ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

EVALUACIÓN DE LA EFICIENCIA DEL PROTOCOLO OLSR SOBRE REDES MANET USANDO EL SIMULADOR NS3

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

DANILO JAVIER CALVACHI MORENO

DIRECTOR: DR. LUIS FELIPE URQUIZA AGUIAR

Quito, junio 2020

AVAL

Certifico que el presente trabajo fue desarrollado por Danilo Javier Calvachi Moreno, bajo mi supervisión.

DR. LUIS FELIPE URQUIZA AGUIAR
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo Danilo Javier Calvachi Moreno, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

DANILO JAVIER CALVACHI MORENO

DEDICATORIA

A mis padres, por el apoyo, consejos y fuerzas que me dan para seguir adelante con mis metas. A mi hermana por su comprensión y apoyo. Sin ustedes no hubiese terminado el trabajo.

AGRADECIMIENTO

A mis padres, Danilo y Ruth, por su apoyo incondicional y sus palabras de aliento. Este trabajo es suyo, los amo.

A mi hermana, Darleen, que me ha acompañado a lo largo de mis estudios.

A mi primo, Rolando Reinoso, por darme palabras de apoyo y ayudarme en ciertos problemas que se presentaron en el desarrollo de este trabajo.

Al Dr. Luis Felipe Urquiza, por su paciencia y disposición a resolver las dudas e inquietudes que existieron a lo largo del desarrollo de este trabajo de titulación.

A la Escuela Politécnica Nacional y sus docentes que han compartido sus enseñanzas y experiencias.

ÍNDICE DE CONTENIDO

AVAL…… I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO..... V
RESUMEN..... X
ABSTRACT..... XI
1. INTRODUCCIÓN.....1
1.1 OBJETIVOS.....2
1.2 ALCANCE.....3
1.3 MARCO TEÓRICO 4
1.3.1 REDES MÓVILES AD-HOC (MANET) 4
1.3.2 CARACTERÍSTICAS DE LAS MANET.....5
1.3.3 LIMITACIONES DE LAS MANET 6
1.3.4 APLICACIONES DE LAS MANET 7
1.3.5 PROTOCOLOS DE ENRUTAMIENTO.....7
1.3.6 OLSR (OPTIMIZED LINK STATE ROUTING PROTOCOL) 8
2. METODOLOGÍA.....18
2.1. ESTUDIO DE NS-3.....19
2.1.1 INTRODUCCIÓN A NS-3.....19
2.1.2 GENERALIDADES DE NS-3.....20
2.1.3 CARACTERÍSTICAS DE NS-3 20
2.1.4 MODELADO DE RED EN NS-3.....21
2.1.5 ESTRUCTURA DE NS-3.....21
2.2. ESTRUCTURA DEL CÓDIGO DE SIMULACIÓN 23

2.2.1 LIBRERÍAS 24
2.2.2 DEFINICIÓN DE VARIABLES 29
2.2.3 ALEATORIEDAD DE EXPERIMENTOS 31
2.2.4 MOVILIDAD DE RED 32
2.2.5 DISPOSITIVOS DE RED 33
2.2.6 CONFIGURACIÓN DE LOS PARÁMETROS DE OLSR 34
2.2.7 DEFINICIÓN DE NODOS 35
2.2.8 ARCHIVOS DE ANÁLISIS 37
2.2.9 ESTADÍSTICAS 37
2.3. PARÁMETROS DE OLSR 40
2.4. ESCENARIO DE PRUEBAS 41
2.4.1 DESCRIPCIÓN DEL ESCENARIO DE PRUEBAS 41
2.4.2 PRUEBAS REALIZADAS 42
2.5. MÉTRICAS A EVALUARSE 43
2.5.1 THROUGHPUT 43
2.5.2 TASA DE ENTREGA DE PAQUETES 43
2.5.3 RETARDO PROMEDIO EXTREMO A EXTREMO 44
2.6. TEST ESTDÍSTICOS 44
3. RESULTADOS Y DISCUSIÓN 45
3.1 ANÁLISIS DE VALORES DE PARÁMETROS POR DEFECTO 45
3.2 ANÁLISIS DE VARIACIÓN DE LOS PARÁMETROS 47
3.2.1 VARIACIÓN HELLO INTERVAL 47
3.2.2 VARIACIÓN TC INTERVAL 52
3.2.3 VARIACIÓN HNA INTERVAL 56
3.3 ANÁLISIS DE VARIACIÓN COMBINADA DE LOS PARÁMETROS 61
3.3.1 PRIMERA VARIACIÓN COMBINADA 61
3.3.2 SEGUNDA VARIACIÓN COMBINADA 66
4. CONCLUSIONES Y RECOMENDACIONES 71

4.1. CONCLUSIONES 71
4.2. RECOMENDACIONES 73
5. REFERENCIAS BIBLIOGRÁFICAS.....74
ANEXOS..……..78

ÍNDICE DE FIGURAS

Figura 1.1. Funcionalidad del protocolo OLSR. 18
Figura 2.1. Arquitectura de NS-3 [26] [27]. 22
Figura 2.2. Modelo básico de red en NS-3 [28]. 23
Figura 3.1. Métricas de evaluación del protocolo OLSR con los parámetros por defecto.....47
Figura 3.2. Throughput vs Hello_Interval para tres densidades de nodos y tres velocidades de desplazamiento de nodos. 48
Figura 3.3. Tasa de entrega de paquetes vs Hello_Interval para tres densidades de nodos y tres velocidades de desplazamiento de nodos. 50
Figura 3.4. Tasa de entrega de paquetes vs Hello_Interval para tres densidades de nodos y tres velocidades de desplazamiento de nodos. 51
Figura 3.5. Throughput vs TC_In para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos. 53
Figura 3.6. Indica la tasa de entrega de paquetes vs TC_In para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos. 55
Figura 3.7. Indica el retardo promedio vs TC_In para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos. 56
Figura 3.8. Indica el throughput vs HNA para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos. 58
Figura 3.9. Tasa de entrega de paquetes vs HNA para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos. 59

Figura 3.10. Retardo promedio vs HNA para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos. 60

Figura 3.11. Métricas con parámetros combinados y por defecto, para tres densidades de nodos a una velocidad de 1.5 m/s.....63

Figura 3.12. Métricas con parámetros combinados y por defecto, para tres densidades de nodos a una velocidad de 3 m/s. 64

Figura 3.13. Métricas con parámetros combinados y por defecto, para tres densidades de nodos a una velocidad de 5 m/s. 65

Figura 3.14. Métricas con parámetros combinados y por defecto, para tres densidades de nodos a una velocidad de 1.5 m/s.....67

Figura 3.15. Métricas con parámetros combinados y por defecto, para tres densidades de nodos a una velocidad de 3 m/s. 69

Figura 3.16. Métricas con parámetros combinados y por defecto, para tres densidades de nodos a una velocidad de 5 m/s. 70

ÍNDICE DE TABLAS

Tabla 1.1. Principales parámetros de OLSR y valores especificados en el RFC 3626 15

Tabla 2.1. Parámetros utilizados para evaluar la eficiencia del protocolo OLSR ..40

Tabla 2.2. Descripción del escenario de pruebas.....42

ÍNDICE DE CÓDIGO

Código 2.1. Librerías.....24

Código 2.2. Variables globales.....30

Código 2.3. Generación aleatoria de las simulaciones. 31

Código 2.4. Escenario de simulación. 32

Código 2.5. Movilidad de los nodos.....33

Código 2.6. Creación de dispositivos de red. 33

Código 2.7. Configuración de OLSR. 34
Código 2.8. Definición de nodos.....35
Código 2.9. Archivos de análisis. 37
Código 2.10. Generación de estadísticas de los paquetes enviados. 38
Código 2.11. Generación de estadísticas de los paquetes recibidos. 38
Código 2.12. Estadísticas básicas del paquete. 39
Código 2.13. Retardo de propagación. 39

RESUMEN

Las redes móviles ad-hoc o MANET (Mobile Ad-hoc NETwork) son redes dinámicas y descentralizadas formadas por estaciones móviles. Las estaciones en las redes MANET generalmente son computadoras portátiles o teléfonos móviles. Uno de los protocolos de enrutamiento utilizados para este tipo de redes es OLSR (Optimized Link State Routing Protocol) debido a que reduce los paquetes de control ya que, en lugar de declarar todos los enlaces de un nodo con sus nodos vecinos, declara solamente un conjunto de estos llamados MRPs (Multipoint Relay Selectors). En el presente estudio se realiza un análisis detallado de la variación de los valores de los parámetros de OLSR: HELLO_INTERVAL, REFRESH_INTERVAL, TC_INTERVAL, y los tiempos límites de validez de la información NEIGHB_HOLD_TIME (Hello), MID_HOLD_TIME (MID) y TOP_HOLD_TIME (TC). Estos parámetros serán estudiados en función de las métricas: throughput, tasa entrega de paquetes y retardo promedio.

El presente trabajo inicia con una descripción de OLSR para poner en perspectiva el uso de este protocolo. Luego, se abordan tres escenarios de pruebas los cuales permitirán evaluar al protocolo. En los tres escenarios se varían el número de los nodos (50, 10, 150 respectivamente) y la velocidad de desplazamiento de estos (1.5, 3, 5 [m/s]).

Los resultados de las simulaciones realizadas, las cuales constan de 10 simulaciones por cada escenario, indican que los parámetros: Hello_*Interval, Mid_Interval, HNA_Interval* y *Tc_Interval* influyen de forma relevante en el desempeño del protocolo OLSR.

PALABRAS CLAVE: MANET, NS-3, protocolo, OLSR, eficiencia, enrutamiento, métricas.

ABSTRACT

Mobile ad-hoc networks or MANET (Mobile Ad-hoc Networks) are dynamic and decentralized networks formed by mobile stations. Stations in MANET are generally laptops or mobile phones. One of the most widely used routing protocols in MANETs is the Optimized Link State Routing Protocol (OLSR). OLSR reduces control packets because a node only declares a subset of the links to its neighboring nodes, called MRPs (Multipoint Relay Selectors), instead of all of them.

This study, a detailed analysis of the variation of the OLSR parameter values is performed: HELLO_INTERVAL, REFRESH_INTERVAL, TC_INTERVAL, and the information validity time limits NEIGHB_HOLD_TIME (Hello), MID_HOLD_TIME (MID) and TOP_HOLD_TIME (TC). These parameters will be studied based on the metrics: throughput, packet delivery rate, and average delay.
The present work begins with a description of OLSR to put into perspective the use of this protocol. Then, the three scenarios that will allow evaluating the protocol are presented. In the three scenarios, the number of the nodes (50, 10, 150 respectively) and the speed of their movement (1.5, 3, 5 [m / s]) are varied.

The results of the simulations carried out, which consist of 10 simulations for each scenario, indicate that the Hello_Interval, Mid_Interval, HNA_Interval and Tc_Interval parameters have a significant influence on the performance of the OLSR protocol.

KEYWORDS: MANET, NS-3, protocol, OLSR, efficiency, routing, metrics

1. INTRODUCCIÓN

Las redes inalámbricas son redes configuradas mediante el uso de la frecuencia de la señal de radio para comunicarse entre computadoras y otros dispositivos de red.

Una MANET es un tipo de red ad-hoc que puede realizar cambios de ubicación y permite configurarse sobre la marcha. Debido a que las MANET son móviles, utilizan conexiones inalámbricas para establecer una conexión a varias redes. Puede ser una conexión WiFi u otro medio, como una transmisión celular o satelital.

Los nodos MANET pueden moverse aleatoriamente a medida que la topología de la red cambia frecuentemente. Cada nodo adquiere el comportamiento de un enrutador cuando reenvía el tráfico a otro nodo específico en la red. Forman una topología autónoma, altamente dinámica con la presencia de uno o múltiples transceptores diferentes entre nodos. El principal desafío para MANET es equipar cada dispositivo para mantener continuamente la información requerida para enrutar el tráfico de manera adecuada [1].

La falta de una estructura fija en una red MANET expone diversas metas. Una de las más relevantes es el enrutamiento, el proceso de encontrar una ruta a lo largo de una red para transmitir datos hacia un destino.

Un protocolo de enrutamiento ad-hoc controla como los nodos eligen el camino óptimo para enviar paquetes entre los dispositivos que se encuentran en la red ad-hoc móvil. La idea es que un nuevo nodo pueda anunciar su presencia y escuchar los anuncios emitidos por los nodos vecinos, cada nodo obtiene información de los nodos cercanos y la forma de enviar datos hacia ellos. El objetivo principal de los protocolos de enrutamiento es establecer una ruta correcta y eficiente entre un par de nodos.

Entre los protocolos más usados en las redes MANET se encuentra el protocolo OLSR (Optimized Link State Routing Protocol) [1], el cual es una optimización del algoritmo clásico de estado de enlace adaptado a los requisitos de una LAN (Local Area Network) inalámbrica móvil. El concepto clave utilizado en el protocolo es el de los repetidores multipunto (MPR). Los MPR son nodos seleccionados que reenvían mensajes de difusión durante la etapa de inundación. Esta técnica genera una *primera optimización* reduciendo sustancialmente la sobrecarga del mensaje en comparación con un mecanismo de inundación clásico, donde cada nodo retransmite cada mensaje cuando recibe el mensaje. En OLSR, la información del estado del enlace se genera solo por los nodos elegidos como MPR. Por lo tanto, se logra una *segunda optimización* minimizando el número de mensajes de control inundados en la red. Como *tercera optimización*, un nodo

MPR puede elegir informar solo enlaces entre sí mismo y sus selectores MPR. Por lo tanto, a diferencia del algoritmo clásico de estado del enlace, la información parcial del estado del enlace se distribuye en la red. Esta información luego se utiliza para calcular la ruta. OLSR proporciona rutas óptimas (en términos de número de saltos). El protocolo es adecuado para redes con un gran número de nodos, ya que la técnica de MPR funciona bien bajo este contexto [2].

Lo mecanismos del protocolo OLSR están definidos por un conjunto de parámetros definidos en el RFC del protocolo (RFC 3626) [3] enfocada para MANET. Los parámetros que más influyen son: los intervalos de tiempo de espera para el reenvío de los mensajes Hello, MID (Multiple Interface Declaration) y TC (Topology Control) (HELLO_INTERVAL, REFRESH_INTERVAL y TC_INTERVAL, respectivamente), los tiempos límites de validez de la información recibida mediante los tres tipos de mensajes los cuales son: NEIGHB_HOLD_TIME (Hello), MID_HOLD_TIME (MID) y TOP_HOLD_TIME (TC); el WILLINGNESS o predisposición de los nodos a cómo actuar como MRP [3].

El estudio se realiza con el propósito de encontrar un desempeño óptimo del protocolo, es decir, encontrar la modificación de sus parámetros más relevantes a través de diferentes escenarios de simulación que se planteen. Este tipo de análisis no se ha realizado anteriormente ya que por lo general suele realizarse un estudio utilizando los valores por defecto, por lo cual pueden existir valores de los parámetros más convenientes que permitan mejorar la eficiencia del protocolo.

1.1 OBJETIVOS

El objetivo general de este Estudio Técnico es:

- Evaluación de la eficiencia del protocolo OLSR sobre redes MANET usando el simulador NS3.

Los objetivos específicos de este Estudio Técnico son:

- Estudiar el protocolo OLSR, incluyendo su funcionamiento, tipo de mensajes e información de mayor importancia para este estudio.
- Analizar de forma detallada los parámetros que afectan la eficiencia del protocolo OLSR en las redes MANET.

- Identificar las mejores variaciones y combinaciones de los parámetros del protocolo OLSR mediante simulación del protocolo OLSR en diferentes escenarios de redes MANET.
- Analizar los resultados obtenidos mediante métricas como: Throughput, tasa de entrega de paquetes y retardo promedio.

1.2 ALCANCE

Este trabajo evaluará la eficiencia del protocolo OLSR, la modificación de sus parámetros más relevantes y sus posibles variaciones en diferentes escenarios para brindar una guía de mejora del mismo. Entre los parámetros que por su funcionalidad son de particular importancia, que se evaluarán al menos son los siguientes: los intervalos de tiempo de espera para el reenvío de los mensajes Hello, MID (Multiple Interface Declaration) y TC (Topology Control) (HELLO_INTERVAL, REFRESH_INTERVAL y TC_INTERVAL, respectivamente), los tiempos límites de validez de la información recibida mediante los tres tipos de mensajes los cuales son: NEIGHB_HOLD_TIME (Hello), MID_HOLD_TIME (MID) y TOP_HOLD_TIME (TC) [8]. Así también, se realizará un estudio que combine los parámetros anteriormente escritos con otros que puedan resultar de interés en el desarrollo de este proyecto.

Las redes ad-hoc presentan limitaciones al ser evaluadas de forma experimental debido a la demanda de gran cantidad de recursos como la cantidad de nodos, tiempo y coste asociado; por lo tanto, los investigadores usan un simulador de red para su estudio.

Para la evaluación de OLSR y modificación de sus parámetros que tengan mayor incidencia en su eficiencia, se usará el software de simulación de redes NS-3. Esta herramienta se utilizará por las ventajas que presenta frente a otros simuladores. Entre las características que destaca NS-3 se incluye la escalabilidad y el uso eficiente de memoria, lo cual permite optimizar el desempeño de las simulaciones el eficiente de memoria [9]. Se variarán los valores de los parámetros del protocolo para encontrar su mayor eficiencia sobre escenarios de redes MANET con diferentes características, por ejemplo, densidad de nodos y velocidad de los mismos. Con el uso de este software se asegura que el manejo y control de datos sea eficiente según sea necesario.

Finalmente, el estudio técnico se centra únicamente en el protocolo OLSR, por lo que no se realizarán comparaciones con otros protocolos de enrutamiento, ya que ese no es el objetivo de este estudio.

1.3 MARCO TEÓRICO

En el presente capítulo se abordarán conceptos para la total comprensión sobre las redes móviles Ad-hoc (MANET), protocolos de enrutamiento y, un claro énfasis en el protocolo de enrutamiento OLSR.

Dentro del análisis de las MANET se describirán principalmente los conceptos básicos, características, aplicaciones y sus principales desafíos que estas presentan; lo cual, permite tener un mejor conocimiento acerca del uso correcto, funcionamiento, ventajas y desventajas que se tendrían en su aplicación.

Después, se ejecuta una rápida revisión de los protocolos de enrutamiento existentes para estas redes, como introducción para el análisis siguiente del protocolo OLSR. Dicho análisis se hará de forma amplia por la importancia que toma el protocolo OLSR en el presente estudio técnico.

1.3.1 REDES MÓVILES AD-HOC (MANET)

Una red móvil ad-hoc (MANET) es un sistema interconectado de nodos inalámbricos que se comunican a través de enlaces inalámbricos con restricciones de ancho de banda. Cada nodo inalámbrico puede funcionar como remitente, receptor o enrutador. Cuando el nodo es un remitente, puede enviar mensajes a cualquier nodo de destino especificado a través de alguna ruta. Como receptor, puede recibir mensajes de otros nodos.

Cuando el nodo funciona como un enrutador, puede retransmitir el paquete al destino o al siguiente enrutador en la ruta. Cuando sea necesario, cada nodo puede almacenar los paquetes en un buffer en espera para la transmisión [4].

Los nodos se mueven al azar; por lo tanto, en un determinado punto en el tiempo, existe una red ah-hoc entre los nodos, dando lugar a una topología de red arbitraria. Las redes MANET se pueden formar dinámicamente de entre un grupo de usuarios inalámbricos y no requieren de una configuración o infraestructura existente.

Estas redes pueden adaptarse y reconfigurarse sobre la marcha de acuerdo con la movilidad de los nodos y las topologías de red cambiantes. Estas características son particularmente atractivas para el uso militar debido a la imprevisibilidad inherente del entorno táctico [5].

1.3.2 CARACTERÍSTICAS DE LAS MANET

Las características principales de las redes MANET son: la falta total de control centralizado, la falta de asociación entre nodos, la movilidad rápida de los hosts, la topología de red que varía dinámicamente, el canal de radio de transmisión compartida, el entorno operativo inseguro, la vulnerabilidad física y la disponibilidad limitada de recursos, tales como: capacidad de procesamiento de la CPU (Central Processing Unit), energía de la memoria, energía de la batería y ancho de banda [4].

Topologías de red dinámicas: los nodos se mueven de forma libre e independientemente en cualquier dirección. La topología inalámbrica de la red puede cambiar con frecuencia y en momentos impredecibles y consiste principalmente en enlaces bidireccionales [5].

Ancho de banda bajo: estas redes tienen menor capacidad y un rango de transmisión más corto que las redes de infraestructura fija. El rendimiento de la comunicación inalámbrica es menor que la comunicación por cable debido al efecto de las condiciones de acceso múltiple, desvanecimiento, ruido e interferencia [6].

Energía de batería limitada: los nodos o hosts funcionan con baterías pequeñas y otros medios de energía agotables. Por lo tanto, la conservación de energía es el criterio de optimización de diseño más importante.

Control descentralizado: debido a enlaces poco confiables, el funcionamiento de las MANET depende de la cooperación de los nodos participantes. Por lo tanto, la implementación de cualquier protocolo que implique una autoridad centralizada o un administrador se vuelve difícil [4].

Comunicaciones poco confiables: la naturaleza de medio compartido y la calidad de canal inestable de los enlaces inalámbricos pueden dar como resultado una alta tasa de pérdida de paquetes y una inestabilidad de enrutamiento, que es un fenómeno común que conduce a caídas de rendimiento en redes de múltiples saltos [7].

Escalabilidad: debido a la capacidad limitada de memoria y procesamiento en los dispositivos móviles, la escalabilidad es un problema clave cuando consideramos un gran tamaño de red [6].

1.3.3 LIMITACIONES DE LAS MANET

Existe una necesidad actual y futura de tecnología dinámica de redes ad-hoc [3]. Sin embargo, esta tecnología de red altamente adaptativa ya que, enfrenta varias limitaciones.

