implementación del protocolo ligero con dos nodo conectados es casi tres veces menor a los obtenidos con ZigBee donde solo existe un dispositivo coordinador, con una topología en estrella y en el cual se obtuvieron tiempos de alrededor de 2.543Seg, comparado con el tiempo promedio de conexión de 878.664mSeg obtenido utilizando un intervalo de tiempo de envío de mensajes de control de 20mSeg. El tiempo máximo de conexión observado fue de 1534.24mSeg.

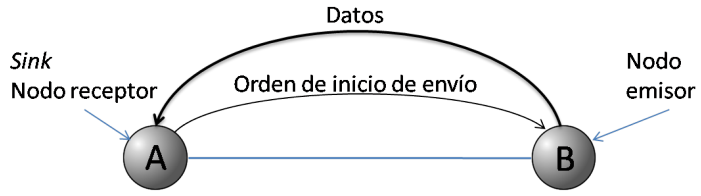
IV.2.8. Entrega exitosa de datos para un salto

Descripción general: Esta prueba se centra en medir el número de bytes de datos entregados con éxito en una comunicación donde se involucran únicamente dos nodos

en la red. El nodo hijo esperará por un mensaje del nodo *sink* que indica que se puede empezar a transmitir los datos, cuando el nodo lo recibe transmite los datos que constan de 1000 mensajes de 80 bytes cada uno en intervalos de 10 mSeg. El nodo *sink* almacena el tiempo transcurrido entre el envío del mensaje que indica el inicio de la transmisión de datos y el último mensaje de datos del nodo conectado. Cada mensaje contiene en cada 2 bytes el número de mensaje y la prueba se realizará para cuatro diferentes intervalos de transmisión de mensajes de control eco, 10mSeg, 20mSeg, 2Seg y 4Seg.

Las tareas relacionadas con esta prueba son:

- Envío de mensajes de datos *unicast*
- Recepción de mensajes de datos *unicast*

Topología	
	

Condiciones iníciales	
1	El nodo A inicia la red y se conecta a él los nodos B, C, D, E, F y G.
2	Los nodos B, C, D, E, F y G están programados para iniciar la capa de red y esperar por un mensaje del nodo *sink* para empezar a enviar los mensajes de datos.
3	El nodo A es el nodo *sink* y está programado para enviar un mensaje de datos y tomar el tiempo empleado para recibir los mensajes de los nodos conectados.

Procedimiento		
Paso	Descripción	Salida
1	El nodo A envía un mensaje de datos en *broadcast.*	1
2	Los nodos reciben el mensaje de A y envían los mensajes de datos	
3	El nodo A cuenta el número de mensajes recibidos de cada nodo	2
4	El nodo A toma el tiempo en el que finaliza la transmisión de los mensajes de cada nodo	3

Salida	
1	Tiempo de inicio de transmisión de mensajes
2	Número de mensajes recibidos
3	Tiempo de fin de transmisión de mensajes

Características relacionadas con las pruebas
Esta prueba permite medir la transferencia efectiva permitida para datos.

Resultado

En la gráfica que se muestra en la Figura 25 se despliega los bits por segundo en promedio obtenidos de las diez pruebas realizadas para cada intervalo de envío de mensajes de datos, las cuales constan del envío de 1000 mensajes cada uno.

Como puede apreciarse en la gráfica y de acuerdo al análisis de varianza de los datos obtenidos (apéndice B), se acepta la afirmación de no haber variación en los bits por segundo para transmisión de datos entre dos nodos utilizando los diferentes intervalos de tiempo para envío de mensajes de control eco.

El número de bytes de datos entregados con éxito obtenido de la implementación del protocolo es 5.7 veces mayor que los 13.4kbps obtenidos con la implementación de ZigBee,

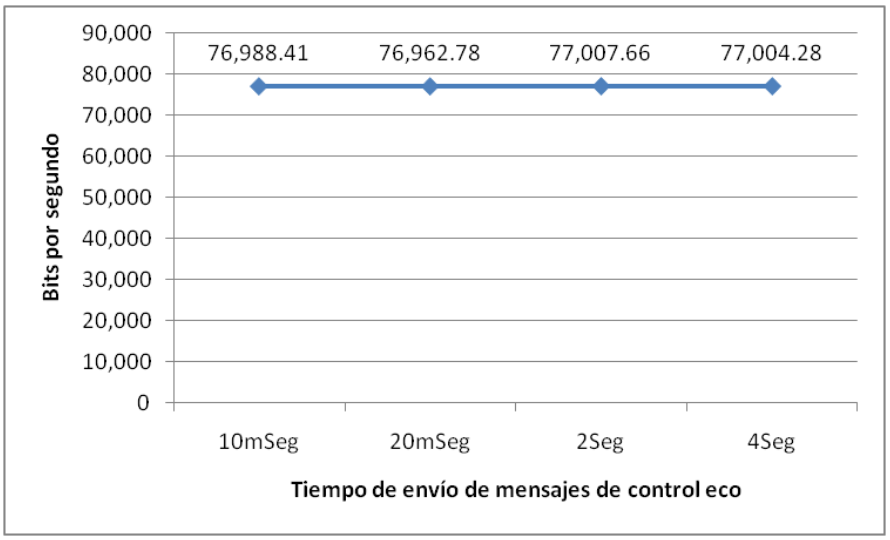


Figura 25: **Promedios de entrega exitosa para un salto.**

donde se utiliza la carga máxima de 80 bytes (Pinedo˜Frausto, 2008).

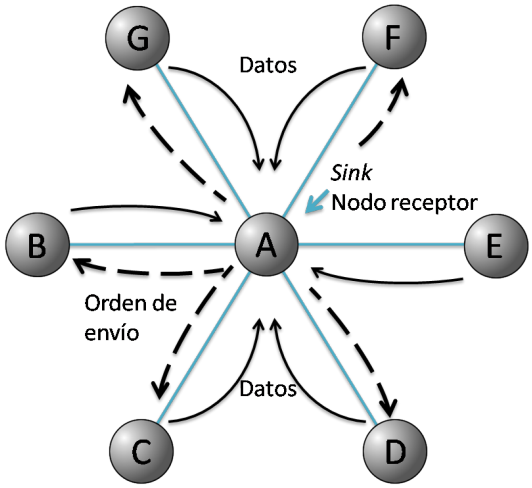
IV.2.9. Entrega exitosa de datos para estrella

Descripción general: Esta prueba se centra en medir el número de bytes de datos entregados con éxito en una comunicación donde se encuentran seis nodos a un salto del *sink*. Los nodos esperarán por un mensaje del nodo *sink* que indica que se puede empezar a transmitir los datos, cuando los nodos lo reciben transmiten mensajes de datos en intervalos de 140mSeg, cada uno consta de 80 bytes. El nodo *sink* almacena el tiempo transcurrido entre el mensaje que indica el inicio de la transmisión de datos y el mensaje de datos número 100 de todos los mensajes recibidos. Cada mensaje contiene en cada 2 bytes el número de mensaje y la prueba se realizará para cuatro diferentes intervalos de transmisión de mensajes de control eco, 10mSeg, 20mSeg, 2Seg y 4Seg.

Las tareas relacionadas con esta prueba son:

- Envío de mensajes de datos *unicast*.

- Recepción de mensajes de datos *unicast*.
- Envío de mensajes de datos *broadcast*.

Topología	
	

Condiciones iníciales	
1	El nodo A inicia la red y se conecta a él los nodos B, C, D, E, F y G.
2	Los nodos B, C, D, E, F y G están programados para iniciar la capa de red y esperar por un mensaje del nodo *sink* para empezar a enviar los mensajes de datos.
3	El nodo A es el nodo *sink* y está programado para enviar un mensaje de datos y tomar el tiempo empleado para recibir los mensajes de los nodos conectados.

