Implementación del protocolo de enrutamiento RPL para LLN mediante Simulink/Matlab

Juan Diego López Triana, Jhon Alexander Meléndez Pérez

Trabajo de Grado para optar por el título de Ingeniero de Sistemas

Director

Pedro Javier Trujillo Tarazona

Mg. Informática

Universidad Industrial de Santander

Facultad de Ingenierías Físico-Mecánicas

Escuela de Ingeniería de Sistemas e Informática

Bucaramanga

2019

Agradecimientos

"En primera instancia quiero agradecer a Dios por darme la fuerza y sabiduría para poder alcanzar mis metas y objetivos.

A mi madre quien siempre ha dado todo de ella para que yo pudiera estar donde estoy y gracias a su apoyo he podido lograr todo lo que me he propuesto en la vida.

A mis hermanas por ser un modelo a seguir y darme la motivación para continuar luchando día a día por mis objetivos.

A mis tías quienes me brindaron su apoyo incondicional aquí en Bucaramanga y siempre me hicieron sentir como en casa estos cinco años a su lado.

Al profesor Pedro por su guía y acompañamiento durante el desarrollo de este proyecto.

Por último, a todos mis amigos que siempre me brindaron su apoyo e hicieron grato todo este tiempo en la universidad."

Juan Diego López Triana

Agradecimientos

“Quiero agradecer a mi orientador, Magister Pedro Javier Trujillo Tarazona, por su disponibilidad para resolvernos las dudas que surgieron en el proceso y darnos la guía necesaria para llevar a cabo este proyecto.

Hubo gran cantidad de personas que me apoyaron en mi estancia en la universidad y realización de este proyecto, de las cuales quiero destacar:

A mis padres Elsa y José por darme su amor y enseñanzas, la cuales fueron el mayor impulso para alcanzar mis metas y querer superarme día a día.

A mis hermanos Julisse y Oliver por la complicidad y brindarme su compañía en los momentos difíciles.

Por último, a mi novia Anyi por darme calma y apoyo moral cada vez que fue necesario.”

Jhon Alexander Meléndez Pérez

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Lista de Apéndices

(Ver apéndices adjuntos en el CD y pueden visualizarlos en la Base de Datos de la Biblioteca UIS)

Lista de Siglas

ANSI:	American National Standards Institute
DAG:	Directed acyclic graph
DAO:	Destination Advertisement Object
DAOAck:	Destination Advertisement Object Acknowledgment
DIO:	DODAG Information Object
DIS:	DODAG Information Solicitation
DODAG:	Destination Oriented Directed Acyclic Graph
FFD:	Full Function Device
IEC:	International Electrotechnical Commission
IEEE:	Institute of Electrical and Electronics Engineers
IETF:	Internet Engineering Task Force
IoT:	Internet of Things
IPv6:	Internet Protocol Version 6
LAN:	Local Area Network
LLN:	Low power and Lossy Network
MAC:	Media Access Control
MINMINAS:	Ministerio de Minas y Energía
MINTIC:	Ministerio de Tecnologías de la Información y las Comunicaciones
MP2P:	Multipoint to Point
NIST:	National Institute of Standards and Technology
P2P:	Point to Point
PHY:	Physical Layer
RFD:	Reduced Function Device
RPL:	Routing Protocol for Low Power and Lossy Networks
TIA:	Telecommunications Industry Association
UPME:	Unidad de Planeación Minero-Energética
WSN:	Wireless Sensor Network

Resumen

TITULO: IMPLEMENTACIÓN DEL PROTOCOLO DE ENRUTAMIENTO RPL PARA LLN MEDIANTE SIMULINK/MATLAB*

AUTOR: JUAN DIEGO LÓPEZ TRIANA, JHON ALEXANDER MELÉNDEZ PÉREZ**

PALARAS CLAVE: RPL, LLN, MATLAB, SIMULINK, SIMEVENTS, TRICKLE

DESCRIPCIÓN

El protocolo de enrutamiento para redes de baja potencia y con pérdidas (RPL) es un protocolo vector distancia diseñado por el grupo de trabajo ROLL (Routing Over Low power and Lossy networks) de la IETF como protocolo esencial en las comunicaciones de datos en redes con estas limitaciones. Este tipo de redes comúnmente se caracterizan por restricciones de procesamiento, memoria y energía presentes en los dispositivos utilizados y por sus enlaces con altas tasas de pérdida, bajas velocidades e inestabilidad.

En este Proyecto de Grado se realizó un estudio de la norma [RFC6550], que define el protocolo RPL, sus funciones objetivo, su algoritmo y el temporizador Trickle. Además, se llevó a cabo la implementación de ciertas características del protocolo, el cual fue acotado en su alcance por razones de tiempo, en un entorno de simulación proporcionado por Matlab y Simulink. En Simulink se desarrolló una librería basada en el toolbox Simevents y en Matlab una interfaz gráfica para hacer uso de dicha librería. Una vez realizada la implementación, se diseñaron, simularon y analizaron una serie de escenarios que permitieron evidenciar las características del protocolo y su funcionamiento.

Finalizando el desarrollo de este proyecto, fue posible concluir que el entorno de simulación de Matlab y Simulink brinda las herramientas necesarias para lograr una implementación del protocolo RPL. Además, gracias a estas herramientas se verificó que RPL es un protocolo destacable en el enrutamiento de redes LLN.

---

*Trabajo de grado

**Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática.

Director: Pedro Javier Trujillo Tarazona, Mg. Informática

Abstract

TITLE: IMPLEMENTATION OF THE RPL ROUTING PROTOCOL FOR LLN BY SIMULINK/MATLAB*

AUTHOR: JUAN DIEGO LOPEZ TRIANA, JHON ALEXANDER MELÉNDEZ PÉREZ**

KEYWORDS: RPL, LLN, MATLAB, SIMULINK, SIMEVENTS, TRICKLE

DESCRIPTION:

The Routing Protocol for Low-Power and Lossy-Networks (RPL) is a distance vector protocol designed for the ROLL (Routing Over Low power and Lossy networks) working group of the IETF as an essential protocol in data communications in networks with these limitations. This type of networks is commonly characterized by processing, memory and energy restrictions present in the devices used and by their links with high rates of loss, low speeds and instability.

In this Degree Project, a study of the standard [RFC6550] was performed, which defines the RPL protocol, its objective functions, its algorithm and the Trickle timer. In addition, some protocol features were implemented, which were limited for time reasons, in a simulation environment provided by Matlab and Simulink. In Simulink, a library based on the toolbox Simevents was developed and in Matlab a graphical interface to make use of this library. Once the implementation was carried out, a series of scenarios were designed, simulated and analyzed, which allowed demonstrating the characteristics of the protocol and its operation.

Finalizing the development of this project, it was possible to conclude that the simulation environment of Matlab and Simulink provides the necessary tools to achieve an implementation of the RPL protocol. In addition, thanks to these tools it was verified that RPL is a remarkable protocol in the routing of LLN networks.

---

*Bachelor Thesis

**Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática.

Director: Pedro Javier Trujillo Tarazona, Mg. Informática

Introducción

En la actualidad, se encuentra un nuevo concepto denominado Internet de las Cosas (IoT), paradigma que pretende interconectar objetos y/o dispositivos a Internet, que van desde sensores muy simples hasta artefactos industriales, residenciales y elementos de uso cotidiano, con el fin de que estos se comuniquen entre sí y que el ser humano pueda llegar a interactuar con ellos. Se presume que en el año 2025 habrá hasta cien mil millones de dispositivos conectados a IoT y que su impacto será de US$ 11.000.000.000.000 (Rose, Eldridge y Chapin, 2015), entre ellos, sensores, medidores y demás dispositivos electrónicos.

Dentro de las tecnologías de comunicaciones de datos candidatas a ser utilizadas en IoT se encuentran las redes LLN, que se caracterizan por operar con dispositivos con limitaciones en potencia, memoria y energía. El IEEE 802.15.4 es un estándar para el funcionamiento de este tipo de redes, reconocido por el NIST y por la IEC como uno de los estándares candidatos para ser utilizado en las comunicaciones de datos de las redes eléctricas inteligentes. En adición a esto, el estudio "Smart Grids Colombia 2030" (MinMinas, MinTic, UPME y ICI, 2016) define las comunicaciones de datos, incluyendo las redes inalámbricas y en particular las basadas en el estándar IEEE 802.15.4, como elemento esencial para las redes inteligentes en Colombia.

Una vez definida las características del tipo de redes a ser usadas en IoT, nace el protocolo RPL definido por la IETF como un protocolo de enrutamiento en IPv6 que pretende mejorar la comunicación de datos dentro de las redes LLN. Se presenta un breve estudio del protocolo RPL con el fin de realizar una implementación de este en el ambiente

de simulación de MATLAB y Simulink, que permita analizar sus principales características y funcionamiento.

El siguiente capítulo se describen los objetivos a desarrollar en el presente libro. Todas las bases teóricas utilizadas para lograr la implementación se encuentran detalladas en el Capítulo 2. En el Capítulo 3 se presenta la metodología adoptada para la parte investigativa y el desarrollo de software de esta tesis. Los requisitos y especificaciones definidos a partir de las diferentes normas que definen el protocolo son descritos en el Capítulo 4. Los detalles de cómo se realizó la implementación del protocolo RPL y las características de esta, se muestran en el Capítulo 5. El modo de uso de la implementación realizada es explicado en el Capítulo 6. La evaluación final de la implementación del protocolo se muestra en el Capítulo 7. En el Capítulo 8 se exponen las conclusiones obtenidas a partir de la investigación del protocolo y la implementación realizada. En el Capítulo 9 se mencionan algunas recomendaciones a tener en cuenta.

1. Objetivos

1.1 Objetivo general

Implementar el protocolo de enrutamiento RPL para redes LLN (Low Power and Lossy Networks) mediante Matlab y Simulink.

1.2 Objetivos específicos

- Identificar y definir los requisitos y especificaciones para la implementación del protocolo de enrutamiento RPL
- Implementar el protocolo de enrutamiento RPL mediante módulos y rutinas de Matlab y Simulink.
- Diseñar y evaluar escenarios y juegos de datos de simulación para la validación del protocolo RPL.

2. Marco de referencia

En este capítulo se describen las bases teóricas estudiadas para el desarrollo de este libro: Internet de las Cosas, estándar IEEE 802.15.4, redes con pérdida y de baja potencia, protocolos de enrutamiento, protocolo RPL, simulación orientada a eventos y herramientas de simulación utilizadas.

2.1 Internet de las cosas (IoT)

Por lo general, el término Internet de las Cosas se refiere a los escenarios en los que la conectividad de red y la capacidad de cómputo se extienden a objetos, sensores y artículos de uso cotidiano que habitualmente no se consideran computadoras, permitiendo que estos dispositivos generen, intercambien y consuman datos con la mínima intervención humana. (Rose, Eldridge y Chapin. 2015)

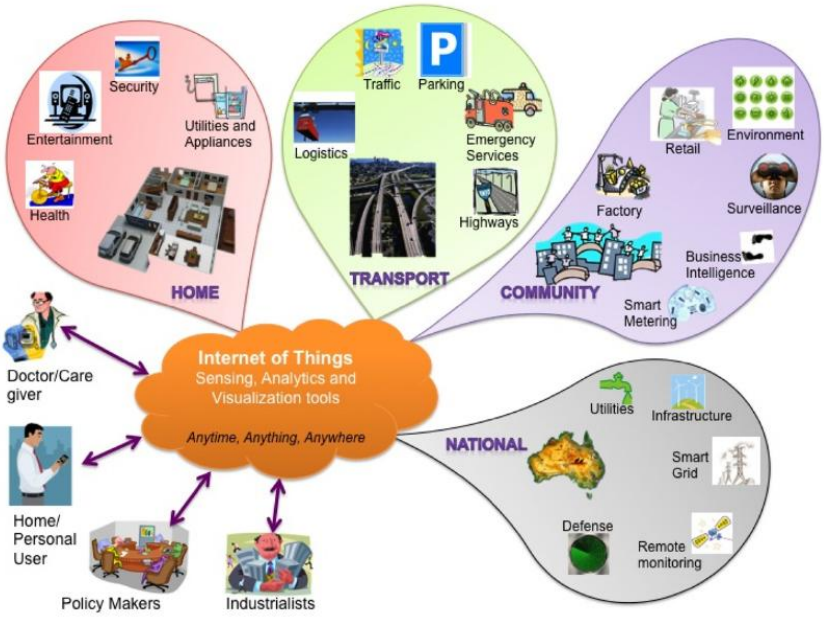


Figura 1. Alcance de IoT. Tomada de Rose, Eldridge y Chapin. (2015).

2.2 Estándar IEEE 802.15.4

El estándar IEEE 802.15.4 define las especificaciones de la capa física (PHY) y la subcapa de control de acceso al medio (MAC) para conectividad inalámbrica de redes de área personal de baja tasa de transmisión de datos. Las características más importantes del estándar IEEE 802.15.4 son la flexibilidad de la red, bajo costo y consumo de energía; este estándar se puede utilizar para muchas aplicaciones domésticas e industriales, donde se requiere una baja tasa de transmisión de datos.

El IEEE 802.15.4 soporta dos clases de dispositivos: dispositivos completamente funcionales (FFD) y dispositivos de funcionalidad reducida (RFD). Todas las redes de área personal (PAN) consisten en al menos un FFD que actúa como coordinador de la PAN, y es el responsable de mantener y administrar esta. Los RFDs son los responsables directos de obtener datos del entorno y enviarlos al coordinador. Dependiendo de los requisitos de aplicación, el estándar IEEE 802.15.4 puede operar en cualquiera de las siguientes topologías: topología en estrella o

topología punto a punto (Kurunathan, Severino, Koubaa y Tovar, 2018).

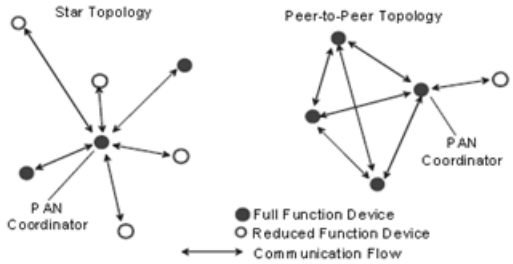


Figura 2. Ejemplos de topología estrella y punto a punto. Tomada de IEEE 802.15.4.

2.3 Redes con pérdida y de baja potencia (LLN)

Las LLN son una clase de redes en las que tanto los enrutadores como su interconexión están limitados. Los enrutadores LLN generalmente operan con limitaciones en la potencia de procesamiento, memoria y energía. Sus interconexiones están caracterizadas por altas tasas de pérdida, baja velocidad de datos e inestabilidad. Las LLN se componen desde unas cuantas decenas hasta miles de enrutadores y en sus flujos de tráfico soportados se incluyen conexiones punto a punto (entre dispositivos dentro del LLN), punto a multipunto (de un punto de control central a un subconjunto de dispositivos dentro del LLN), y multipunto a punto (de dispositivos dentro del LLN hacia un punto de control central) (Winter et al., 2012).

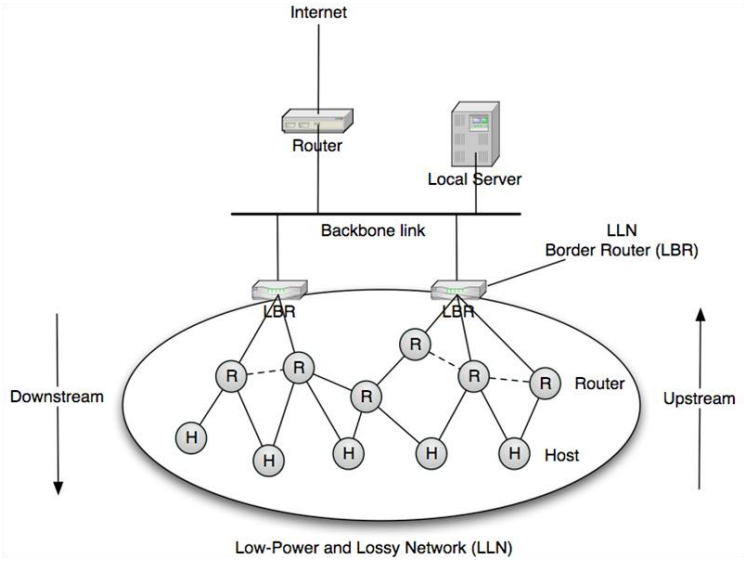


Figura 3. Red LLN. Tomada de Rowe, (2012).

2.4 Protocolos de enrutamiento

Un protocolo de enrutamiento es un conjunto de reglas, estructuras de datos y mensajes, utilizados por los enrutadores para determinar las rutas más apropiadas en las que deben reenviar los paquetes hacia sus destinos previstos.

Un enrutador es un dispositivo electrónico y/o software que conecta al menos dos redes y remite paquetes entre ellas de acuerdo con la información en los encabezados de paquetes y tablas de enrutamiento. El enrutamiento es el proceso de mover paquetes a través de una red de un host. La tabla de enrutamiento es una base de datos en un enrutador que almacena y actualiza las ubicaciones (direcciones) de otros dispositivos de red y las rutas más eficientes para alcanzar otros dispositivos.