- **El rendimiento disminuye con más saltos**

  Los nodos que están en el rango de transmisión entre sí pueden enviar los paquetes de datos directamente. Sin embargo, cuando el nodo necesita enviar datos a un nodo no vecino, los paquetes de datos deberán enviarse a través de una secuencia de múltiples saltos, con los nodos intermedios actuando como enrutadores. Esto indica un aumento en el número de saltos. El rendimiento disminuirá rápidamente cuando aumente el número de saltos [8].

- **El rendimiento disminuye con el aumento de la movilidad**

  Los nodos altamente móviles generarán más gastos generales debido a los cambios frecuentes de topología. Esto se debe al aumento en el número de transmisiones de paquetes de enrutamiento, debido a la necesidad de determinar nuevas rutas después de las rutas fallidas. Cuando se utiliza la tabla de enrutamiento, cada nodo mantiene una lista de destinos disponibles, así como la cantidad de saltos requerido para llegar a cada destino. Los cambios en la topología se reflejarán en la tabla de enrutamiento [9]. Cualquier cambio en la tabla de enrutamiento se transmite a todos los demás nodos. Esto impone una sobrecarga mayor en la red general. Cuando la sobrecarga es grande, un porcentaje menor del paquete va hacia la transmisión de datos, lo que resulta en un menor rendimiento.

- **Retardo**

  La demora es el tiempo promedio que un paquete emplea desde el momento en que sale una fuente para el momento en que llega a su destino. Como se vió anteriormente, es necesario mantener los nodos ocupados con la transmisión y recepción de paquetes para aumentar el rendimiento de la red. Esto, a su vez, significará que la cola de cada nodo no siempre está vacía, lo que lleva a un retraso más largo.

1.3.4 APLICACIONES DE LAS MANET

Hay muchas aplicaciones de MANET, su dominio es diverso, desde redes pequeñas y estáticas que están limitadas por fuentes de energía hasta redes móviles, dinámicas y de gran escala [10]. Ejemplos significativos incluyen el establecimiento de comunicación dinámica y eficiente para: entornos militares / de campo de batalla centrados en la red, operaciones de emergencia / rescate, operaciones de socorro en casos de desastre, sistemas de transporte inteligentes, conferencias, redes de sensores móviles tolerantes a fallas, hogares inteligentes, monitoreo de pacientes, control ambiental y otras aplicaciones sensibles a la seguridad [4]. La mayoría de estas aplicaciones exigen garantías de seguridad específicas y una comunicación confiable. Algunas aplicaciones bien conocidas son:

- **Operaciones tácticas militares:** para el establecimiento rápido y posiblemente a corto plazo de comunicaciones militares y despliegues de tropas en entornos hostiles y / o desconocidos.
- **Operaciones de búsqueda y rescate:** para la comunicación en áreas con poco o ningún soporte de infraestructura inalámbrica.
- **Operaciones de socorro en casos de desastre:** para la comunicación en entornos donde la infraestructura existente se destruye o se deja inoperable.
- **Aplicación de la ley:** para una comunicación segura y rápida durante las operaciones de aplicación de la ley.
- **Uso comercial:** para permitir la comunicación en exposiciones, conferencias y grandes reuniones. Para algunos escenarios de negocios. Después de todo, a menudo ocurre que las personas necesitan tener reuniones externas para cooperar e intercambiar información sobre un proyecto determinado [6].

1.3.5 PROTOCOLOS DE ENRUTAMIENTO

Los nodos MANET realizan las funciones de enrutamiento por sí mismos. Debido al rango de transmisión inalámbrica limitado, el enrutamiento generalmente consiste en múltiples saltos. Por lo tanto, los nodos dependen unos de otros para reenviar paquetes a los destinos. La naturaleza de las redes coloca dos requisitos fundamentales en los protocolos de enrutamiento [1]. Primero, tienen que ser distribuidos. Y, en segundo lugar, dado que los cambios de topología son frecuentes, deben calcular múltiples rutas sin

bucles mientras mantiene una comunicación con un mínimo overhead. Según el tiempo en que tarde en descubrir una de ruta, los protocolos de enrutamiento para MANET se clasifican en tres categorías generales [1, 11]:

- Protocolos de enrutamiento Proactivos
- Protocolos de enrutamiento Reactivos
- Protocolos de enrutamiento Híbridos

Los protocolos proactivos también se conocen como protocolos de enrutamiento impulsados por tablas. Cada nodo móvil tiene una tabla de enrutamiento separada que contiene la información de las rutas a todos los posibles nodos móviles de destino. Dado que la topología en la red ad-hoc móvil es dinámica, estas tablas de enrutamiento se actualizan periódicamente a medida que cambia la topología de la red. Tiene una limitación, no funciona bien para las redes grandes ya que las entradas en la tabla de enrutamiento se vuelven demasiado grandes y necesitan contener la información de ruta a todos los nodos posibles [12].

Los protocolos reactivos son conocidos también como protocolos de enrutamiento bajo demanda. En este tipo de enrutamiento, la ruta se descubre solo cuando se requiere / necesita. El descubrimiento de la ruta ocurre inundando los paquetes de solicitud de ruta a través de la red móvil. Consiste en dos fases principales, a saber, descubrimiento de ruta y mantenimiento de ruta [11].

Los protocolos híbridos básicamente combinan las ventajas de los protocolos de enrutamiento reactivo y proactivo. Los protocolos híbridos son de naturaleza adaptativa y se adaptan según la zona y la posición de los nodos móviles de origen y destino.

Dentro de los protocolos proactivos se encuentra el protocolo OLSR (Optimized Link State Routing Protocol). A continuación, se revisa de forma detallada este protocolo al ser el centro de análisis de este estudio técnico.

1.3.6 OLSR (OPTIMIZED LINK STATE ROUTING PROTOCOL)

Optimized Link State Routing Protocol, OLSR, es un protocolo proactivo que se basa en la optimización de los protocolos clásicos Link-State, basados en estado de enlace.

El protocolo OLSR funciona de forma correcta en redes con alta demanda de usuarios (nodos) y con una topología cambiante. Para llevar un control, se intercambian

periódicamente mensajes de tal forma que aprende la topología de la red y el estado de los nodos vecinos. Como se sabe, el intercambio de tantos paquetes puede congestionar la red, lo que supone un grave problema en las comunicaciones [13]. Para solucionar este problema, OLSR utiliza la técnica de MRP (Multipoint Relay). Gracias a esta técnica se reduce el número de retransmisiones.

1.3.6.1 Descripción general del protocolo

OLSR hereda el uso del algoritmo de estado de enlace, usando el envío conocido como *shortest path first.* El intercambio de información topológica con otros nodos de red es periódico, y cada nodo mantiene la topología de toda la red. Las rutas son calculadas basadas en la información de topología que cada nodo contiene [3]. En un protocolo proactivo, la información de enrutamiento está disponible inmediatamente cuando esta se necesita.

OLSR disminuye la sobrecarga de la inundación del tráfico mediante el uso de solo los nodos seleccionados, llamados MPR, para retransmitir mensajes de control. Esta técnica reduce de manera significativa el número de retransmisiones requerido para inundar un mensaje a todos los nodos en la red. En segundo lugar, OLSR requiere que solo se inunde el estado de enlace parcialmente para proporcionar rutas más cortas. En OLSR, solo los nodos MRP son responsables del envío del control de tráfico, destinado para difusión dentro de toda la red [13]. Los MRPs proveen un mecanismo eficiente para la inundación de control de tráfico reduciendo el número de transmisiones requeridas [3].

OLSR puede optimizar la reactividad de los cambios topológicos, reduciendo el intervalo de tiempo máximo para la transmisión de mensajes de control de forma periódica. Además, como OLSR mantiene continuamente rutas a todos destinos en la red, el protocolo es beneficioso para patrones de tráfico donde un gran subconjunto de nodos se comunica con otro gran subconjunto de nodos, y donde los pares {fuente, destino} están cambiando con el tiempo [3]. El protocolo es adecuado para redes grandes y densas, debido a que la optimización realizada con MPR funciona bien en este contexto. Cuanto más grande y más densa es una red, mayor optimización se puede lograr en comparación con el clásico algoritmo estado de enlace.

OLSR está diseñado para funcionar de forma completamente distribuida y no depende de ningún componente central. El protocolo no requiere transmisión confiable de mensajes de control: cada nodo envía periódicamente mensajes de control, y por lo tanto puede sufrir pérdidas razonables de algunos de esos mensajes [3]. Dichas pérdidas ocurren con frecuencia en las redes radiales, debido a colisiones u otros problemas de transmisión.

Además, OLSR no requiere entrega secuenciada de mensajes. Cada mensaje de control contiene un número de secuencia que se incrementa para cada mensaje. Así, el destinatario de un mensaje de control puede, si requiere, identificar fácilmente qué información es más reciente, incluso si los mensajes han sido reordenados durante la transmisión.

1.3.6.2 Terminología de OLSR

En esta sección se define la terminología usada en OLSR que aún no ha sido detallada [3]:

Nodo: Un router MANET que utiliza el protocolo OLSR.

- **Interfaz OLSR:** Un dispositivo de red que participa en un MANET que ejecuta OLSR. Un nodo puede tener varias interfaces OLSR, a cada interfaz se le asigna una dirección IP (Internet Protocol) única.

- **Interfaz no OLSR:** Un dispositivo de red que no participa en una MANET que ejecuta OLSR. Un nodo puede tener varias interfaces no OLSR (inalámbricas y / o cableadas). La información de enrutamiento de estas interfaces puede ser agregada en el dominio de enrutamiento OLSR.

- **Nodo de interfaz OLSR único:** Un nodo que tiene una única interfaz OLSR, que participa en un dominio de enrutamiento OLSR.

- **Nodo de interfaz OLSR múltiple:** Un nodo que tiene múltiples interfaces OLSR, que participa en un dominio de enrutamiento OLSR.

- **Dirección principal:** Dirección principal de un nodo, que se utilizará en el control de tráfico como la "dirección de origen" de todos los mensajes emitidos por este nodo. Es la dirección de una de las interfaces OLSR del nodo. Un nodo de interfaz OLSR único debe utilizar la dirección de su único interfaz OLSR como la dirección principal. Un nodo de interfaz OLSR múltiple debe elegir una de sus direcciones de interfaz OLSR como su “dirección principal”. No tiene importancia qué dirección

se elige, sin embargo, un nodo debe usar siempre la misma dirección como su dirección principal.

- **Nodo vecino:** Un nodo X es un nodo vecino del nodo Y si el nodo Y puede escuchar el nodo X (es decir, existe un enlace entre una interfaz OLSR en el nodo X y un interfaz OLSR en Y).

- **Vecino de 2 saltos:** Un nodo escuchado por un vecino.

- **Vecino estricto de 2 saltos:** Un vecino de 2 saltos que no es el nodo en sí o un vecino del nodo, y además es vecino de un vecino, con disposición diferente del campo WILL_NEVER.

- **Repetidor multipunto (MPR):** Un nodo que es seleccionado por su vecino de 1 salto, para "retransmitir" todos los mensajes de difusión que recibe de X, siempre que el mensaje no sea un duplicado y que el campo TTL del mensaje es mayor que uno.

- **Selector de repetidor multipunto (selector MPR, MS):** Un nodo que ha seleccionado su vecino de 1 salto, nodo X, como su relé multipunto, se llamará un selector de relé multipunto de nodo X.

- **Enlace:** Un enlace es un par de interfaces OLSR (de dos nodos diferentes) susceptibles de escucharse unos a otros. Se dice que un nodo tiene un enlace a otro nodo cuando una de sus interfaces tiene un enlace a una de las interfaces del otro nodo.

- **Enlace simétrico:** Un enlace bidireccional verificado entre dos interfaces OLSR.

- **Enlace asimétrico:** Un enlace entre dos interfaces OLSR, verificado en una sola dirección.

1.3.6.3 MRP (Multi Point Relay)

La idea de los MRPs es disminuir la sobrecarga de los mensajes de inundación en la red disminuyendo la retransmisión redundante en la misma región [11]. Cada nodo en la red selecciona un conjunto de nodos en su barrido simétrico (1-hop) en el cual puede retransmitir sus mensajes [11]. Este conjunto de nodos seleccionados es llamado “Multipoint Relay” (MRP). Los vecinos del nodo N que no están en su conjunto MRP, reciben y procesan mensajes de broadcast, pero no retransmite los mensajes de broadcast recibidos desde el nodo N [3].

Cada nodo selecciona su propio conjunto MPR de entre su barrido simétrico de 1 salto. Este conjunto se selecciona de manera que cubra todos los nodos simétricos de 2 saltos. El conjunto MPR de N, llamado MPR (N), cumple la siguiente condición: cada nodo en el vecindario simétrico estricto de 2 saltos de N debe tener un enlace simétrico hacia MPR (N). Cuanto más pequeño es un conjunto MPR, habrá menos resultados de sobrecarga del tráfico de control del protocolo de enrutamiento [3].

Cada nodo mantiene información acerca del conjunto de vecinos que han sido seleccionados como MPR. Dicho conjunto es llamado "Multipoint Relay Selector set" (conjunto selector MPR) de un nodo. Un nodo adquiere esta información de mensajes HELLO periódicos recibidos de los vecinos.

Un mensaje de broadcast, destinado a ser difundido en toda la red, proveniente de cualquiera de los selectores MPR del nodo N supone que será retransmitido por el nodo N, si N aún no lo ha recibido. Este conjunto puede cambiar con el tiempo (es decir, cuando un nodo selecciona otro conjunto MPR) y es indicado por los nodos selectores en sus mensajes de HELLO.

1.3.6.4 Parámetros de OLSR

OLSR intercambia periódicamente diferentes mensajes para mantener la información de topología de toda la red ante la presencia de movilidad y fallas [14]. OLSR RFC [3] define un conjunto de constantes que regulan el uso de mecanismos OLSR. La especificación establece que la configuración de parámetros se realiza por nodo y propone un conjunto de valores predeterminados. A continuación, se presenta un resumen de parámetros más

relevantes que pueden tener alguna influencia en el rendimiento de la red y sus valores según lo propuesto por defecto en la especificación OLSR [15] [3].

- Los tiempos de espera antes de reenviar cada tipo de mensaje, *HELLO_INTERVAL, REFRESH_INTERVAL, TC_INTERVAL,* respectivamente.
- El "tiempo de validez" de la información recibida para cada tipo de mensaje, *NEIGHB_HOLD_TIME, MID_HOLD_TIME* y *TOP_HOLD_TIME.*
- La voluntad de un nodo para actuar como MRP.
- El tiempo durante el cual los MRPs registran información sobre los paquetes reenviados, *DUP_HOLD_TIME.*

Hello Interval

Este atributo especifica el intervalo de tiempo entre dos paquetes de saludo. Los paquetes de saludo son necesarios para mantener adyacencias entre nodos. Los paquetes de saludo contienen información del vecino de 1 salto y del vecino de 2 saltos. El objetivo principal de esta optimización es reducir la cantidad de tráfico de enrutamiento y también minimizar un período de tiempo medido, como un tiempo entre abandonar el rango de transmisión del nodo vecino actual y estar registrado en la lista de vecinos de otro nodo móvil. El valor predeterminado definido en [3] es de 2 segundos.

Topology Control Interval

El intervalo de Control de topología (TC) especifica el período de tiempo entre los mensajes de Control de topología. Los mensajes TC son generados por nodos MPR para transportar información de topología / conectividad. Estos mensajes llenan la tabla de topología de cada nodo. Esta información se utiliza para los cálculos de la tabla de enrutamiento [3]. El valor predeterminado definido en [3] es de 5 segundos.

Neighbor Hold Time Interval

Este parámetro especifica el tiempo de caducidad del enlace. Esto normalmente es 3 veces el valor del intervalo de saludo [3]. Con cada mensaje de saludo que llega, el temporizador de vencimiento del enlace se reinicia. Si no se recibe un mensaje de saludo

durante este tiempo, se considera que el enlace está perdido. Si todos los enlaces a un nodo vecino se han perdido, el vecino se declara inalcanzable. La pérdida de paquetes o el paquete recibido con un error son los problemas más frecuentes en las redes inalámbricas. La pérdida de paquetes es, causada por el nodo móvil que abandona el rango de transmisión del segundo nodo. Además, los algoritmos comunes de corrección de errores de reenvío (FEC) implementados en tecnologías de redes móviles pueden corregir al menos un bit en el mensaje.

Topology Hold Time Interval

Este atributo especifica el tiempo de vencimiento de las entradas en la tabla de topología. Este atributo generalmente se establece en 3 veces el valor del intervalo TC predeterminado [3]. Las entradas de la tabla de topología se actualizan mediante mensajes TC en función de su dirección de origen y número de secuencia.

Willingness

La disposición de un nodo puede establecerse en cualquier valor entero de 0 a 7, y especifica qué tan dispuesto está un nodo a reenviar tráfico en nombre de otros nodos. Los nodos, por defecto, tendrán una disposición *WILL_DEFAULT. WILL_NEVER* indica un nodo que no desea transportar tráfico para otros nodos, por ejemplo, debido a limitaciones de recursos (como tener poca batería). *WILL_ALWAYS* indica que siempre se debe seleccionar un nodo para transportar tráfico en nombre de otro nodo, por ejemplo, debido a la abundancia de recursos (como fuente de alimentación permanente, interfaces de alta capacidad a otros nodos).

Un nodo puede cambiar dinámicamente su disposición a medida que cambian sus condiciones. Una posible aplicación sería, por ejemplo, para un nodo, conectado a una fuente de alimentación permanente y con baterías completamente cargadas, para anunciar una voluntad de *WILL_ALWAYS*. Al desconectarse de la fuente de alimentación permanente, se anuncia una voluntad de *WILL_DEFAULT*. A medida que se agota la capacidad de la batería, la disposición se reduciría aún más. Primero al valor intermedio entre *WILL_DEFAULT y WILL_LOW*, luego a *WILL_LOW* y finalmente a *WILL_NEVER*, cuando la capacidad de la batería del nodo ya no soporta el transporte de tráfico extranjero.

Por lo mencionado anteriormente todos los nodos que se emplean en la simulación tienen un valor de Willingness = 3.

WILL_NEVER = 0

WILL_LOW = 1

WILL_DEFAULT = 3

WILL_HIGH = 6

WILL_ALWAYS = 7

Otros parámetros que pueden influir en el rendimiento de OLSR son los siguientes: MPR_COVERAGE, que permite definir la cantidad de MRPs que debe cubrir un nodo; TC_REDUNDANCY, que ajusta el conjunto de enlaces anunciados en un mensaje TC, y enlaza los parámetros de histéresis que afectan el establecimiento del enlace y los procedimientos de falla [15]. Estos parámetros no son considerados para este estudio técnico, pero quedan remarcados para futuros estudios.

A continuación, la tabla 1.1 representa los valores estandarizados de los parámetros del protocolo OLSR los cuales están especificados en su RFC 3626 [14].

Tabla 1.1. Principales parámetros de OLSR y valores especificados en el RFC 3626

Parámetro	Valor estándar	Rango
HELLO_INTERVAL	2.0 s	R ε [1.0, 30.0]
REFRESH_INTERVAL	2.0 s	R ε [1.0, 30.0]
TC_INTERVAL	5.0 s	R ε [1.0, 30.0]
WILLINGNESS	3	Z ε [0,7]
NEIGHB_HOLD_TIME	3 × HELLO INTERVAL	R ε [3.0, 100.0]
TOP_HOLD_TIME	3 × TC_INTERVAL	R ε [3.0, 100.0]
MID_HOLD_TIME	3 × TC_INTERVAL	R ε [3.0, 100.0]
DUP_HOLD_TIME	30.0 s	R ε [3.0, 100.0]

En base a los parámetros de OLSR, se lleva a cabo un estudio detallado de la variación de estos de manera aislada y, a su vez, se analiza las posibles combinaciones entre estos, acción que deriva una gran cantidad de simulaciones que pueden incrementar exponencialmente debido a la cantidad de experimentos que pueda resultar de dicho análisis.

El estudio descrito anteriormente se efectúa con el objetivo de encontrar un desempeño óptimo del protocolo. Este tipo de análisis no se lo ha hecho anteriormente ya que generalmente se usan los valores por defecto, por lo que puede haber valores de los parámetros más convenientes para un escenario o para otro que permitan un mejor desempeño de protocolo

1.3.6.5 Mensajes OLSR

La funcionalidad principal del protocolo OLSR se la realiza mediante el uso de diferentes de mensajes:

- HELLO
- Topology Control (TC)
- Multiple Interface declaration (MID)
- Host and Network Association (HNA)

A continuación, se da una breve explicación de cada uno de los mensajes mencionados anteriormente.

Hello

Los mensajes HELLO se transfieren entre otros nodos en la red (1 salto de distancia). Estos son usados para proveer la detección de vecinos y señalización de selección MRP [14] .

Topology Control (TC)

Los mensajes Topology Control son generados periódicamente por los MRPs para indicar por cuál de los otros nodos ha sido seleccionado como su MRP. Esta información es almacenada en la información de topología de cada nodo, el cual es usado para calcular la tabla de enrutamiento [14].

Luego estos mensajes son enviados a otros nodos en la red. Los mensajes TC son diferenciados por asignar números de secuencia.

Multiple Interface Declaration (MID)

Los mensajes MID contienen información de las interfaces de los nodos de red los cuales son empleados para participar en la comunicación de la red. Dicha información es necesaria ya que los nodos pueden tener múltiples interfaces con distintas direcciones que participan en las comunicaciones [14].

Host and Network Association (HNA)

"Host and Network Association" (HNA) proporciona la información de enrutamiento externa, dando la posibilidad de enrutamiento a las direcciones externas. El mensaje HNA proporciona información sobre la red y las direcciones de máscara de red, de modo que el host OLSR puede considerar que el host anunciante puede actuar como una puerta de entrada al conjunto de direcciones anunciantes. El HNA se considera como una versión generalizada del mensaje TC con la única diferencia de que el mensaje TC puede informar sobre la cancelación de ruta, mientras que la información del mensaje HNA se elimina sólo después de tiempo de vencimiento [16].