Procedimiento		
Paso	Descripción	Salida
1	El nodo A envía la orden de inicio con un mensaje de datos en *broadcast.*	1
2	Los nodos reciben el mensaje de A y envían los mensajes de datos	
3	El nodo A cuenta el número de mensajes recibidos de cada nodo	2
4	El nodo A toma el tiempo en el que finaliza la transmisión de los mensajes de cada nodo	3

Salida	
1	Tiempo de inicio de transmisión de mensajes
2	Número de mensajes recibidos
3	Tiempo de fin de transmisión de mensajes

Características relacionadas con las pruebas
Esta prueba permite medir la transferencia efectiva permitido para datos.

Resultado

La gráfica que se muestra en la Figura 26 despliega los bits por segundo en promedio recibidos por el nodo *sink*, este es el resultado obtenido de las diez pruebas realizadas para cada intervalo de envío de mensajes de datos.

De acuerdo con la gráfica de la Figura 26 y al análisis de varianza de los datos obtenidos (apéndice B), se puede afirmar que existe variación en el número de bytes de datos que llegan por segundo utilizando diferentes intervalos de envío de mensajes de control eco.

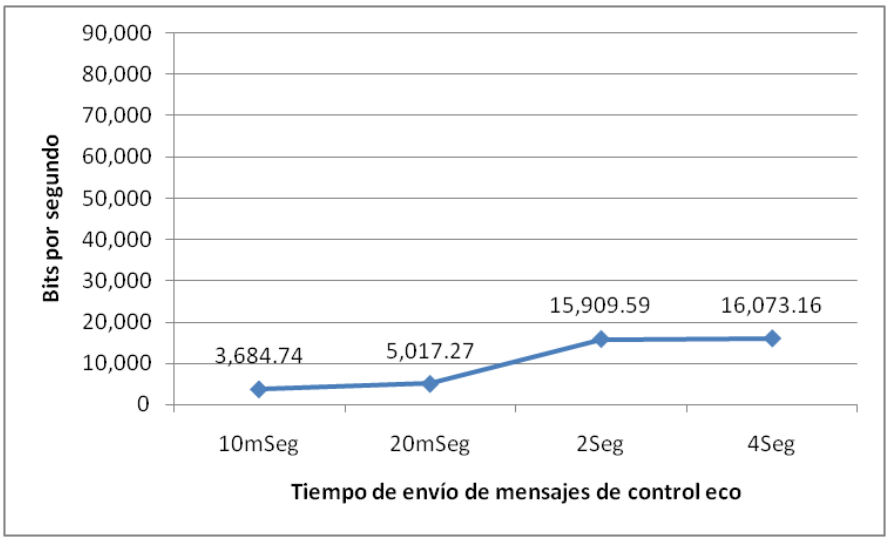


Figura 26: **Promedios de entrega exitosa para estrella.**

IV.2.10. Consumo de energía

En esta sección se realiza un análisis con las variables más importantes que influyen en la potencia requerida para mantener a un dispositivo que implementa el protocolo ligero de red, variando los intervalos de envío de mensajes de control eco. Se asume que no hay perdida en el medio físico, los mensajes son enviados una vez por la capa de red y siempre son entregados a su destino, a consecuencia de esto no hay desconexiones que implique reiniciar el procedimiento de asociación a la red.

La potencia total requerida es la suma de la potencia necesaria para envío y recepción de mensajes de control y la potencia empleada en el tiempo de espera, por lo tanto:

$$P = P_{Trabajando} * Porcentaje_{TiempoTrab} + P_{Ocioso} * Porcentaje_{TiempoOcioso} \quad (1)$$

donde:

$P_{Trabajando}$ es la energía requerida por el nodo cuando se encuentra realizando tareas de mantenimiento de la red, éstas son el envío y recepción de mensajes de control eco y respuesta de eco.

$Porcentaje_{TiempoTrab}$ es el porcentaje del tiempo total que consume el nodo en realizar tareas de mantenimiento de la conexión.

P_{Ocioso} es la energía requerida por el nodo cuando se encuentra sin hacer tareas referentes al mantenimiento de la red.

$Porcentaje_{TiempoOcioso}$ es el porcentaje del tiempo total que consume el nodo sin hacer operaciones para el mantenimiento de la capa de red.

La energía requerida por el nodo para realizar tareas de mantenimiento de la conexión (el envío y recepción de mensajes de control eco y respuesta de eco) está dado por:

$$P_{Trabajando} = P_{MCU} + P_{TXRX} + P_{Perifericos} \tag{2}$$

donde:

P_{MCU} es la potencia requerida por el procesador cuando se encuentra en ciclos de trabajo.

P_{TXRX} es la potencia requerida por el transceptor cuando se esta utilizando. Esta dada por:

$$P_{TXRX} = \frac{1}{2}(P_{TX}) + \frac{1}{2}(P_{RX}) \tag{3}$$

donde P_{TX} es la potencia necesaria para transmitir y P_{RX} es la potencia requerida para recibir. Se multiplican por $\frac{1}{2}$ debido a que para cada mensaje de control eco transmitido se espera recibir un mensaje de control respuesta de eco con la misma longitud, por lo que se emplea la mitad del tiempo transmitiendo y la otra mitad recibiendo.

$P_{Perifericos}$ es la potencia requerida para mantener los puertos o dispositivos periféricos trabajando.

La potencia requerida por el nodo cuando se encuentra en espera de realizar tareas de mantenimiento de la conexión está dado por:

$$P_{Ocioso} = P_{MCUocioso} + P_{TXRXocioso} + P_{PerifericosOcioso} \tag{4}$$

donde:

$P_{MCUocioso}$ es la potencia requerida por el microcontrolador cuando el nodo se encuentra en espera de realizar tareas de mantenimiento de la conexión.

$P_{TXRXocioso}$ es la potencia requerida por el transceptor cuando el nodo se encuentra en espera de realizar tareas de mantenimiento de la conexión.

$P_{PerifericosOcioso}$ es la potencia requerida por para mantener los puertos o dispositivos periféricos trabajando cuando el nodo se encuentra en espera de realizar tareas de mantenimiento de la conexión.

El porcentaje del tiempo en el que el nodo se encuentra realizando tareas de mantenimiento se obtiene de:

$$Porcentaje_{TiempoTrab} = t_{TX}\,(NumHijos + 1)\,\frac{1}{T_{Msj}} + t_{RX}\,(NumHijos + 1)\,\frac{1}{T_{Msj}} \tag{5}$$

donde:

t_{TX} es el tiempo necesario para el envío de un mensaje de control.

t_{RX} es el tiempo necesario para la recepción de un mensaje de control.

$NumHijos$ es el número esperado de dispositivos conectados al nodo analizado.

T_{Msj} es el intervalo de tiempo entre el envío de los mensajes de control en segundos.

El porcentaje del tiempo total que consume el nodo sin hacer operaciones para el mantenimiento de la capa de red se obtiene de:

$$Porcentaje_{TiempoOcioso} = 1 - Porcentaje_{TiempoTrab} \tag{6}$$

donde:

$Porcentaje_{TiempoTrab}$ es el porcentaje del tiempo en el que el nodo se encuentra realizando tareas de mantenimiento.

A continuación se realiza un análisis de la potencia requerida para mantenimiento de la conexión con los dispositivos utilizados en el desarrollo del prototipo. Estos dispositivos son MC13192 *Sensor Applications Reference Design* (SARD) los cuales cuentan con las siguientes características eléctricas (MC13192):

- Potencia requerida por la MCU es de 17.55mW en modo de espera y realizando tareas de mantenimiento de capa de red.
- Potencia típica requerida para transmitir 81mW.
- Potencia típica requerida para recibir 99.9mW.
- Potencia requerida por el transceptor en modo de espera es de 1.35mW.
- Para este ejemplo se supone que no hay periféricos, entonces la potencia requerida para estos dispositivos es igual a 0 W.