Parte del trabajo de un protocolo de enrutamiento es especificar cómo los enrutadores reportan cambios y comparten información con los otros enrutadores de la red para actualizar sus tablas de enrutamiento, permitiendo así que las redes se ajustan dinámicamente a las condiciones cambiantes (por ejemplo, cambios en topologías de red y patrones de tráfico).

Existen dos tipos principales de algoritmos para enrutamiento IP: Vector distancia y estado de enlace. El primero determina el mejor camino en base a qué tan lejos está el destino, comúnmente en términos del número de saltos. El segundo utiliza métodos más sofisticados tomando en consideración variables de enlace tales como ancho de banda, demora, fiabilidad y carga (Routing protocol definition by The Linux Information Project, 2005).

2.5 Protocolo de enrutamiento para redes con pérdida y baja potencia (RPL)

RPL es un protocolo de enrutamiento vector distancia diseñado para redes LLN IPv6, que especifica cómo construir un Grafo Acíclico Orientado a Destino (DODAG), utilizando una función objetivo y un conjunto de métricas y/o restricciones. La función objetivo es una combinación de mediciones para calcular la "mejor" trayectoria hacia un nodo. Podría haber varias funciones objetivo operando en la misma red de nodos, debido a que las implementaciones varían enormemente con objetivos diferentes y una única malla de red puede llevar tráfico con diversos requisitos de calidad de la ruta (Vasseur et al., 2011).

Las principales funciones del protocolo RPL son:

- Proporcionar un mecanismo mediante el cual se admite el tráfico multipunto a punto, punto a multipunto y punto a punto.
- Separar el procesamiento de paquetes y envío con el objetivo de optimizar el enrutamiento. Ejemplos de este objetivo incluyen minimizar la energía, minimizar de latencia o satisfacer las limitaciones.
- Proporcionar un mecanismo para divulgar información sobre la topología de la red formada dinámicamente.

Los requisitos del protocolo RPL son:

- Verificar la alcanzabilidad de un enrutador antes de que este sea usado como pariente, mediante mecanismos que son disparados durante la fase de selección de parientes, para verificar las propiedades del enlace y la alcanzabilidad de un vecino.
- Necesita de un mecanismo externo para acceder y transportar a cierta información de control, denominada “Información de paquete RPL”, en paquetes de datos.

2.5.1 Terminología. La terminología pertinente que abarca el protocolo de enrutamiento RPL es la siguiente:

- **Grafo acíclico dirigido (DAG):** Un DAG o Grafo acíclico dirigido es un grafo que tiene la propiedad que todas sus aristas están orientadas de tal forma que no existen ciclos. Es decir, todas sus aristas están contenidas en rutas orientadas y terminados en uno o más nodos raíces.
- **Raíz DAG:** Es un nodo dentro del DAG que no tiene arista de salida. Como el grafo es acíclico, por definición, todos los DAGs deben tener al menos una raíz DAG y todas las rutas terminan en una raíz DAG.
- **Grafo acíclico dirigido orientado a el destino (DODAG):** Se refiere a un DAG orientado en un solo destino, es decir, hacia la raíz DAG que es el único nodo que no tiene aristas de salida.
- **Raíz DODAG:** Es la raíz DAG de un DODAG. Esta puede actuar como un enrutador en el borde del DODAG; en particular, puede agregar rutas en el DODAG y puede redistribuir rutas DODAG entre otros protocolos de enrutamiento.
- **Nodo Hoja:** Se refiere a cualquier dispositivo presente en una topología RPL que no ha sido configurado para ser la raíz DODAG, debido a sus métricas y/o restricciones.
- **Ruta hacia arriba y hacia abajo:** La primera se refiere a la dirección de envío de

paquetes desde los nodos hojas hacia las raíces y la segunda a la dirección de envío de paquetes desde las raíces hacia los nodos hojas. Ambas rutas siguen la terminología común utilizada en los grafos y de la búsqueda en profundidad, en donde los vértices más lejanos a la raíz son "más profundos" o "están abajo" y los vértices más cercanos son "menos profundos" o "están arriba".

- **Objetivo DODAG:** El Objetivo DODAG es un objetivo específico de la aplicación donde se implementa el protocolo RPL, este es definido fuera del alcance de RPL. Cualquier nodo que origine un DODAG deberá conocer este objetivo para decidir si este puede cumplirlo o no. Un objetivo común es la construcción del DODAG de acuerdo a una función objetivo específica y usarse para mantener la conectividad entre un conjunto de hosts.
- **Función objetivo (OF):** Se refiere a una función que define, como los nodos RPL seleccionan y optimizan las rutas dentro de una instancia RPL. Esta traduce una o más métricas y restricciones, que a su vez están definidas, en un valor llamado "rango", que se refiere a la distancia abstracta desde la raíz hasta un nodo determinado.
- **Rango:** El rango de un nodo es una representación escalar de la posición del nodo dentro de una versión del DODAG. El Rango es usado para evitar y detectar ciclos. El cálculo exacto del Rango es determinado por la Función Objetivo.
- **Instancia RPL:** Una instancia RPL es un conjunto de uno o más DODAG que comparte un mismo objetivo. Cada instancia RPL opera de forma independiente respecto a otras instancias RPL y con una única función objetivo y modo de operación.
- **DODAG en tierra:** Se dice que un DODAG está en tierra cuanto la raíz DODAG puede satisfacer el Objetivo de la aplicación.

- **DODAG flotante:** Se dice que un DODAG es flotante cuando no "está en tierra", es decir, un DODAG flotante no espera tener las propiedades requeridas para satisfacer el objetivo del DODAG. Este puede, sin embargo, proveer conectividad hacia otros nodos dentro del DODAG.

(Winter et al., 2012)

2.5.2 Mensajes de control. Para el proceso de creación y el mantenimiento del DODAG, RPL usa una serie de mensajes de control del tipo ICMPv6. Estos mensajes constan de un encabezado seguido del cuerpo del mensaje. El cuerpo del mensaje está compuesto por la base del mensaje y posiblemente por una serie de opciones como se ilustra en la Figura 4.

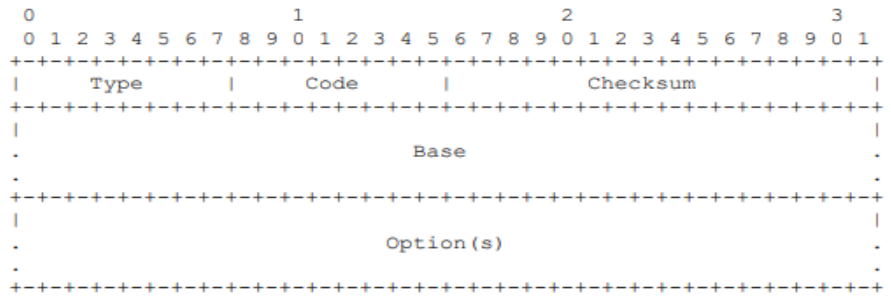


Figura 4. Estructura de mensaje RPL. Tomada de Winter et al., (2012)

Los diferentes mensajes de control utilizados por el protocolo RPL se detallan a continuación.

- **DODAG Information Object (DIO):** Este tipo de mensajes de control son creados y enviados por los nodos y llevan información que le permite a un nodo hoja descubrir una instancia RPL, aprender sus parámetros de configuración y seleccionar su nodo padre. También son utilizados para el mantenimiento del DODAG.
- **Destination Advertisement Object (DAO):** Este tipo de mensaje de control es usado para propagar información del destino en dirección hacia arriba a lo largo del DODAG. En el modo Storing, este mensaje es unidifusión desde los hijos hacia sus

padres. En el modo Non-storing, es unidifusión hacia la raíz. Además, permite a los nodos hojas pedir la unión a un DODAG después de haber escuchado un mensaje DIO.

- **Destination Advertisement Object Acknowledgment (DAOAck):** Un mensaje DAOAck es un paquete unidifusión enviado por un nodo raíz como confirmación a la solicitud de unión del DODAG, hecha por un nodo hoja.
- **DODAG Information Solicitation (DIS):** Este mensaje puede ser usado por un nodo para solicitar un mensaje DIO. Su uso es análogo al descubrimiento de vecinos que hace un enrutador en IPv6. Un nodo puede usar un mensaje DIS para explorar su vecindario en busca de DODAG cercanos. También es utilizado como una respuesta inmediata a una inconsistencia de la red.

(Winter et al., 2012)

2.5.3 Modos de operación. El protocolo RPL soporta dos modos de operación, Storing y Non-Storing; cada instancia RPL debe ser configurada para trabajar con alguno de estos dos modos. Estos modos de operación determinan la manera en que los nodos pueden o no almacenar las rutas hacia otros nodos.

En ambos modos de operación, al recibir un mensaje DIO, después de realizar el procedimiento de selección de padre, los nodos hoja almacenan la ruta del padre seleccionado en su tabla de enrutamiento. Siempre que llegue un mensaje con destino desconocido, el nodo lo enviará hacia su padre utilizando la ruta previamente almacenada.

En una instancia trabajando en el modo Storing, los nodos hoja son capaces de almacenar las rutas hacia otros nodos diferentes a su nodo padre. Esto lo hacen al recibir mensajes DAO, que son enviados en dirección ascendente hacia el nodo raíz. Para poder cumplir con esta

funcionalidad, los nodos presentes en la topología deben tener la suficiente capacidad de almacenamiento para poder mantener una tabla de enrutamiento.

A diferencia del modo Storing, el modo Non-storing fue pensado para ambientes en los cuales los nodos de la topología no cuentan con la suficiente capacidad de almacenamiento para mantener una tabla de enrutamiento completa. En este modo de operación, los nodos hoja almacenan únicamente la ruta hacia su respectivo nodo padre. Por este motivo, todo mensaje que pase por un nodo que no sea el destino, es enviado de padre en padre hasta llegar al nodo raíz, el cual es el encargado de calcular la ruta completa hacia el destino.

(Winter et al., 2012)

2.5.4 Métricas de encaminamiento. La construcción de una topología RPL se realiza de acuerdo a diferentes funciones objetivo específicas, las cuales están configuradas con ciertas métricas y/o restricciones. A continuación, se resumen cuatro de las métricas más utilizadas en el protocolo RPL.

- **Hop Count:** En redes, el conteo de saltos es el número total de dispositivos intermedios, como enrutadores, a través de los cuales debe pasar un determinado paquete de datos entre el origen y el destino, en lugar de ir directamente sobre un solo enlace. El conteo de saltos es considerado la medida básica de distancia en una red. En otras palabras, esta es una medida aproximada de la distancia entre dos dispositivos. ("What is a Hop Count", 2019)
- **Expected transmission count (ETX):** El ETX es una medida de estimación del número de transmisiones (incluyendo retransmisiones) que se necesitan para enviar un paquete sobre un enlace y que este sea recibido en su destino sin errores. El ETX pude variar de uno hasta infinito, donde uno representa un medio de transmisión perfecto, e

infinito un enlace completamente no funcional. La fórmula para calcular esta estimación es la siguiente:

$$ETX = \frac{1}{df * dr}$$

Donde *df* es la tasa de entrega de envió correcto esperada de un paquete y *dr* la tasa de entrega inversa, es decir la probabilidad de que un paquete sea transmitido con error (Parissidis, Karaliopoulos, Baumann, Spyropoulos y Plattner, 2016).

- **Node Energy:** La métrica de energía del nodo hace referencia a la eficiencia energética que tiene un nodo en una topología dada. Este tipo de métricas es altamente requerido en redes de sensores de gran escala, que mantiene el principio de que un nodo debe seleccionar como padre al nodo con mayor eficiencia energética y así poder asegurar que sus futuros mensajes llegarán al destino. (Liu, Sheng, Yin, Ali y Roggen, 2017)
- **Node Stability:** Esta métrica hace referencia a la estabilidad que tiene un nodo para permanecer conectado a un DODAG. Dado que en el protocolo RPL los diferentes mensajes de control (DIO, DIS, DAO) son responsables de la construcción y el mantenimiento de la topología DODAG, la tasa de transmisión de mensajes de control implica que un nodo es lo suficientemente estable para ser seleccionado como padre. Por ejemplo, enviar paquetes hacia nodos que con frecuencia transmiten mensajes DIS no es una buena decisión, ya que el envío de mensajes DIS implica que los nodos pierden conexión frecuentemente. (Yang, Guo, Orlik, Parsons y Ishibashi, 2014)

En la Tabla 1 se presenta una comparativa entre diferentes métricas de encaminamiento descritas anteriormente.

Tabla 1

Comparativas métricas de enrutamiento

Métrica de encaminamiento	Parámetros	Características clave
Hop Count	Conteo de salto entre dos nodos	A menor conteo de saltos, el paquete viaja a través de menos nodos intermediarios.
ETX	Conteo de transmisión esperada de paquetes entre dos nodos	La tasa de envío de paquetes es alta, lo cual incrementa el retraso en la recepción de estos.
Energy	Nivel de energía restante en los nodos	Incrementa la longevidad de la red y eventualmente el consumo de energía distribuida.
Stability	Número de mensajes DIO, DIS y DAO en la red	La estabilidad aumenta a medida que el número de mensajes de control se reduce.

2.5.5 Temporizador Trickle. El mecanismo del Temporizador Trickle es usado por los mensajes DIO, este ha sido enfatizado como una parte importante del mecanismo de control del protocolo RPL, que es el encargado de detectar y responder frente a inconsistencias e inestabilidades de la red. El temporizador Trickle es implementado basado en el algoritmo Trickle y su funcionamiento se basa en el cálculo del tiempo en el cual un nodo puede transmitir paquetes de datos, a menos que escuche otras transmisiones que sugieran que su transmisión es redundante. El temporizador se ejecuta para un intervalo definido y tiene tres parámetros de configuración: El tamaño mínimo del intervalo *Imin,* el tamaño máximo del intervalo *Imax* y una constante de redundancia *k*. Además, cuenta con tres variables: el tamaño de intervalo actual *I*, el tiempo dentro del intervalo actual *t*, y un contador *c*.

El funcionamiento del algoritmo Trickle es el siguiente:

1. Al comenzar la ejecución del algoritmo se establece un valor *I* comprendido entre el

intervalo [*Imin*, *Imax*]. El algoritmo empezara en este intervalo.

2. Cuando un intervalo se inicia, se reinicia a 0 el contador y se establece un tiempo aleatorio para dicho intervalo, cuyo valor está comprendido entre [*I/2, I*] y terminará en *I*.
3. Si se obtiene una transmisión consistente, el algoritmo incrementará su contador.
4. En un momento *t* del intervalo, se transmiten paquetes si y sólo si el contador es menor que la constante de redundancia *k*.
5. Cuando el intervalo *I* expire, se duplicará el tamaño del intervalo. Si el nuevo tamaño del intervalo es mayor que el tiempo especificado en *Imax*, se actualizará el tamaño del intervalo *I* por el valor de *Imax*
6. Si se detecta una transmisión inconsistente y el intervalo *I* es menor que *Imin*, se reinicia el temporizador del algoritmo. Una vez reiniciado, se establecerá el valor de *Imin* al que tenía *I* volviéndose al paso 2. Si el valor de *I* es igual al de *Imin* y la transmisión es inconsistente, no se hará nada, aunque podría reiniciarse el temporizador si ocurriera un evento externo.

(Levis, Clausen, Hui, Gnawali, & Ko, 2011)

En la Figura 5 se detalla el diagrama de flujo del algoritmo Trickle.

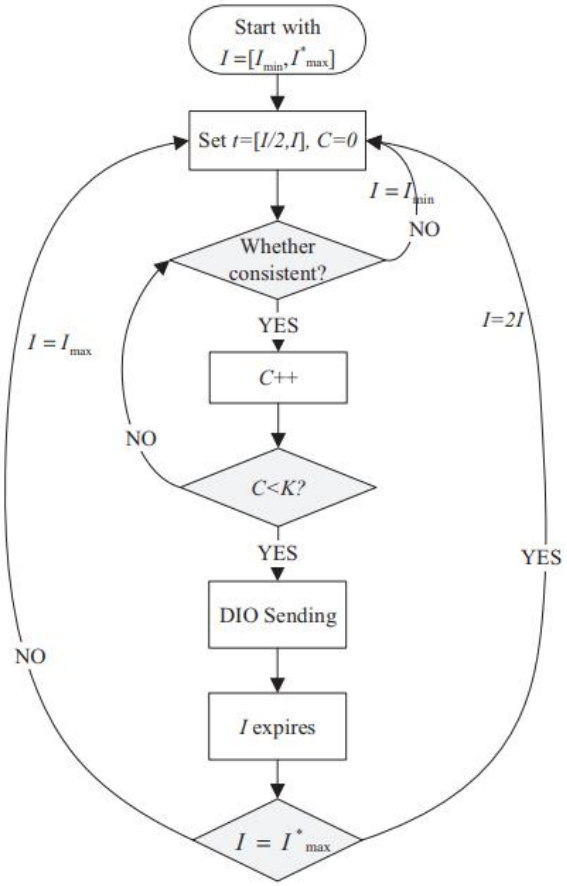


Figura 5. Diagrama de flujo del algoritmo Trickle. Tomado de Liu et al. (2017)

2.5.6 Algoritmo del protocolo. Una descripción general del algoritmo con el que el protocolo RPL construye el DODAG:

1. Algunos nodos son configurados para ser nodos raíz y contendrán toda la información del DODAG.
2. Los nodos seleccionados como raíz deben anunciar su presencia, afiliación con un DODAG, costo de enrutamiento, métrica y demás propiedades del DODAG, enviando mensajes de multidifusión DIO a todos los demás nodos.
3. Los nodos hoja escucharán los mensajes DIO y usarán su información para unirse a un nuevo DODAG, iniciando el proceso de selección de nodos padres o para

mantener un DODAG existente. Las decisiones de cada nodo dependen de la función objetivo configurada y el rango de sus nodos vecinos.