1.3.6.6 Funcionalidad de OLSR

En esta sección se describe el funcionamiento general de protocolo.

Los paquetes en OLSR se comunican mediante UDP (User Datagram Protocol). El puerto 698 ha sido asignado por IANA (Internet Assigned Numbers Authority) para uso exclusivo por el protocolo OLSR [17] [11].

OLSR está dividido en un "núcleo" de funcionalidad, que siempre se requiere para que el protocolo trabaje y un conjunto de funciones auxiliares.

El núcleo; específica un protocolo capaz de proporcionar enrutamiento en una MANET independiente. Una función auxiliar proporciona funcionalidad adicional, que puede ser aplicable en escenarios específicos, por ejemplo, en el caso que un nodo proporcione conectividad entre la MANET y otro dominio de enrutamiento [3].

Todas las funciones auxiliares son compatibles, en la medida que el (sub) conjunto de funciones auxiliares puede implementarse con el núcleo. Además, el protocolo permite nodos heterogéneos, es decir, nodos que implementan diferentes subconjuntos de las funciones auxiliares. El propósito de dividir el funcionamiento de OLSR en un núcleo de funcionalidad y un conjunto de funciones auxiliares es proporcionar un protocolo simple y fácil de comprender [3].

Funcionamiento

La funcionalidad principal de OLSR define el comportamiento de un nodo, equipado con interfaces OLSR que participan en la MANET y se ejecutan OLSR como protocolo de enrutamiento. Esto incluye una especificación de mensajes OLSR y su transmisión a través de la red, así como: detección de enlaces, difusión de topología y cálculo de ruta.

Específicamente el núcleo está formado por los siguientes componentes:

En la figura 1.1 se detalla como el protocolo OLSR realiza las detecciones de enlace y vecinos y; además, la selección y señalización de los MRPs.

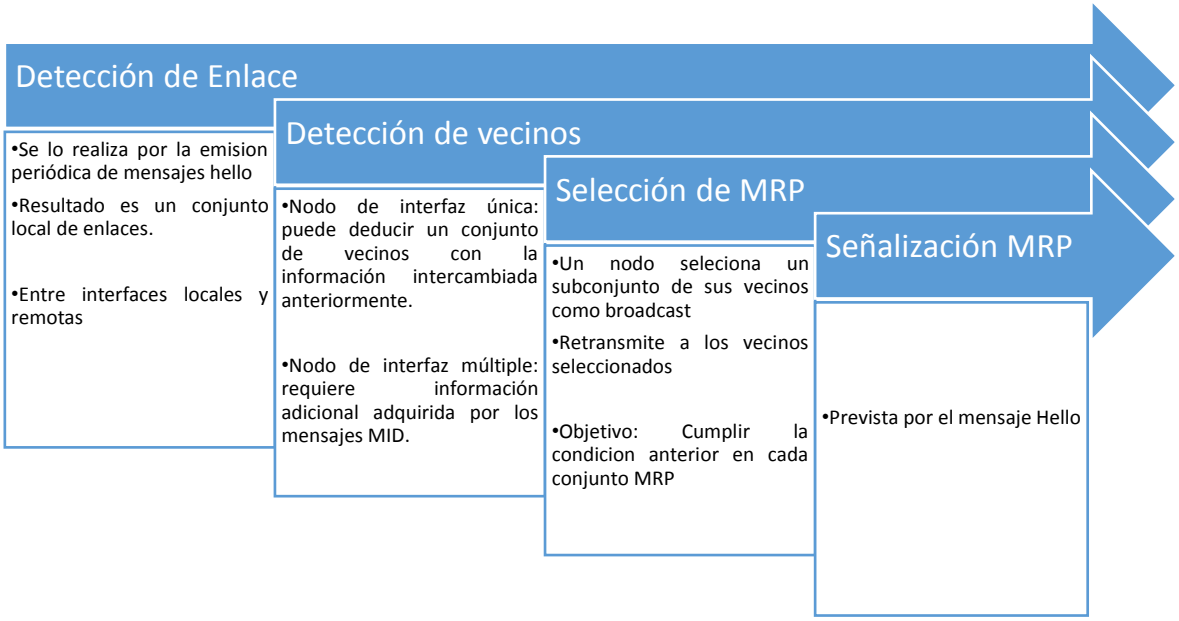


Figura 1.1. Funcionalidad del protocolo OLSR.

Difusión de mensajes TC

Estos mensajes son difundidos con el propósito de proveer a cada nodo en la red con suficiente información de estado de enlace para permitir el cálculo de la ruta.

Cálculo de la ruta

Dada la información de estado de enlace adquirida a través del intercambio periódico de mensajes, así como la configuración de la interfaz de los nodos, la tabla de enrutamiento puede ser calculada para cada nodo. La principal noción para este mecanismo es la relación MRP.

2. METODOLOGÍA

El presente capítulo describe la metodología de trabajo desarrollara en el presente estudio técnico. Este capítulo se encuentra dividido en secciones las cuales permiten mantener un trabajo organizado acorde a los objetivos planteados en el capítulo anterior.

Como primer punto, se realiza una revisión del simulador de red (Network Simulator) versión 3, NS-3, donde se lleva a cabo una breve introducción acerca del simulador, sus generalidades y características.

Como segundo punto, se detalla la estructura del código de simulación utilizado para obtener los resultados. Cabe recalcar que el script utilizado para desarrollar este estudio técnico se basa en un código que ya está implementado en el simulador el cual se llama wifi-example-sim.cc. Al script mencionado se realizaron varias modificaciones para que se puedan cumplir los objetivos de este estudio. Estas modificaciones se detallan en la sección 2.2.

A continuación, se presenta el escenario de pruebas utilizado para tener los resultados de las simulaciones que se realizan en este estudio.

Posteriormente, se enumeran las métricas con las cuales se evalúa la eficiencia del protocolo OLSR, estas son: throughput, tasa de entrega de paquetes y retardo promedio.

Finalmente, se describe el escenario de pruebas. En este se encuentran todos los parámetros y características que se utilizaron para crear dicho escenario y poder realizar las simulaciones necesarias.

2.1. ESTUDIO DE NS-3

Las redes ad-hoc no pueden ser estudiadas de forma abierta debido a la gran demanda de recursos como el número de nodos, tiempo, coste asociado; por lo tanto, se ha optado por utilizar simuladores de red que facilitan el estudio de estas redes. Este tipo de herramienta basa su funcionamiento en un conjunto de códigos que permiten el estudio de protocolos de enrutamiento en varios escenarios de forma controlada y eficiente.

2.1.1 INTRODUCCIÓN A NS-3

NS (Network Simulator) es un nombre para una serie de simuladores de red de eventos discretos , específicamente NS-1, NS-2 y NS-3. Todos son simuladores de redes de eventos discretos, utilizados principalmente en investigación [18] y enseñanza. En el proceso de desarrollo de NS-3, se decidió abandonar por completo la compatibilidad con NS-2 [19]. El nuevo simulador se escribiría desde cero, utilizando el lenguaje de programación C ++ . El desarrollo de NS-3 comenzó en julio de 2006.

Network Simulator 3 es un software de código abierto, que principalmente, el sistema de destino necesita tener un entorno de desarrollo de software para construir primero las bibliotecas, y luego construir el programa de usuario [20] .

Usando NS-3 se pueden crear conexiones PointToPoint, Wireless, CSMA, etc. entre nodos. Cuando las tarjetas de red están habilitadas en los dispositivos, se agregan diferentes parámetros en los canales (es decir, la ruta del mundo real utilizada para enviar datos) que son la velocidad de datos, el tamaño del paquete, etc. Ahora se usa la aplicación para generar tráfico y enviar los paquetes usando estas aplicaciones.

2.1.2 GENERALIDADES DE NS-3

En este capítulo echamos un vistazo superficial al simulador NS-3, destinado a sustituir a NS-2. El proyecto es administrado por el Consorcio NS-3, y todos los materiales están disponibles en nsnam.org .

Para configurar una simulación simple, un script NS-2 Tcl tenía que crear nodos y enlaces, crear "agentes" de conexión de red unidos a los nodos y crear aplicaciones generadoras de tráfico asociadas a los agentes. Lo mismo se aplica a NS-3, pero además cada nodo debe configurarse con sus interfaces de red, y cada interfaz de red debe tener asignada una dirección IP [21].

2.1.3 CARACTERÍSTICAS DE NS-3

NS-3 ofrece características especiales que se pueden utilizar para integraciones de la vida real. Algunas de estas características son [22]:

- **Rastreo de los nodos:** NS-3 permite seguir el rastro de las rutas de los nodos, lo que ayuda a conocer cuántos datos se envían o reciben. Los archivos de rastreo se generan para monitorear estas actividades.
- **NetAnim (Network Animator):** Es una versión animada de cómo será la red en la vida real y cómo transferirán los datos entre nodos.
- **Archivo pcap:** NS-3 ayuda a generar un archivo pcap que se puede utilizar para obtener toda la información de los paquetes (por ejemplo, número de secuencia, IP de origen, IP de destino, etc.). Estos pcaps se pueden ver usando una herramienta de software conocida como Wireshark [22].
- Los atributos se utilizan para configurar la mayoría de los modelos de elementos de red con un conjunto razonable de valores predeterminados (como el TTL inicial de tiempo de vida valor especificado cuando se crea un nuevo paquete IPv4).
- Los objetos auxiliares están diseñados para ayudar y ocultar algunos de los detalles de varias acciones necesarias para crear y ejecutar una simulación NS-3.

Por ejemplo, *Point to Point Helper* proporciona un método sencillo para crear una red punto a punto.

2.1.4 MODELADO DE RED EN NS-3

Como en prácticamente todas las herramientas de simulación de red, el simulador NS-3 tiene modelos para todos los diversos elementos de red que comprenden una red informática. En particular hay modelos para [22]:

- Los *nodos de red*, que representan tanto sistemas finales como computadoras de escritorio y portátiles, así como enrutadores, concentradores y conmutadores de red.
- *Dispositivos de red*, representan el dispositivo físico que conecta un nodo al canal de comunicaciones. Esto podría ser una simple tarjeta de red Ethernet o un dispositivo inalámbrico IEEE 802.11 más complejo.
- *Canales de comunicación* que representan el medio utilizado para enviar la información entre dispositivos de red. Estos pueden ser de fibra óptica, enlaces a puntos, medios compartidos basados en difusión como Ethernet o espectro inalámbrico utilizado para comunicaciones inalámbricas.
- *Protocolos de comunicaciones*, que modelan la implementación de las descripciones de protocolos que se encuentran en los diversos documentos de solicitud de comentarios de Internet, así como los protocolos experimentales más nuevos aún no estandarizados.
- *Encabezados de protocolo* que son subconjuntos de los datos encontrados en los paquetes de red y tienen formatos específicos para cada uno de los objetos del protocolo.
- *Los paquetes de red* son la unidad fundamental de intercambio de información en una red. Casi siempre un paquete de red contiene uno o más encabezados de protocolo que describen la información que necesita la implementación del protocolo en los puntos finales y varios saltos en el camino.

2.1.5 ESTRUCTURA DE NS-3

NS-3 tiene una estructura modular, como se indica en la Figura 2.1. Las capas inferiores contienen el modelo básico común de todo el simulador, mientras que las capas superiores están destinadas a agregar detalles en el escenario simulado,

como: protocolos de red , dispositivos y tecnologías. Finalmente, en la parte superior del modelo, el "helper" define interfaces útiles para que el usuario implemente las características del escenario. Cada elemento representado en la figura 2.1 se describe brevemente a continuación [23] [24]:

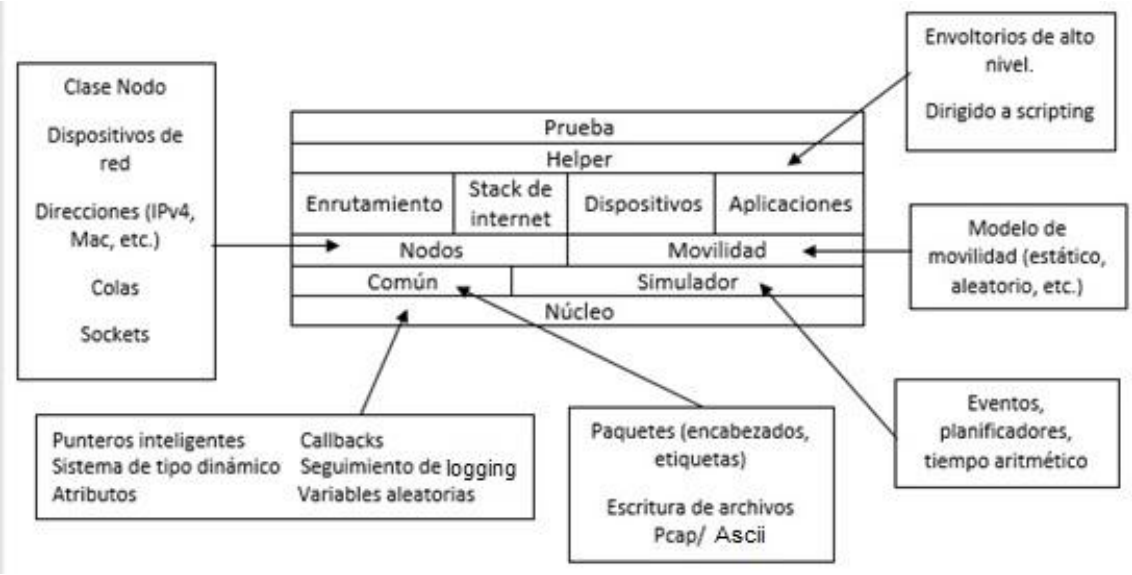


Figura 2.1. Arquitectura de NS-3 **[25] [26]**.

- **Núcleo**: incluye todos los componentes básicos necesarios para crear la simulación (generador de variables aleatorias, seguimiento de eventos y mecanismos de registro).
- **Común**: contiene todos los elementos que definen cada paquete transmitido dentro del simulador.
- **Simulador**: programa los eventos gestionando los temporizadores de simulación.
- **Nodo**: tiene como objetivo crear todos los nodos de red dentro de la simulación. También permite la definición de pilas de protocolos , interfaces de red y aplicaciones para cada nodo.
- **Movilidad**: contiene diferentes patrones de movilidad que pueden usarse en un escenario móvil para simular los movimientos del dispositivo.
- **Routing**: permite la utilización de diferentes protocolos de enrutamiento dentro de la red simulada
- **Stack de Internet**: permite la instalación de los protocolos TCP (Transmission Control Protocol/Internet Protocol) / IPv4 y de IPv6 en cada nodo simulado.

- **Dispositivos**: define todas las interfaces de red disponibles dependientes de la tecnología de un nodo.

Modelo básico de red de NS-3

En la figura 2.2 se aprecia el modelo básico de red de NS-3. Se pueden observar dos nodos en NS-3, los cuales permiten tener métodos para representar dispositivos informáticos en las simulaciones tales como: Host, router, switch. Los nodos son equipos a los cuales se les pueden agregar funcionalidades; de esta manera, se adicionan aplicaciones, pilas de protocolos, tarjetas de red, etc.

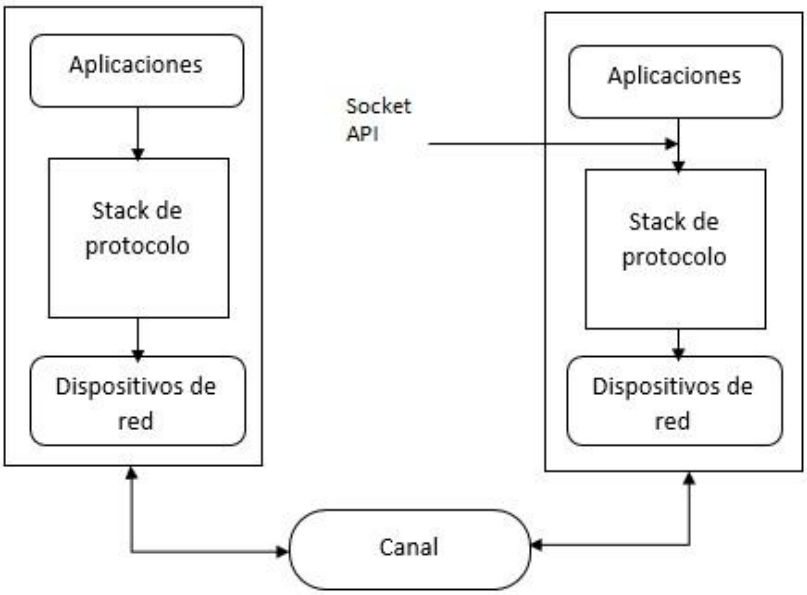


Figura 2.2. Modelo básico de red en NS-3 **[27]**.

2.2. ESTRUCTURA DEL CÓDIGO DE SIMULACIÓN

Esta parte del estudio técnico es de suma importancia, ya que se procede a detallar el script de simulación utilizado. El código, wifi-example-sim.cc es el script al cual se realiza una serie de modificaciones a fin de cumplir con los objetivos del presente estudio técnico, entre los cuales están:

- La evaluación de la eficiencia del protocolo OLSR sobre redes MANET usando el simulador NS-3
- Estudiar el protocolo OLSR, incluyendo su funcionamiento, tipo de mensajes e información de mayor importancia para este estudio, analizar de forma detallada los parámetros que afectan la eficiencia del protocolo OLSR en las redes MANET.

- Identificar las mejores variaciones y combinaciones de los parámetros del protocolo OLSR mediante simulación del protocolo OLSR en diferentes escenarios de redes MANET.
- Analizar los resultados obtenidos mediante métricas como: throughput, tasa de entrega de paquetes y retardo promedio.

2.2.1 LIBRERÍAS

El código realizado comienza con la declaración de los módulos utilizados en este programa. Estos módulos son, en realidad, llamadas a conjuntos de librerías más grandes, donde se encuentra la declaración de las clases que componen los diferentes módulos del simulador [28].

El código 2.1 presenta la sección correspondiente al código de simulación en el cual se defines los módulos utilizados por el código de simulacióng y los módulos adicionales que se utilizaron para cumplir con las particularidades del estudio.

Código 2.1. Librerías.

```
1  #include <ctime>
2  #include <iostream>
3  #include <sstream>
4  #include "ns3/core-module.h"
5  #include "ns3/network-module.h"
6  #include "ns3/mobility-module.h"
7  #include "ns3/wifi-module.h"
8  #include "ns3/internet-module.h"
9  #include "ns3/olsr-module.h"
10 #include "ns3/olsr-routing-protocol.h"
11 #include "ns3/olsr-helper.h"
12 #include "ns3/ipv4-list-routing-helper.h"
13 #include "ns3/ipv4-static-routing-helper.h"
14 #include "ns3/applications-module.h"
15 #include "ns3/flow-monitor-module.h"
16 #include "ns3/stats-module.h"
```

```
17 #include "wifi-example-apps.h"
18 #include "ns3/ipv4.h"
```

A continuación, se describen las librerías de interés utilizadas para el presente estudio:

Core module

Este es el núcleo de NS-3, lo cual permite realizar simulaciones a través de clases. Estas permiten realizar el modelamiento de los protocolos, canales y dispositivos que se utilizan en NS-3. Las clases y módulos más importantes [29] [30]:

- *SmartPointer::Ptr:* permite el uso de punteros inteligentes. Proporciona todas las interfaces proporcionadas por punteros normales con algunas excepciones. Durante la construcción, posee la memoria y la libera cuando sale del alcance. De esta manera, el programador es libre de administrar la memoria asignada dinámicamente.
- *Object:* una clase que permite administrar memoria y generación de objetos. En C++, Object es una entidad del mundo real, por ejemplo, silla, automóvil, bolígrafo, móvil, computadora portátil, etc. En otras palabras, el objeto es una entidad que tiene estado y comportamiento
- *Simulator::Run()*: es una función global a ejecutar la simulación en NS-3 a través de eventos previamente programados. La simulación se ejecutará hasta uno de [31]:
  - Ya no hay eventos presentes.
  - El usuario llamado **Simulator :: Stop**
  - El usuario llamó **Simulator :: Stop** con un tiempo de detención y el tiempo de vencimiento del próximo evento a procesar es mayor o igual que el tiempo de detención.
- *Simulator::Stop():* función global que detiene la simulación luego que han transcurrido los segundos indicados en el argumento de la función. Dice al simulador que el evento de llamada debe ser el último ejecutado.
- *Simulator::Destroy():*este método generalmente se invoca al final de una simulación para evitar informes falsos positivos por parte de un verificador de fugas. .

- CommandLine: permite modificar valores por defecto mediante la línea de comandos.

Network module

Un módulo de red es un módulo de software que implementa una función específica en una pila de red, como una interfaz de enlace de datos, un protocolo de transporte o una aplicación de red. Un módulo de red puede ser un módulo de proveedor , un módulo de cliente o ambos, dependiendo de dónde se encuentre en la pila de red. Las clases que se utilizaron en este estudio son [30] [32]:

- *Node, NodeContainer:* por lo general, los ayudantes NS-3 operan en más de un nodo a la vez. Por ejemplo, un dispositivo auxiliar puede querer instalar dispositivos en una gran cantidad de nodos similares. Los métodos de instalación auxiliar generalmente toman un **NodeContainer** como parámetro. .
- *Packet:* cada paquete de red tiene un búfer de bytes. El búfer de bytes almacena el contenido serializado de los encabezados y trailers agregados a un paquete.
- *NetDevice:* esta interfaz define la API a la que deben acceder las capas IP y ARP para administrar una instancia de una capa de dispositivo de red.

Mobility module

El soporte de movilidad en *NS-3* incluye [32]:

- Una fuente de rastreo "notificador de cambio de curso" que se puede utilizar para registrar oyentes a los cambios de curso de un modelo de movilidad
- Una serie de clases auxiliares que se utilizan para colocar nodos y configurar modelos de movilidad (incluidos analizadores para algunos formatos de definición de movilidad).