El tiempo teórico para transmitir un mensaje de control que consta de 192 bits es de 912μSeg, por que se utilizan 250kbps y se emplea un tiempo de transición del estado en espera a transmitir de 144μSeg. Este también es utilizado como el tiempo necesario para recibir un mensaje. En los 192 bits se incluyen los bits de encabezado de la capa de enlace de datos, la capa física y los 5 bytes del mensaje de control de la capa de red. El número esperado de dispositivos conectados a la red a través del nodo son 4, utilizando un tiempo entre cada envío de mensajes de control de 10mSeg, 20mSeg, 2Seg y 4Seg como en las

pruebas anteriores. Con estas características se obtiene los resultados de la Tabla I.

Tabla I: **Potencia requerida por los dispositivos.**

Intervalo entre mensajes de control	Potencia Requerida
• 10mSeg	100.16 mW
• 20mSeg	59.53 mW
• 2Seg	19.31 mW
• 4Seg	19.10 mW

Para que estos resultados sean más palpables, en la Tabla II se muestra el tiempo de vida esperado con la utilización de baterías con capacidades de 1500mAH y 3000mAH, por ser estos los amperajes que se aproximan a las baterías AA comunes.

Tabla II: **Días de vida útil con baterías de 1500mAH y 3000mAH.**

Intervalo entre	Días de funcionamiento	
mensajes de control	1500mAH	3000mAH
• 10mSeg	1.65	3.37
• 20mSeg	2.83	5.67
• 2Seg	8.74	17.48
• 4Seg	8.83	17.68

Se puede concluir que el tiempo de operación de los dispositivos esta limitado principalmente por la potencia requerida de la MCU que se encuentra siempre activa. Aquí existe un área de oportunidad para mejorar el protocolo y extender la vida útil de los dispositivos energizados por baterías (por ejemplo, usar el modo de ahorro de energía de los radio presente en el estándar IEEE 802.15.4 o los ciclos de bajo consumo de los microcontroladores).

IV.3. Conclusiones

Las características de la especificación del protocolo propuesto, permiten comunicación entre capas de aplicación de diferentes dispositivos a través de un pequeño conjunto de directivas que conforman el API especificada por el protocolo. Esta comunicación puede ser a través de múltiples saltos haciendo uso reducido de recursos, requiriendo poca memoria para almacenamiento de programa, no necesita mensajes de descubrimiento de rutas y administración de números de secuencia para las rutas activas.

Con los resultados obtenidos de las pruebas realizadas podemos concluir que la implementación del protocolo permite el envío de mensajes de datos a través de múltiples saltos con espacio de tiempo entre cada mensaje de 20mSeg o hasta 4 mensajes de datos consecutivos para no saturar el *buffer* de salida en el nodo emisor.

El tiempo de respuesta promedio es dependiente del intervalo de envío de los mensajes de control, debido a la mayor utilización del canal con mensajes de control eco en menores intervalos de envío de estos mensajes. Según los resultados de las pruebas, se puede llegar a esperar hasta alrededor de un segundo por la respuesta de un paquete de datos sin importar el intervalo de envío de los mensajes de control, pero en promedio es menor a 100mSeg.

En el mejor de los casos, la recuperación de ruta es menor de un segundo por el tiempo requerido para reconectar la capa de enlace de datos, que necesita poco menos de 900mSeg. Comparados con 120mSeg requeridos para la recuperación de ruta por la implementación de ZigBee, es relativamente grande.

Para la formación de la red el tiempo se mantiene en poco menos de 900mSeg para distinto número de nodos, mientras que para la implementación de ZigBee el tiempo de

formación de la red es de 2.54Seg.

La Tabla III contiene un resumen de los valores obtenidos con la implementación del protocolo ligero propuesto y los de ZigBee. Como se esperaba por la mayor robustez proporciona por ZigBee, se tiene un mejor desempeño con el protocolo propuesto, exceptuando el tiempo de recuperación de ruta que es de 120mSeg para ZigBee.

Tabla III: **Comparación entre la implementación del protocolo propuesto y ZigBee.**

Característica	Protocolos	
	Protocolo ligero (Utilizando 20mSeg como intervalo de mensajes de control)	ZigBee
• Tiempo de respuesta	36.039mSeg	48mSeg
• Tiempo de recuperación de ruta	966.15mSeg	120mSeg
• Tiempo de formación de red	886.28mSeg	2,540mSeg
• Entrega exitosa	76.9kbps	13.4kbps

Capítulo V

Conclusiones

V.1. Conclusiones

Con la necesidad de protocolos de comunicación abiertos para redes de monitoreo y control que permitan la interconexión de dispositivos para distintas aplicaciones con bajos requerimientos de transmisión de datos, se han desarrollado desde el protocolo Synkro de Freescale, que recientemente se ha abierto para ayudar a la estandarización de control en aplicaciones de entretenimiento, hasta ZigBee, un protocolo ampliamente utilizado que se actualiza para encajar en un conjunto mayor de aplicaciones.

Este trabajo de tesis provee una solución de comunicación multisalto dentro de la misma red de área personal y puede ser utilizada en aplicaciones de monitoreo y control que requieren bajas prestaciones. Estas aplicaciones pueden ser para domótica, seguridad como detectores de humo, dispositivos de entretenimiento como juguetes, monitoreo de ambientes exteriores, entre otras.

Para el desarrollo de estas aplicaciones se presenta una API que consta de los procedimientos para iniciar y finalizar la capa de red y enviar y recibir datos , esta es parecida a la utilizada por TCP/IP, la cual es conocida por una gran cantidad de programadores de

sistemas de red. La API propuesta puede facilitar la incursión de estos programadores en el ambiente de sistemas embebidos y de bajas prestaciones.

Las aplicaciones que utilicen este protocolo ligero, obtendrán un mejor desempeño que al utilizar ZigBee en cuanto a transferencia efectiva de datos se refiere, sin la robustez ofrecida por ZigBee. Por otra parte, se requiere menos memoria para almacenamiento de programas que ZigBee y dependiendo de la configuración de envío de mensajes de control será la cantidad de energía requerida.

Los objetivos planteados para esta tesis fueron cumplidos. Por medio de los objetivos específicos, identificar los servicios provistos por la capa de enlace de datos e identificar y establecer los servicios que debe proveer la capa de red, se logró alcanzar el objetivo general, diseñar e implementar un protocolo de red, ligero para redes inalámbricas de monitoreo y control que utilizan el estándar IEEE 802.15.4, cumpliendo con los requisitos establecidos: tamaño del código que requiera poco espacio en memoria; sistema de direccionamiento con soporte para hasta 1024 nodos; enrutamiento multisalto; autoformación de redes; comunicación *unicast* y *broadcast*; y proveer una interfaz para desarrollar aplicaciones sobre él.

V.2. Aportaciones

La principal aportación de este trabajo es una especificación abierta de un protocolo de red, el cual permite encaminar directamente mensajes de datos a su destino sin necesidad de tablas de rutas y la administración requerida de éstas como el uso de memoria y temporizadores para rutas. Además, la especificación está enfocada en permitir movilidad de los nodos, para satisfacer esta característica de las redes inalámbricas de monitoreo y control. La especificación del protocolo propone un modelo de asignación de direcciones que permite el envío de mensajes *broadcast* sin la utilización de números de secuencia y la

administración requerida para ellos. Este protocolo utiliza el estándar IEEE 802.15.4 que es implementado por varias plataformas de *hardware*, por lo que se tiene una puerta abierta para la implementación del protocolo en distintas plataformas.