4. Los nodos proveen entradas en la tabla de enrutamiento para los destinos especificados en los mensajes DIO. Un nodo que decide unirse a un DODAG puede aprovisionar uno o más nodos padres como el siguiente salto para una ruta determinada y un número de otras rutas externas para la instancia asociada, pero solo enviara mensajes por un nodo padre preferido. Dependiendo del modo de operación (Storing o Non-storing) los nodos hojas también almacenarán los destinos provenientes de mensajes DAO.

(Winter et al., 2012)

2.6 Simulación orientada a eventos

Para comprender la idea principal de un simulador orientado a eventos es necesario conocer las siguientes definiciones:

- **Entidad:** Es una abstracción de un tema de interés en particular. Esta es descrita por sus atributos. El término *objeto* es usado comúnmente como un sinónimo.
- **Sistema:** Es un conjunto de entidades relacionadas entre sí con el fin de cumplir un propósito.
- **Sistema discreto:** Es un sistema cuyo estado, está definido por el estado de todas las entidades del sistema, cambia solo en un punto discreto en el tiempo. El cambio del estado es disparado por la ocurrencia de un evento.

La idea de un simulador orientado a eventos es saltar de un evento al siguiente, por lo que la ocurrencia de un evento puede ocasionar cambios en el estado del sistema, así como la generación de un nuevo evento llamado “avisos de eventos en el futuro”. La Figura 6 detalla este

principio de una forma gráfica.

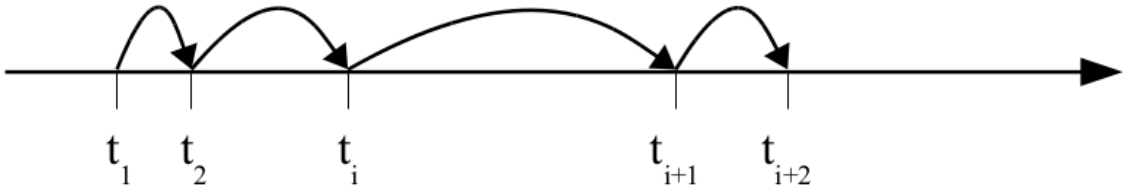


Figura 6. Principio de la simulación orientada a eventos discretos. Tomada de Wehrle, Mesut y Gross. (2010).

En general los simuladores basados en eventos discretos comparten los siguientes componentes:

- Estado del sistema.
- Reloj que proporciona el tiempo actual durante la simulación.
- Lista de eventos futuros.
- Contadores estadísticos.
- Rutina de inicialización.
- Rutina de tiempo que adelanta el reloj al tiempo de ocurrencia del siguiente evento en la lista de eventos futuros.
- Rutina de evento quees llamada cuando un evento particular ocurre.

(Wehrle, Mesut y Gross, 2010)

El algoritmo central de un simulador orientado a eventos discretos se define con el diagrama de flujo detallado en la Figura 7.

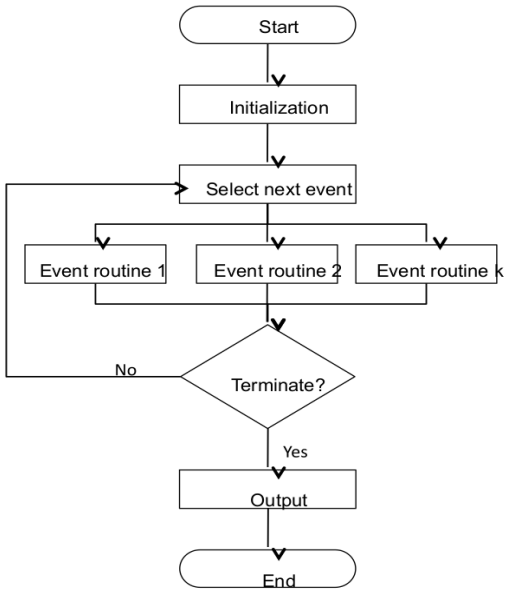


Figura 7. Diagrama de flujo del algoritmo central de un simulador basado en eventos. Tomado de Wehrle et al. (2010).

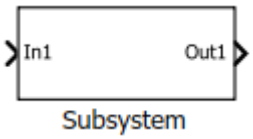
2.7 Herramientas de simulación

2.7.1 Matlab. Matlab es una plataforma de programación diseñada específicamente para ingenieros y científicos. El corazón de Matlab es el lenguaje Matlab, un lenguaje basado en matrices que permiten la expresión más natural de las matemáticas computacionales. Matlab combina un entorno de escritorio sintonizado para el análisis iterativo y los procesos de diseño con un lenguaje que expresa matemáticas de arreglos y matrices directamente. Con Matlab se pueden: analizar datos, desarrollar algoritmos, crear modelos y aplicaciones. El lenguaje, las aplicaciones y las funciones matemáticas integradas le permiten al usuario explorar rápidamente múltiples enfoques para llegar a una solución. Matlab permite llevar las ideas de la investigación a la producción mediante el despliegue en aplicaciones empresariales y dispositivos integrados (Embebidos), así como la integración con Simulink y el diseño basado en modelos. (“What is MATLAB?”, 2018)

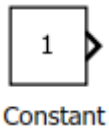
2.7.2 Simulink. Simulink es un producto adicional a MATLAB, que proporciona un entorno interactivo y gráfico para modelado, simulación y análisis de sistemas dinámicos. Permite la construcción rápida de prototipos virtuales para explorar conceptos de diseño en cualquier nivel con el mínimo esfuerzo. Para el modelado, Simulink proporciona una interfaz gráfica de usuario (GUI) para la construcción de modelos como diagramas de bloques. Incluye una completa biblioteca de bloques predefinidos para ser utilizados en la construcción de modelos gráficos. El usuario es capaz de producir el funcionamiento de un modelo, que de lo contrario requeriría horas para ejecutar en un entorno de laboratorio. Soporta sistemas lineales y no lineales, modelados en tiempo continuo, muestreo de tiempo, o híbrido de los dos. Por último, y no menos importante, Simulink está integrado con MATLAB y los datos se pueden compartir fácilmente entre los programas ("Simulink Tutorial", n.d.).

Para crear un modelo de simulación en Simulink se deben agregar bloques al diagrama, conectar y configurar cada uno de estos. Es posible asignar valores numéricos a los parámetros de cada bloque para controlar los cálculos del bloque durante la simulación y modificar su apariencia. Los bloques utilizados en la implementación del protocolo RPL fueron los siguientes:

- **Subsystem:** Permite crear un modelo jerárquico que contiene un subconjunto de bloques dentro de sí mismo. Es utilizado para reemplazar un conjunto de bloques agrupándolos en un solo subsistema para simplificar el modelo principal.

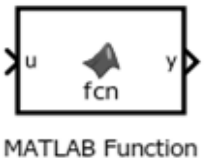


- **Constant:** Permite generar un valor constante real o complejo. También puede generar una salida escalar, vectorial o matricial.



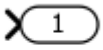
- **Scope:** Es utilizado para desplegar señales generadas durante la simulación.

- **Matlab Function:** Permite escribir una función de Matlab para usar en un modelo de Simulink. La función de Matlab creada se ejecuta y genera código para el codificador objetivo de simulink.

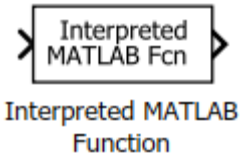


- **Input port:** Provee un puerto de entrada para un subsistema o modelo.

- **Output port:** Provee un puerto de salida para un subsistema o modelo



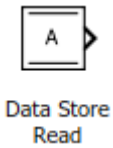
- **Matlab Interpreted Function:** Aplica una función específica de Matlab o una expresión para una entrada. La salida de la función debe coincidir con las dimensiones de salida del bloque.



- **Data Store Memory:** Define e inicializa un almacén de datos compartidos con un determinado nombre, el cual es una región de memoria utilizada por los bloques "Data Store Write" y el "Data Store Read" configurados con el mismo nombre.

- **Data Store Write:** Guarda el valor de su entrada en un almacén de datos específico. Cada operación hecha por este bloque sobrescribe el valor almacenado anteriormente.

- **Data Store Read:** Copia el valor de un dato almacenado a su puerto de salida. Más de un "Data Store Read" puede leer datos de un mismo "Data Store Memory".

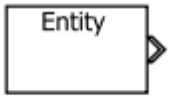


2.7.3 SimEvents. SimEvents ® es un toolbox de Simulink que proporciona un motor de simulación de eventos discretos y una biblioteca de componentes para analizar modelos de sistemas impulsados por eventos y optimizar las características de rendimiento como la latencia, el rendimiento y la pérdida de paquetes. Las colas, los servidores, los conmutadores y otros bloques predefinidos le permiten modelar enrutamiento, procesar retrasos y establecer

prioridades para la programación y la comunicación. Con SimEvents puede estudiar los efectos de la temporización de tareas y el uso de recursos en el rendimiento de sistemas de control distribuido, arquitecturas de software y hardware y redes de comunicación. También es utilizado para realizar investigaciones operativas de toma de decisiones relacionadas con la previsión, la planificación de la capacidad y la gestión de la cadena de suministro.

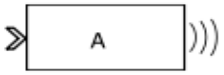
Los bloques de Simevents utilizados en la implementación del protocolo fueron los siguientes:

- **Generador de entidades:** Permite generar entidades. Las entidades son elementos discretos que pueden ser definidos en una simulación de eventos discretos, estas pueden llevar información de tipo escalar o vector. El significado de una entidad depende el modelo simulado, las entidades pueden representar clientes en un sistema de colas, información de paquetes desde un emisor hacia un receptor o cualquier elemento que se quiera definir.



Entity Generator

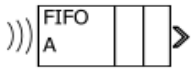
- **Multidifusor de entidades:** Permite difundir entidades a través del modelo creado. Una entidad que llega a este bloque es clonada y enviada con una etiqueta de multidifusión. Cada bloque Cola configurado con la misma etiqueta recibirá todas las entidades que sean enviadas.



Entity Multicast

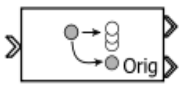
- **Cola receptora de multidifusión:** Almacena entidades en una cola basándose en el orden

de llegada como prioridad. La entidad que esté al comienzo de la cola se envía cuando el bloque siguiente esté listo para aceptarlo. Este tipo de bloque recibirá todas las entidades enviadas por un bloque Multidifusor de entidades configurado con la misma etiqueta.



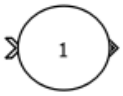
Multicast Receive Queue

- **Replicador de entidades:** Permite replicar entidades. Cuenta con dos salidas, una para enviar las entidades replicadas y la entidad original.



Entity Replicator

- **Servidor de entidades:** En una simulación de eventos discretos, un bloque Servidor almacena entidades por un periodo de tiempo denominado “Tiempo de servicio”. Una vez acabado el tiempo de servicio, la entidad es enviada a través del puerto de salida del bloque. El bloque puede servir múltiples entidades simultáneamente y enviar cada entidad por su puerto de salida, a menos que este esté ocupado.



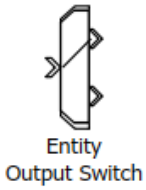
Entity Server

- **Terminador de entidades:** Permite aceptar y destruir entidades. Este bloque es utilizado para representar el fin de una entidad en el modelo.

Entity Terminator

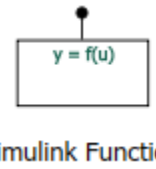
- **Interruptor de salida de entidades:** Permite seleccionar un puerto para la salida de una

entidad. Se puede programar un criterio apropiado para la elección del puerto de salida.



- **Transmisor de mensaje:** Permite leer una señal de entrada y crear y enviar un mensaje hacia un bloque.

- **Función de Simulink:** Es un bloque Subsistema preconfigurado como un punto de partida para definir gráficamente una función con otros bloques de Simulink. Este bloque provee una interfaz de texto para hacer los llamados a la función.



3. Marco metodológico

La metodología utilizada en el desarrollo de este trabajo se basó en tres fases principales. Primero una fase de profundización teórica, luego una fase de implementación y simulación y por último una fase de pruebas integrales y evaluación final de la implementación realizada.

3.1 Fase 1: Profundización teórica

En esta fase se realizó una revisión del protocolo, sus características y funcionamiento, además se realizó la búsqueda de herramientas y documentación proporcionadas por Matlab y Simulink para llevar a cabo la implementación.

Las actividades realizadas en esta fase fueron las siguientes:

1. Revisión detallada de las características y funcionamiento del protocolo RPL.
2. Identificación de requisitos del modelo de simulación del protocolo RPL.
3. Búsqueda y selección de herramientas proporcionadas por Matlab y Simulink que permitan la implementación del modelo.
4. Revisión y formación teórica sobre programación orientada a eventos.

3.2 Fase 2: Implementación y simulación

En esta fase se utilizó una metodología de desarrollo de software iterativa e incremental para la implementación del modelo de simulación del protocolo RPL, en la cual, a partir de los requisitos identificados en la primera fase, se planificó el proyecto en distintos bloques temporales que se le denominaron iteraciones. En cada iteración, se realizó cambios en el diseño y se agregaron nuevas funcionalidades y capacidades al modelo, así sucesivamente hasta que se cumplieron los requisitos previamente definidos. Dado que una de las características fundamentales de esta metodología es que el cliente esté involucrado a lo largo de todo el proyecto, el director de proyecto de grado hizo las veces de cliente, el cual, en cada iteración entregó una realimentación de los resultados obtenidos para avanzar a la siguiente. De igual manera, en cada iteración se llevó a cabo la documentación necesaria del código realizado y pequeñas pruebas unitarias de este.

Las actividades realizadas en esta fase fueron las siguientes:

1. Definición de escenarios y juegos de datos del modelo de simulación del protocolo

RPL.

2. Revisión de la incorporación de los requisitos en las especificaciones.
3. Implementación del modelo de simulación del protocolo RPL.
4. Pruebas unitarias del código implementado.
5. Ajustes a los módulos de código en caso de ser necesario.
6. Documentación de la implementación realizada.

3.3 Fase 3: Pruebas y evaluación final

En esta fase se llevó a cabo una serie de pruebas integrales de la implementación realizada, con el fin de verificar su correcto funcionamiento, evaluarla, y realizar el reporte final del trabajo. Además, se realizó la documentación del uso de las herramientas empleadas en la simulación.

Las actividades realizadas en esta fase fueron las siguientes:

1. Diseño y ejecución de pruebas integrales para la evaluación del modelo de simulación del protocolo RPL.
2. Revisión del cumplimiento de especificaciones y logro de objetivos.
3. Ajustes a los módulos de código en caso de ser necesario luego de las pruebas integrales.
4. Elaboración de la documentación final del uso del modelo de simulación implementado.
5. Reporte y divulgación de los resultados.

3.4 Diagrama metodológico

La Figura 8, resume en general la metodología utilizada para lograr los objetivos especificados.

Figura 8. Diagrama metodológico.

4. Requisitos y especificaciones

En este capítulo se detallan los requisitos funcionales y no funcionales definidos a partir de las normas publicadas por la IETF que definen el funcionamiento y características del protocolo RPL. Además, se muestran las especificaciones alcanzadas por la implementación final respecto a los requisitos definidos.

Debido al alcance del proyecto y el tiempo para su desarrollo, no se definieron ni implementaron características relacionadas a factores de seguridad del protocolo RPL.

4.1 Requisitos funcionales

Identificación del requisito	RF01
Nombre de requisito	Mensajes de control
Descripción	La implementación debe contar con el envío y procesamiento de los mensajes de control DIO, DAO, DIS y de manera opcional DAOAck. (Winter et al., 2012)
Prioridad	Alta

Identificación del requisito	RF02
Nombre de requisito	Modo de operación
Descripción	La implementación debe soportar al menos uno de los modos de operación del protocolo RPL, Non-storing o Storing. (Winter et al., 2012)
Prioridad	Alta

Identificación del requisito	RF03
Nombre de requisito	Uso de métricas
Descripción	La implementación debe hacer uso de al menos una de las métricas especificadas para realizar el cálculo de caminos en redes LLN. (Vasseur et al. 2012)
Prioridad	Alta

Identificación del requisito	RF04
Nombre de requisito	Temporizador Trickle
Descripción	La implementación debe hacer uso de un temporizador basado en el algoritmo Trickle, para detectar y responder frente a inconsistencias de la red. (Levis et al. 2011)
Prioridad	Alta

Identificación del requisito	RF05
Nombre de requisito	Tipo de trafico
Descripción	La implementación debe soportar los diferentes tipos de flujo de tráfico de redes, como punto a punto, punto a multipunto y multipunto a punto, para el envío de mensajes. (Winter et al., 2012)

Prioridad	Alta

Identificación del requisito	RF06
Nombre de requisito	Función objetivo
Descripción	La implementación debe permitir a los nodos hoja, seleccionar un nodo padre de un conjunto de nodos vecinos. (Winter et al., 2012)
Prioridad	Alta

Identificación del requisito	RF07
Nombre de requisito	Múltiples instancias
Descripción	La implementación debe contar con la funcionalidad de simular múltiples instancias del protocolo RPL con diferentes funciones objetivo. (Winter et al., 2012)
Prioridad	Alta

Identificación del requisito	RF09
Nombre de requisito	Pérdida de mensajes
Descripción	La implementación debe contar con un factor de pérdida de mensajes, que permita mostrar el comportamiento del protocolo RPL en estas circunstancias.
Prioridad	Alta

Identificación del requisito	RF08
Nombre de requisito	Carga desde archivo externo
Descripción	La implementación debe contar con la funcionalidad de poder cargar una topología definida desde un archivo

	externo, para poder crear juegos de datos y comparar resultados.
Prioridad	Media

4.2 Requisitos no funcionales

Identificación del requisito	RNF01
Nombre de requisito	Interfaz de la implementación
Descripción	La implementación debe contar con una interfaz gráfica que permita correr simulaciones sin tener conocimientos de Matlab o Simulink.
Prioridad	Media

Identificación del requisito	RNF02
Nombre de requisito	Presentación de resultados
Descripción	La implementación debe permitir observar los resultados del protocolo de una forma comprensible una vez acabada cada simulación.
Prioridad	Media

4.3 Especificaciones alcanzadas

La Tabla 2 detalla el estado final de la implementación realizada respecto a los requisitos descritos anteriormente.