El modelo utilizado para este estudio es *Random Waypoint Mobility* ya que este es un modelo sintético comúnmente utilizado para la movilidad, por ejemplo, en redes ad-hoc. Es un modelo elemental que describe el patrón de movimiento de nodos independientes mediante términos simples.

Cada objeto comienza haciendo una pausa en el tiempo cero durante la duración gobernada por la variable aleatoria "Pausa". Después de una pausa, el objeto elegirá un nuevo punto de referencia (a través del *PositionAllocator*) y una nueva velocidad aleatoria a través de la variable aleatoria "Velocidad", y comenzará a moverse hacia el

punto de referencia a una velocidad constante. Cuando llega al destino, el proceso comienza de nuevo (haciendo una pausa).

Wifi module

Los nodos *NS-3* pueden tener una colección de objetos NetDevice, tal como una computadora real tiene tarjetas de interfaz separadas para Ethernet, Wifi, Bluetooth, etc. Esta sección describe el dispositivo Net - WifiSNet y modelos relacionados. Al agregar objetos WifiNetDevice a *los* nodos *NS-3*, se pueden crear modelos de infraestructura basada en 802.11 y redes ad-hoc [32]. Las clases utilizadas en el script son:

- *WifiNetDevice:* esta clase es usada en NS-3 para modelar el funcionamiento de una tarjeta de red inalámbrica basada en el estándar IEEE 802.11.
- *WifiChannel:* Ns3::WifiChannel es una clase base abstracta que permite implementar diferentes canales.
- *WifiMacHelper:* crear capas MAC para un *ns3 :: WifiNetDevice* . Esta clase puede crear MAC de tipo *ns3 :: ApWifiMac , ns3 :: StaWifiMac y ns3 :: AdhocWifiMac* . Su propósito es permitir que un *WifiHelper* configure e instale objetos *WifiMac* en una colección de nodos. Por defecto, crea una capa MAC Adhoc sin QoS
- *YansWifiChannelHelper:* esta clase permite administrar y crear objetos de canal wifi para el modelo yans. El objetivo de esta clase es facilitar la creación de un objeto de canal que implemente el modelo de canal de yans (Yet Another Network Simulator).
- *YansWifiPhy:* usa el modelo de capa 802.11 PHY. Este PHY implementa un modelo de 802.11a. Este modelo PHY depende de un modelo de pérdida y retraso de canal provisto por las *clases ns3 :: PropagationLossModel y ns3 :: PropagationDelayModel* , los cuales son miembros de la clase *ns3 :: YansWifiChannel .*
- *YansWifiPhyHelper:* facilita la creación y administración de objetos PHY para el modelo yans. Las trazas Pcap y ASCII generadas por los métodos EnableAscii y EnablePcap: definidos en esta clase corresponden a las trazas de nivel PHY y nos llegan a través de *WifiPhyHelper.*

- *WifiHelper*: ayuda a crear objetos *WifiNetDevice*. Esta clase puede ayudar a crear un gran conjunto de objetos *WifiNetDevice* similares y a configurar un gran conjunto de sus atributos durante la creación.

Internet module

Este módulo [59] provee clases para que los nodos trabajen con la pila de protocolos de la arquitectura TCP/IPv4, incluyen protocolos como: IPv4, TCP, UDP, entre otros. Se encuentra en el directorio src/Internet [30]. Las clases utilizadas son las siguientes [32]:

- *InternetStackHelper:* agrega la funcionalidad IP / TCP / UDP a los nodos existentes. Este asistente permite el rastreo pcap y ascii de eventos en la pila de Internet asociada a un nodo. Esta clase agrega instancias de estos objetos, por defecto, a cada nodo las cuales se destacan:

- Un TCP basado en la fábrica de TCP proporcionada
- Enrutamiento **ipv4** (un objeto de enrutamiento de lista, un objeto de enrutamiento global y un objeto de enrutamiento estático)
- Enrutamiento **ipv6** (un objeto de enrutamiento estático)

- *Ipv4AddressHelper:* una clase auxiliar para hacer la vida más fácil mientras se realiza una simple asignación de dirección IPv4 en scripts. Esta clase es un generador de direcciones IPv4 muy simple. No tiene noción de que las direcciones IP sean parte de un espacio de direcciones global.
- *Ipv4InterfaceContainer:* por lo general, las interfaces NS-3 Ipv4Interfaces se instalan en dispositivos que utilizan un asistente de dirección Ipv4 .

OLSR module

Este módulo es de suma importancia, debido a que el presente estudio se basa en el análisis de los parámetros que afectan a la eficiencia del protocolo OLSR. Aquí se encuentran todo el API necesarios para realizar las respectivas variaciones descritas anteriormente. El simulador NS-3 modela el protocolo OLSR en base al rfc 3626 [3].

Applications module

Conceptualmente, una aplicación tiene cero o más objetos *ns3 :: Socket* asociados con ella, que se crean utilizando la API de creación de Socket de la capacidad Kernel. La API del objeto Socket está modelada según la conocida interfaz de sockets BSD, aunque está algo simplificada para su uso con NS-3 . Además, cualquier llamada de socket que

normalmente se "bloquearía" en sockets normales volverá inmediatamente en NS-3 . Se define un conjunto de "llamadas ascendentes" que se llamarán cuando la llamada de bloqueo anterior salga normalmente [31] [32]:

Stats module

El módulo de estadísticas incluye algunas características útiles para facilitar la recopilación de datos de los experimentos.

En particular, se incluyen las siguientes características [31] [32]:

- El marco central y dos recopiladores de datos básicos: un contador y un observador min / max / avg / total.
- Extensiones de aquellos para trabajar fácilmente con tiempos y paquetes.
- Salida de texto sin formato con formato para OMNet ++.
- Salida de la base de datos utilizando SQLite, un motor SQL independiente, liviano y de alto rendimiento.
- Metadatos obligatorios y abiertos para describir y trabajar con ejecuciones.

- *CounterCalculator:* crea contadores para que estos sigan la trayectoria recorrida por los paquetes enviados y recibidos.
- *PacketCounterCalculator:* estadística utilizada para contar paquetes.
- *PacketSizeMinMaxAvgTotalCalculator:* estadística utilizada para recopilar estadísticas de tamaño de paquete: mínimo, máximo y promedio.
- *TimeMinMaxAvgTotalCalculator:* estadística utilizada para los datos correspondientes a los tiempos de propagación máximos, mínimos, totales y promedios.
- *DataOutputInterfaces:* las clases en el grupo de salida de datos se utilizan para recopilar, preparar y generar datos para la salida posterior en un formato específico, por ejemplo, Omnet ++, SQLite, etc.

2.2.2 DEFINICIÓN DE VARIABLES

A continuación, se presenta la definición de las variables globales utilizadas a lo largo del código de simulación. Estas se pueden modificar de acuerdo a las pruebas

planteadas para realizar el análisis del protocolo OLSR. El código 2.2 muestra la parte del código de simulación en la cual están definidas estas variables utilizadas en el *script* del presente estudio técnico.

Código 2.2. Variables globales.

```
1  int size = 50;
2  int totalTime = 200;
3  int startTime = 0;
4  double helloInterval = 2;
5  double refreshInterval = 2;
6  double tcInterval = 5;
7  double dupHoldTime = 30;
8  bool files = true;
9  int port = 4000;
10 int packetSize = 1024;
11 int numPackets = 400;
```

A continuación, se detalla por qué se utilizaron estas variables.

Variables utilizadas:

- *Size:* variable utilizada para indicar el número de nodos en cada simulación.
- *totalTime:* variable utilizada para indicar el tiempo, en segundos, que durará la simulación.
- *startTime:* variable utilizada para indicar el tiempo, en segundos, en el que inicia la simulación.
- *helloInterval:* variable utilizada para modificar el parámetro de configuración *Hello_Interval* de OLSR. Este determina el tiempo, en segundos, en el que se envían los mensajes Hello.
- *refreshInterval:* variable utilizada para modificar el parámetro de configuración *Refresh_Interval* de OLSR. Indica el tiempo en segundos en el cual cada enlace y vecino debe anunciarse al menos una vez.

- *tcInterval:* variable utilizada para modificar el parámetro de configuración *Tc_Interval* de OLSR. Indica el periodo de tiempo en segundos en el cual un conjunto de enlaces anunciados debe estar completo.
- *files:* variable utilizada para generar los archivos de análisis de los diferentes escenarios de simulación.
- *port:* variable utilizada para indicar el puerto seleccionado en el que se enviarán los paquetes.
- *munPackets:* variable utilizada para indicar el número de paquetes que se enviarán

2.2.3 ALEATORIEDAD DE EXPERIMENTOS

Las simulaciones NS-3 se pueden configurar para producir resultados deterministas o aleatorios. Si la simulación NS-3 está configurada para usar una semilla determinista fija con el mismo número de ejecución, debería dar la misma salida cada vez que se ejecuta [31].

Un caso de uso típico es ejecutar una simulación como una secuencia de ensayos independientes, para calcular estadísticas en un gran número de ejecuciones independientes. El usuario puede cambiar la semilla global y volver a ejecutar la simulación, o puede avanzar el estado de subtransmisión del RNG, que se conoce como incremento del número de ejecución.

En el código 2.3 se presenta el fragmento de código en el cual se produce la aleatoriedad de las simulaciones.

Código 2.3. Generación aleatoria de las simulaciones.

```
1   srand(time(NULL));
2   SeedManager::SetSeed(rand());
3   SeedManager::SetRun(rand());
```

La línea 1 descrita en el código 2.3 permite generar números aleatorias mediante la llamada a la función *srand ().*

La generación de números aleatorios adquiere gran relevancia para un programador, pudiendo usarse para distintas tareas de las cuales se pueden citar:

- Construcción preliminar de programas, en los que a falta de datos definitivos introducimos datos aleatorios.
- Simular procesos aleatorios.
- Verificación de programas, haciendo pruebas con datos aleatorios.

Para obtener aleatoriedad en múltiples ejecuciones de simulación, se debe establecer la semilla de manera diferente o establecer el número de ejecución de manera diferente. Para establecer una semilla, se debe llamar a ns3 :: SeedManager :: SetSeed () al comienzo del programa; para establecer un número de ejecución con la misma semilla, se debe llamar a ns3 :: SeedManager :: SetRun () al comienzo del programa.

2.2.4 MOVILIDAD DE RED

A continuación, se presenta la sección del código de simulación en la cual se utilizó para configurar la movilidad de los nodos para los escenarios de simulación.

En el código 2.4 se define el espacio de simulación

Código 2.4. Escenario de simulación.

```
1 ObjectFactory posicion;
2 posicion.SetTypeId("ns3::RandomRectanglePositionAllocator");
3 posicion.Set("X",StringValue("ns3::UniformRandomVariable[Min=0.0|Max=500.0]"));
4 posicion.Set("Y",StringValue("ns3::UniformRandomVariable[Min=0.0|Max=500.0]"));
```

En la línea 1 se realiza la definición del objeto *pos.* A continuación, en la línea 2, se utiliza la función *ns3::RandomRectanglePositionAllocator,* la cual permite asignar posiciones aleatorias dentro de un rectángulo de acuerdo con un par de variables aleatorias.

En las líneas 3 y 4 se utiliza la clase *ns3::UniformRandomVariable,* esta clase admite la creación de objetos que devuelven números aleatorios de una distribución uniforme fija. También es compatible con la generación de números aleatorios únicos a partir de varias distribuciones uniformes.

El rango de salida es $[min, \max)$ para valores de coma flotante, (max *excluidos*) y $[min, \max]$ (max *incluidos*) para valores integrales. Se ha fijado un área de 500 m x 500 m como área en la que se moverán los nodos.

En el código 2.5 se puede apreciar la movilidad aleatoria de los nodos.

Código 2.5. Movilidad de los nodos.

```
1 mobilityAdhoc.SetMobilityModel("ns3::RandomWaypointMobilityModel","Speed",
2 StringValue (ssSpeed.str()), "Pause", StringValue (ssPause.str()),
3 "PositionAllocator", PointerValue(taPositionAlloc));
```

En la línea 1, se puede apreciar el modelo de movilidad utilizado en cual es *ns3::RandomWaypointMobilityModel.*

En este modelo cada objeto comienza haciendo una pausa en el tiempo cero durante la duración gobernada por la variable aleatoria "Pausa". Después de una pausa, el objeto elegirá un nuevo punto de referencia (a través del PositionAllocator) y una nueva velocidad aleatoria a través de la variable aleatoria "Velocidad", y comenzará a moverse hacia el punto de referencia a una velocidad constante. Cuando llega al destino, el proceso comienza de nuevo (haciendo una pausa).

La variable *ssSpeed* es de suma interés ya que este permite definir la velocidad de los nodos, esto puede influir en el funcionamiento del protocolo OLSR.

2.2.5 DISPOSITIVOS DE RED

En el código 2.6 se exponen la parte de código en la cual se crean y asignan los dispositivos de red a los nodos implementados en la simulación.

Código 2.6. Creación de dispositivos de red.

```
1 WifiHelper wifi;
2 WifiMacHelper wifiMac;
3 wifiMac.SetType ("ns3::AdhocWifiMac");
4 YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
5 YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
6 wifiPhy.SetChannel (wifiChannel.Create ());
7 wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager", "DataMode",
8 StringValue("OfdmRate6Mbps"), "RtsCtsThreshold", UintegerValue (0));
9 NetDeviceContainer nodeDevices = wifi.Install (wifiPhy, wifiMac, nodes);
```

Se puede apreciar el uso de las clases helper en diferentes partes del código presente, con estos helper se pueden crear dispositivos desde su capa física (líneas 1, 4, 5, 6, 7) y MAC (líneas 2 y 3).

A nivel físico se utiliza el modelo YANS, el cual fue descrito en la sección 2.2.1.

A nivel de la subcapa MAC se utiliza el estándar de la comunicación y el modo *ad-hoc.* Se configuró el modo mencionado anteriormente ya que es de suma importancia para este estudio.

En la línea 9, se utiliza un contenedor de dispositivos llamado *nodeDevices* el cual es un objeto de la clase *NetDeviceContainer.* Por último, para crear los dispositivos se utilizan los estándares de capa física y MAC creados anteriormente.

2.2.6 CONFIGURACIÓN DE LOS PARÁMETROS DE OLSR

La sección de código siguiente define el protocolo OLSR y sus parámetros establecidos. Esta sección es de suma importancia en el código de simulación, ya que en esta se puede modificar los valores de los parámetros que se han seleccionado para llevar a cabo las simulaciones sobre diferentes situaciones.

En el código 2.7 se pueden ver las líneas de código que permitirán variar el valor de los parámetros de OLSR.

Código 2.7. Configuración de OLSR.

```
1 OlsrHelper olsr;
2 olsr.Set("HelloInterval",TimeValue(Seconds(helloInterval)));
3 olsr.Set("TcInterval", TimeValue(Seconds(tcInterval)));
4 olsr.Set("MidInterval", TimeValue(Seconds(tcInterval)));
5 olsr.Set("HnaInterval", TimeValue(Seconds(tcInterval)));
6 olsr.Set("Willingness", EnumValue(3));
```

En la línea 1 se instancia un objeto de la clase *OlsrHelper,* el cual permitirá acceder a sus parámetros para poder modificarlos. En las siguientes líneas (2, 3, 4, 5, 6) se establecen los valores de los parámetros de configuración de OLSR. Estos inicialmente tienen el valor por defecto el cual esta estandarizado en su respectivo estándar (RFC 3626).

2.2.7 DEFINICIÓN DE NODOS

En la sección presentada a continuación, el código 2.8 define tanto el nodo emisor como el nodo receptor.

El nodo emisor (desde línea 1 hasta línea 8) se definen atributos propios para este nodo, los cuales son: el puerto escucha del nodo por el cual se establecerá la comunicación, el tamaño que tendrán los paquetes para ser enviados, el intervalo de tiempo en el que se enviarán los paquetes, el número de paquetes que se enviarán y el tiempo en el que se iniciará la aplicación.

Por otro lado, en el nodo receptor (desde línea 9 hasta línea 13) simplemente se define el puerto por el cual escuchará la comunicación, el puerto debe ser el mismo definido en el nodo emisor.

En la línea 1 se crea un objeto llamado *appSource*, el cual crea un objeto de la clase Node que servirá para crear al nodo emisor. Desde la línea 2 hasta la línea 8 se crea el nodo emisor; en la línea 2 se crea un objeto del tipo *Sender* el cual servirá para acceder a los parámetros del nodo a configurar (a través de la función *SetAttribute*) los cuales son (líneas 3-6): un número entero que permite definir el número de puerto, un número entero para el tamaño del paquete, el intervalo de tiempo entre cada paquete (este es un string que hace un llamado a la clase *ConstantRandomVariable*), un valor entero que define el número de paquetes en el nodo.

En la línea 7 se agrega el nodo *sender.* En la línea 8 se define el tiempo en el que se iniciará la aplicación. En las líneas 9-12 se crea el nodo receptor. En este solo se define el número el puerto por el cual se establecerá la comunicación (línea 11). Finalmente se define el tiempo en el que se iniciará la aplicación en este nodo (línea 13).

Código 2.8. Definición de nodos.

```
1 Ptr<Node> appSource = NodeList::GetNode (0); // emisor
2 Ptr<Sender> sender = CreateObject<Sender>();
3 sender ->SetAttribute("Port", UintegerValue(port)); //puerto escucha
4 sender ->SetAttribute("PacketSize", UintegerValue(packetSize));
  sender ->SetAttribute("Interval",
5 StringValue("ns3::ConstantRandomVariable[Constant=0.5]"));
6 sender ->SetAttribute("NumPackets", UintegerValue(numPackets));
7 appSource->AddApplication (sender);
8 sender->SetStartTime (Seconds (1));
```

```
9  Ptr<Node> appSink = NodeList::GetNode (9); //receptor
10 Ptr<Receiver> receiver = CreateObject<Receiver>();
11 receiver ->SetAttribute("Port", UintegerValue(port)); //puerto escucha
12 appSink->AddApplication (receiver);
13 receiver->SetStartTime (Seconds (0));
```

A continuación, se presenta algunos parámetros utilizados para realizar las simulaciones correspondientes:

- Nodo emisor: 0, Nodo receptor: 9

  El número del nodo fue seleccionado de forma arbitraria, esto sirve para que las simulaciones realizadas tengan un nodo origen y destino los cuales serán analizados para obtener los datos.

- Puerto de comunicación: 4000

  Un número de puerto es un entero sin signo de 16 bits, que va de 0 a 65535. Para TCP, el número de puerto 0 está reservado y no se puede usar, mientras que, para UDP, el puerto de origen es opcional y un valor de cero significa que no hay puerto. Un proceso asocia sus canales de entrada o salida a través de un socket de Internet, que es un tipo de descriptor de archivo, con un protocolo de transporte, una dirección IP y un número de puerto. Los números de puerto varían de 0 a 65535.

  - Los puertos conocidos varían de 0 a 1023.
  - Los puertos registrados son 1024 a 49151.
  - Los puertos dinámicos (también llamados puertos privados) son 49152 a 65535.

- Tamaño del paquete a transmitir: 1024 bytes, número de paquetes a transmitir: 400.

- Tiempo de transmisión entre cada paquete: 0.5 segundos.

  Se usa este valor para que en el tiempo total de la simulación (200 segundos) se puedan enviar los 400 paquetes y queden dentro del rango de la simulación.

2.2.8 ARCHIVOS DE ANÁLISIS

Para tener un análisis más detallado sobre los resultados de las simulaciones realizadas, se procede a la creación de archivos que permitirán analizar el funcionamiento del protocolo OLSR. Estas son trazar pcap, tablas de enrutamiento entre vecinos OLSR, retardos de envió de paquetes, etc. En el código 2.9 se expone la sección de código en la cual se procede a la creación de dichos archivos de análisis.

Código 2.9. Archivos de análisis.

```
1  AsciiTraceHelper ascii;
2  wifiPhy.EnableAsciiAll(ascii.CreateFileStream("olsr.tr"));
   wifiPhy.EnablePcapAll(std::string("olsr-node"));
3
   Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper>
   ("olsr.routes", std::ios::out);
4  olsr.PrintRoutingTableAllEvery(Seconds (2),routingStream);
   Ptr<OutputStreamWrapper> neighborStream = Create<OutputStreamWrapper>
5  ("olsr.neighbors", std::ios::out);
6  olsr.PrintNeighborCacheAllEvery(Seconds (2), neighborStream);
```

En general, las líneas de código anteriores permiten la creación de 4 archivos los cuales son: olsr.tr, olsr-node.pcap, olsr.routes y olsr.heighbors. Al momento de ejecutar la simulación estos archivos se crean en el directorio ns3/ns-allinone-3.27/ns-3.27. Estos se utilizan para observar el comportamiento y funcionamiento del protocolo OLSR, por ejemplo, los archivos .tr y pcap los cuales permiten observar la transmisión y recepción de las tramas que se realizan en los nodos. Los archivos .route y .neighbors permiten observar las tablas de enrutamiento propia de cada nodo y la de sus vecinos respectivamente.

2.2.9 ESTADÍSTICAS

Para la obtención de las estadísticas que permitirán un mejor análisis el script seleccionado posee métodos incluidos que permiten la obtención de estos. Los resultados son guardados en un archivo con extensión .data que se encuentra en el mismo directorio mencionado en la sección 2.2.8.

En el código 2.10 se puede observar detalladamente como se realiza la creación de estos ya que es de suma importancia para este estudio.

Código 2.10. Generación de estadísticas de los paquetes enviados.

```
1  DataCollector data;
   Ptr<PacketCounterCalculator> appTx =
2  CreateObject<PacketCounterCalculator>();
3  appTx->SetKey ("sender-tx-packets");
4  appTx->SetContext ("node[0]");
5  Config::Connect ("/NodeList/0/ApplicationList/*/$Sender/Tx",
6  MakeCallback (&PacketCounterCalculator::PacketUpdate, appTx));
7  data.AddDataCalculator (appTx);
```

En la línea uno se crea un objeto del tipo *DataCollector* para almacenar toda la información obtenida en la simulación.