Para el desarrollo de la especificación del protocolo se implementó un prototipo funcional para validar que lo establecido en el protocolo es posible realizarlo. Esta implementación es de código libre[1] y puede ser utilizado para desarrollo de aplicaciones comerciales o investigaciones del área académica que puedan extenderlo añadiendo funcionalidades y mayor robustez. La utilización del protocolo evita el pago de licencias de utilización que tienen otros protocolos como ZigBee.

V.3. Trabajo futuro

Existe mucho trabajo por realizar en el diseño de protocolos de red para redes inalámbricas de monitoreo y control, y en cuanto a la especificación e implementación del protocolo ligero de red se identifican principalmente las siguientes:

- Investigar tiempos adecuados de envío de mensajes de control para reducir el consumo de energía sin afectar de forma importante el desempeño.
- Identificar e integrar al protocolo de red técnicas que permitan bajo consumo de energía.
- Mejorar la implementación del protocolo para permitir la utilización de menor memoria de datos y de almacenamiento de programa.
- El protocolo se enfoca en la conexión de nodos dentro de una misma PAN, por lo que se requiere de mecanismos para interconectar varias PAN.

[1] El código se puede obtener en http://code.google.com/p/fruitfly/

- De la misma manera, es necesario contar con métodos para interconectar redes de bajas prestaciones que utilicen el protocolo ligero con redes de mayor desempeño y extensión.
- Disminuir el tiempo de recuperación de ruta.
- Permitir que la red se mantenga viva aunque el nodo *sink* no se encuentre en el enlace.

Bibliografía

- 802154MACPHYSoftware. 2006. *802.15.4 MAC PHY Software Reference Manual, Rev. 1.4*. Freescale Semiconductor.
- Akkaya, K. y Younis, M. 2005. A survey of routing protocols in wireless sensor networks. *Elsevier Ad Hoc Network Journal*, 3(3):325–349 p.
- Akyildiz, I. y Kasimoglu, I. 2004. Wireless sensor and actor networks: Research challenges. *Ad Hoc Networks Journal (Elsevier).Octubre de 2004*, 2(4):351–367 p.
- Akyildiz, I. F., Su, W., Sankarasubramaniam, Y. y Cayirci, E. 2002. A Survey on Sensor Networks. *IEEE Communications Magazine. Agosto de 2002*, 40(8):102–114 p.
- Arch Rock. 2008. Arch Rock website. Http://www.archrock.com/ (consultado en marzo de 2008).
- Cerpa, A., Elson, J., Estrin, D., Girod, L., Hamilton, M. y Zhao, J. 2001. Habitat monitoring: application driver for wireless communications technology. *SIGCOMM Comput. Commun. Rev.*, 31(2 supplement):20–41 p. ISSN 0146-4833. doi:http://doi.acm.org/10.1145/844193.844196.
- CORIE. 2007. CORIE website. Http://www.ccalmr.ogi.edu/CORIE/ (consultado en diciembre de 2007).
- Cuomo, F., Luna, S. D., Monaco, U., y Melodia, T. 2007. Communications, 2007. ICC '07. IEEE International Conference on. *Communications, 2007. ICC '07. IEEE International Conference on*, 3271–3276 p.
- Freescale. 2008. Freescale website. Http://www.freescale.com/ (consultado en junio de 2008).
- HCS08. 2007. *HCS08 Family Reference Manual, Rev. 2*. Freescale Semiconductor.
- IEEE 802.15.4. 2003. WPart 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). URL `http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf`.
- ISO/IEC7498-1. 1994. *Information technology - open systems interconnection - basic reference model: The basic model*. ISO, Geneva, Switzerland.
- Jennic. 2008. JenNet Network Protocol Stack. URL: `http://www.jennic.com/` (consultado en enero de 2008).
- Kushalnagar, N., Montenegro, G. y Schumacher, C. 2007. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919 (Informational). URL `http://www.ietf.org/rfc/rfc4919.txt`.

- Langendoen, K.G., Baggio, A. y Visser, O.W. 2006. Murphy Loves Potatoes: Experiences from a Pilot Sensor Network Deployment in Precision Agriculture. In *14th Int. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*. ISBN 1-4244-0054-6, 1–8 p. URL `http://www.st.ewi.tudelft.nl/%7Ekoen/papers/WPDRTS06.pdf`.

- MC13192. 2008. *MC13192 2.4 GHz Low Power Transceiver for the IEEE® 802.15.4 Standard, Technical Data*. Freescale Semiconductor.

- Microchip Technology. 2008. Microchip Technology website. Http://www.microchip.com/ (consultado en febrero de 2008).

- Montenegro, G., Kushalnagar, N., Hui, J. y Culler, D. 2007. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard). URL `http://www.ietf.org/rfc/rfc4944.txt`.

- Mulligan, G. 2007. The 6LoWPAN architecture. In *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*. ACM, New York, NY, USA. ISBN 978-1-59593-694-3, 78–82 p. doi:http://doi.acm.org/10.1145/1278972.1278992.

- NanoStack. 2008. 6lowpan IPv6 and IEEE 802.15.4 protocol stack. URL: `http://www.nanostack.org` (consultado en octubre de 2008).

- Pinedo Frausto, E. 2008. *Análisis de la aplicación de redes ZigBee en entornos industriales de monitoreo y control*. Master's thesis, CICESE, Ensenada, BC.

- San Juan Software. 2007. PopNet^{TM} the easy, economical alternative for wireless sensor an control networks. URL: `http://www.sanjuansw.com/pub/SJS 12530006 Map PopNet Data Sheet.pdf` (consultado en diciembre de 2007).

- SMAC. 2006. *Simple Media Access Controller (SMAC) User's Guide, Rev. 1.5*. Freescale Semiconductor.

- Stankovic, J. 2004. Research challenges for wireless sensor networks. *SIGBED Rev.*, 1(2):9–12 p. ISSN 1551-3688. doi:http://doi.acm.org/10.1145/1121776.1121780.

- Synapse. 2007. Synapse Datasheet SNAP. URL: `http://www.synapse-wireless.com` (consultado en diciembre de 2007).

- TinyOS. 2007. TinyOS. web. Http://www.tinyos.net/.

- Weiser, M. 1991. The Computer for the Twenty-First Century. *Scientific American*, 265(3):94–104 p.

- YC. Chang, JL. Chen, Z.-S.L. 2006. Cluster Based Self-Organization Management Protocols for Wireless Sensor Networks. *IEEE Transactions on Consumer Electronics*, 52(1):75–80 p.

- ZigBee Alliance. 2007. ZigBee website. Http://www.zigbee.org/ (consultado en diciembre de 2007).

Apéndice A

Manual de FruitFly

FruitFly es la implementación desarrollada del protocolo ligero propuesto. En esta sección se proporciona información suficiente para desarrollar aplicaciones con FruitFly como protocolo de red.

A.1. Herramientas utilizadas

Existe gran cantidad de herramientas para desarrollo sobre el estándar IEEE 802 .15.4, pero se seleccionó utilizar productos de Freescale por que se cuenta *hardware* y *software* para trabajar.

A.1.1. *Hardware*

El hardware utilizado es MC13192 *Sensor Applications Reference Design* (SARD) el cual contiene 4kBytes de memoria RAM, 60kBytes de memoria no volátil, la velocidad del procesador puede configurarse hasta 20MHz, cuenta con 4 LEDs y 4 botones, acelerómetros para tres dimensiones, puerto de depuración de programas, puerto serie DB-9 para comunicación con el nodo, interfaz de radio frecuencia, puede ser alimentado de energía por baterías de 9 V (6LR61) o por u adaptador de 9 V de corriente directa, si los acelerómetros no son utilizados se pueden utilizar baterías con 2-3.4 V.