Tabla 2
Comparativa requisitos contra especificaciones

Requisito	Logrado	Estado final
RF01	Sí	La implementación realiza el envío y procesamiento de los mensajes de control DIO, DAO, DIS y DAOAck.
RF02	Sí	La implementación cuenta con la funcionalidad de realizar simulaciones utilizando tanto el modo Storing como el Non-Storing.
RF03	Sí	La implementación cuenta con cuatro métricas de encaminamiento (Hop Count, ETX, Energy y Stability) para realizar simulaciones.
RF04	Sí	La implementación cuenta con un temporizador Trickle que permite el envío constante de mensajes DIO para mantenimiento y control de la red.
RF05	Sí	La implementación hace uso de los tres tipos de tráficos necesarios para el protocolo RPL. (Punto a punto, punto a multipunto y multipunto a punto)
RF06	Sí	La implementación cuenta con un método que permite a partir de una métrica configurada, calcular el rango de un nodo para realizar el proceso de selección de padre.
RF07	Sí	La implementación permite correr una simulación hasta con cuatro instancias diferentes.
RF08	Sí	La implementación permite cargar una topología definida desde un archivo de texto externo, el cual debe seguir el formato indicado.
RF09	Sí	La implementación permite configurar el porcentaje de pérdida de mensaje en cada nodo de la topología.
RNF01	Sí	La implementación cuenta con una interfaz gráfica amigable y de fácil uso.
RNF02	Sí	La implementación permite visualizar diferentes tipos de resultados de las simulaciones realizadas. Resultados como el grafo generado, tiempos de envío y recepción de mensajes, latencia etc.

5. Implementación del protocolo en Matlab y Simulink

En este capítulo se describe la forma como fue implementado el protocolo RPL en ambiente de simulación de Matlab y Simulink. La implementación realizada es un simulador basado en eventos, en la cual se utilizó Simulink para desarrollar una librería basada en el toolbox SimEvents, que realiza todos los procesos internos y envío de mensajes que se llevan a cabo en el protocolo RPL. Además, se utilizó Matlab para desarrollar una interfaz gráfica que permite hacer uso de esta librería de una forma sencilla y accesible a cualquier usuario, incluso sin tener conocimientos de Simulink.

5.1 Librería de Simulink

La librería de Simulink consta de tres bloques principales, que en el fondo son tres subsistemas que almacenan diferentes estructuras de bloques y en conjunto implementan el protocolo RPL en su núcleo. Los bloques desarrollados permiten la inclusión de ciertos parámetros de entrada en su máscara, que son globales dentro de todo el subsistema y determinan el comportamiento que tendrá la simulación.

5.1.1 Bloque nodo raíz. El bloque nodo raíz representa el único nodo en la topología configurado para servir de enrutador hacia otros DODAG. Este bloque es el encargado de almacenar y transmitir todos los parámetros iniciales de la simulación. Además, realiza el envío del primer mensaje de control en la simulación, dando inicio a la construcción del DODAG. En una simulación se configura únicamente un solo bloque de nodo raíz. Su estructura principal se basa en una serie de bloques que procesan y envían los diferentes mensajes de control del protocolo RPL**.** El aspecto que tiene el bloque del nodo raíz se muestra en la Figura 9.

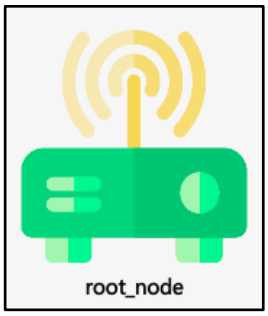


Figura 9. Bloque nodo raíz.

Los parámetros iniciales y estructura principal del bloque nodo raíz se detallan a continuación.

5.1.1.1 Parámetros iniciales.

- **Id:** Define el identificador único del nodo en la red.
- **Radio:** Define el alcance en metros que tiene el nodo para enviar mensajes.
- **Posición X:** Define la posición espacial en el eje x del nodo.
- **Posición Y:** Define la posición espacial en el eje y del nodo.
- **Posición Z:** Define la posición espacial en el eje z del nodo.
- **CPU_time:** Define el tiempo de procesamiento de cómputo del nodo.
- **Objective_function:** (Vector) Define las funciones objetivo que se utilizan para las diferentes instancias de la simulación.
- **Mode:** (Vector) Define los modos de operación que utilizaran las diferentes instancias en la simulación.
- **Channels:** Define en representación decimal los canales habilitados en cada nodo.
- **NumInstances:** Define el número de instancias que se están trabajando en la simulación actual.
- **NumNodes:** Define el número de nodos hoja que contiene la topología.
- **Seed:** Se utiliza para cambiar de forma aleatoria la semilla de todos los nodos, debido

a que en Simulink, esta se inicializa en 0 para todos los bloques.

5.1.1.2 Procesamiento de mensajes. El procesamiento de mensajes en el bloque de nodo raíz comienza con un bloque Cola FIFO que almacena los diferentes mensajes que llegan al nodo y procesa los que puede recibir. Los mensajes que un nodo puede escuchar están definidos por la variable de alcance del nodo y los canales habilitados en el emisor y el receptor.

El siguiente bloque en la secuencia es un bloque Interruptor, donde los mensajes que se deben descartar son enviados por el puerto 1 hacia un bloque Terminador y los mensajes que el nodo sí debe procesar son enviados por el puerto 2 hacia un bloque Servidor. El bloque Servidor determina el tiempo que tarda el mensaje en el medio antes de llegar al nodo. Si al nodo le corresponde procesar este mensaje lo envía a un siguiente bloque Servidor, que añade el nodo emisor a la tabla de vecinos del nodo raíz, de lo contrario descarta el mensaje enviándolo hacia un bloque Terminador.

El siguiente bloque en la estructura es otro bloque Servidor que se encarga de hacer efectivo el tiempo de procesamiento del nodo, además realiza el cálculo de la latencia del mensaje en la red y lo añade a la interfaz gráfica.

Una vez sale el mensaje del bloque Servidor, pasa por un bloque Interruptor que, dependiendo del tipo de mensaje, lo envía por un puerto u otro.

Si se trata de un mensaje DIO es enviado por el puerto 1 para ser procesado, pero no se llevará a cabo ninguna acción con este, ya que el nodo raíz no necesita unirse a ningún otro DODAG. Si se trata de un mensaje DAO es enviado por el puerto 2, el mensaje es procesado y se almacena el nodo emisor del mensaje en la tabla de rutas del nodo raíz. Además, se hace un llamado al método para enviar un mensaje DAOAck que permite confirmar la conexión del nodo hoja que envió el mensaje en el DODAG. Si se trata de un mensaje DIS, el mensaje es enviado

por el puerto 3 para ser procesado y se hace un llamado al método para enviar un respectivo mensaje DIO con la información del DODAG en el que se encuentra el nodo raíz. En el último caso, si llega un mensaje DAOAck, el mensaje es enviado por el puerto 4, en cuyo caso no se realiza ningún tipo de acción debido a que no es posible recibir un mensaje de este tipo siendo el nodo raíz el único capaz de generarlos.

El aspecto que tiene la estructura principal del nodo raíz se muestra en la Figura 10.

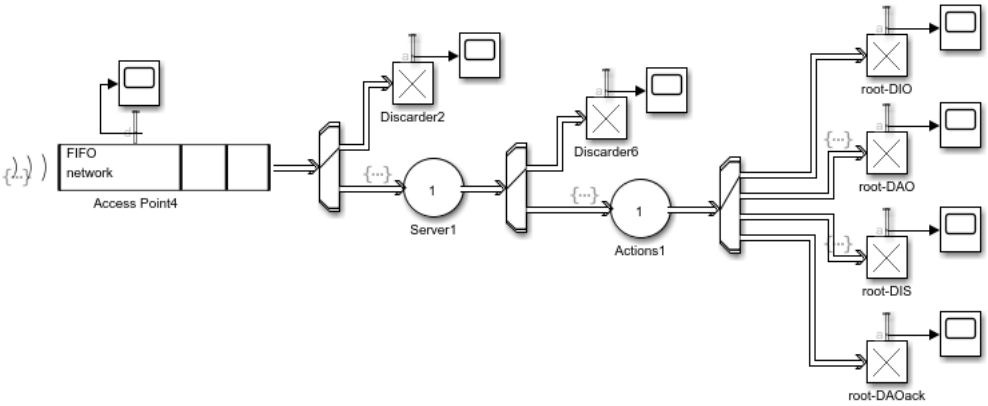


Figura 10. Estructura principal del nodo raíz.

5.1.1.3 Envío de mensajes. Para realizar el envío de mensajes, el bloque del nodo raíz cuenta con 3 estructuras fundamentales.

La primera estructura es el temporizador Trickle, que fue implementado por un bloque Generador de Entidades que envía mensajes multidifusión con un método de generación basado en tiempo. Este proceso se realiza sucesivamente hasta alcanzar el tiempo máximo permitido para el temporizador y que el generador se agote. Si el temporizador Trickle se agota y un nodo aún no se ha conectado a la red, existe la posibilidad de que este transmita un mensaje DIS, solicitando un DIO de respuesta, con lo que el temporizador Trickle se reiniciara, ya que detecta que la red no ha convergido.

Debido a que el simulador está limitado a un número máximo de cuatro instancias, esta estructura se repite por cada una de ellas. Si el simulador corre con menos de cuatro instancias, los generadores imprescindibles, no se activarán y todos sus mensajes serán descartados.

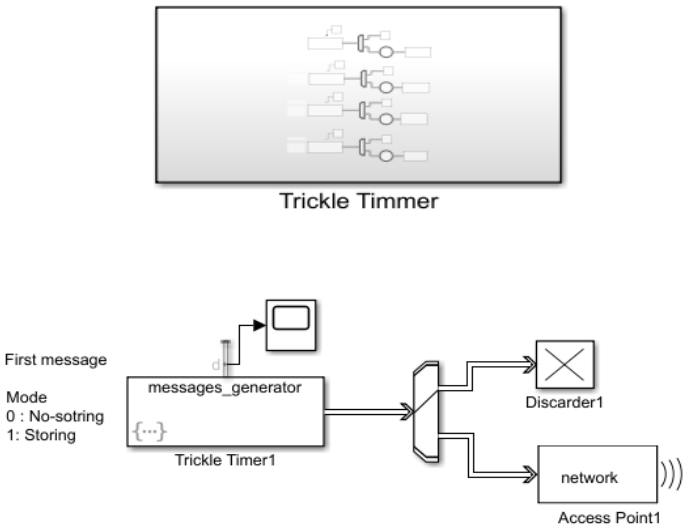


Figura 11. Estructura temporizador Trickle.

La segunda estructura es utilizada para realizar el envío de un mensaje DIO en respuesta a un mensaje DIS que el nodo raíz haya escuchado. Esta estructura se basa en una función de Simulink que manda una señal a un bloque Generador de Entidades basado en eventos, para que este genere un nuevo mensaje DIO.

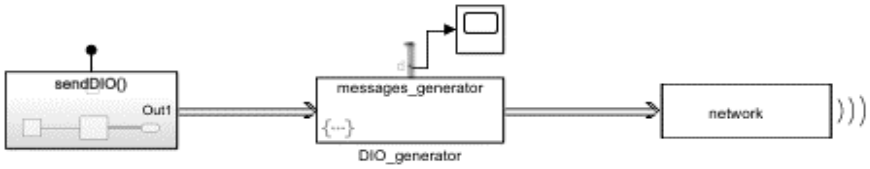


Figura 12. Estructura para el envío de mensajes DIO del bloque nodo raíz.

La última estructura se utiliza para enviar mensajes DAOAck en respuesta a la recepción de un mensaje DAO.

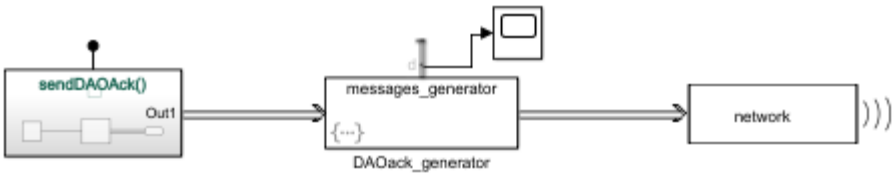


Figura 13. Estructura para el envío de mensajes DAOAck del bloque nodo raíz.

La estructura dentro de los subsistemas que anteceden a los bloques Generados de Entidades, en las estructuras de envió de mensajes, está compuesto por un bloque Constante que envía una señal a un bloque de Envío de mensaje y que se conecta a un puerto de salida del subsistema.

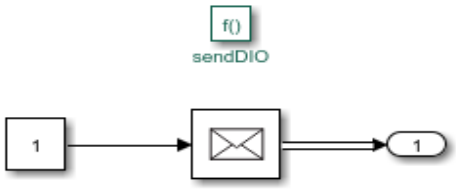


Figura 14. Estructura interna de subsistemas para envío de mensajes.

5.1.1.4 Métodos o funciones. Además de las estructuras para la recepción y envió de mensajes, el bloque de nodo raíz también cuenta con funciones o métodos que le permiten realizar ciertos procesos necesarios para su funcionamiento dentro de la simulación.

Existen dos tipos estructuras de los métodos utilizados por el bloque, estructuras de procesos y estructuras de almacenamiento de datos. Ambos tipos de estructura están compuestos por subsistemas que actúan como funciones de Simulink y en su interior contienen una serie de bloques que en conjunto realizan una determinada función.

Como se observa en la Figura 15, el núcleo de las estructuras para la ejecución de una función de procesos es un bloque de Función de Matlab, que en su interior contiene toda la programación necesaria para realizar dicha función.

El segundo tipo de estructura utiliza los bloques de escritura y lectura de datos para guardar ciertos valores que el nodo puede llegar a necesitar en procesos posteriores. Este tipo de

estructura se muestra en la Figura 16.

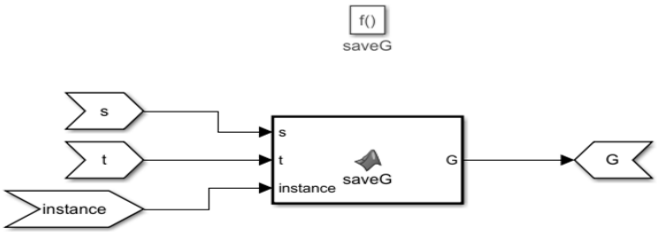


Figura 15. Ejemplo de estructura interna de función de procedimiento.

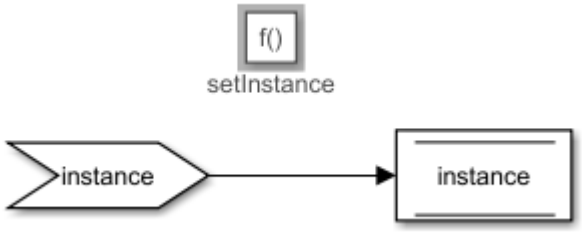


Figura 16. Ejemplo de estructura interna de función de almacenamiento.

Los métodos que posee en su interior el bloque Nodo Raíz son los siguientes:

- **setDestination(destination):** Almacena temporalmente el destino para el envío de un mensaje.
- **getDestination():** Obtiene el destino almacenado anteriormente.
- **setInstance(instance):** Almacena temporalmente la instancia en la que está trabajando la simulación.
- **getInstance():** Obtiene el valor de instancia almacenado anteriormente.
- **getRoute(G):** Retorna el camino más corto para llegar a un nodo cuando se opera bajo el modo Non-storing.
- **saveG(s, t):** Almacena y retorna el grafo acíclico construido a partir de los vectores de conexión entre nodos.

5.1.2 Bloque nodo hoja. El bloque nodo hoja representa cualquier dispositivo usado en una

topología de red LLN que no cuenta con las suficientes características para funcionar como nodo raíz. En una simulación pueden existir uno o más bloques de este tipo. El aspecto que tiene el bloque nodo hoja en la librería se muestra en la Figura 17.

Figura 17. Bloque nodo hoja.