Este contador (línea 2) rastrea cuántos paquetes se generan. Esto está conectado directamente a una señal de rastreo proporcionada por la clase *Sender*.

En el código 2.11 se indica la generación de estadísticas para el nodo receptor.

Código 2.11. Generación de estadísticas de los paquetes recibidos.

```
1  Ptr<CounterCalculator<> > appRx =CreateObject<CounterCalculator<> >();
2  appRx->SetKey ("receiver-rx-packets");
3  appRx->SetContext ("node[9]");
4  receiver->SetCounter (appRx);
5  data.AddDataCalculator (appRx);
```

En esta parte el contador para los paquetes recibidos es manipulado directamente por uno de los objetos personalizados en la simulación (la aplicación receptora). El objeto receptor recibe un puntero al contador y llama a su método Update () cada vez que llega un paquete.

En el código 2.12 se presenta la sección del script en la cual permite la obtención de estadísticas básicas de un paquete.

Código 2.12. Estadísticas básicas del paquete.

```
1 Ptr<PacketSizeMinMaxAvgTotalCalculator> appTxPkts =
2 CreateObject<PacketSizeMinMaxAvgTotalCalculator>();
3 appTxPkts->SetKey ("tx-pkt-size");
4 appTxPkts->SetContext ("node[0]");
5 Config::Connect ("/NodeList/0/ApplicationList/*/$Sender/Tx",
6 MakeCallback
7 (&PacketSizeMinMaxAvgTotalCalculator::PacketUpdate,
8 appTxPkts));
9 data.AddDataCalculator (appTxPkts);
```

Este DataCalculator se conecta directamente a la traza de transmisión proporcionada por nuestra aplicación de remitente. Registra algunas estadísticas básicas sobre los tamaños de los paquetes recibidos (min, max, avg, número total de bytes).

En el código 2.13 se presenta la parte del código en la que se obtiene el retardo de propagación.

Código 2.13. Retardo de propagación.

```
1 Ptr<TimeMinMaxAvgTotalCalculator> delayStat =
2 CreateObject<TimeMinMaxAvgTotalCalculator>();
3 delayStat->SetKey ("delay");
4 delayStat->SetContext (".");
5 receiver->SetDelayTracker (delayStat);
6 data.AddDataCalculator (delayStat);
```

Aquí se manipula directamente otro DataCollector que rastrea retardos mínimos, máximos, totales y promedio de propagación durante la simulación.

2.3. PARÁMETROS DE OLSR

OLSR está regulado por un conjunto de parámetros definidos por su respectiva RFC 3626 [3]. Dado que una transmisión de paquetes más intensiva conduce a un mayor consumo de energía y un agotamiento más rápido de la batería. Este trabajo estudia los parámetros de OLSR más relevantes que puedan afectar su eficiencia; entre los cuales se pueden destacar:

- **HELLO_INTERVAL:** Determina el tiempo entre los mensajes de saludo enviados. Se incluye en los mensajes de saludo, de modo que un nodo sepa el tiempo esperado hasta que se reciba el próximo saludo del mismo nodo.
  Valor por defecto: 2 segundos. Se utilizaron los valores de variación: 0.5, 1, 2 y 4.
- **TC_INTERVAL**. Los mensajes TC pueden ser parciales como los mensajes de saludo. TC_INTERVAL es el período en el que debe completarse el conjunto de enlaces anunciados.
  Valor por defecto: 5 segundos. Se utilizaron los valores de variación: 1.25, 2.5, 5 y 10.
- **HNA_INTERVAL.** Especifica el intervalo de tiempo entre dos mensajes HNA consecutivos. Los mensajes HNA solo se emitirán cuando un nodo OLSR tenga interfaces no OLSR.
  Valor por defecto: 5 segundos. Se utilizaron los valores de variación: 1.25, 2.5, 5 y 7,5.
- **NEIGHB_HOLD_TIME**. Indica durante cuánto tiempo la información proporcionada en un mensaje de saludo debe considerarse como válido.
  Valor por defecto: 3 * HELLO_INTERVAL.
- **TOP_HOLD_TIME**. Este parámetro es análogo al anterior, definiendo el período de validez para la información del mensaje TC.
  Valor por defecto: 3 * TC_INTERVAL.

Tabla 2.1. Parámetros utilizados para evaluar la eficiencia del protocolo OLSR

PARÁMETRO	VALOR POR DEFECTO	FIJO/VARIABLE
Size	-	Variable
HELLO_INTERVAL	2	Variable
REFRESH_INTERVAL	2	Fijo
TC_INTERVAL	5	Variable
MID_INTERVAL	TC_INTERVAL	Variable

HNA_INTERVAL	TC_INTERVAL	Variable
NEIGHB_HOLD_TIME	3 x REFRESH_INTERVAL	Variable
TOP_HOLD_TIME	3 x TC_INTERVAL	Variable
DUP_HOLD_TIME	30	Fijo
MID_HOLD_TIME	3 x MID_INTERVAL	Variable
HNA_HOLD_TIME	3 x HNA_INTERVAL	Variable

2.4. ESCENARIO DE PRUEBAS

En el escenario del presente estudio técnico se lo realizó considerando variaciones de los principales parámetros del protocolo OLSR, siendo estas de manera independiente o en combinaciones entre ellos mismos para indagar en estas combinaciones que permitan aumentar la eficiencia del protocolo bajo diferentes escenarios de red, tales como: variación de la densidad de los nodos, velocidad de los nodos.

Para la variación de los nodos se utiliza tres tipos de densidades diferentes (50, 100 y 150 nodos) dentro de la misma área de simulación; esto permite simular el movimiento de las personas en áreas abiertas como calles, parques o plazas que pueden estar llenas (al medio día) o casi vacías (en la noche o tarde); además, para cada tipo de densidad de nodos también se trabajara con diferentes velocidades (1.5, 3 y 5 m/s), estas representan el movimiento de las personas cuando están caminando, corriendo o en bicicleta en estos espacios de gran aforo.

Para este escenario general se realizarán variaciones para tres casos: análisis de parámetros con valores por defecto, análisis de variación individual de los parámetros y análisis de variación de parámetros combinados. Con estos resultados se buscan los valores de los parámetros que permitan mejorar la eficiencia del protocolo OLSR.

2.4.1 DESCRIPCIÓN DEL ESCENARIO DE PRUEBAS

El estudio de simulación es conducido con la implementación de OLSR el cual corre en la versión 3.27 de NS-3.

En el escenario de simulación se encuentra el área en la que se moverán los nodos además de su respectiva velocidad. Primero se considera un área fija con una dimensión de 250.000 m^2. Es decir 500 metros en el eje X y 500 metros en el eje Y; además, se manejan simulaciones en las cuales los nodos se desplazan a tres velocidades diferentes (1.5, 3 y 5 m/s) mencionadas en la sección 2.4.

Se considera una red que contiene n números de nodos: n = 50 para simular una red de baja densidad, n = 100 para simular una red de mediana densidad y n = 150 para simular una red de alta densidad. Toda la simulación corre en 200 segundos con un intervalo entre el envío de paquetes de 0.5 segundos. El tamaño de los paquetes enviados es de 1024 bytes y el número de paquetes a enviar son 400.

Todas las variables nombradas anteriormente se mencionaron previamente en la sección 2.2. La tabla 2.2 describe el escenario de pruebas que se considera para las simulaciones.

Tabla 2.2. Descripción del escenario de pruebas

Parámetros	**Valor**
Área de simulación	250.000m^2
Velocidad de los nodos	1.5, 3 5 (m/s)
Número de nodos	50, 100, 150
Modelo de movilización	*Random Rectangle Position Allocator*
Tipo de tráfico	UDP
Tamaño del paquete	1024 bytes
Número de paquetes	400
Intervalo entre envío de paquetes	0.5 s
Tiempo de simulación	200 s

2.4.2 PRUEBAS REALIZADAS

A continuación, se explican las pruebas que se realizaran en el presente estudio técnico las cuales permitirán evaluara la eficiencia del protocolo OLSR.

- **Análisis de variación individual de los parámetros**

En este escenario se utilizan los valores estandarizados en su respectiva RFC [3] lo cual permitirá verificar como trabaja el protocolo OLSR con sus valores por defecto, los resultados de estos se analizarán a través den las métricas descritas en la sección 2.5.

- **Análisis de parámetros con valores por defecto**

En esta sección se varían los valores de los parámetros mencionados anteriormente en el alcance del trabajo con la finalidad de hallar los valores adecuados que permitan incrementar su eficiencia en los escenarios descritos en la sección 2.3.

Para lograr esto se considerará valores tanto por encima como por debajo de los parámetros de configuración estandarizados en su RFC [3] que son de interés para este estudio técnico. Por lo tanto, se realizarán simulaciones con los nuevos valores para poder realizar una comparativa entre cada uno de los resultados obtenidos.

- **Análisis de variación de parámetros combinados**

Partiendo del escenario anterior que permite variar los valores de los parámetros individualmente, se crea la necesidad de analizar la combinación de los parámetros que más influyen en la eficiencia del protocolo OLSR. Este análisis es muy importante ya que a través de este se puede concluir si la eficiencia aumenta o no al variar un conjunto de parámetros.

2.5. MÉTRICAS A EVALUARSE

Para analizar los resultados obtenidos a través de las simulaciones se deben considerar ciertas métricas que influyen directamente con el desempeño del protocolo, las cuales se listan a continuación.

2.5.1 THROUGHPUT

El throughput es una medida de cuántas unidades de información puede procesar un sistema en un período de tiempo determinado. Se aplica ampliamente a los sistemas que van desde diversos aspectos de los sistemas informáticos y de red hasta las organizaciones [33]. También el throughput se refiere al desempeño de las tareas por un servicio o dispositivo informático durante un período específico. Mide la cantidad de trabajo completado contra el tiempo consumido y puede usarse para medir el rendimiento de un procesador, memoria y / o comunicaciones de red la, la ecuación 2.1 permite ejemplificar la incorporación de una fórmula.

$$Troughput = \frac{Datos\ entregados\ exitosamente}{Tiempo\ de\ transmision}\ [Kbps] \qquad (2.1)$$

2.5.2 TASA DE ENTREGA DE PAQUETES

“La tasa promedio de entrega es la fracción de paquetes que llegan a su destino por el número total de paquetes enviados en la red, generalmente se expresa en porcentaje” [34], la ecuación 2.2 permite ejemplificar la incorporación de una fórmula.

$$Tasa\ de\ entrega = \frac{Paquetes\ recibidos}{Paquetes\ enviados}\ X\ 100 \quad (2.2)$$

2.5.3 RETARDO PROMEDIO EXTREMO A EXTREMO

"El retardo promedio extremo a extremo es el tiempo que demora un paquete en recorrer una ruta desde un punto fuente en la red hasta un punto final, donde el retardo del paquete i-ésimo se expresa como $Tr_j - Ts_i$. Asumiendo que j paquetes se entregaron a su destino, mostrando en la siguiente ecuación la manera de calcularlo " [34], la ecuación 2.2 permite ejemplificar la incorporación de una fórmula.

$$Retardo = \frac{\sum jTr_j - Ts_j}{Paquetes\ entregados}\ [s] \quad (2.3)$$

2.6. TEST ESTDÍSTICOS

Además de las métricas mencionadas anteriormente se realizó un análisis estadístico con todos los valores obtenidos, la función utilizada fue T.TEST (matriz1, matriz2, colas, tipo) en Excel, esta función calcula la probabilidad asociada con la prueba T de Student. La función se usa comúnmente para probar la probabilidad de dos muestras que tienen poblaciones subyacentes con la misma media. Se utilizó un número de cola igual a 1 y un tipo de prueba t igual a 3 (Prueba t de varianza desigual de dos muestras) [35]. Se utilizó esta función con todos los valores obtenidos al realizar las variaciones comparándolos con los resultados obtenidos de los valores por defecto, si el valor obtenido al realizar la comparación es menor al 5% quiere decir que el valor escogido permite obtener resultados diferentes a los obtenidos por defecto, si es mayor al 5% quiere decir que son iguales.

3. RESULTADOS Y DISCUSIÓN

En el presente capítulo se explican las diferentes pruebas realizadas con los parámetros previamente descritos en la sección 2.3 estos son: HELLO_INTERVAL, TC_INTERVAL, MID_INTERVAL, HNA_INTERVAL.

Cabe recalcar que los parámetros: NEIGHB_HOLD_TIME (Hello), MID_HOLD_TIME (MID) y TOP_HOLD_TIME (TC) varían de forma directamente proporcional con los parámetros mencionados en el párrafo anterior; por lo tanto, al variar dichos parámetros se varían también estos.

3.1 ANÁLISIS DE VALORES DE PARÁMETROS POR DEFECTO

Este punto trata sobre las pruebas realizadas con los valores por defecto de los parámetros del protocolo OLSR para diferentes densidades de nodos y para las tres velocidades.

La razón por la cual se realizan estas simulaciones es para evaluar la eficiencia que presenta el protocolo OLSR. Esto permite realizar una comparación con la eficiencia obtenida mediante los cambios que se realicen a los valores de los parámetros.

A continuación, la figura 3.1 indica las métricas utilizadas para la evaluación del protocolo OLSR las cuales fueron descritas en la sección 2.5 (throughput, tasas de entrega de paquetes, retardo promedio extremo a extremo).

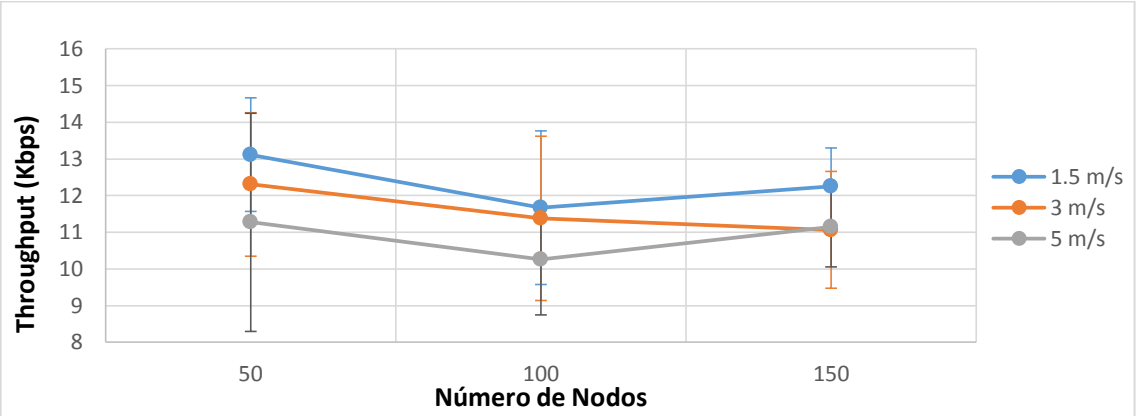
En la figura 3.1a se observar que los valores de los parámetros por efecto entregan una mayor throughput con una densidad de 50 nodos a una velocidad de 1.5 m/s; esto se contrasta al incrementar la velocidad de desplazamiento, ya que se puede apreciar una caída en el throughput. Si bien el throughput decae al incrementar la densidad y velocidad de los nodos, a una velocidad menor se sigue obteniendo un mayor valor de throughput en comparación del obtenido al aumentar la velocidad de desplazamiento.

De la misma manera, en la figura 3.1b, algo similar pasa con la tasa de entrega de paquetes. A velocidades de desplazamiento bajas con una densidad baja de nodos se obtiene una menor pérdida de paquetes en comparación a lo obtenido a altas velocidades, al igual que la figura anterior si bien la pérdida incrementa a medida que se aumenta la velocidad de desplazamiento y la densidad de los nodos para una velocidad de 1.5 m/s se sigue obteniendo mejores resultados en comparación con los anteriores

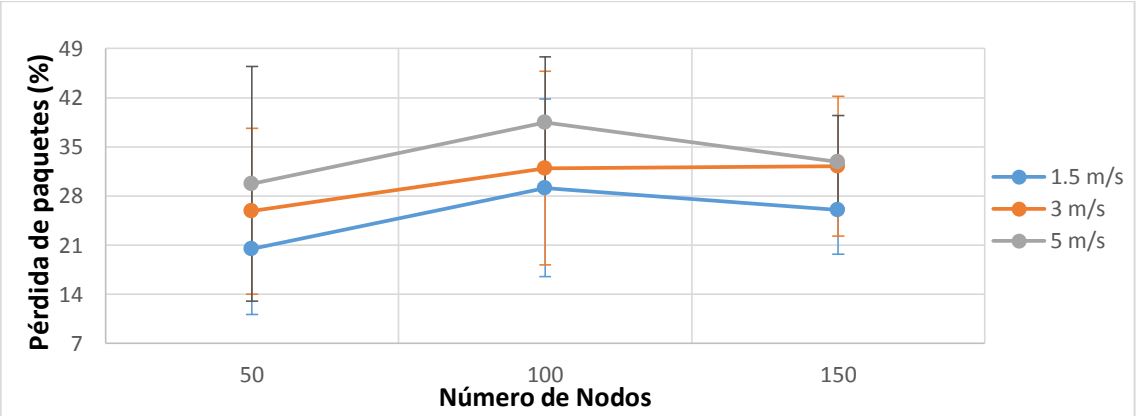
escenarios. Lo cual afirma la confiabilidad del protocolo al tener una red con un número menor de nodos.

En el caso de la figura 3.1c, se puede visualizar un incremento en el retardo al incrementar el número de nodos en la red, lo cual tiene sentido ya que los mensajes del protocolo OLSR deben atravesar una mayor cantidad de nodos para establecer las rutas o mantener las rutas de la topología; además, como existe un mayor número de nodos existe mayor contención, dicho de otra manera, existe más pelea por acceder al medio lo que involucra más retardo y más colisiones.

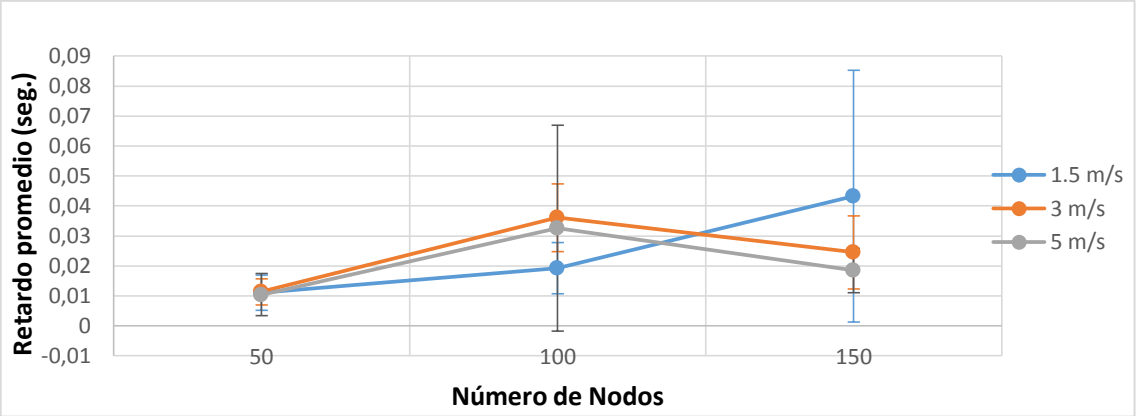
De forma general, el protocolo OLSR con sus valores de parámetros por defecto presenta un mejor comportamiento para densidades de nodos bajas. Cada resultado descrito anteriormente se comparará con los generados en las secciones posteriores.



a) Throughput vs el número de nodos para tres velocidades de desplazamiento de los nodos.



b) Tasa de entrega de paquetes vs el número de nodos para tres velocidades de desplazamiento de los nodos.



c) Retardo promedio vs el número de nodos para tres velocidades de desplazamiento de los nodos.

Figura 3.1. Métricas de evaluación del protocolo OLSR con los parámetros por defecto.

3.2 ANÁLISIS DE VARIACIÓN DE LOS PARÁMETROS

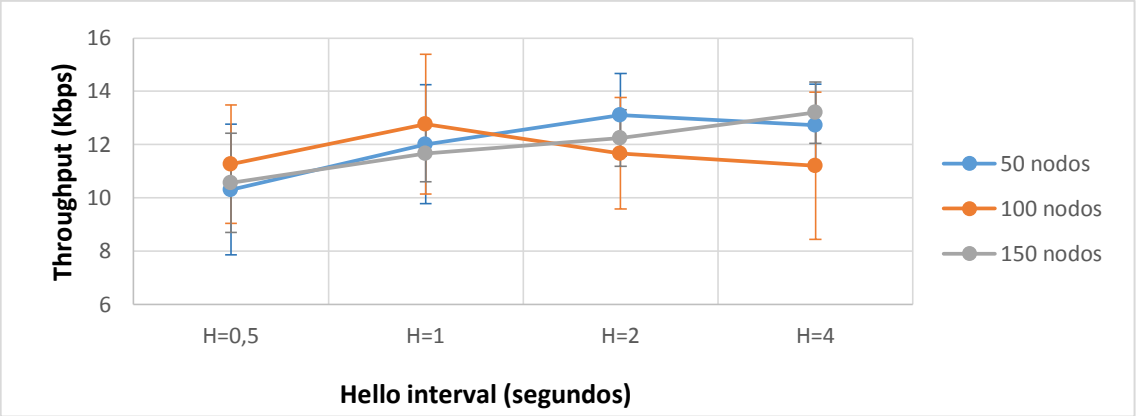
En este estudio técnico se analiza cómo influye cada uno de los parámetros de configuración de OLSR dentro de la eficiencia de este protocolo; por lo cual, existe la necesidad de contrastar los datos del comportamiento por defecto con los resultados obtenidos en esta sección.

A continuación, se presentan los parámetros que se evaluaron para conseguir una mejora en la eficiencia del protocolo OLSR.