A.1.2. *Software*

Para el desarrollo del prototipo se utiliza la implementación del estándar IEEE 802.15.4 versión 2003 desarrollada por Freescale.

Se utilizó *Freescale BeeKit Wireless Connectivity Toolkit* versión 1.1.0.0 para configurar y tener un código base de esta implementación. Aunque para tener el código no se necesita licencia, para compilarlo sí. Se utiliza *CodeWarrior* como IDE (*Integrated Development Environment*).

BeeKit provee código base para utilizar la implementación del estándar IEEE 802.15.4 de Freescale en ensamblador, C y C++. El utilizado para este prototipo es C.

A.2. Instalación

Esta sección muestra como instalar el código fuente de FruitFly en un proyecto de CodeWarrior y utilizarlo en el envío y recepción de mensajes por la red.

A.2.1. Paso 1: Crear proyecto

Se utiliza la herramienta BeeKit versión 1.1.0.0 para crear y configurar una solución basada en la pila de Freescale y a partir de esta solución obtener un proyecto para utilizarlo con el IDE Freescale CodeWarrior for Microcontrollers V6.1.

Para crear la solución se utiliza el código base *BeeKit MAC Codebase 1.0.0*, el cual provee una implementación de la versión 2003 del estándar IEEE 802.15.4, además de ciertas utilidades para el control de los dispositivos. Se debe seleccionar en Tipos de Proyecto *MAC Demo Applications* y en plantilla *MyWirelessApp Demo 08 (Coordinator)*.

Después de crear la solución se procede a configurar los parámetros referentes al tipo de dispositivo a utilizar. Para el desarrollo de FruitFly se utilizó el tipo de plataforma *MC13192-SARD*. Los parámetros restantes debe estar como sigue: modulo UART habilitado a 19200 kb/s, el canal seleccionado fue el 26, la dirección extendida no se modifica, todos los parámetros restantes quedan como están, excepto `Beacon Order` que debe ser `0x0F` para inhabilitar los mensajes *beacon*.

Terminada la configuración en BeeKit exportamos la solución y ahora utilizamos CodeWarrior donde la importamos creando un proyecto nuevo. En el nuevo proyecto agregamos los archivos contenidos en FruitFly.zip[1], es decir, la carpeta *Network*, la agregamos al proyecto de CodeWarrior para poder utilizarla.

A.2.2. Paso 2: Eliminar de funciones redundantes

FruitFly redefine varias funciones y variables que se encuentran en el proyecto creado por BeeKit en el archivo *\Application\Source\MApp.c*, por lo cual es necesario eliminarlas. Estas funciones y variables son:

```
static uint8_t App_StartScan(uint8_t scanType);
static void App_HandleScanEdConfirm(nwkMessage_t *pMsg);
static uint8_t App_StartCoordinator(void);
static uint8_t App_HandleMlmeInput(nwkMessage_t *pMsg);
static uint8_t App_SendAssociateResponse(nwkMessage_t *pMsgIn);
static void App_HandleMcpsInput(mcpsToNwkMessage_t *pMsgIn);
static void App_TransmitUartData(void);
static uint8_t App_WaitMsg(nwkMessage_t *pMsg, uint8_t msgType);
static void App_InitSecurity(void);
```

[1] FruitFly.zip se puede obtener en http://code.google.com/p/fruitfly/

```
uint8_t MLME_NWK_SapHandler(nwkMessage_t * pMsguint8_t);
MCPS_NWK_SapHandler(mcpsToNwkMessage_t *pMsguint8_t);
ASP_APP_SapHandler(aspToAppMsg_t *pMsg);
```

Como el paquete *Network* se encarga de manipular la capa MAC, debemos quitar las llamadas a las funciones de la capa MAC de la función `void AppTask(event_t events)` y destinarla a tareas de aplicación y no de red. Las variables que debemos remover porque ya no son utilizadas por este archivo y son requeridas en el paquete *Network* son:

```
static const uint8_t maShortAddress[2];
static const uint8_t maPanId[2];
static const uint8_t mSecurityLevel;
static const uint8_t mSecurityMode;
static uint8_t mLogicalChannel;
static uint8_t maDeviceShortAddress[2];
static uint8_t maDeviceLongAddress[8];
static nwkToMcpsMessage_t *mpPacket;
static uint8_t mMsduHandle;
static uint8_t mcPendingPackets;
static anchor_t mMlmeNwkInputQueue;
static anchor_t mMcpsNwkInputQueue;
uint8_t gState;
```

Además de la declaración de estas variables debemos eliminar las referencias a ellas, para evitar errores al compilar. Incluimos el archivo *NetworkAPI.h* en el archivo *Application* \ *Source* \ *MApp.c* para poder sustituir las llamadas a las funciones `Init_802_15_4()`, `Init_MacExtendedAddress ()`, `MSG_InitQueue ( & mMlmeNwkInputQueue )` y `MSG_InitQueue ( & mMcpsNwkInputQueue )` por `Nwk_Init()`.

Eliminamos las declaraciones de las constantes `mDefaultValueOfChannel_c` y `mDefault ValueOfExtendedAddress_c` del archivo *\Application \Configure \ApplicationConf.h* porque ahora son utilizadas por el paquete *Network*. En el archivo *\Application \Init \MApp _Init.c* eliminamos la función `void Init_MacExtendedAddress( void )` para evitar errores al compilar y porque ahora el paquete *Network* se encarga de esta función. En el archivo *\Application \Source \MApp.h* quitamos los siguientes macros:

```
mDefaultValueOfShortAddress_c
mDefaultValueOfPanId_c
mDefaultValueOfMaxPendingDataPackets_c
mDefaultValueOfBeaconOrder_c
mDefaultValueOfSuperframeOrder_c
mDefaultValueOfSecurityLevel_c
mDefaultValueOfSecurityMode_c
```

También las siguientes enumeraciones de este mismo archivo:

```
enum { stateInit,
stateScanEdStart,
stateScanEdWaitConfirm,
stateStartCoordinator,
stateStartCoordinatorWaitConfirm,
stateListen
};
typedef enum{
evtInit = 1,
evtStartScan = 2,
evtMessageFromMLME = 4,
evtMessageFromMCPS = 8,
evtStartCoordinator = 16,
```

```
evtListen = 32
};
enum {
errorNoError,
errorWrongConfirm,
errorNotSuccessful,
errorNoMessage,
errorAllocFailed,
errorInvalidParameter,
errorNoScanResults
};
```

Además renombramos la variable `gState` a `gStateApp` porque esta es utilizada por el paquete *Network* y provocará errores en el compilado del proyecto.

A.2.3. Paso 3: Crear tarea

El proyecto creado por BeeKit utiliza un planificador de tareas para calendarizarlas, por esto se requiere crear una tarea para que el protocolo de pueda hacer uso del procesador. Entonces, modificamos el macro `TS_TaskCreate` que se encuentra en el archivo *\SSM\TS\Interface\TS_Configuration.h*, redefiniéndolo de la siguiente manera:

```
#define TS_TaskCreate \
TS_Create(gIdleTaskPriority_c, TS_IdleTask); \
/* Esta linea se agrega al macro para crear la tarea de red.*/ \
TS_Create(gNwkTaskPriority_c, NwkLayer_Main); \
TS_Create(gMacTaskPriority_c, Mlme_Main); \
TS_Create(gZAppTaskIDPriority_c, AppTask); \
TS_Create(gUartTaskIDPriotity_c, UART_Main); \
TS_Create(gTimerTaskPriority_c, TMR_Task);
```

Para poder tener acceso a la función `NwkLayer_Main()` se debe incluir la definición de esta con la instrucción `#include "NetworkLayer.h"` al mismo archivo *\SSM\TS\Interface\TS_Configuration.h.*

A.2.4. Paso 3: Modificar número de temporizadores

FruitFly utiliza temporizadores para su operación, por lo tanto, se debe reservar una cantidad extra de estos modificando el macro gSoftwareTimerCount_c que se encuentra en el archivo *\PLM\Interface\TMR_Interface.h* redefiniéndolo de la siguiente manera:

```
#define gSoftwareTimerCount_c gNumberOfBeeStackTimers_c + \
gNumberOfApplicationTimers_c + \
16 /* nTotalChild + 2 */
```

Ahora, antes poder enviar y recibir datos por la red, falta iniciar la red con la directiva con `Nwk_Start()`, esta puede invocarse al presionar por primera vez una tecla o al iniciar el programa de aplicación. Después de esto podemos enviar datos con `Nwk_Send()` o recibir datos si hay en el *buffer* con `Nwk_Receive()`, también se puede obtener la dirección corta que fue asignada al dispositivo con `Nwk_getShortAddress()` o la dirección extendida con `Nwk_getExtendedAddress()`.