Al igual que el bloque de nodo raíz, los bloques de nodos hoja también cuentan con ciertos parámetros iniciales de configuración interna y una estructura principal que realiza el procesamiento y envío de mensajes la cual se muestra en la Figura 18.

5.1.2.1 Parámetros iniciales.

- **ID:** Define el identificador único del nodo en la red.
- **Radio:** Define el alcance en metros que tiene el nodo para enviar mensajes.
- **Posición X:** Define la posición espacial en el eje x del nodo.
- **Posición Y:** Define la posición espacial en el eje y del nodo.
- **Posición Z:** Define la posición espacial en el eje z del nodo.
- **ETX:** Define el valor de la métrica ETX proporcionada por la capa física.
- **Stability:** Define el valor de la métrica “estabilidad” del nodo.
- **Energy:** Define el valor de la métrica “eficiencia energética” del nodo.
- **Imin:** Define el mínimo valor para el intervalo del algoritmo Trickle.

- **Imax:** Define el máximo valor para el intervalo del algoritmo Trickle.
- **K:** Define la constante de redundancia del algoritmo Trickle.
- **CPU_time:** Define el tiempo de procesamiento de cómputo del nodo.
- **Channels:** Define el número de canales habilitados para transmitir mensajes.
- **NumInstances:** Define el número de instancias que se están trabajando en la simulación.
- **Seed:** Se utiliza para cambiar de forma aleatoria la semilla de todos los nodos debido a que esta se inicializa en 0 por defecto para simulink en cada uno de sus bloques.

5.1.2.2 Procesamiento de mensajes. La secuencia de bloques comienza con un bloque Cola Multidifusión FIFO que se encarga de escuchar los mensajes de la red y descartar aquellos que el nodo no debería escuchar, utilizando el terminador conectado al siguiente bloque Switch.

Los mensajes que el nodo reciba pasarán primero por un bloque Servidor que filtra los mensajes a los cuales el nodo no deba realizar ninguna acción. En este punto se realiza el proceso de selección de nodo padre, con el fin de poder descartar mensajes DIO con métricas o rangos peores a los obtenidos mediante el padre actual, como, por ejemplo, recibir un mensaje DIO de rango mayor.

Una vez salgan los mensajes del bloque anterior, son enviados a otro bloque Servidor que se encarga de decidir qué hacer con este y modificar sus campos, dependiendo de su tipo, para luego hacer su posterior lectura o reenvío. Además, añade los respectivos emisores de los mensajes a la tabla de vecinos del nodo y de cumplir ciertas condiciones, lo añadirá también a su tabla de posibles padres.

Si se trata de un mensaje DIO, se guarda la información necesaria para enviar posteriormente

un mensaje DAO y activar el temporizador Trickle, con esto, además, empezar a enviar DIOs con el fin de difundir la información del DODAG.

Si se trata de un mensaje de tipo DAO, puede ocurrir dos escenarios. Trabajando en el modo Non-storing, el nodo sólo enviará el mensaje a su nodo padre. En caso contrario, trabajando en el modo Storing, el nodo además almacenará el nodo emisor a su tabla de rutas.

Cuando llega un mensaje de tipo DIS, en caso de que el nodo ya haga parte de algún DODAG, este responderá con un mensaje DIO, informando al emisor la existencia del DODAG al que pertenece.

Para los mensajes DAOAck, si el destino del mensaje concuerda con el identificador del nodo, este recibe y termina el mensaje, por otro lado, si el nodo no es el destino, lo reenvía a través de la ruta especificada por la raíz o por su tabla de rutas, dependiendo del modo de operación configurado en la instancia.

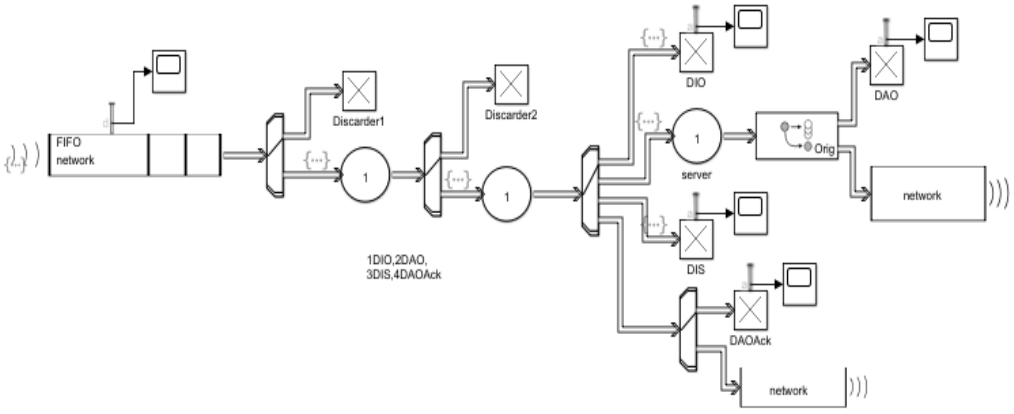


Figura 18. Estructura principal bloque nodo hoja.

5.1.2.3 Envío de mensajes. Para realizar el proceso de envío de mensajes el bloque nodo hoja cuenta con 3 estructuras principales. Estas estructuras se muestran en las Figuras 19, 20 y 21.

La primera estructura se utiliza para realizar el envío periódico de mensajes DIS, dependiendo del valor definido en el parámetro de entrada para esta función. Una vez el nodo haya entrado en

algún DODAG, el envío de mensajes DIS se detendrá.

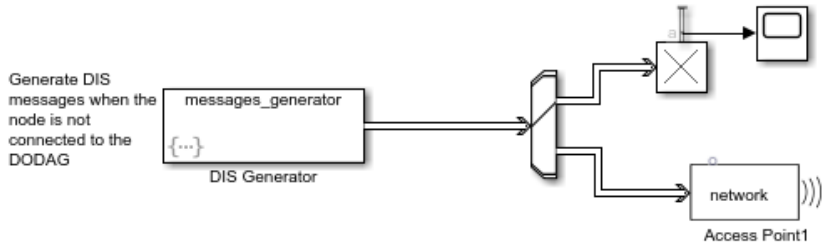


Figura 19. Estructura para envío de mensajes DIS del bloque nodo hoja.

La segunda estructura se utiliza para realizar el envío de mensajes DAO hacia el nodo raíz.

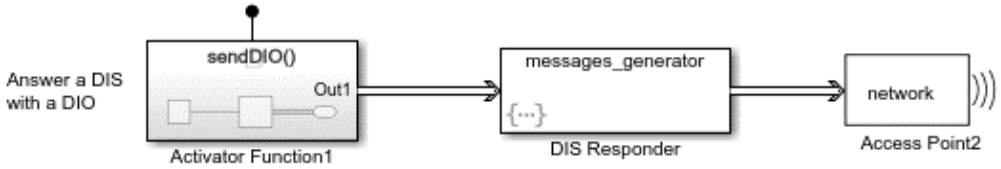


Figura 20. Estructura para envío de mensajes DAO del bloque nodo hoja.

La tercera estructura se utiliza para enviar un mensaje DIO en respuesta a un mensaje DIS.

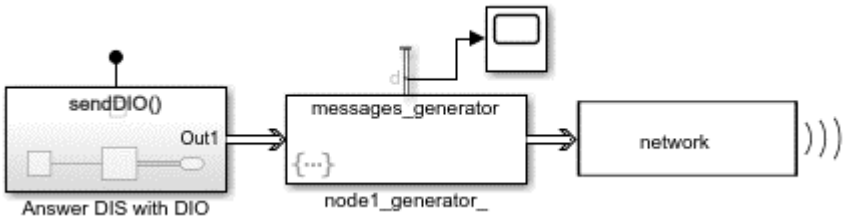


Figura 21. Estructura para envío de mensajes DIO del bloque nodo hoja.

El temporizador Trickle en el bloque de nodo hoja se implementó de la misma manera que la descrita en el nodo raíz, con la única de diferencia que solo se activará cuando el nodo hoja se conecte a un DODAG. Esta estructura se muestra en la Figura 11.

5.1.2.4 Métodos o funciones Al igual que en el bloque de nodo raíz, los bloques de nodo hoja también cuentan con ciertos métodos para realizar un procedimiento o almacenar valores necesarios para su funcionamiento. Estos métodos tienen la misma estructura que los descritos

para el nodo raíz.

Los métodos utilizados por los bloques nodo hoja son los siguientes:

- **SetInstance(instance):** Cambia temporalmente el valor de la instancia con la que está trabando la simulación.
- **getInstance():** Obtiene un valor de instancia almacenado con anterioridad.
- **setMode(mode):** Cambia temporalmente el valor del modo de operación con el que está trabajando la simulación.
- **getMode():** Obtiene un valor del modo de operación almacenado con anterioridad.
- **setObjectiveFunction(of):** Cambia temporalmente el valor de la función objetivo con el que está trabajando la simulación.
- **getObjectiveFunction():** Obtiene un valor de función objetivo almacenado con anterioridad.
- **setParent(parent):** Cambia el ID del nodo padre seleccionado.
- **getParent():** Obtiene el valor del padre seleccionado.
- **setHopcnt():** Cambia el valor del número de saltos.
- **getHopcnt():** Obtiene el valor del número de saltos almacenado.
- **setConnection(connection):** Cambia el valor el estado de conexión a un DODAG.
- **getConnection():** Obtiene el valor de estado de la conexión a un DODAG.
- **setTrickleActive(active):** Cambia el valor de estado del contador Trickle.
- **getTrickleActive():** Obtiene el valor de estado del contador Trickle.
- **objectiveFunction(active, of, metric, instance, rank):** Calcula el rango del

nodo teniendo en cuenta la función objetivo con la que esté trabajando la simulación.

5.1.3 Bloque de funciones generales. Además de los métodos detallados en los bloques de nodo raíz y nodo hoja, estos también utilizan algunos métodos generales que realizan una determinada función en ambos. Estos métodos se colocaron en un subsistema aparte para mejorar el rendimiento de las simulaciones y debe ser importado obligatoriamente en toda simulación. Los métodos que contiene este subsistema son los siguientes:

- **setConvergenceTime():** Asigna el tiempo de convergencia del protocolo.
- **drawMessage(instance, time, type, src, dest):** Dibuja el mensaje en el visualizador de eventos de la GUI.
- **addOrGetNextHop(instance, id, nexthop, hopCnt):** Obtiene o almacena el siguiente salto para el destino de un mensaje.
- **addToParentsTable(instance, id, parent):** Añade un nodo a la tabla de posibles padres.
- **addToNeighborsTable(instance, id, neighbor):** Añade un nodo a la tabla de vecinos.
- **showMessage(entity, type):** Presenta al usuario el formato del mensaje de control dependiendo de su tipo (DIO, DAO, DIS y DAOAck), además de esto señala quien envió, quien recibe y en qué tiempo fue entregado el mensaje.
- **deleteMessage(instance, node, dest):** Elimina un nodo de la tabla de rutas de un determinado nodo.
- **getTime():** Obtiene el tiempo actual de la simulación.
- **printValue(value):** Escribe en consola el valor ingresado.
- **plotGraph(G):** Manipula la figura que el grafo.

5.2 Interfaz gráfica

La interfaz gráfica de la implementación se desarrolló utilizando el Diseñador de Aplicaciones de Matlab, se compone de 3 secciones principales: Menú superior, panel de instancias y tabla de nodos. El aspecto global de la implementación se muestra en la Figura 22.

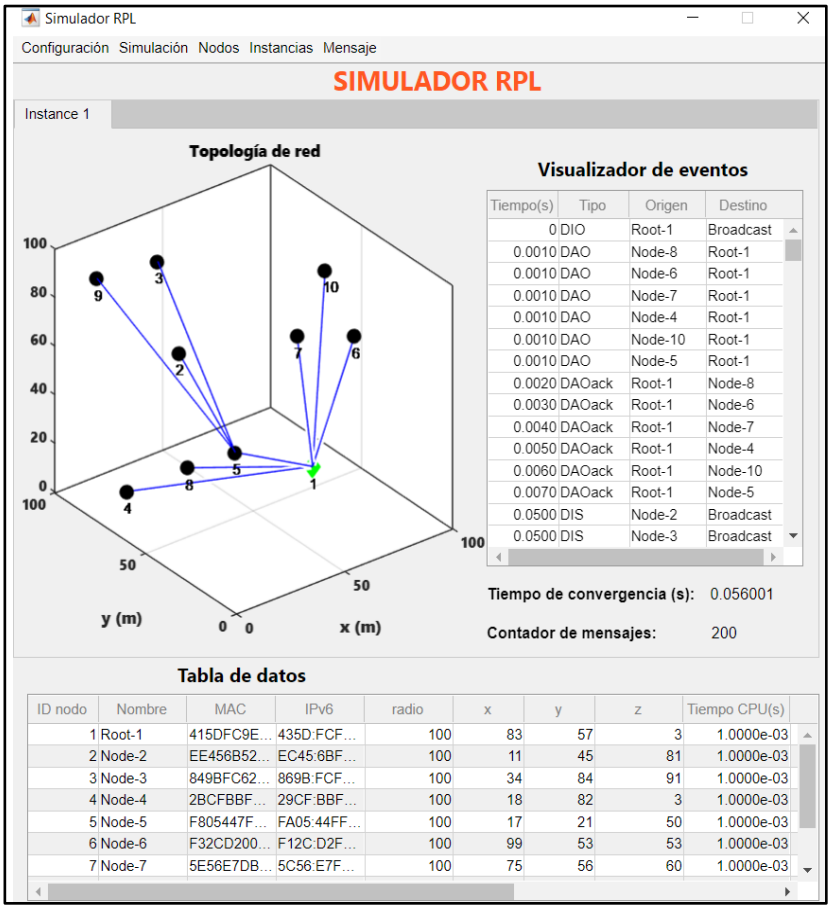


Figura 22. Interfaz gráfica de la implementación.

5.2.1 Menú superior. El menú superior permite el acceso a diferentes submenús, opciones y ventanas que permiten ajustar parámetros de la simulación.

5.2.1.1 Menú de configuración. En este menú, seleccionando la primera opción, se abre un cuadro de diálogo que permite ajustar algunos parámetros generales de la simulación, parámetros como el tiempo de simulación, el tipo de vista (2D o 3D) y las dimensiones que tendrá el área de la topología. El cuadro de diálogo de configuración general se muestra en la Figura 23.

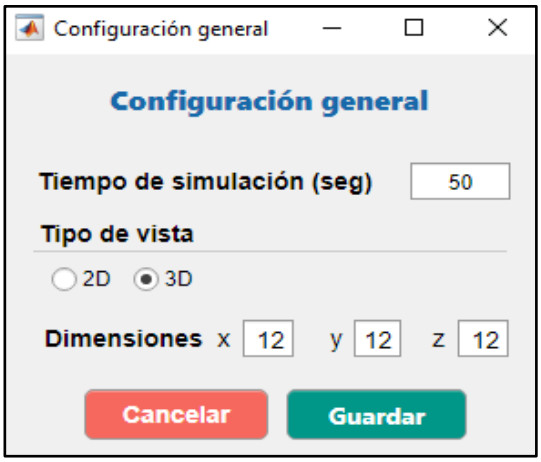


Figura 23. Cuadro de diálogo de configuración general.

5.2.1.2 Menú de simulación. En este menú se pueden llevar a cabo tres acciones: iniciar la simulación actual, ver algunos datos recolectados después de haber finalizada una simulación, como se muestra en la Figura 24 y finalmente, enviar un mensaje de usuario a través del DODAG, como se muestra en la Figura 25.

5.2.1.3 Menú de nodos. Permite agregar nodos ya sean de tipo raíz u hoja, editar y eliminar estos una vez han sido creado mediante la interfaz mostrada en la Figura 26, una última opción, permite cargar datos de los nodos desde un archivo de texto externo con el formato requerido.

5.2.1.4 Menú de instancias. Este menú permite crear, configurar y eliminar instancias en la simulación. Cada instancia cuenta con un submenú para ajustar sus parámetros, ver Figura 27, ver las tablas generadas (tablas de rutas, tablas de vecinos y tablas de posibles padres), como ejemplo de una tabla de rutas ver Figura 28, y finalmente limpiar las tablas.