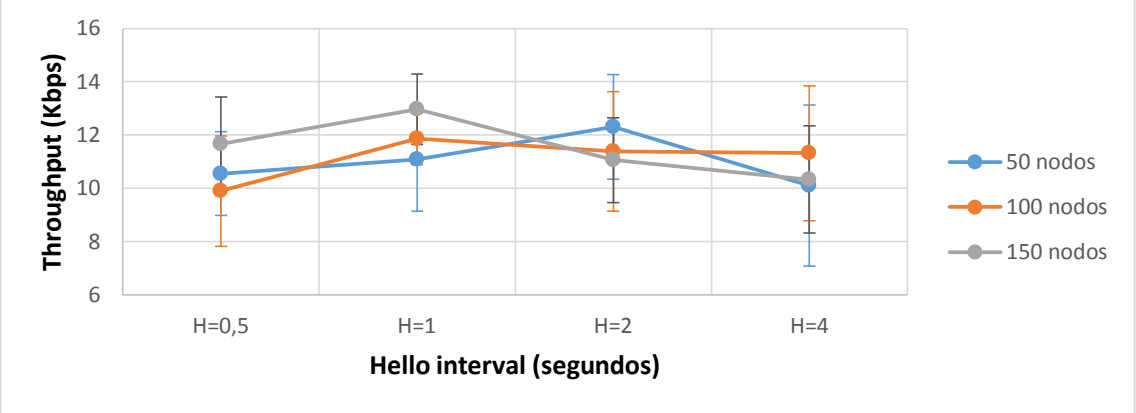
3.2.1 VARIACIÓN HELLO INTERVAL

El Hello_Interval (H) especifica el tiempo en el cual se envían los mensajes de hello; los cuales sirven para obtener información de los vecinos; además, los mensajes hello se utilizan para encontrar la información sobre el estado del vínculo y los vecinos del host. Por tanto, en esta variación se pretende conocer el valor más adecuado para optimizar el envío de estos mensajes; su valor por defecto es 2 segundos, se realizaron pruebas con los valores 0.5, 1, 2 y 4 segundos para este parámetro.

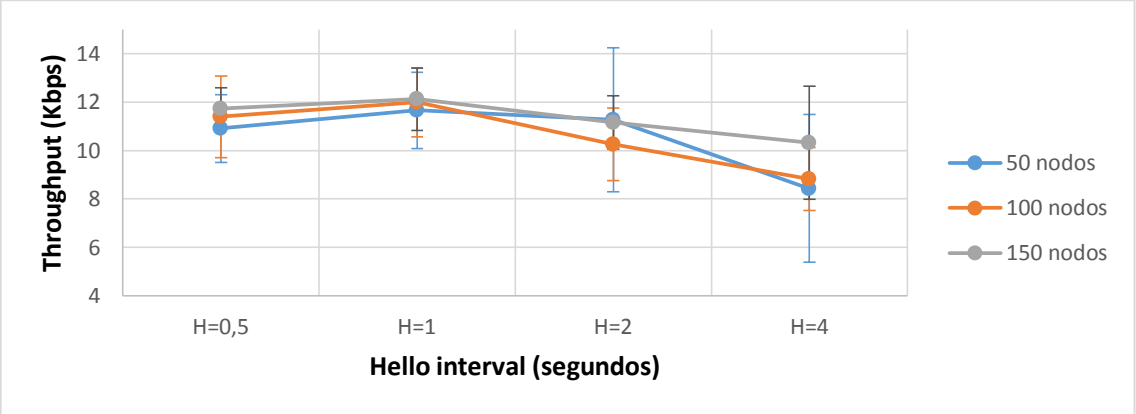
La figura 3.2 indica el *throughput* vs H para las tres densidades y tres velocidades (desplazamiento) de los nodos.



a) Throughput vs H para tres densidades de nodos a la velocidad de 1.5 m/s



b) Throughput vs H para tres densidades de nodos a la velocidad de 3 m/s



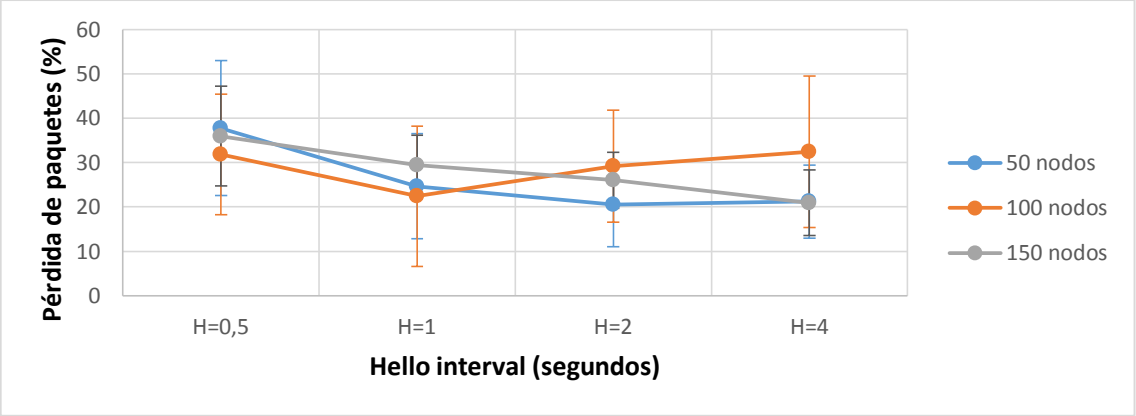
c) Throughput vs H para tres densidades de nodos a la velocidad de 5 m/s

Figura 3.2. *Throughput vs Hello_Interval* para tres densidades de nodos y tres velocidades de desplazamiento de nodos.

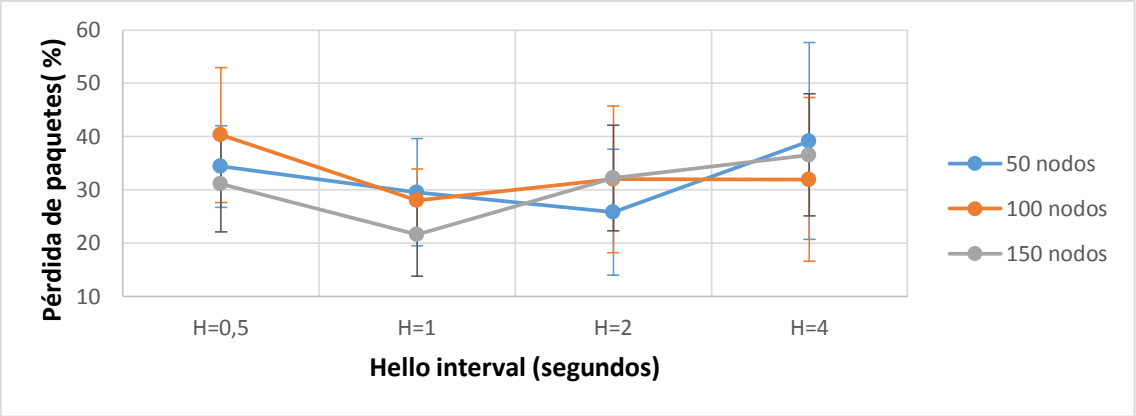
De la figura 3.2, se observa que el mejor throughput se obtiene cuando el valor de Hello_Interval es igual a 1 (en las velocidades de 3 y 5 m/s), es decir, un segundo por debajo del tiempo establecido en su RFC. De esta manera, la modificación de este parámetro hace que el protocolo envié mensajes Hello de forma más rápida para que de tal forma pueda descubrir a sus vecinos y así seleccionar cuál de ellos será seleccionado como MRP. También se observa que el mayor throughput se obtiene cuando disminuye la velocidad de desplazamiento de los nodos (1.5 m/s) lo que tiene sentido ya que una velocidad menor indica menor cambio de topología y por lo tanto menos ruptura de rutas en la red.

Cabe recalcar que, por ejemplo, en la gráfica c), estadísticamente se obtuvo que a la velocidad de desplazamiento de 5 m/s y a una densidad de 150 nodos todas las variaciones sean iguales. Debido a que este tiempo influye en que tan rápido un nodo anuncia sus vecinos a otros nodos y al haber mayor número de nodos a una velocidad muy alta se pueden perder paquetes Hello en este proceso. Por lo cual para este caso se escogería un Hello Interval más alto porque involucra menos intercambios de mensajes.

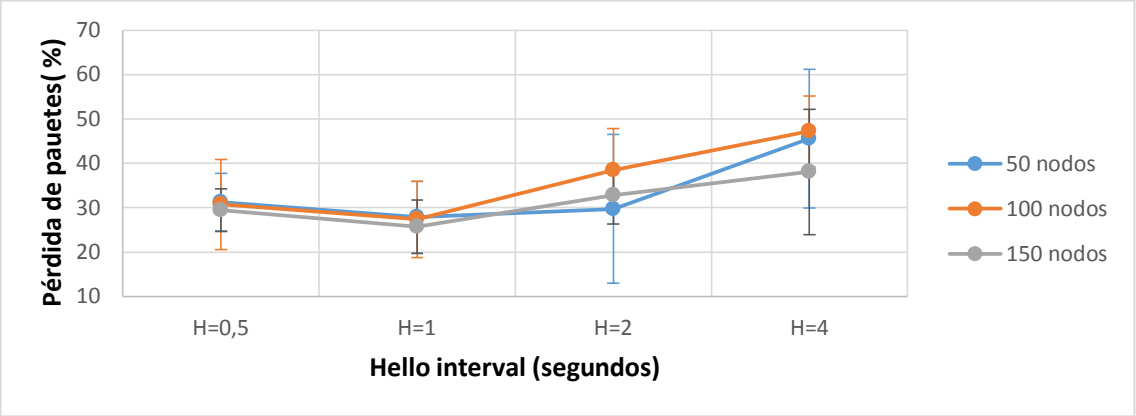
La figura 3.3 indica la tasa de entrega de paquetes vs H para las tres densidades y tres velocidades (desplazamiento) de los nodos.



a) Tasa de entrega de paquetes vs H para tres densidades de nodos a la velocidad de 1.5 m/s



b) Tasa de entrega de paquetes vs H para tres densidades de nodos a la velocidad de 3 m/s

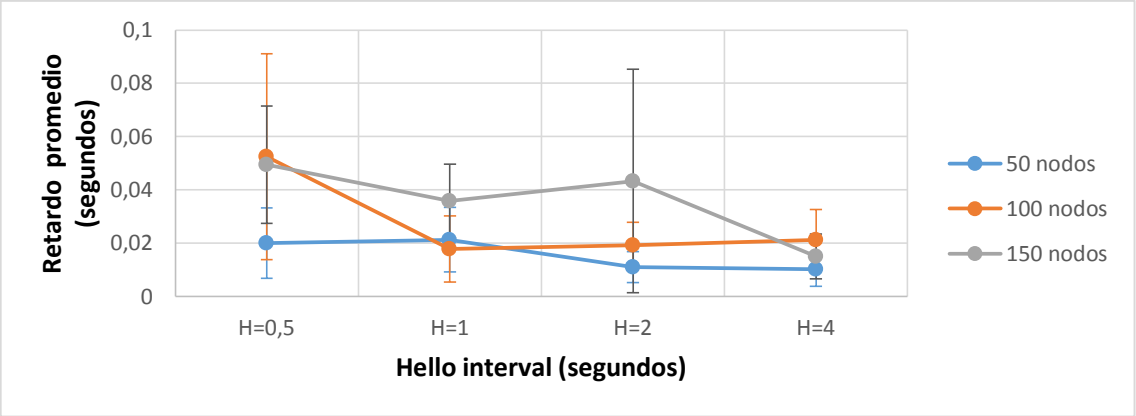


c) Tasa de entrega de paquetes vs H para tres densidades de nodos a la velocidad de 5 m/s

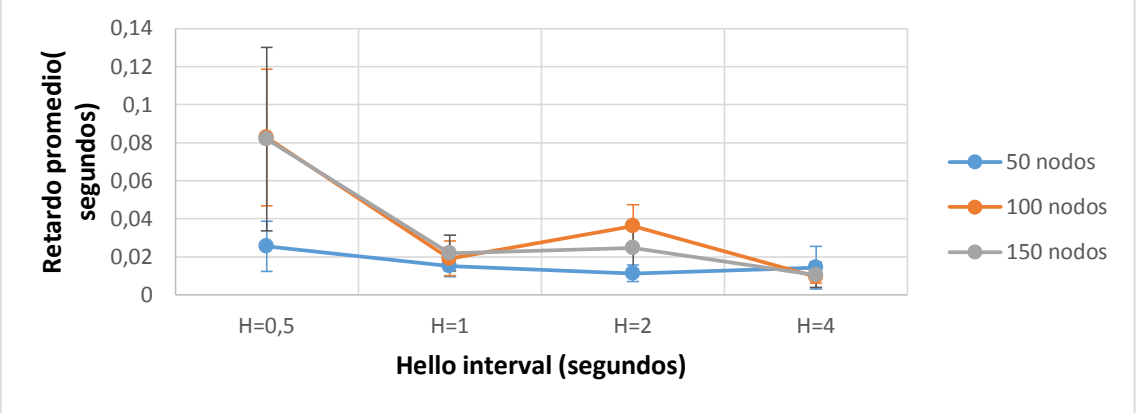
Figura 3.3. Tasa de entrega de paquetes vs Hello_Interval para tres densidades de nodos y tres velocidades de desplazamiento de nodos.

A partir de la figura 3.3 se puede establecer que un menor tiempo de envío de mensajes Hello existen menos pérdidas a bajas velocidades y a medida que se aumenta la velocidad las pérdidas también lo hacen, pero siguen siendo menores a las obtenidas con el valor por defecto (2 segundos). Para una densidad de 150 nodos y al valor de H=1 se registra una baja pérdida en comparación al obtenido cuando H=2.

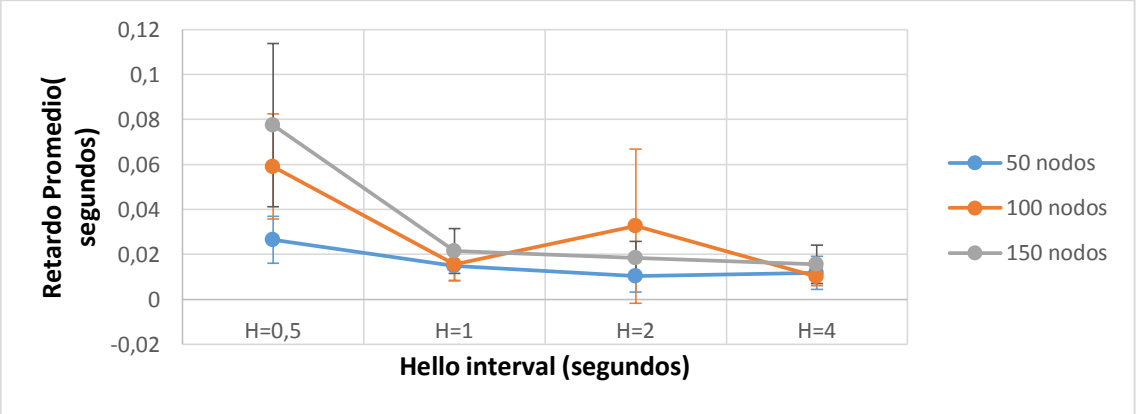
La figura 3.4 indica el retardo promedio vs H para las tres densidades y tres velocidades (desplazamiento) de los nodos.



a) Retardo promedio vs H para tres densidades de nodos a la velocidad de 1.5 m/s



b) Retardo promedio vs H para tres densidades de nodos a la velocidad de 3 m/s



c) Retardo promedio vs H para tres densidades de nodos a la velocidad de 5 m/s

Figura 3.4. Tasa de entrega de paquetes *vs* Hello_Interval para tres densidades de nodos y tres velocidades de desplazamiento de nodos.

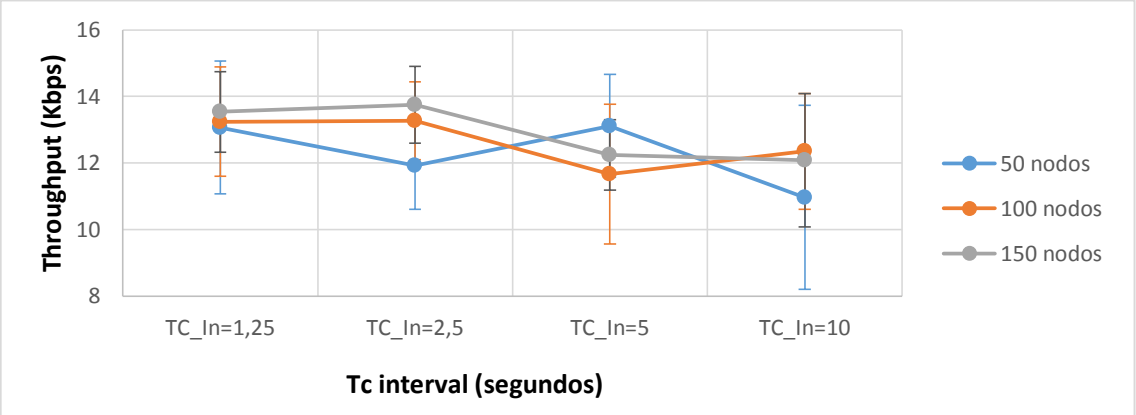
De la figura 3.4, de forma general se puede observar que el valor por defecto de 2 segundos genera menos retardo para todas las velocidades. Con el valor de 1 segundo se obtienen un poco más de retardo, pero se aumenta el throughput y disminuye la pérdida de paquetes para altas velocidades de desplazamiento de los nodos.

3.2.2 VARIACIÓN TC INTERVAL

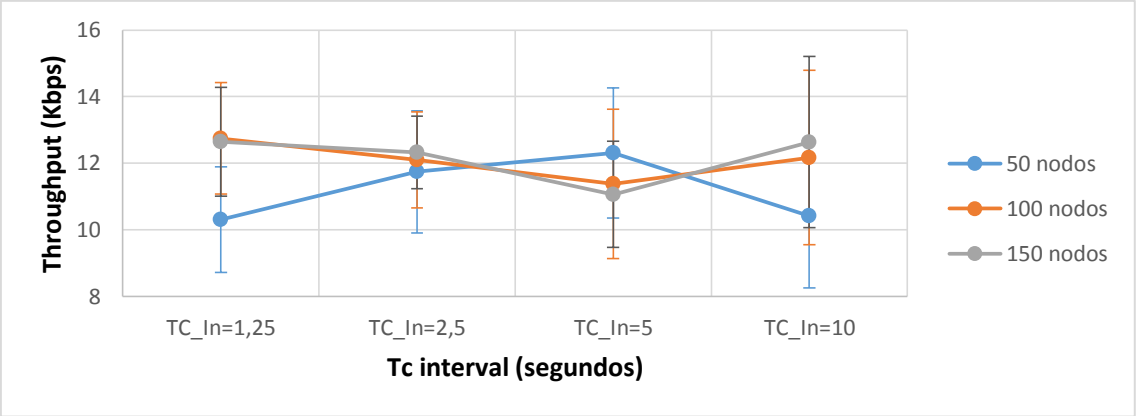
El parámetro *TC_Interval* (TC_In) representa la frecuencia con la que se genera un mensaje TC. En OLSR, la tasa de actualizaciones de estado topológico es la tasa de envío de mensajes TC. Los mensajes de TC son una de las mayores fuentes de overhead en OLSR, ya que están inundadas en toda la red, pero son esenciales para mantener conocimiento de conectividad consistente de la red completa.

Su valor por defecto es de 5 segundos, por lo que se realizaron pruebas con los valores de 1.25, 2.5 y 10 segundos respectivamente.

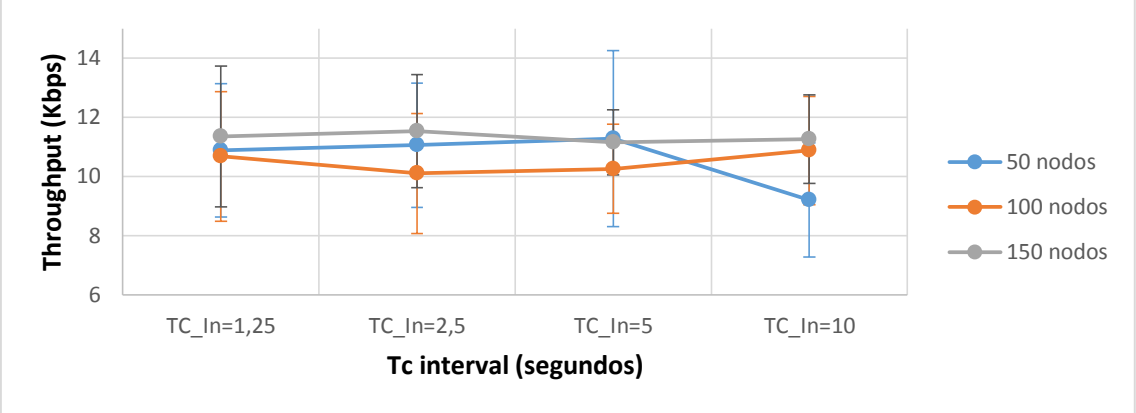
La figura 3.5 presenta el throughput vs TC_In para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos.



a) Throughput vs TC_In para tres densidades de nodos a la velocidad de 1.5 m/s



b) Throughput vs TC_In para tres densidades de nodos a la velocidad de 3 m/s



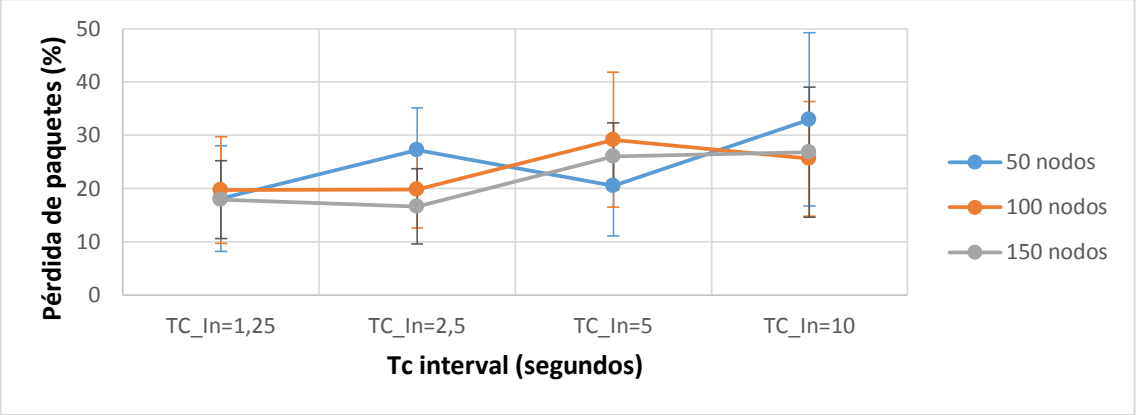
c) Throughput vs TC_In para tres densidades de nodos a la velocidad de 5 m/s

Figura 3.5. Throughput vs TC_In para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos.