A.3. Interfaz de desarrollo de aplicaciones

La interfaz para el desarrollo de aplicaciones sirve para interactuar con la capa de red solicitando servicios de envío o recepción de información. Consta de las siguientes funciones `Nwk_Start ()`, `Nwk_Stop ()`, `Nwk_Reset ()`, `Nwk_Send ()`, `Nwk _Receive ()`, `Nwk_get ExtendedAddress ()` y `Nwk_getShortAddress ()`.

void Nwk_Start(void) Este procedimiento inicia el motor del protocolo de red. Se encarga

de identificar las redes y dispositivos dentro del alcance, para asociarse a ellas y con esto obtener una dirección de enlace de datos, que también es utilizada en la capa de red.

No recibe ni devuelve datos.

void Nwk_Stop(void) Este procedimiento detiene el motor del protocolo de red. Inhabilita los temporizadores utilizados en el mantenimiento de la conexión. Al detener los *timers* permite a la capa de aplicación poner al nodo en ciclos de bajo consumo.

No recibe ni devuelve datos.

void Nwk_Reset(void) Este procedimiento detiene el motor del protocolo de red y lo reinicia eliminando los mensajes almacenados en los *buffer* de entrada y de salida.

No recibe ni devuelve datos.

uchar Nwk_Send(uchar *netAddrDest, uchar * data, uchar length) La función `Nwk_Send()` permite enviar datos a través de la red hasta llegar a la dirección `netAddrDest`.

Los parámetros que se envían son:

- netAddrDest: es un puntero a la dirección de memoria donde se encuentra almacenada la dirección destino, en *little-endian*.
- data: es un puntero a la dirección de memoria donde inician los datos a enviar a través de la red.
- length: es la cantidad de bytes que se desean enviar por la red. Con este parámetro se puede saber donde terminan los datos a partir de la dirección de data.

Valores de retorno:

- Sin error: 0x00
- Datos nulos: 0x01
- Longitud de datos cero: 0x02
- Sin memoria para alojar mensaje: 0x03
- Conexión no establecida: 0x04

void Nwk_Receive(nAppData_t * data) La función `Nwk_Receive()` permite recuperar los datos de la capa de red. Para poder ser utilizado se debe reservar espacio en memoria para almacenar esos datos y enviar donde empieza esa memoria reservada a la función.

Los parámetros que se envían son:

- `data`: es la posición de inicio de memoria reservada para almacenar los datos recibidos de la red. data es de tipo `nAppData_t`. `nAppData_t` consta de los siguientes campos:

  `networkAddress[2]` Son 2 bytes en *little-endian* utilizados para almacenar la dirección de red fuente del paquete de datos.

  `length` Es un byte que indica la longitud de los datos recibidos.

  `data[nSizeDataInPackets_c]` Es un arreglo de tamaño `nSizeDataInPackets_c` que puede ser hasta 97 bytes. Solo los primeros bytes indicados por length son los datos recibidos de la red.

void Nwk_getExtendedAddress(uint8_t * address) La función `Nwk_getExtendedAddress()` permite obtener la dirección extendida del nodo, esta consta de 64 bits que identifican al nodo.

Los parámetros que se envían son:

- `address`: es la posición de inicio de memoria reservada para almacenar la dirección extendida del dispositivo en *little-endian*, esta debe ser de 64 bits.

void Nwk_getShortAddress(uint8_t * address) La función `Nwk_getShortAddress()` permite obtener la dirección corta del nodo, esta consta de 16 bits que identifican al nodo en la red.

Los parámetros que se envían son:

- `address`: es la posición de inicio de memoria reservada para almacenar la dirección corta del dispositivo en *little-endian*, esta debe ser de 16 bits.

A.4. Aplicación de ejemplo

Esta sección muestra como escribir un programa utilizando FruitFly por medio de una pequeña aplicación que envía un mensaje de datos al nodo *sink* cada 10 segundos.

Una vez que se encuentra configurado el proyecto en CodeWarrior podemos hacer uso de FruitFly para comunicación en la red. El procedimiento para enviar y recibir datos por la red es el siguiente:

- Primero se deben inicializar la pila por medio de la instrucción `Nwk_Start()` e iniciar el motor de la red.
- Después se puede enviar mensajes de datos por la red con la función `Nwk_Send()` o recibir con la función `Nwk_Receive()`.

Como en todo sistema de red, se debe programar los dos extremos que se comunican, el emisor y el receptor. Para este ejemplo el emisor es un nodo hijo y el receptor es el nodo *sink*.

A.4.1. Programa emisor

Para inicializar la capa de red se debe especificar cual es la dirección extendida del nodo por medio del macro `mDefaultValueOfExtendedAddress_c`, esto se hace en el archivo *\Network\Configure\NetworkLayerConf.h*, para fines de este ejemplo lo definimos como sigue:

```
#define mDefaultValueOfExtendedAddress_c 0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,\
0xAA,0xAA
```

En ese mismo archivo indicamos que el nodo no será el *sink* con la siguiente instrucción:

```
#define nIsSink FALSE
```

Después, modificamos el cuerpo de la función `App_HandleKeys` del archivo *\Application\Source\MApp.c* agregando la instrucción `Nwk_Start()` y la instrucción para enviar datos cada 10 segundos:

```
static void App_HandleKeys(key_event_t events)
{
        /*Declara variable para identificador de temporizador. */
       uint8_t idTimer;

       /*Inicia la capa de red. */
       Nwk_Start();

       /*Busca un temporizador disponible. */
       idTimer=20;
       while(TMR_IsTimerOff(idTimer)==FALSE){
            idTimer++;
```

```
        }

        /*Inicia el temporizador para llamar a la función enviarDatos
         cada 10000 milisegundos. */
        TMR_StartInterval(idTimer,10000,enviarDatos);
}
```

Ahora creamos la función `enviarDatos()` que será llamada cada 10000 milisegundos en el mismo archivo:

```
void enviarDatos(uint8_t event){
     /*Variable para almacenar la dirección destino del mensaje.
    El mensaje debe entregarse al nodo sink, por lo tanto se indica
    la dirección 0,0,0,0.*/
     uint8_t dstAdd[2]={0x00,0x00};

     /*Variable para almacenar los datos que se enviarán por la red.
    En este caso es la dirección extendida asignada al nodo. */
     uint8_t dat[8]={mDefaultValueOfExtendedAddress_c};

     /*Llamada a la función de red para envío de mensajes. */
     Nwk_Send(dstAdd,dat,8);
 }
```