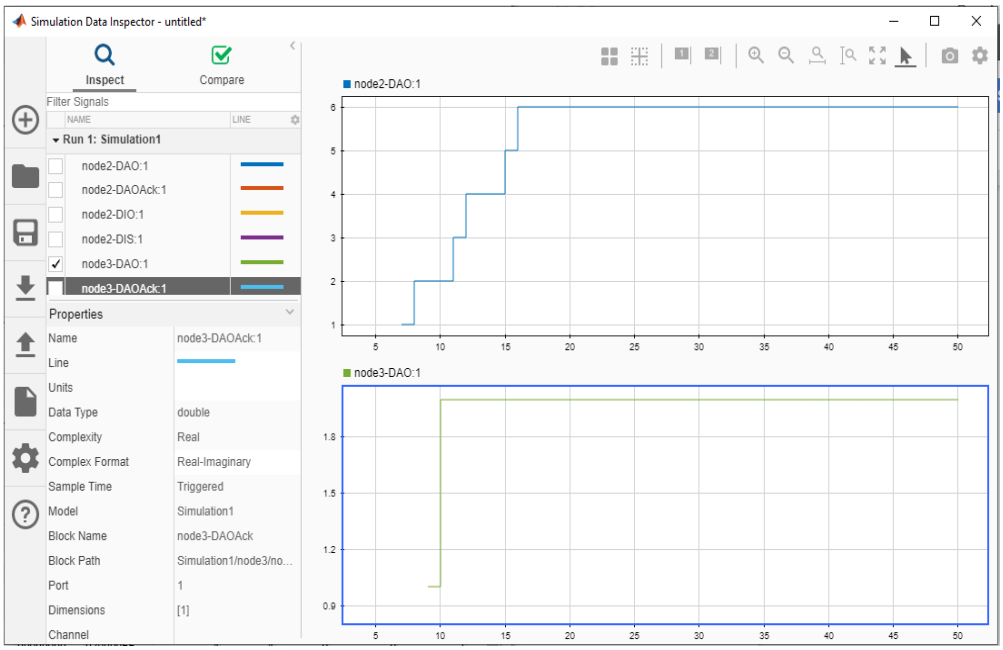


Figura 24. Inspector de datos de Simulink

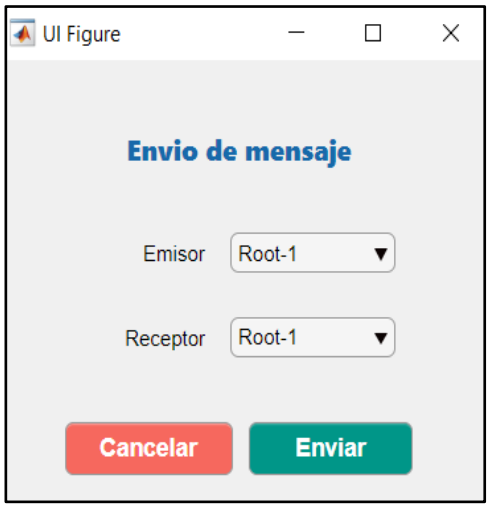


Figura 25. Interfaz para el envío de mensajes.

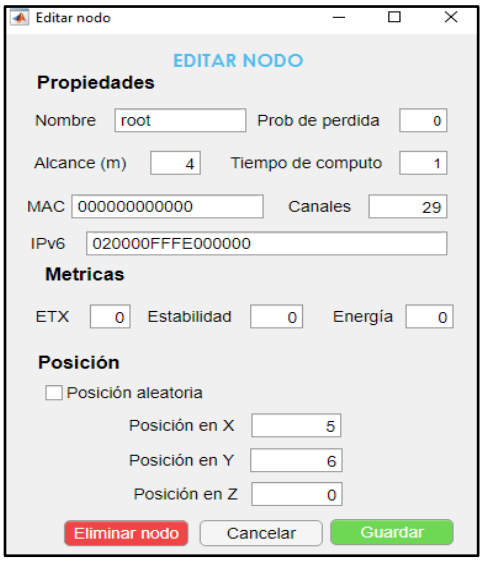


Figura 26. Cuadro de diálogo de edición de nodo.

Figura 27. Cuadro de diálogo de configuración de instancia

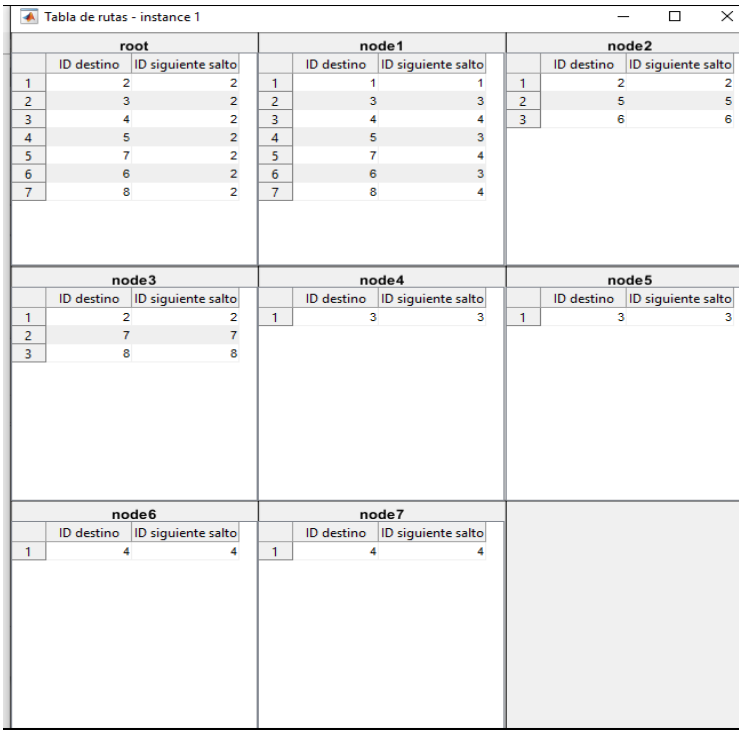


Figura 28. Tabla de rutas generada por la implementación.

5.2.2 Panel de instancias. Es un grupo de pestañas que contienen los resultados de la simulación para cada instancia configurada.

La parte izquierda de la interfaz gráfica contiene un diagrama de coordenadas cartesianas en tres dimensiones, que representara el DODAG generado para la instancia que se encuentre seleccionada. El nodo raíz se representa con un rombo verde y los nodos hoja con un círculo negro. Un ejemplo de un grafo generado en este diagrama se muestra en la Figura 29.

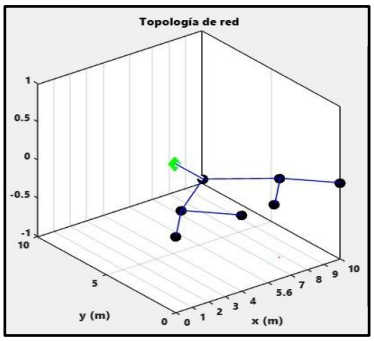


Figura 29. Diagrama cartesiano del grafo generado.

En la parte derecha, la interfaz gráfica cuenta con un visualizador de mensajes que se llenará con los diferentes mensajes que se envíen durante la simulación. La apariencia del visualizador de eventos se muestra en la Figura 30. Al seleccionar cualquiera de los mensajes dentro del visualizador, se abre un cuadro de diálogo con información más detallada de este mensaje, como se muestra en la Figura 31.

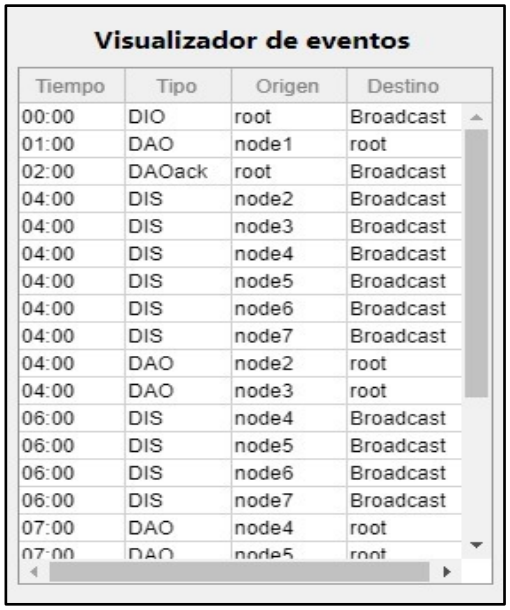


Figura 30. Visualizador de eventos.

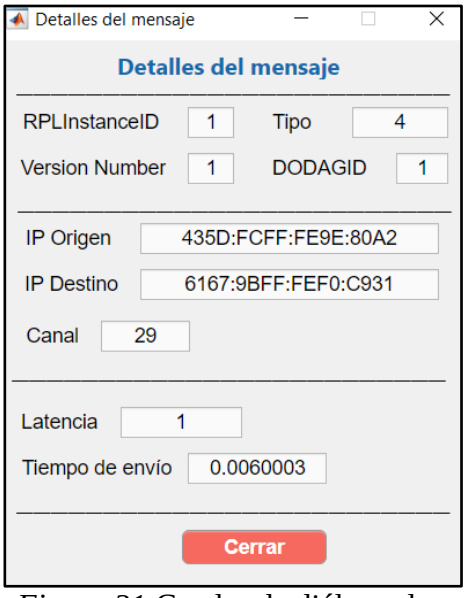


Figura 31. Cuadro de diálogo de detalles del mensaje.

5.2.3 Datos de nodos. En la parte inferior de la interfaz gráfica se encuentra una tabla que contiene toda la información de cada uno de los nodos configurados para la simulación. La apariencia de la tabla de nodos se muestra en la Figura 32.

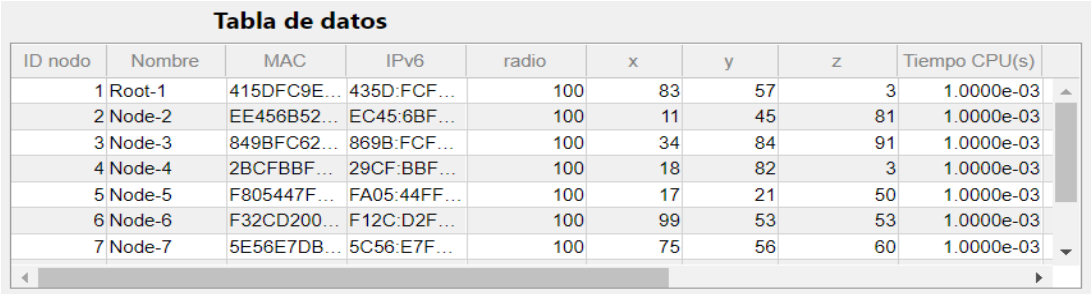


Figura 32. Tabla de datos de los nodos en la simulación.

5.2.4 Formato de datos. Para poder cargar una topología de nodos desde un archivo de texto externo, se deben seguir las siguientes especificaciones:

1. Cada nodo ocupa una línea en el archivo.
2. Cada atributo de los nodos debe ir separado por un espacio.
3. Los atributos deben seguir el siguiente orden: Id, nombre, dirección MAC, posición X, posición Y, posición Z, tiempo de cómputo, métrica ETX, métrica de estabilidad, métrica de nivel energético, canales habilitados y probabilidad de pérdida de mensajes.

Ejemplo: 1 Root-1 EB:71:CE:DE:9C:A0 100 54 63 95 0.001 0 0 0 29 0

Nota: El archivo debe ser de texto y debe estar ubicado en la carpeta raíz del proyecto.

6. Modo de uso

En este capítulo se detallan los pasos a seguir para realizar una simulación.

1. Instale el Simulador RPL en Matlab. Este proceso puede demorar unos minutos, ya que se descargaran e instalaran todas las dependencias necesarias para el uso del simulador.
2. Inicie la interfaz gráfica del simulador, escribiendo el comando RPL_GUI en la terminal de Matlab.
3. Una vez iniciada la interfaz gráfica, en el menú superior, en la opción *Configuración*, seleccione *Editar configuración* y ajuste los parámetros generales de la simulación.
4. (Opcional) Si desea configurar más de una instancia, en el menú superior, en la opción *Instancias*, seleccione *Nueva instancia* y añada tantas instancias del protocolo como desee, ajustando los parámetros de cada instancia.

5. En el menú superior, en la opción *Nodos*, seleccione *Añadir nodo raíz* y añada un nodo raíz a la simulación, ajustando sus parámetros iniciales. (En cada simulación sólo puede existir un nodo raíz)
6. En el menú superior, en la opción *Nodos*, seleccione *Añadir nodo hoja* y añada tantos nodos hojas como desee, ajustando los parámetros iniciales de cada nodo.
7. (Opcional). Puede omitir los pasos 4 y 5 cargando toda la información de la topología desde un archivo de texto externo. Cargue los datos de la topología utilizando en el menú superior la opción *Nodos*, y seleccione la opción de *Cargar datos*, se abrirá un cuadro de diálogo para seleccionar el archivo. (El archivo de texto debe seguir el formato descrito en el Capítulo 5)
8. Seleccione *Comenzar Simulación* en el menú *Simulación.*
9. Una vez terminada la simulación, podrá ver los resultados.

Nota: El Apéndice A es un video explicativo para correr una simulación simple.

7. Evaluación de la implementación

En este capítulo se describen los escenarios planteados para comprobar el funcionamiento de la implementación realizada, en base a los requerimientos funcionales y no funciones definidos en el Capítulo 5. Además, se muestran los resultados obtenidos a partir de estos escenarios.

Las especificaciones del equipo de cómputo que se utilizó en las simulaciones y las versiones del software se detallan en la Tabla 3 y en la Tabla 4.

Tabla 3
Especificaciones equipo de cómputo.

Tipo	Referencia
Procesador	Intel Core i5 8400 2.8 GHz - 4.0 GHz
Arquitectura del procesador	64 bits, procesador x64
Memoria RAM	16 Gb a 2666 MHz
Almacenamiento	SSD 240 Gb

Tabla 4
Versiones del software utilizado.

Software	Versión
Sistema operativo	Windows 10
Matlab	2018b
Simulink	9.2
SimEvents	5.5
Communications Toolbox	7.0

Se diseñaron 5 escenarios de topologías urbanas con nodos estáticos para realizar las simulaciones. Las dimensiones de los escenarios se definieron tomando en cuenta las áreas aproximadas de barrios pequeños de la ciudad de Bucaramanga. Los detalles de los escenarios planteados se resumen en la Tabla 5.

Tabla 5
Escenarios simulados

Escenario	Numero de Simulaciones	Dimensiones
1	4	100 x 100 m
2	2	100 x 100 x 100 m
3	5	100 x 100 x 100 m

4	5	500 x 500 x 500 m

7.1 Escenario 1: Múltiples instancias

El primer escenario se diseñó para comprobar el funcionamiento de la implementación utilizando múltiples instancias del protocolo RPL con diferentes funciones objetivo.

7.1.1 Diseño del escenario. En este escenario se corrieron cuatro simulaciones variando el número de instancias en cada una de ellas. Para lograr una mejor visualización de los grafos generados en cada instancia, se posicionaron todos los nodos en la coordenada 0 del eje Z. Además, se configuraron los nodos con un radio de alcance de 40 metros y con un porcentaje de pérdida de mensajes del 0%, debido a que no es el propósito de esta simulación comprobar el efecto de la pérdida de mensajes. Se generaron los valores de las métricas de forma aleatoria para cada nodo. En cuanto al modo de operación, todas las instancias fueron configuradas con el modo Non-storing, debido a que este no afecta el objetivo del escenario. Los detalles de las simulaciones realizadas y las características la topología utilizada se resumen en la Tabla 6 y en la Tabla 7.

Tabla 6
Simulaciones del escenario 1.

Simulación	Numero de instancias	Métricas usadas
1	1	Stability
2	2	Stability, Energy
3	3	Stability, Energy, ETX
4	4	Stability, Energy, ETX, Hop Count

Tabla 7
Características topología utilizada en el escenario 1.

Característica	Valor
Numero de nodos	15
Posicionamiento de nodos	Aleatorio (X, Y) - 0 (Z)
Alcance de nodos	40 (m)
Tiempo de cómputo de nodos	0.001 (s)
Porcentaje de pérdida	0%
Canales	1
Modo de operación	Non-Storing

El archivo de datos de la topología utilizada para las simulaciones de este escenario se adjunta en el Apéndice B.

7.1.2 Resultados obtenidos. La Tabla 8 resume los tiempos de convergencia obtenidos para todas las simulaciones realizadas en el escenario 1. Los grafos generados para las tres primeras simulaciones fueron idénticos a obtenidos en la simulación 4, los cuales se muestran de la Figura 33 a la Figura 36.

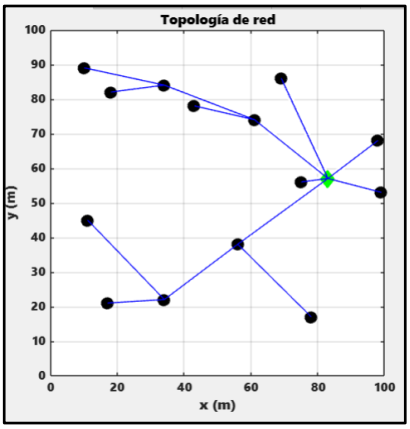


Figura 33. DODAG generado mediante Hop Count en el escenario 1.

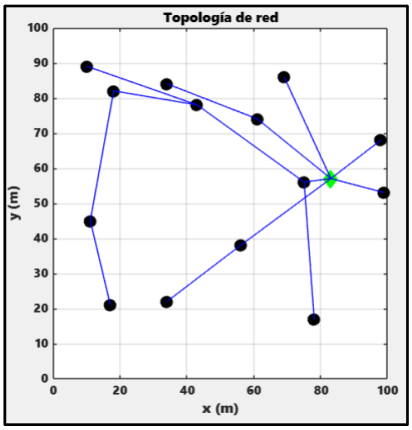


Figura 34. DODAG generado mediante ETX en el escenario 1.

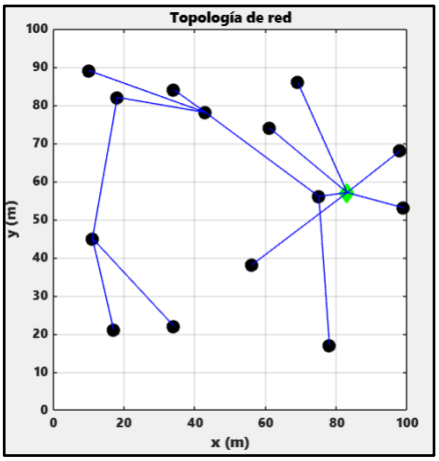


Figura 35. DODAG generado mediante Stability en el escenario 1.