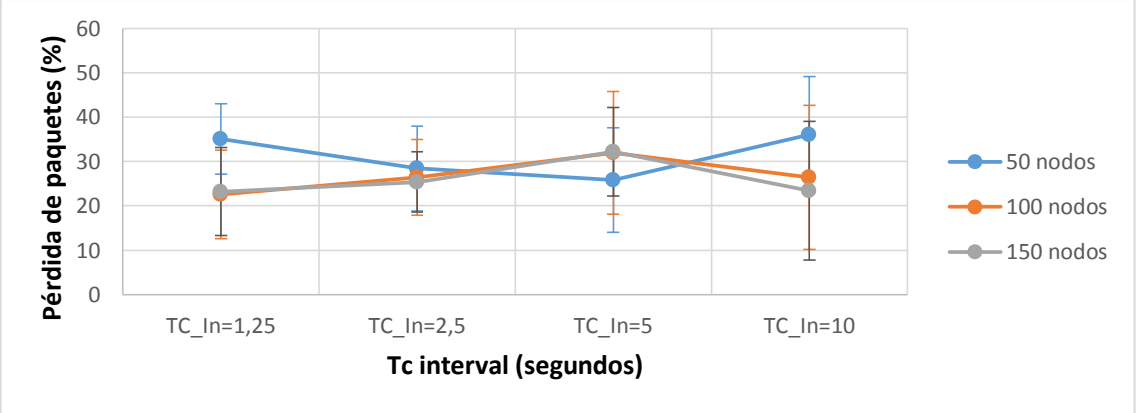
La figura 3.5 expone como varía el throughput a medida que se disminuye o aumenta este parámetro, se puede apreciar que el mayor throughput se obtiene cuando se disminuye el valor de dicho parámetro a la mitad, ya que; los mensajes TC se generan y retransmiten para inundar de información topológica en toda la red solo a través de nodos MRP, y al reducir el tiempo de este intervalo estos mensajes proporcionaran a cada nodo en la red información de estado de enlace de forma más rápida para permitir el cálculo de la ruta.

A medida que incrementa la velocidad de los nodos, el throughput disminuye debido a que a mayor velocidad de desplazamiento de los nodos y menor tiempo (TC_Interval) se pueden perder paquetes por el corto período y el nodo MRP no proporcionaría la información suficiente para realizar el cálculo de la ruta.

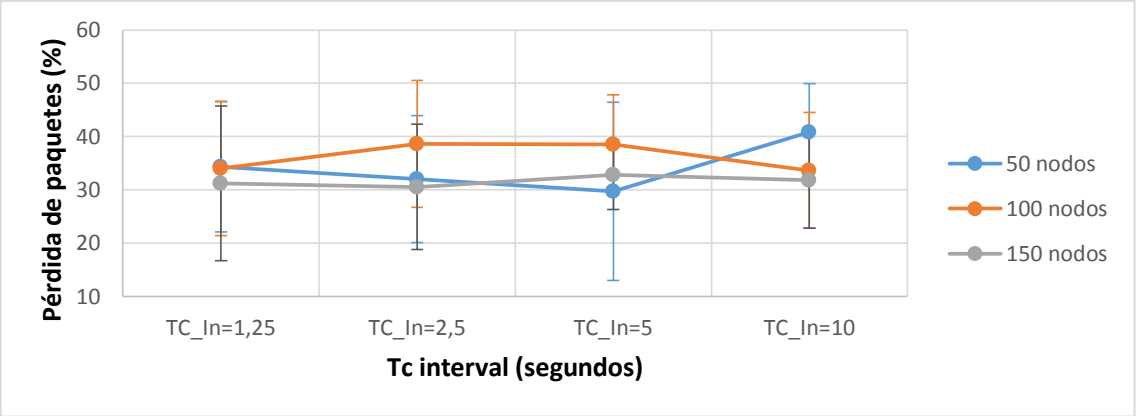
La figura 3.6 indica la tasa de entrega de paquetes vs TC_Interval para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos.



a) Tasa de entrega de paquetes vs TC_In para tres densidades de nodos a la velocidad de 1.5 m/s



b) Tasa de entrega de paquetes vs TC_In para tres densidades de nodos a la velocidad de 3 m/s

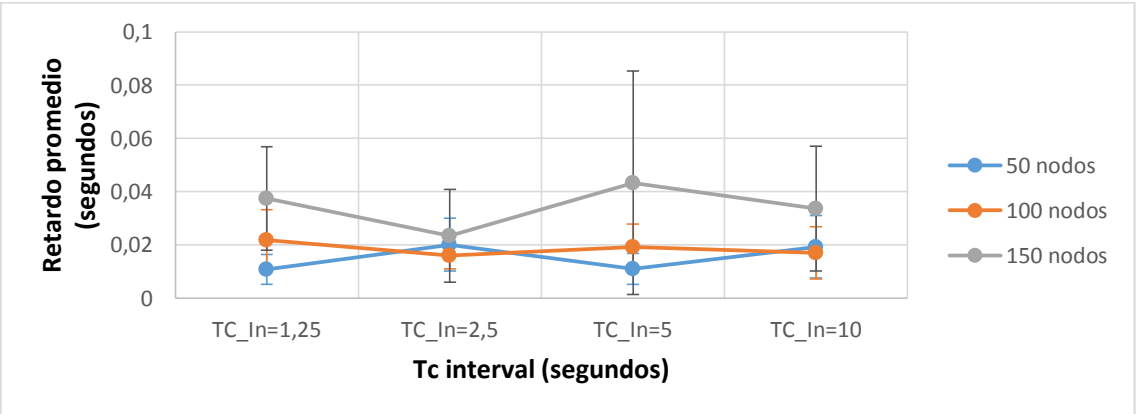


c) Tasa de entrega de paquetes vs TC_In para tres densidades de nodos a la velocidad de 5 m/s

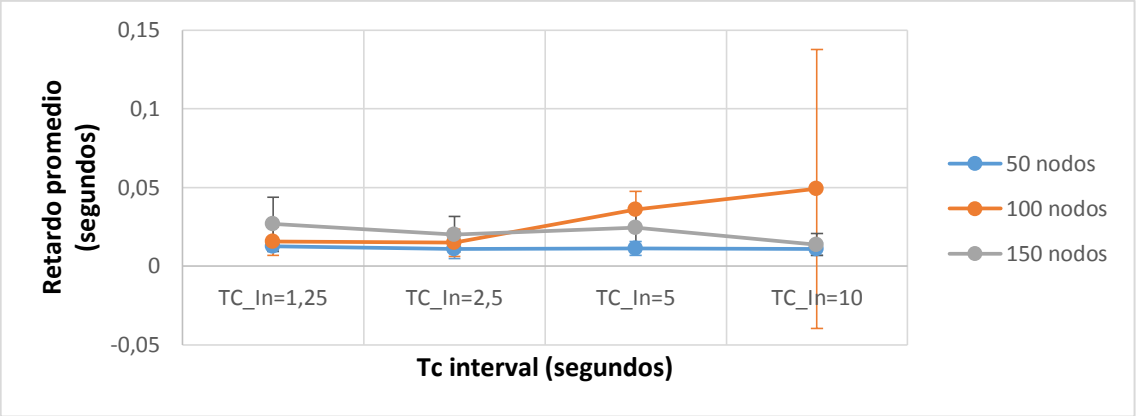
Figura 3.6. Indica la tasa de entrega de paquetes vs TC_In para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos.

En la figura 3.6 se muestra la pérdida de paquetes, se aprecia que disminuyendo el intervalo TC a la mitad se obtiene menor pérdida a velocidades bajas de desplazamiento de nodos.

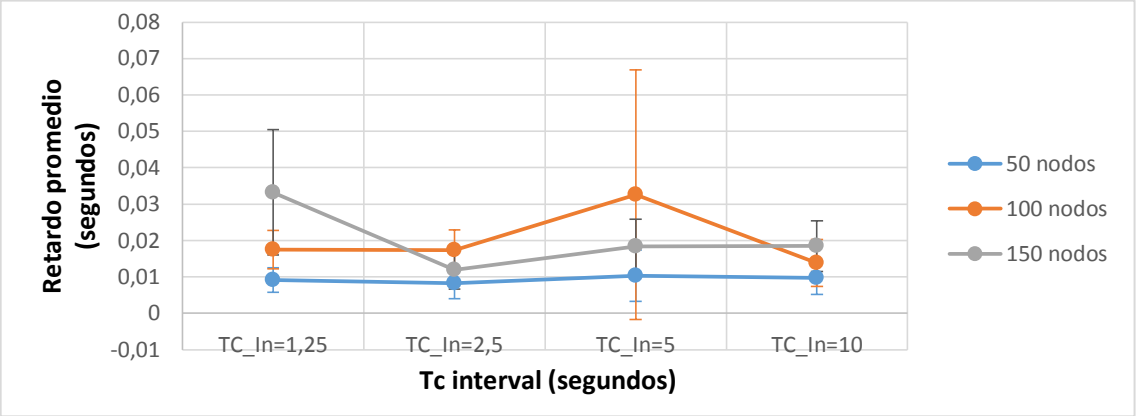
La figura 3.7 indica el retardo promedio vs TC_In para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos. En la misma se puede apreciar que el valor de 2.5 segundos es el óptimo para mejorar este parámetro, si bien estadísticamente los valores de variaciones considerados son iguales, este valor sobresale debido a que se obtiene un mayor throughput y menores pérdidas a diferencia de los demás valores.



a) Retardo promedio vs TC_In para tres densidades de nodos a la velocidad de 1.5 m/s



b) Retardo promedio vs TC_In para tres densidades de nodos a la velocidad de 3 m/s



c) Retardo promedio vs TC_In para tres densidades de nodos a la velocidad de 5 m/s

Figura 3.7. Indica el retardo promedio vs TC_In para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos.

3.2.3 VARIACIÓN HNA INTERVAL

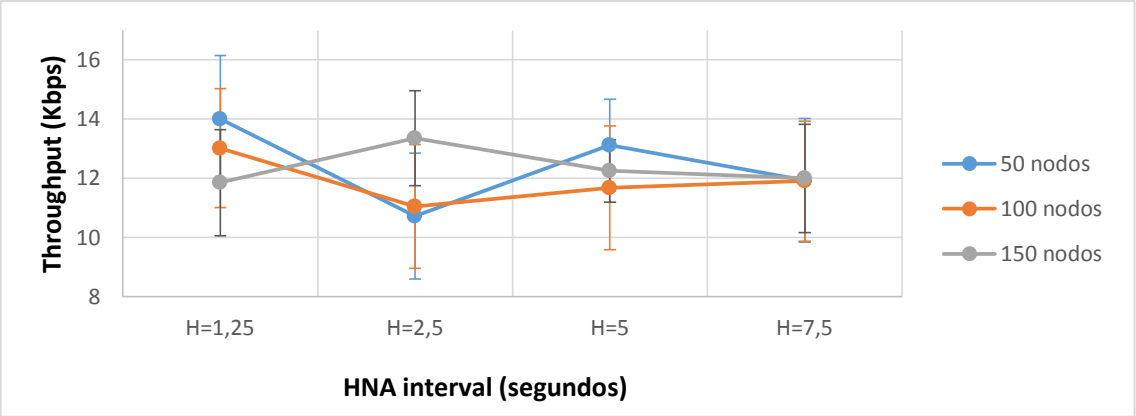
El intervalo HNA especifica el tiempo entre dos mensajes HNA consecutivos. Los mensajes HNA serán enviados solo cuando un nodo OLSR tenga una interfaz no OLSR. Por ejemplo: un nodo puede estar equipado con varias interfaces, algunas de las cuales no participan en la MANET OLSR. Estas interfaces no-OLSR pueden ser conexiones punto a punto a otros hosts o pueden conectarse a redes separadas. Para proporcionar conexión desde la interfaz MANET OLSR a esta interfaz no-OLSR, un nodo debe poder inyectar información de ruta externa a la MANET OLSR.

El mensaje "Host and Network Association" (HNA) proporciona la información de ruteo externa dando la posibilidad de rutear direcciones externas. El mensaje HNA proporciona

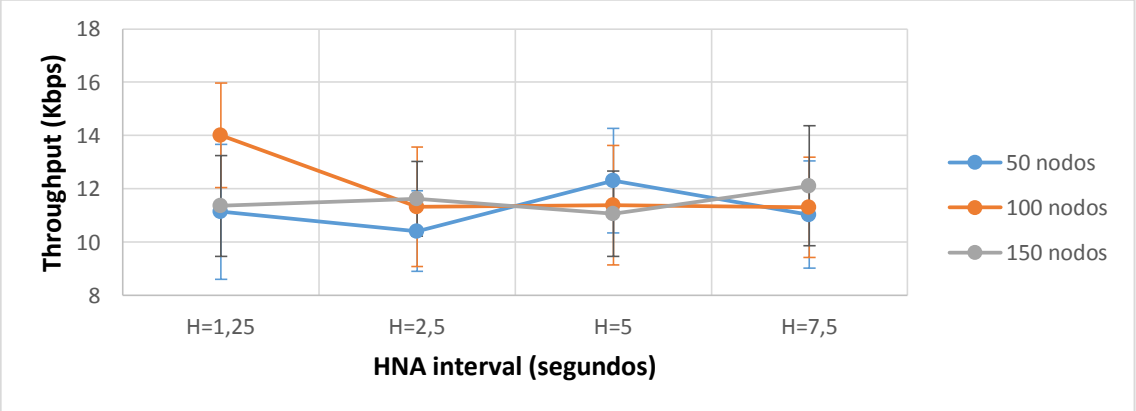
información sobre las direcciones de red y de máscara de red, de modo que el host OLSR pueda considerar que el host que anuncia puede actuar como puerta de enlace para anunciar un conjunto de direcciones.

Su valor por defecto es de 5 segundos, se consideraron los siguientes valores para realizar las pruebas: 1.25, 2.5, 5 y 7.5 segundos.

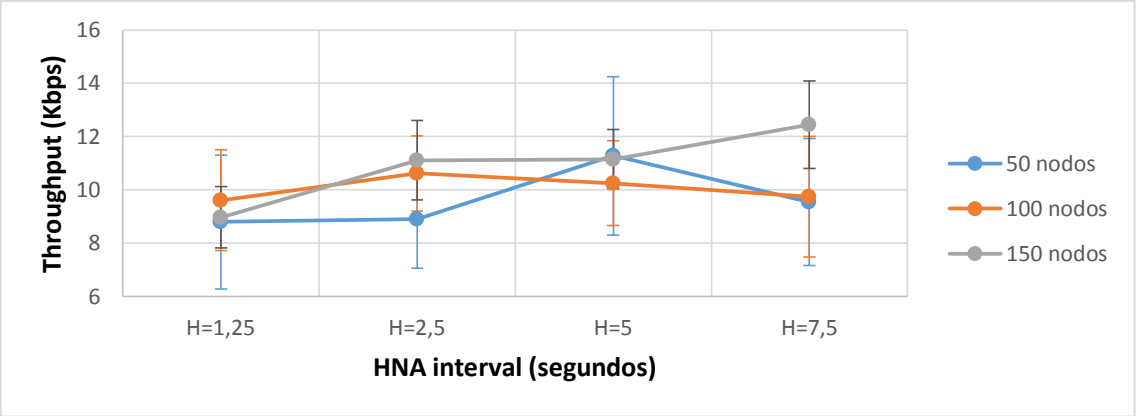
La figura 3.8 indica el throughput vs HNA para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos



a) Throughput vs HNA para tres densidades de nodos a la velocidad de 1.5 m/s



b) Throughput vs HNA para tres densidades de nodos a la velocidad de 3 m/s

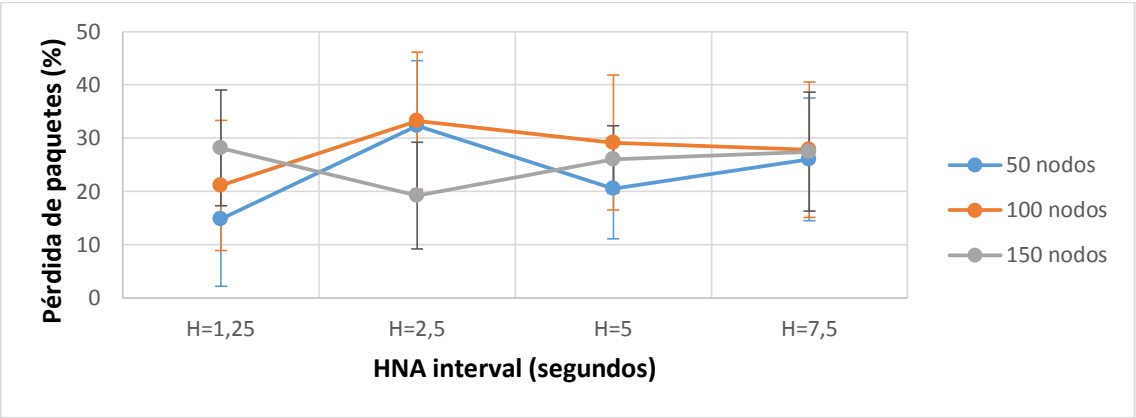


c) Throughput vs HNA para tres densidades de nodos a la velocidad de 5 m/s

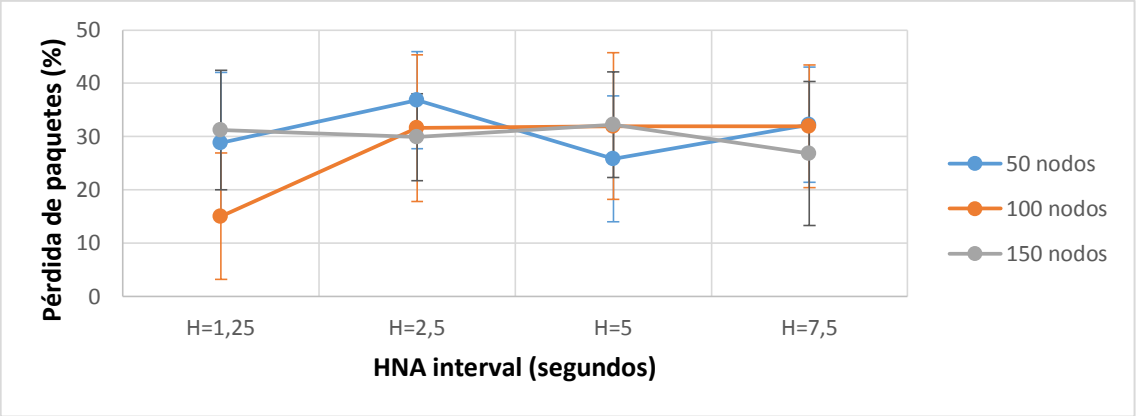
Figura 3.8. Indica el throughput vs HNA para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos.

En la figura 3.8 se destaca que el mayor throughput se obtiene cuando el parámetro toma un valor igual a 1.25 segundos (para las velocidades de 1.5 y 3 m/s) esto es debido a que permite a los nodos que poseen interfaces no OLSR anuncien de manera más rápida sus direcciones y máscaras de subred y así realizar un enrutamiento más eficiente y rápido. En la figura 3.8 c, el throughput decae con el valor mencionado anteriormente esto se debe a que los nodos se desplazan de forma más rápida por lo cual puede generar pérdida de mensajes.