A.4.2. Programa receptor

Para crear el programa para el nodo receptor haremos las siguientes modificaremos al programa del emisor. Cambiamos la dirección extendida en el archivo *Network**Configure**NetworkLayerConf.h* como sigue:

```
#define mDefaultValueOfExtendedAddress_c 0xBB,0xBB,0xBB,0xBB,0xBB,0xBB,\
0xBB,0xBB
```

Ahora, en ese mismo archivo indicamos que el nodo será el *sink* con la siguiente instrucción:

```
#define nIsSink TRUE
```

Posteriormente, modificamos la función `App_HandleKeys` del archivo *\Application\Source\MApp.c* indicando que se inicie FruitFly y después se ponga en espera de mensajes. La función queda como sigue:

```
static void App_HandleKeys(key_event_t events)
{
       /*Inicia la capa de red. */
       Nwk_Start();

       /*Inicia que llame a la función AppTask() cuando este disponible
        el procesador. */
       TS_SendEvent(gZappTaskID_c, evtListen);
}
```

Por último, en la función `AppTask()` indicamos que se lean los mensajes recibidos de la red y los envíe a través de la interfaz UART (*Universal Asynchronous Receiver-Transmitter*), también conocida como puerto serie. La función es la siguiente:

```
void AppTask(event_t events) {
    /*Declaración de variable para almacenar mensajes recibidos. */
    nAppData_t datos;
```

```
    /*Inicializa la longitud de los datos a cero indicando que
    esta vacío.*/
    datos.length=0;

    /*Lee el mensaje de la red, si hay en el buffer de entrada.*/
    Nwk_Receive(&datos);

    /*Comprueba si se obtuvo un mensajes, si es así la longitud
    no debe ser cero.*/
    if (datos.length!=0){

      /*Si se leyó un mensaje de la red, se escribe en el
      puerto serial.*/
      UartUtil_PrintHex(&datos.networkAddress,2,gPrtHexNewLine_c);

    }

    /*Inidica que vuelva a llamar a la función AppTask() cuando este
     disponible el procesador. */
    TS_SendEvent(gZappTaskID_c, evtListen);

}
```

Lo que resta es grabar los programas en las memorias de los nodos y ponerlos a trabajar.

Apéndice B

Análisis de datos

En este apéndice se presenta el análisis estadístico de los datos arrojados por las pruebas de comportamiento realizadas. El motivo de este análisis es identificar en que pruebas y consiguientemente, que parámetros se ven afectados por variaciones en el intervalo de envío de mensajes de control.

B.1. Tiempo de respuesta en un salto

Los tiempos de respuesta obtenidos en la prueba se resumen en la Tabla IV y en la Tabla V se muestra el análisis análisis de varianza.

En el análisis de varianza se observa que el valor calculado para F es igual a 7.476597 y es mayor que el valor crítico para F que es igual a 2.607132. Esto indica que existe diferencia significativa entre los grupos que en este caso son los diferentes tiempos de envío de mensajes de control eco.

B.2. Tiempo de respuesta en múltiple saltos

Los tiempos de respuesta obtenidos en la prueba se resumen en la Tabla VI y en la Tabla VII se muestra el análisis análisis de varianza.

Tabla IV: **Resultados obtenidos en la prueba tiempos de respuesta en un salto.**

Estadístico muestral	10mSeg	20mSeg	2Seg	4Seg
Media	49.712352	36.039072	27.331024	24.524416
Error estándar	5.317666081	4.172692666	3.613889416	3.138078684
Mediana	20.928	15.744	14.72	15.024
Moda	15.952	14.768	15.952	15.952
Desviación estándar	168.1593665	131.952128	114.2812177	99.23476117
Varianza muestral	28277.57255	17411.36409	13060.19671	9847.537824
Curtosis	33.43272452	57.35689056	78.67119875	106.5900256
Coeficiente de asimetría	5.935833005	7.681049088	8.970868422	10.40854571
Rango	1083.952	1086.4	1057.424	1063.552
Mínimo	8.272	8.224	8.368	11.792
Máximo	1092.224	1094.624	1065.792	1075.344
Suma	49712.352	36039.072	27331.024	24524.416
Cuenta	1000	1000	1000	1000
Nivel de confianza (95.0%)	10.43507639	8.188247637	7.091684876	6.157981769

Tabla V: **Análisis de varianza para tiempos de respuesta en un salto.**

Origen de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	Probabilidad	Valor crítico para F
Entre grupos	384652.3	3	128217.4	7.47659	5.46348E-05	2.607132
Dentro de los grupos	68528075	3996	17149.17			
Total	68912727	3999				

En el análisis de varianza se observa que el valor calculado para F es igual a 14.75862 y es mayor que el valor crítico para F que es igual a 2.607132. Esto indica que existe diferencia significativa entre los grupos que en este caso son los diferentes tiempos de envío de mensajes de control eco.

Tabla VI: **Resultados obtenidos en la prueba tiempos de respuesta en dos saltos.**

Estadístico muestral	10mSeg	20mSeg	2Seg	4Seg
Media	90.037936	80.195936	45.68216	47.225152
Error estándar	6.834247534	6.662224367	4.867663773	4.97447746
Mediana	46.064	35.752	22.144	22.464
Moda	69.984	27.232	22.144	20.864
Desviación estándar	216.117883	210.6780328	153.9290441	157.3067894
Varianza muestral	46706.93935	44385.23352	23694.15061	24745.426
Curtosis	30.79294019	22.15377091	40.64044482	38.65287232
Coeficiente de asimetría	5.317750825	4.841139903	6.521791939	6.367800718
Rango	2438.992	1597.792	1066.64	1065.92
Mínimo	17.36	4.976	15.552	16.784
Máximo	2456.352	1602.768	1082.192	1082.704
Suma	90037.936	80195.936	45682.16	47225.152
Cuenta	1000	1000	1000	1000
Nivel de confianza (95.0%)	13.41112699	13.0735588	9.552018224	9.761623144

Tabla VII: **Análisis de varianza para tiempos de respuesta en dos saltos.**

Origen de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	Probabilidad	Valor crítico para F
Entre grupos	1544472.115	3	514824	14.75862	1.48E-09	2.607132
Dentro de los grupos	139392217.7	3996	34882.94			
Total	140936689.8	3999				

B.3. Tiempo de recuperación de ruta

Los tiempos recuperacion de ruta obtenidos en la prueba se resumen en la Tabla VIII y en la Tabla IX se muestra el análisis análisis de varianza.

En el análisis de varianza se observa que el valor calculado para F es igual a 15.19290619 y es mayor que el valor crítico para F que es igual a 2.866266. Esto indica que existe diferencia significativa entre los grupos que en este caso son los diferentes tiempos de envío de mensajes de control eco.

Tabla VIII: **Resultados obtenidos en la prueba tiempos de recuperación de ruta.**

Estadístico muestral	10mSeg	20mSeg	2Seg	4Seg
Media	910.4576	966.1536	2080.2432	2699.3648
Error estándar	39.10683126	75.15158654	191.318341	397.5846453
Mediana	959.424	918.512	2203.776	3098.888
Moda	#N/A	#N/A	#N/A	#N/A
Desviación estándar	123.6666589	237.6501832	605.0017157	1257.273042
Varianza muestral	15293.44251	56477.6096	366027.076	1580735.501
Curtosis	-0.489177774	-0.373821146	-1.357935693	-1.01713141
Coeficiente de asimetría	-0.946494037	0.747642788	-0.229635653	-0.35502527
Rango	333.04	680.672	1682.384	3791.392
Mínimo	688.704	683.024	1145.36	695.152
Máximo	1021.744	1363.696	2827.744	4486.544
Suma	9104.576	9661.536	20802.432	26993.648
Cuenta	10	10	10	10
Nivel de confianza (95.0%)	88.46579828	170.0046994	432.7921545	899.3989513

Tabla IX: **Análisis de varianza para tiempos de recuperación de ruta.**

Origen de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	Probabilidad	Valor crítico para F
Entre grupos	23000544	3	7666848	15.19291	1.50106E-06	2.866266
Dentro de los grupos	18166803	36	504633.4			
Total	41167347	39				

B.4. Tiempo de formación de red con dos nodos

Los tiempos formación con dos nodos obtenidos en la prueba se resumen en la Tabla X y en la Tabla XI se muestra el análisis análisis de varianza.