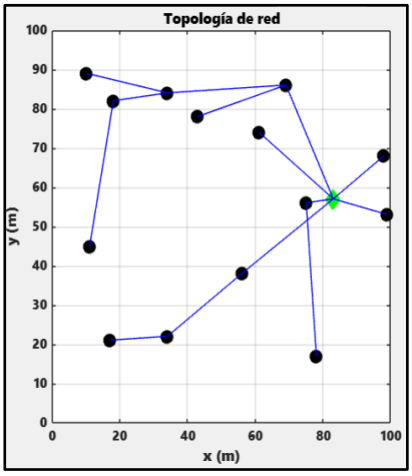


Figura 36. DODAG generado mediante Energy en el escenario 1.

Tabla 8

Tiempos de convergencia del escenario 1.

Métrica / Simulación	Stability	Energy	ETX	Hop Count
1	0.2085 (s)	N/A (s)	N/A	N/A
2	0.4057 (s)	0.21246 (s)	N/A	N/A
3	0.41571 (s)	0.23625 (s)	0.33771 (s)	N/A
4	0.44921 (s)	0.271 (s)	0.41846 (s)	0.2115 (s)

7.1.3 Análisis de resultados. A partir de los datos obtenidos de cada simulación, se compararon los tiempos de convergencia para las instancias con métrica de estabilidad, contra el número de instancias que habían configurado en cada simulación, esto debido a que esta era la única métrica presente en las 4 simulaciones. Esta comparación se observa en la figura 37.

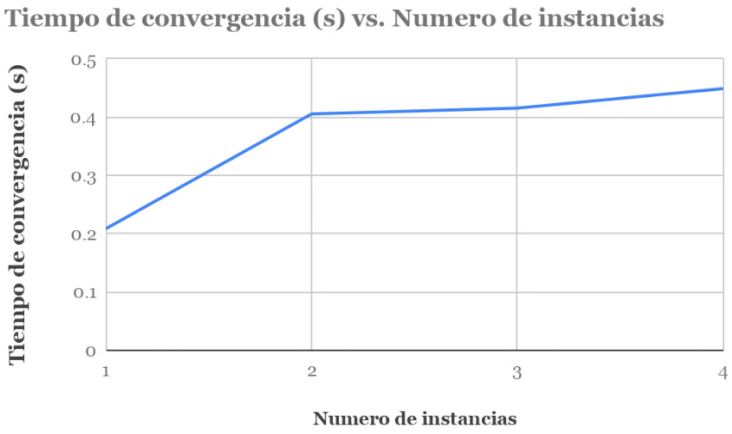


Figura 37. Tiempo de convergencia vs número de instancias en el escenario 1.

A partir de la comparativa anterior, se pudo observar que el tiempo de convergencia entre la simulación 1 y 2 se vio significativamente afectado por el tráfico presente en la red. Aunque la diferencia entre los tiempos de convergencia de las simulaciones 3 y 4 también aumentó, esta diferencia no fue tan grande como la anterior. De este comportamiento se infiere que, a mayor número de instancias, los tiempos de convergencia también serán mayores, debido a que cada instancia maneja sus propios mensajes de control de forma independiente, incrementando tráfico de la red.

Por otro lado, en base a los grafos generados por la última simulación, configurada con el número máximo de funciones objetivo disponibles en la implementación, se observó que, dependiendo de la métrica y los valores asignados en los nodos para esta, el protocolo generó un DODAG distinto. Lo anterior permite observar que el protocolo RPL fue diseñado para transportar tráfico con diferentes necesidades.

7.2 Escenario 2: Modo de operación

El escenario 2 fue diseñado con el fin de comprobar el funcionamiento de la implementación con los diferentes modos de operación del protocolo RPL, Storing y Non-storing.

7.2.1 Diseño del escenario. En este escenario se corrieron dos simulaciones con una única instancia, utilizando la métrica de ETX y cada una con un modo de operación distinto (Storing y Non-Storing). Los detalles de las simulaciones se resumen en la Tabla 9.

Tabla 9
Simulaciones escenario 2.

Simulación	N° instancias	Métrica usada	Modo de operación
1	1	ETX	Storing
2	1	ETX	Non-Storing

Se configuró una topología de 20 nodos con posiciones aleatorias, un radio de 60 metros de alcance y una probabilidad de pérdida de mensajes del 0%. Los detalles de la topología utilizada en la simulación se resumen en la Tabla 10.

Tabla 10
Características topología utilizada en el escenario 2.

Característica	Valor
Numero de nodos	20
Posicionamiento de nodos	Aleatoria
Alcance de nodos	60 (m)
Tiempo de cómputo de nodos	0.001 (s)
Porcentaje de pérdida	0%
Canales	1

El archivo de datos de la topología utilizada para las simulaciones de este escenario se adjunta

en el Apéndice C.

7.2.2 Resultados obtenidos. Tanto para la simulación con el modo Storing y Non-storing, se obtuvo un tiempo de convergencia de 0.273 segundos, se enviaron un total de 775 mensajes y el DODAG generado se muestra en la Figura 38. Las tablas de rutas generadas para cada simulación se adjuntan en los apéndices L y M.

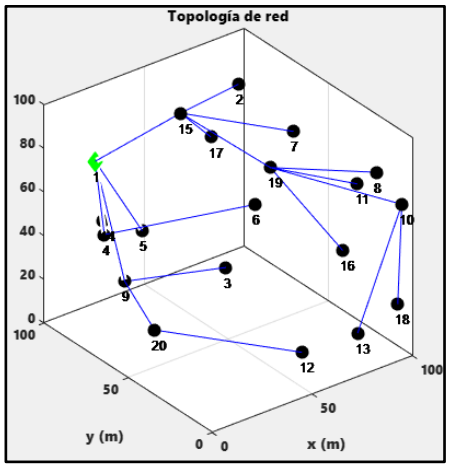


Figura 38. DODAG generado en el escenario 2.

Una vez terminada la construcción del DODAG en cada simulación, se envió un mensaje, seleccionando el nodo 16 como origen y nodo 2 como destino. Esto con el fin de analizar la diferencia en el funcionamiento de los modos de operación. En la primera simulación, la ruta obtenida para llegar de origen a destino fueron los nodos 16, 19, 15, 2, por otro lado, en la segunda simulación, la ruta obtenida fue a través de los nodos 16, 19, 15, 1, 15 y 2.

7.2.3 Análisis de resultados. En el escenario 2 se pudo observar que, aunque el comportamiento en las tablas de enrutamiento fue diferente en las simulaciones, en ambas se obtuvo el mismo tiempo de convergencia para la red y se generó el mismo grafo. De lo anterior se puede inferir que independientemente del modo de operación en el que trabaje una instancia

RPL, la red va a converger al mismo tiempo y de igual forma, ya sea que esté usando un modo u otro.

Con los resultados obtenidos de las tablas de rutas en las simulaciones, se notó que el comportamiento que tuvieron los nodos fue diferente, tanto en el modo Storing como en el modo Non-Storing. En el modo Storing, los nodos intermedios entre el nodo emisor y el nodo raíz, fueron almacenando la ruta hacia el nodo emisor. Por el contrario, en el modo Non-storing, estos nodos solo almacenaron la ruta de su respectivo nodo padre.

Donde sí se puedo ver una diferencia entre los modos de operación fue en el mensaje enviado al finalizar la construcción del DODAG, donde utilizando el modo Storing, el número de saltos que tuvo que hacer el mensaje para llegar al destino fue inferior al que se hizo en el modo Non-storing. Esto es posible ya que, en el primero de los modos, un nodo intermediario entre el nodo raíz y el nodo emisor puede decidir si enviar el mensaje directamente al destino, si este lo conoce en su tabla de rutas. A diferencia del modo Non-storing en el que un mensaje tiene que viajar hasta el nodo raíz para poder encontrar su destino.

7.3 Escenario 3: Pérdida de mensajes

El escenario 3 fue diseñado para comprobar el funcionamiento de la pérdida de mensajes y cómo una topología RPL responde ante en estas circunstancias en la implementación realizada.

7.3.1 Diseño del escenario. Para este escenario se corrieron cinco simulaciones, con una única instancia, utilizando el conteo de saltos como métrica de enrutamiento y el modo de operación Non-storing. En cada simulación se varió el porcentaje de pérdida de mensajes que tenían los nodos. Además, se dejó un tiempo de simulación de 50 segundos para todas las simulaciones. Los detalles de las simulaciones realizadas en este escenario se resumen en la

Tabla 11.

Tabla 11
Simulaciones escenario 3.

Simulación	Porcentaje de pérdida
1	0 %
2	10 %
3	20 %
4	30%
5	50%

Se configuró una topología de 25 nodos para todas las simulaciones. La posición de los nodos fue escogida de manera aleatoria y todos con un rango de alcance de 70 metros. Los detalles de la topología utilizada para este escenario se resumen en la Tabla 12.

Tabla 12
Características topología utilizada en el escenario 3

Característica	Valor
Numero de nodos	25
Posicionamiento de nodos	Aleatoria
Alcance de nodos	70 (m)
Tiempo de cómputo de nodos	0.001 (s)
Canales	1

Tiempo de simulación	50s

Los archivos de datos de las topologías utilizadas en este escenario se adjuntan del Apéndice D al Apéndice G.

7.3.2 Resultados obtenidos. Los tiempos de convergencia obtenidos para las simulaciones realizadas del escenario 3 se detallan en la Tabla 13. Los DODAG generados en cada simulación se muestran de la Figura 39 a la Figura 43.

Tabla 13

Tiempos de convergencia de simulaciones en el escenario 3.

Simulación	Tiempo de convergencia (s)
1	0.068
2	0.19996
3	0.46524
4	1.9035
5	4.4098

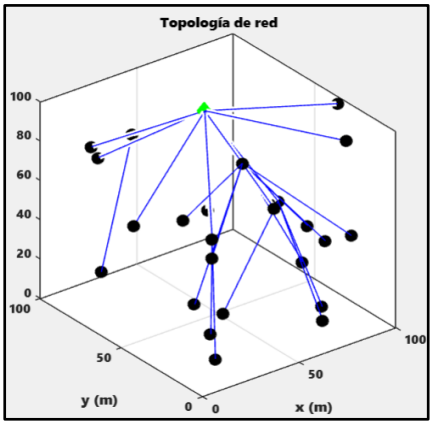


Figura 39. DODAG generado con 0 % de probabilidad de pérdida en el escenario 3.

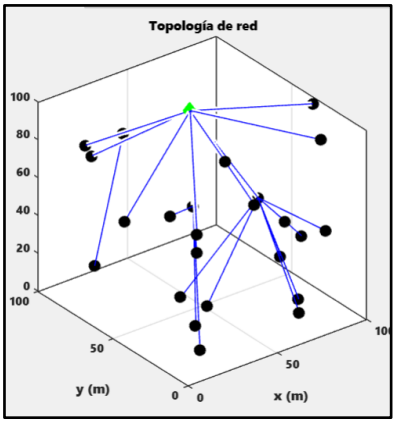


Figura 40. DODAG generado con 10 % de probabilidad de pérdida en el escenario 3.

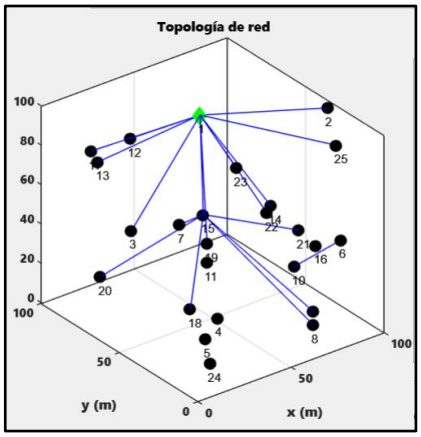


Figura 41. DODAG generado con 20 % de probabilidad de pérdida en el escenario 3.

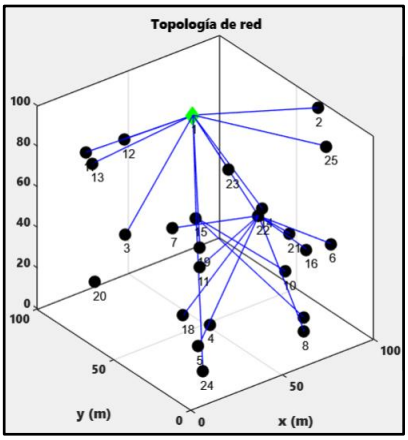


Figura 42. DODAG generado con 50 % de probabilidad de pérdida en el escenario 3

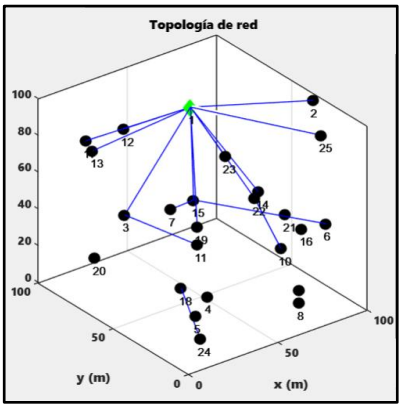


Figura 43. DODAG generado con 60 % de probabilidad de pérdida en el escenario 3

7.3.3 Análisis de resultados

En las dos primeras simulaciones, las cuales fueron las únicas en llegar a la convergencia de la red, se pudo observar que la probabilidad de pérdida de los mensajes afecta directamente el tiempo de convergencia, haciendo que, a mayor probabilidad de pérdida, se de mayor tiempo de convergencia. Lo anterior se debe a que el DODAG tiene que esperar la generación de nuevos mensajes de control para compensar los mensajes que se perdieron en el proceso de construcción. El encargado de realizar está función es el temporizador Trickle.

Con las simulaciones 3, 4 y 5 quedo en evidencia que, si existe una probabilidad de perdida

demasiada alta en los nodos de la topología es muy posible que no se llegue a una convergencia de la red. Configurando parámetros de Imin e Imax del temporizador Trickle con valores suficientes, esta convergencia podría llegar a darse, pero en un tiempo sustancialmente mayor.

7.4 Escenario 4: Rendimiento del protocolo y de la implementación

El escenario 4 se diseñó con el fin de medir y analizar el tiempo de convergencia del protocolo RPL y de estudiar el rendimiento de la implementación realizada en el ambiente de simulación de Matlab y Simulink.

7.4.1 Diseño del escenario. En este escenario se corrieron cuatro simulaciones, con una única instancia, usando el conteo de saltos como métrica de encaminamiento y el modo de operación Non-storing. Se configuró un tiempo de simulación de 5 segundos y en cada simulación se varió las dimensiones y el número de nodos presente en la topología. Los detalles de las simulaciones se resumen la Tabla 14.

Tabla 14
Simulaciones escenario 4.

Simulación	N° de nodos	Tiempo de simulación (s)	Dimensiones (m)
1	10	5	100 x 100 x 100
2	20	5	200 x 200 x 200
3	30	5	300 x 300 x 300
4	40	5	300 x 300 x 300

Se configuraron los nodos de la topología con posiciones aleatorias, un radio de alcance de 100 metros, una probabilidad de pérdida de mensajes del 0% y un tiempo de computó de 0.001 segundos. Los detalles de la topología utilizada se resumen en la Tabla 15.

Tabla 15
Características topología utilizada en el escenario 4.

Característica	Valor
Posicionamiento de nodos	Aleatoria
Alcance de nodos	100 (m)
Tiempo de cómputo de nodos	0.001 (s)
Probabilidad de pérdida	0 %
Canales	1

Los archivos de datos de las topologías utilizadas en este escenario se adjuntan del Apéndice H al Apéndice K.

7.4.2 Resultados obtenidos. Los tiempos de convergencia y número total de mensajes enviados obtenidos para las simulaciones realizadas del escenario 4 se detallan en la Tabla 16. Los grafos generados para cada simulación se muestran de la Figura 44 a la Figura 47. Los tiempos aproximados de compilación e inicialización de la librería de Simulink obtenidos para cada simulación se detallan en la Tabla 17.

Tabla 16
Tiempos de convergencia simulaciones escenario 4.

Simulación	Tiempo de convergencia (s)	N° mensajes enviados
1	0.05525	164
2	0.112	382
3	0.33375	669
4	0.42785	863

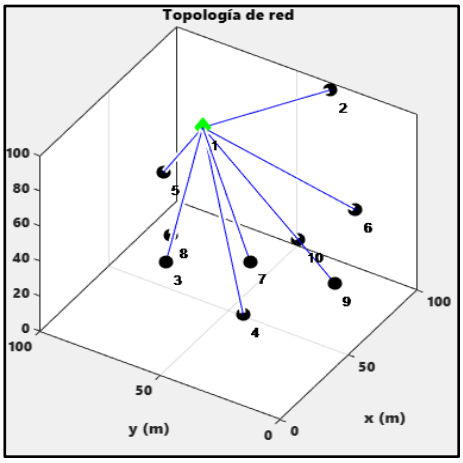


Figura 44. DODAG generado con 10 nodos en el escenario 4.