En el análisis de varianza se observa que el valor calculado para F es igual a 0.730943 y es menor que el valor crítico para F que es igual a 2.838745. Esto indica que no existe diferencia significativa entre los grupos que en este caso son los diferentes tiempos de envío de mensajes de control eco.

Tabla X: **Resultados obtenidos en la prueba tiempos de formación de red con dos nodos.**

Estadístico muestral	10mSeg	20mSeg	2Seg	4Seg
Media	879.7936	886.2848	876.2784	865.92
Error estándar	8.859355526	10.04377485	10.91910103	14.02519942
Mediana	873.448	896.264	866.736	871.368
Moda	#N/A	#N/A	#N/A	#N/A
Desviación estándar	28.01574206	31.76120484	34.52922926	44.35157479
Varianza muestral	784.8818034	1008.774133	1192.267674	1967.062187
Curtosis	0.297999639	-0.613794142	-0.595426122	-1.211923098
Coeficiente de asimetría	0.775894996	-0.745026838	0.841652379	0.21949098
Rango	92.08	94	97.056	127.36
Mínimo	838.912	830.544	840.784	814.288
Máximo	930.992	924.544	937.84	941.648
Suma	8797.936	8862.848	8762.784	8659.2
Cuenta	10	10	10	10
Nivel de confianza (95.0%)	20.04125452	22.72059717	24.70072256	31.72720525

Tabla XI: **Análisis de varianza para tiempos de formación de red con dos nodos.**

Origen de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	Probabilidad	Valor crítico para F
Entre grupos	2456.725	3	818.9082	0.730943	0.539646	2.838745
Dentro de los grupos	44813.77	40	1120.344			
Total	47270.5	43				

B.5. Tiempo de formación de red con tres nodos

Los tiempos formación con tres nodos obtenidos en la prueba se resumen en la Tabla XII y en la Tabla XIII se muestra el análisis análisis de varianza.

En el análisis de varianza se observa que el valor calculado para F es igual a 1.847441 y es menor que el valor crítico para F que es igual a 2.866266. Esto indica que no existe diferencia significativa entre los grupos que en este caso son los diferentes tiempos de envío de mensajes de control eco.

Tabla XII: **Resultados obtenidos en la prueba tiempos de formación de red con dos nodos.**

Estadístico muestral	10mSeg	20mSeg	2Seg	4Seg
Media	957.8368	992.504	844.344	842.5936
Error estándar	70.99240415	88.10422199	9.400944369	7.910807487
Mediana	853.184	878.664	847.688	842.28
Moda	#N/A	#N/A	#N/A	#N/A
Desviación estándar	224.4976937	278.610013	29.72839636	25.01616979
Varianza muestral	50399.21447	77623.53933	883.7775502	625.8087509
Curtosis	1.283155229	1.219373941	-1.263090448	0.862583925
Coeficiente de asimetría	1.721295117	1.676900767	-0.23008345	0.670362796
Rango	560.544	725.888	83.76	87.28
Mínimo	823.104	808.352	801.664	806.592
Máximo	1383.648	1534.24	885.424	893.872
Suma	9578.368	9925.04	8443.44	8425.936
Cuenta	10	10	10	10
Nivel de confianza (95.0%)	160.5959752	199.3055964	21.2664136	17.89548978

Tabla XIII: **Análisis de varianza para tiempos de formación de red con dos nodos.**

Origen de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	Probabilidad	Valor crítico para F
Entre grupos	179477.5	3	59825.84	1.847441	0.15605	2.866266
Dentro de los grupos	1165791	36	32383.09			
Total	1345269	39				

B.6. Entrega exitosa de datos para un salto

El número de bits por segundo entregados existosamente en la prueba se resumen en la Tabla XIV y en la Tabla XV se muestra el análisis análisis de varianza.

En el análisis de varianza se observa que el valor calculado para F es igual a 1.953259 y es menor que el valor crítico para F que es igual a 2.866266. Esto indica que no existe diferencia significativa entre los grupos que en este caso son los diferentes tiempos de envío de mensajes de control eco.

Tabla XIV: **Resultados obtenidos en la prueba tiempos de formación de red con dos nodos.**

Estadístico muestral	10mSeg	20mSeg	2Seg	4Seg
Media	76988.41	76962.78	77007.66	77004.28
Error estándar	14.18687	24.05658	6.04858	6.418258
Mediana	77002.67	76995.11	77002.97	76996.82
Moda	#N/A	#N/A	#N/A	#N/A
Desviación estándar	44.86282	76.07357	19.12729	20.29631
Varianza muestral	2012.673	5787.188	365.8532	411.9403
Curtosis	6.720645	-0.09386	0.16501	-1.22205
Coeficiente de asimetría	-2.47751	-1.13583	0.88522	0.289829
Rango	153.7898	211.8249	61.24051	58.70265
Mínimo	76869.12	76815.54	76984.22	76974.29
Máximo	77022.91	77027.36	77045.46	77033
Suma	769884.1	769627.8	770076.6	770042.8
Cuenta	10	10	10	10
Nivel de confianza (95.0%)	32.09293	54.41975	13.68284	14.51911

Tabla XV: **Análisis de varianza para tiempos de formación de red con dos nodos.**

Origen de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	Probabilidad	Valor crítico para F
Entre grupos	12565.79	3	4188.595	1.953259	0.138482	2.866266
Dentro de los grupos	77198.89	36	2144.414			
Total	89764.68	39				

B.7. Entrega exitosa de datos para estrella

El número de bits por segundo entregados existosamente en la prueba se resumen en la Tabla XVI y en la Tabla XVII se muestra el análisis análisis de varianza.

En el análisis de varianza se observa que el valor calculado para F es igual a 195.8571 y es mayor que el valor crítico para F que es igual a 2.866266. Esto indica que existe diferencia significativa entre los grupos que en este caso son los diferentes tiempos de envío de mensajes de control eco.

Tabla XVI: **Resultados obtenidos en la prueba tiempos de formación de red con dos nodos.**

Estadístico muestral	10mSeg	20mSeg	2Seg	4Seg
Media	3684.745	5017.271	15909.59	16073.16
Error estándar	542.0557	367.9574	467.5254	529.8739
Mediana	3733.349	5265.409	16613.31	16725.86
Moda	#N/A	#N/A	#N/A	#N/A
Desviación estándar	1714.131	1163.583	1478.445	1675.608
Varianza muestral	2938244	1353926	2185800	2807663
Curtosis	-1.43386	-1.10501	-0.99276	4.736582
Coeficiente de asimetría	0.232562	-0.44818	-0.8703	-2.18516
Rango	4599.69	3318.893	3821.943	5283.035
Mínimo	1671.269	3187.711	13307.87	11852.06
Máximo	6270.959	6506.604	17129.81	17135.1
Suma	36847.45	50172.71	159095.9	160731.6
Cuenta	10	10	10	10
Nivel de confianza (95.0%)	1226.215	832.3774	1057.616	1198.658

Tabla XVII: **Análisis de varianza para tiempos de formación de red con dos nodos.**

Origen de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	Probabilidad	Valor crítico para F
Entre grupos	1.36E+09	3	4.55E+08	195.8571	2.41E-22	2.866266
Dentro de los grupos	83570696	36	2321408			
Total	1.45E+09	39				