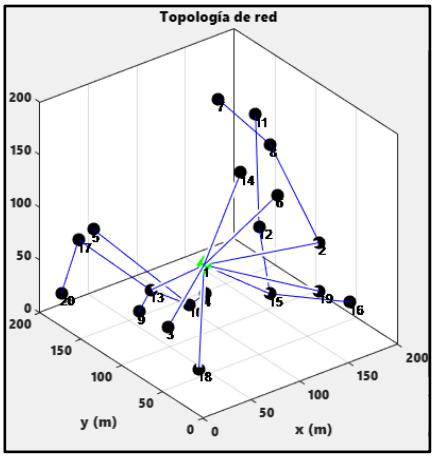


Figura 45. DODAG generado con 20 nodos en el escenario 4.

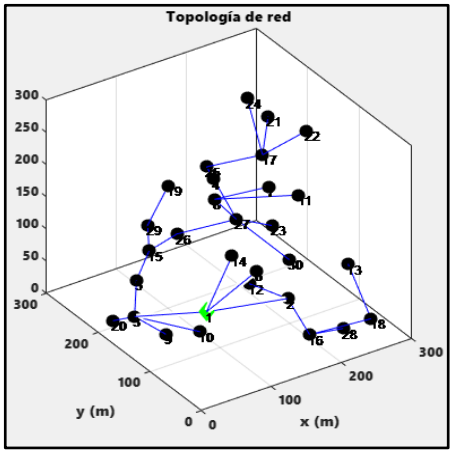


Figura 46. DODAG generado con 30 nodos en el escenario 4.

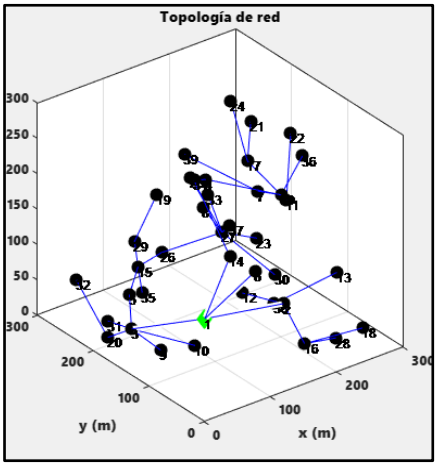


Figura 47. DODAG generado con 40 nodos en el escenario 4.

Tabla 17

Tiempos de compilación de simulaciones en el escenario 4.

Simulación	Tiempo de compilación e inicialización (m)
1	5
2	15
3	30
4	60

Mientras se corrían la simulación se utilizó el Administrador de Tareas de Windows para

medir el uso de recursos de este entorno de simulación. En la Tabla 18, se detallan las mediciones del uso promedio de CPU y la máxima memoria RAM utilizada por las simulaciones realizadas.

Tabla 18

Uso de recursos del escenario 4.

Simulación	Porcentaje uso de CPU	Memoria RAM utilizada
1	25 %	2.095 MB
2	25. 7 %	3.500 MB
3	25.9 %	4.490 MB
4	26.2 %	5.160 MB

7.4.3 Análisis de resultados. Con el escenario final se observó una aproximación del desempeño que tiene el protocolo RPL en un entorno con un número de nodos mayor.

Basado en los resultados obtenidos se observó que hay una relación directamente proporcional entre el número de nodos de la topología y el tiempo de convergencia de la red. Entre mayor sea el número de nodos de la topología, el tiempo de convergencia también será mayor. De igual manera ocurre con el número total de mensajes enviados en la simulación, ya que entre más nodos existan en la topología, el flujo de tráfico de mensajes será mucho mayor. Estas relaciones se pueden observar en la Figura 48 y en la Figura 49.

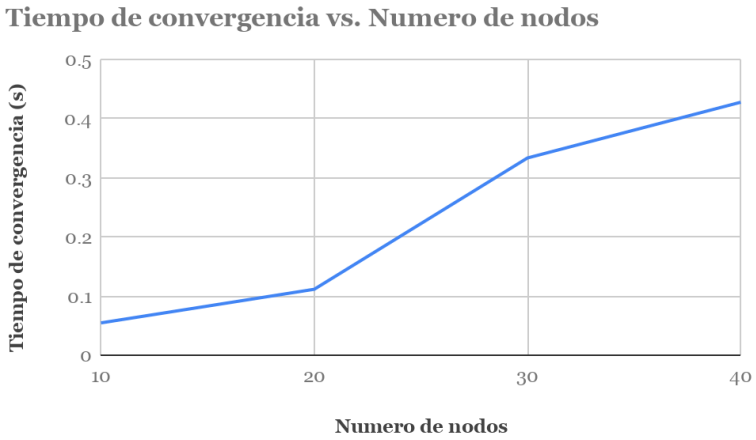


Figura 48. Tiempo de convergencia vs. Número de nodos.

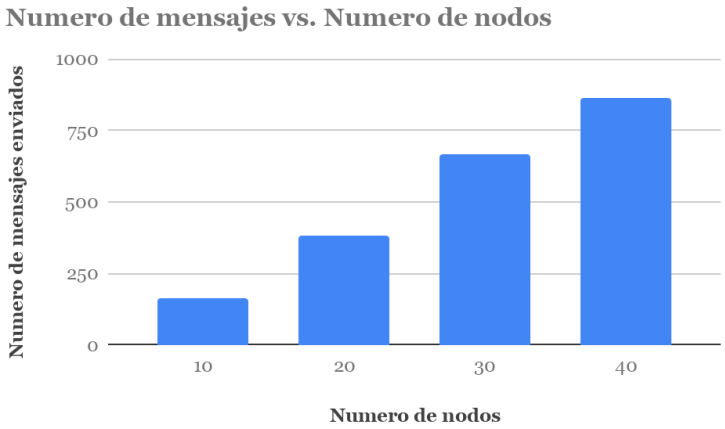


Figura 49. Número de mensajes enviados vs. Número de nodos.

Por otro lado, se observó que el número de nodos presentes en la topología afecta de manera significativa el tiempo de compilación de la implementación realizada. Esta relación se puede observar en la Figura 50.

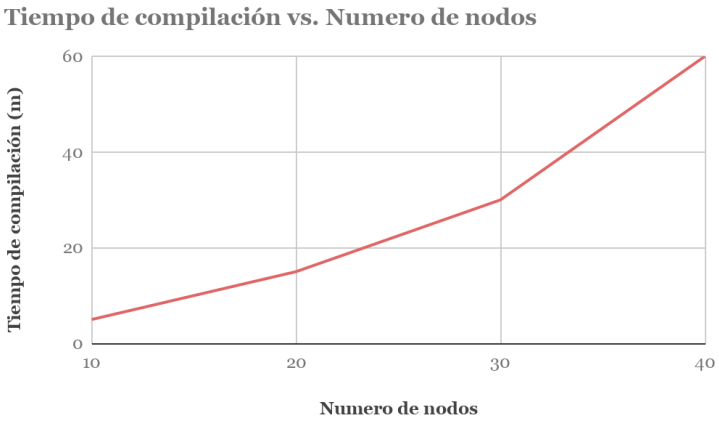


Figura 50. Tiempo de compilación vs. número de nodos.

También se estudió la relación que existía entre la cantidad de nodos presentes en la topología y los recursos de la máquina que utilizaba la implementación mientras se corrían las simulaciones. Se encontró que existe una relación directamente proporcional entre el número de nodos en la topología y el uso de memoria RAM utilizada. A diferencia del uso del CPU el cual fue aproximadamente el mismo en todas las simulaciones. Las relaciones anteriores se muestran en la Figura 51 y en la Figura 52.

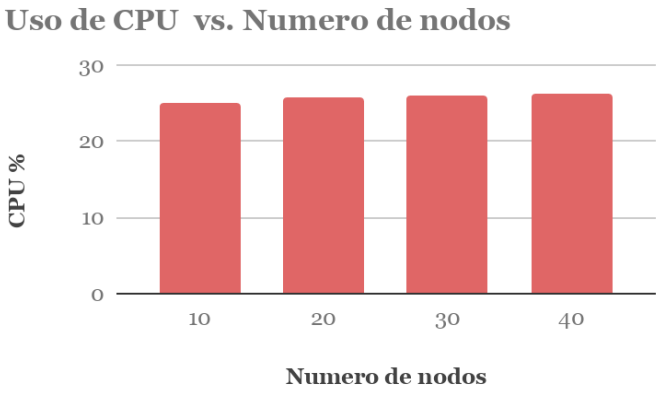


Figura 51. Uso de CPU vs. Número de nodos.

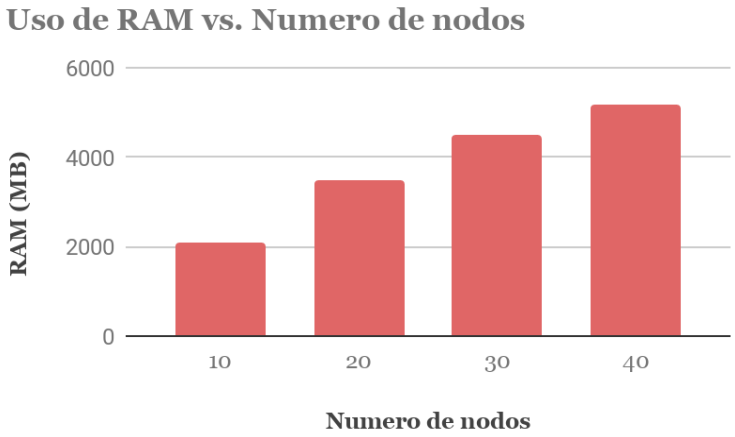


Figura 52. Uso de RAM vs. Número de nodos.

8. Conclusiones

Con base en los artículos y normas estudiadas, las simulaciones realizadas y el análisis de estas, se pudo llegar a las siguientes conclusiones:

- Matlab, Simulink y Simevents poseen las herramientas necesarias para implementar el comportamiento del protocolo RPL y sus características, siendo el primer el modelo de red desarrollado con estas herramientas.
- La métrica de encaminamiento utilizada en una instancia RPL determinará el DODAG que representa la red. Esto es de gran utilidad cuando un usuario final requiere enviar diferentes tipos de tráfico con un objetivo específico.
- El modo Storing genera menor tráfico en la red, pero requiere que los nodos tengan la capacidad de almacenamiento necesaria para mantener sus tablas de enrutamiento, la cual es una de las limitaciones más comunes en los dispositivos utilizados en este tipo de redes, por lo que se acude al modo Non-Storing.
- La probabilidad de pérdida de mensajes presente en los nodos de la topología afecta

directamente el tiempo de convergencia y el flujo de tráfico presente en las redes LLN. En algunos casos donde esta probabilidad sea demasiado alta, puede ser que no se llegue a una convergencia total de la red.

- El número de nodos presente en la topología RPL tiene una relación directa con el tiempo de convergencia. Además, afecta sustancialmente el tiempo que tardan las simulaciones en concluir.
- La implementación realizada es de gran utilidad para entender las características, funcionamiento y el desempeño del protocolo RPL. Gracias a esta, se abre un amplio panorama para posibles estudios futuros en este campo de investigación.
- Teniendo en cuenta los escenarios planteados y los resultados obtenidos, se pudo verificar que los requisitos funcionales y no funcionales definidos en el Capítulo 5 de este libro, se cumplieron satisfactoriamente.

9. Recomendaciones

- Una segunda versión de la implementación podría realizarse a partir de lo desarrollado en este proyecto, tratando de reducir el tiempo de compilación e inicialización de las simulaciones, optimizando las estructuras utilizadas en la librería RPL hecha en Simulink. Además, se podrán añadir características que por tiempo y complejidad fueron omitidas.
- Debido a que en las simulaciones con múltiples instancias el tráfico de mensajes de control es considerable, es necesario hacer un estudio para determinar en qué campos de aplicación es recomendable usar esta funcionalidad.
- Se debe hacer un estudio a profundidad de la capacidad que tiene el ambiente de simulación

de Matlab y Simulink para aprovechar los recursos del sistema, ya que en la implementación realizada nunca se llegó a utilizar al 100% de estos

- Sería conveniente realizar un estudio, comparando los tiempos de compilación de la implementación realizada contra otros entornos de simulación, como Contiki, Omnet++, etc. para verificar y validar el rendimiento de esta implementación.

Referencias bibliográficas

Georgios Parissidis, Merkourios Karaliopoulos, Rainer Baumann, Thrasyvoulos Spyropoulos, Bernhard Plattner, (2016). Routing metrics for Wireless Mesh Networks.

Heile, B., Liu, B., Zhang, M., & Perkins, C. (2017). Wi-SUN FAN Overview. Tools.ietf.org. Recuperado el 25 de Abril del 2018, https://tools.ietf.org/id/draft-heile-lpwan-wisun-overview-00.html

IEEE Standard for Low-Rate Wireless Networks. (2015). IEEE. http://dx.doi.org/10.1109/ieeestd.2016.7460875

JP. Vasseur, Ed., M. Kim, K. Pister, N. Dejean, D. Barthel. (2012). RFC 6551 - Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks. Tools.ietf.org. Recuperado el 26 de Enero del 2019, https://tools.ietf.org/html/rfc6551

JP Vasseur, Navneet Agarwal, Jonathan Hui, Zach Shelby, Paul Bertrand, Cedric Chauvenet. (2011). RPL: The IP routing protocol designed for low power and lossy networks.

Kurunathan, H., Severino, R., Koubaa, A., & Tovar, E. (2018). IEEE 802.15.4e in a Nutshell: Survey and Performance Evaluation. Recuperado el 25 de Abril del 2018, https://ieeexplore.ieee.org/document/8278196/

MathWorks - Fabricantes de MATLAB y Simulink. (2018). La.mathworks.com. Recuperado el 2 de Mayo del 2018, https://la.mathworks.com/products.html

Muneer Bani Yassein, Shadi Aljawarneh, Esra'a Masa'deh, Baraq Ghaleb, Raja'a Masa'deh, (2017). A New Dynamic Trickle Algorithm for Low Power and Lossy Networks.

O.G. Gnawali, P.L. Levis. (2010). Internet-Draft - The ETX Objective Function for RPL. Recuperado el 26 de Enero del 2019, https://tools.ietf.org/id/draft-gnawali-roll-etxof-00.html

P. Levis, Ed., T. Clausen, J. Hui, O. Gnawali, J. Ko. (2011). RFC 6206 - The Trickle Algorithm. Tools.ietf.org. Recuperado el 26 de Enero del 2019, https://tools.ietf.org/html/rfc6206

P. Thubert, Ed. (2012). RFC 6552 - Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). Tools.ietf.org. Recuperado el 26 de Enero del 2019, https://tools.ietf.org/html/rfc6552

Rose, K., Eldridge, S., & Chapin, L. (2015). LA INTERNET DE LAS COSAS—UNA BREVE RESEÑA. Cdn.prod.internetsociety.org. Recuperado el 25 de Abril del 2018, https://cdn.prod.internetsociety.org/wp-content/uploads/2017/09/report-InternetOfThings-20160817-es-1.pdf

Routing protocol definition by The Linux Information Project. (2005). *Linfo.org*. Recuperado el 1 Mayo del 2018, from http://www.linfo.org/routing_protocol.html

Rowe, A. (2012). 6LoWPAN - Nano-RK. Nanork.org. Recuperado el 25 de Abril del 2018 http://www.nanork.org/projects/nanork/wiki/6LoWPAN

Simulink Documentation - MathWorks América Latina. (2019). Recuperado el 26 de Enero de 2018, https://la.mathworks.com/help/simulink/index.html

Simulink Tutorial. Ewh.ieee.org. Recuperado el 25 de Abril del 2018, https://ewh.ieee.org/r1/ct/sps/PDF/MATLAB/chapter8.pdf

Support, P., Software, C., 15.5M&T, C., & Guides, C. (2019). Routing Protocol for LLN (RPL). Recuperado el 26 de Enero de 2019. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/rpl/configuration/15-mt/rpl-15-mt-book.html#concept_56DDF281D4494D4FB807695739F4D142

Vasseur, J., Agarwal, N., Hui, J., Shelby, Z., Bertrand, P., & Chauvenet, C. (2011). *RPL: The IP routing protocol designed for low power and lossy networks*. *Ipso-alliance.org*. Recuperado el 1 de Mayo del 2018, http://www.ipso-alliance.org/wp-content/media/rpl.pdf

Wehrle, K., Mesut, G., & Gross, j. (2010). *Modeling and tools for network simulation*. Berlin: Springer.

What is a Hop Count? - Definición de Techopedia. (2019). Recuperado el 12 de Febrero de 2019, https://www.techopedia.com/definition/26127/hop-count

What is MATLAB? (2018). *La.mathworks.com*. Recuperado el 2 de Mayo del 2018, https://la.mathworks.com/discovery/what-is-matlab.html

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., & Levis, P. et al. (2012). RFC 6550 - RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. Tools.ietf.org. Recuperado el 25 de Abril del 2018, https://tools.ietf.org/html/rfc6550

Xin Yang, Jianlin Guo, Philip Orlik, Kieran Parsons, Koichi Ishibashi, (2014). Stability Metric Based Routing Protocol for Low-Power and Lossy Networks.

Xiyuan Liu, Zhengguo Sheng , Changchuan Yin, Falah Ali, Daniel Roggen. (2017). Performance Analysis of Routing Protocol for Low Power and Lossy Networks (RPL) in Large Scale Networks.