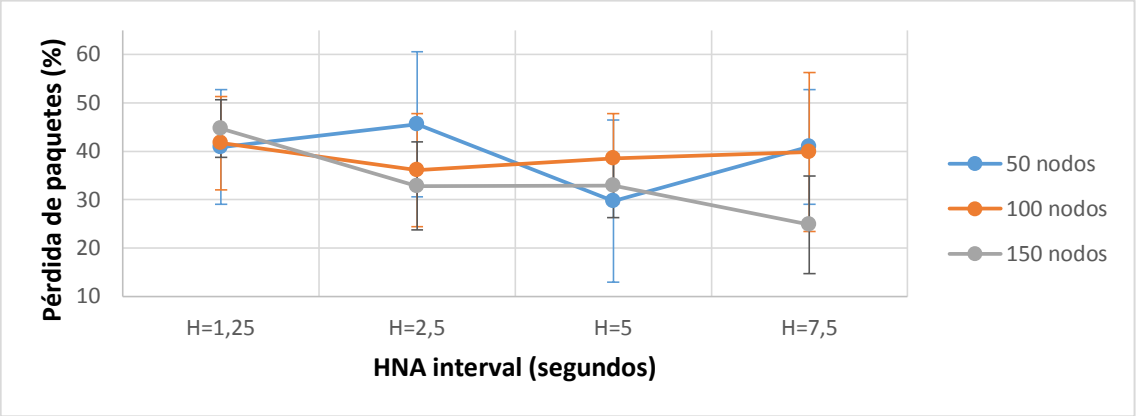
La figura 3.9 indica la tasa de entrega de paquetes vs HNA para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos.



a) Tasa de entrega de paquetes vs HNA para tres densidades de nodos a la velocidad de 1.5 m/s



b) Tasa de entrega de paquetes vs HNA para tres densidades de nodos a la velocidad de 3 m/s

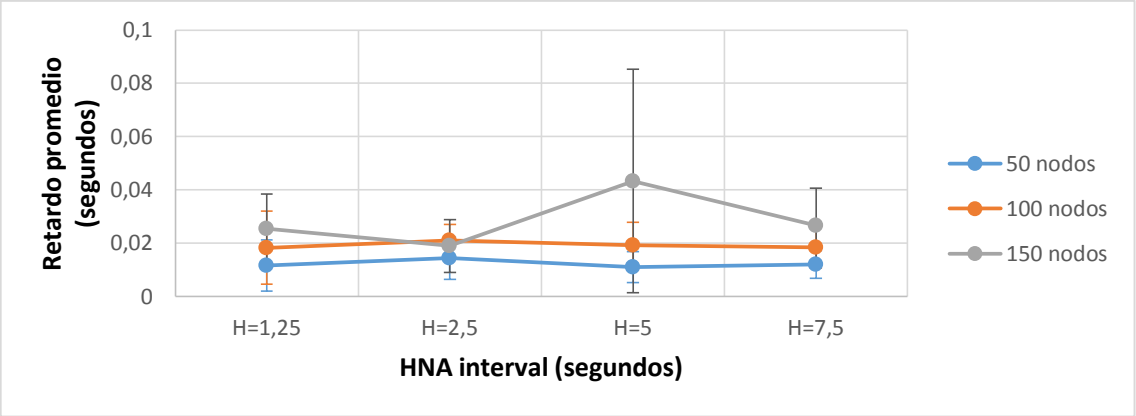


c) Tasa de entrega de paquetes vs HNA para tres densidades de nodos a la velocidad de 5 m/s

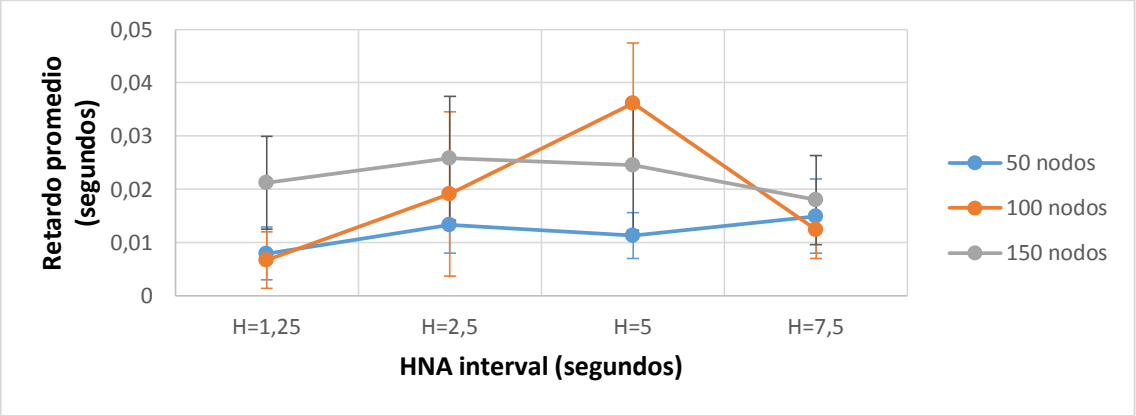
Figura 3.9. Tasa de entrega de paquetes vs HNA para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos.

De la misma manera que se pudo apreciar con el throughput, el valor con el que se obtiene menos pérdida de paquetes es el de 1.25 para velocidades bajas de desplazamiento de nodos. Para la velocidad de 5 m/s se observa un claro incremento en la tasa de entrega de paquetes, al estar a una mayor velocidad los mensajes HNA pueden tardar en llegar al destino lo cual haría pensar al nodo origen que la ruta es inalcanzable y podría descartarla.

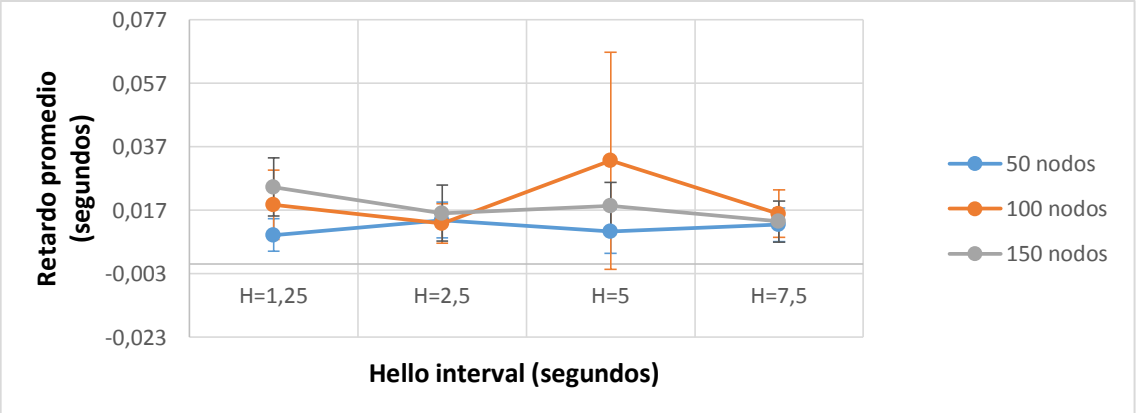
La figura 3.10 indica el retardo promedio vs HNA para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos.



a) Retardo promedio vs HNA para tres densidades de nodos a la velocidad de 1.5 m/s



b) Retardo promedio vs HNA para tres densidades de nodos a la velocidad de 3 m/s



c) Retardo promedio vs HNA para tres densidades de nodos a la velocidad de 5 m/s

Figura 3.10. Retardo promedio vs HNA para tres densidades de nodos y para tres velocidades de desplazamiento de los nodos.

En la figura 3.10 de forma general se puede destacar que al igual que con las dos métricas anteriores, el valor de 1.25 segundos genera menos retardo en el protocolo esto concuerda con los resultados obtenidos con el parámetro de pérdida de paquetes, aunque estadísticamente tomando en cuenta los intervalos de confianza usar cualquiera de los valores expuestos en la gráfica tiene el mismo resultado, pero al usar el valor de 1,25 se destaca que los paquetes se pueden entregar de manera más rápida haciendo que se puede identificar de mejor manera aquellos nodos que no son OLSR para agregarlos y utilizarlos al momento de realizar el cálculo de la ruta.

3.3 ANÁLISIS DE VARIACIÓN COMBINADA DE LOS PARÁMETROS

En la presente sección se busca identificar la combinación de parámetros óptimos del protocolo OLSR para diferentes escenarios MANET teniendo en cuenta los resultados obtenidos en las secciones previas.

Este análisis combinará los parámetros, los cuales, al ser analizados de forma individual, influyen de forma significativa en la eficiencia del protocolo OLSR. Los parámetros mencionados son: *Hello_Interval, TC_Interval, Mid_Interval, HNA_Interval*.

Las combinaciones que se presentan posteriormente se realizaron para cada velocidad de desplazamiento de los nodos, es decir, para las velocidades 1.5, 3 y 5 (m/s).

3.3.1 PRIMERA VARIACIÓN COMBINADA

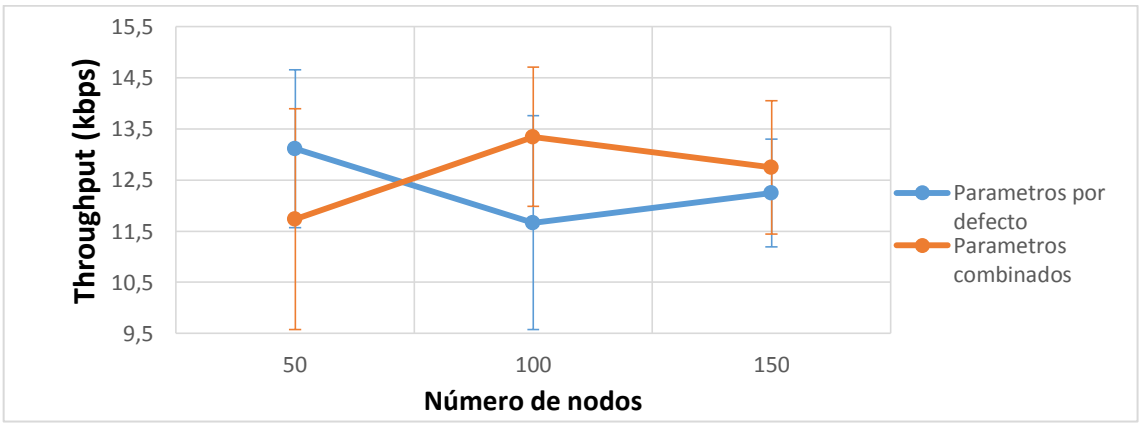
Esta combinación se propone debido a que se ha visualizado un incremento en el throughput cuando el valor del Hello_Interval es igual a 1 y TC_Interval igual a 2,5.

Esta combinación presenta un tiempo menor (Hello_Interval) en el cual se realizará el intercambio de mansajes Hello, estos permiten reducir la cantidad de tráfico de enrutamiento y el tiempo para registrar un nodo en la lista de vecinos. Dado que el valor de 2 segundos puede ser muy alto cuando se maneja densidades de nodos muy altas el retardo puede ser muy significativo y no se puede alcanzar el nodo destino lo cual generara una mayor pérdida de paquetes.

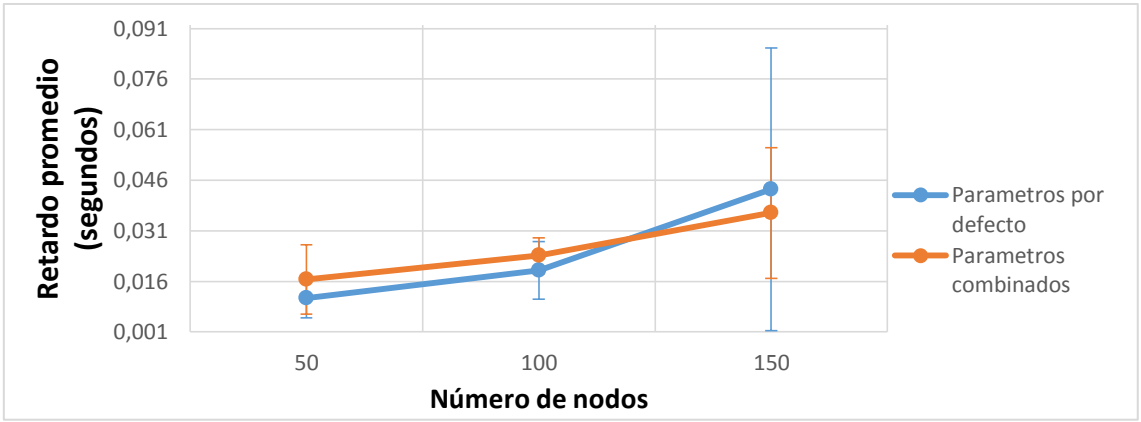
También se presenta un periodo de tiempo menor (TC_Interval) para la entrega de mensajes TC los cuales permiten transportar información de topología y conectividad. Estos dos valores funcionan mejor a velocidades bajas (1.5 y 3 m/s) y no degradan (no perjudican) en velocidades altas (5 m/s); por esta razón, se analizó esta variación con

esas tres velocidades y con las tres densidades de nodos (50, 100 y 150); esto se puede observar en las figuras 3.11 – 3.13.

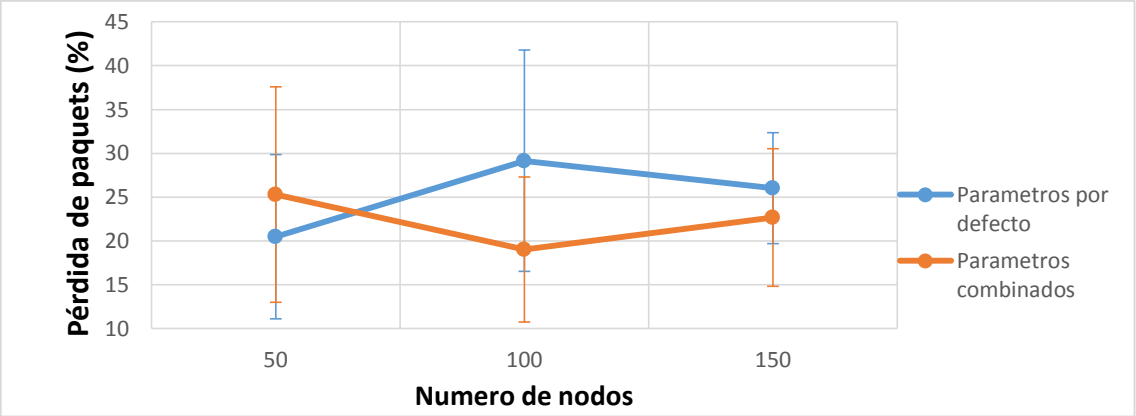
Las Figuras 3.11 – 3.13 indican el desempeño de las simulaciones realizadas considerando estas variaciones.



a) Throughput



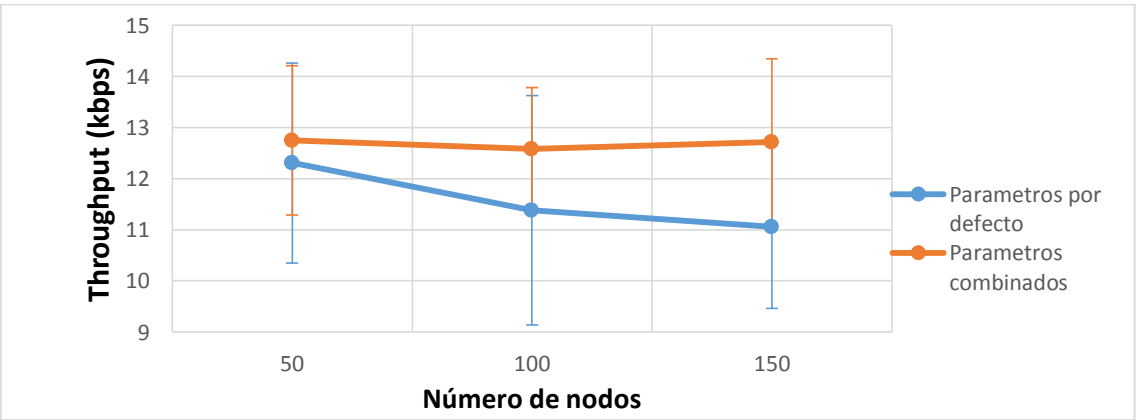
b) Retardo promedio



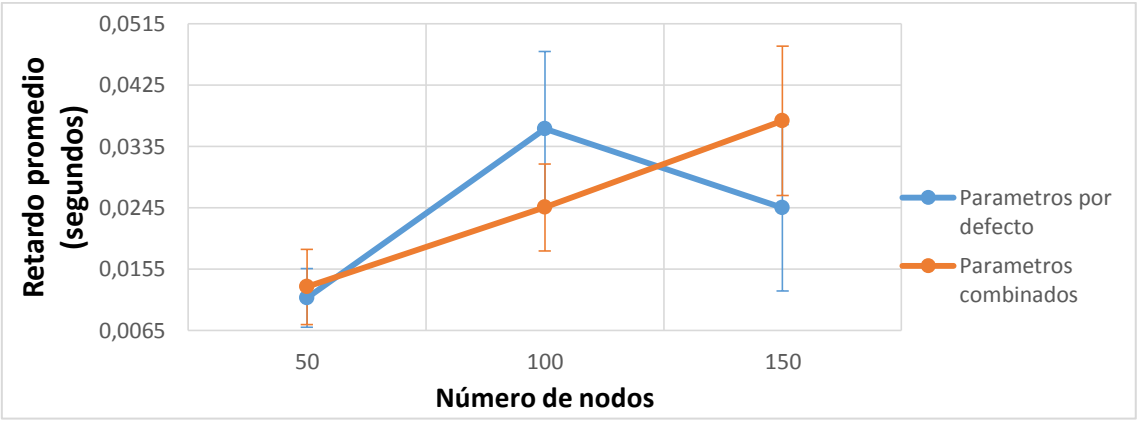
c) Pérdida de paquetes

Figura 3.11. Métricas con parámetros combinados y por defecto, para tres densidades de nodos a una velocidad de 1.5 m/s.

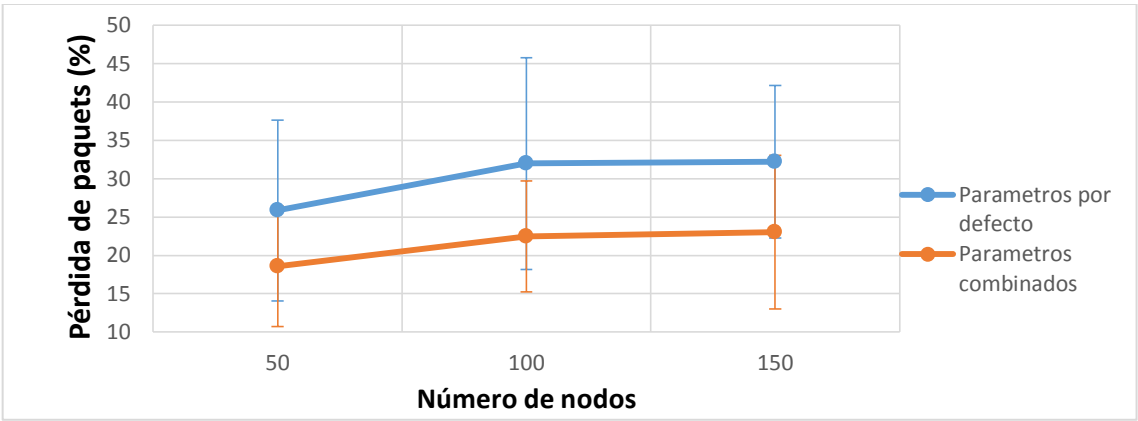
La figura 3.11 muestra cómo trabaja el protocolo OLSR con sus parámetros combinados a una velocidad de 1.5 m/s, analizando los resultados se observa que se obtiene mayor eficiencia en comparándola con sus valores por defecto; esto es, se obtiene mayor throughput y menos pérdida de paquetes. El retardo que se puede alcanzar con esta combinación no es muy significativo al obtenido con sus parámetros por defecto; se puede obtener más retardo, pero se gana mayor throughput en la red.



a) Throughput



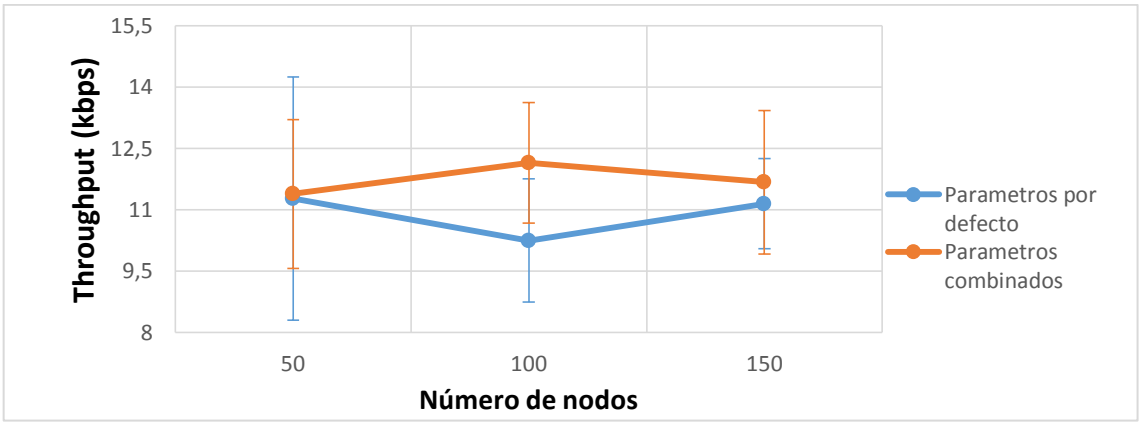
b) Retardo promedio



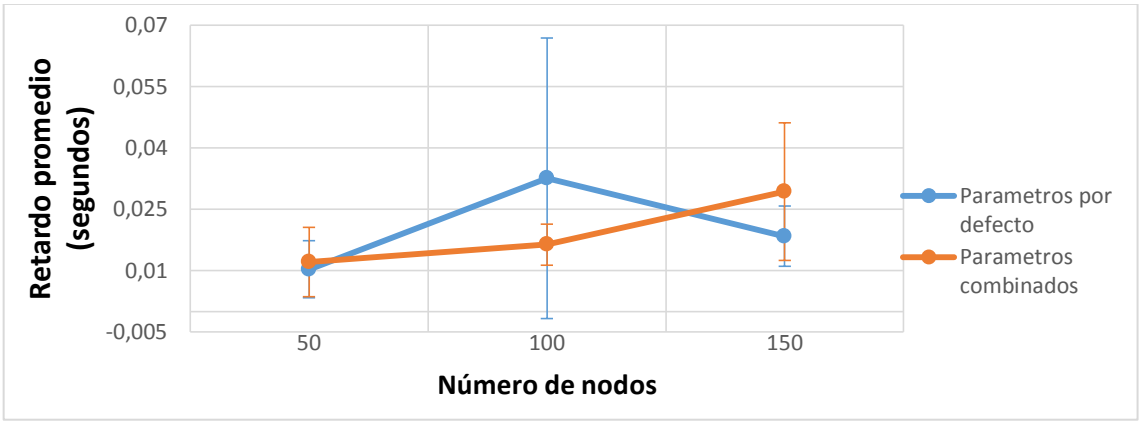
c) Pérdida de paquetes

Figura 3.12. Métricas con parámetros combinados y por defecto, para tres densidades de nodos a una velocidad de 3 m/s.

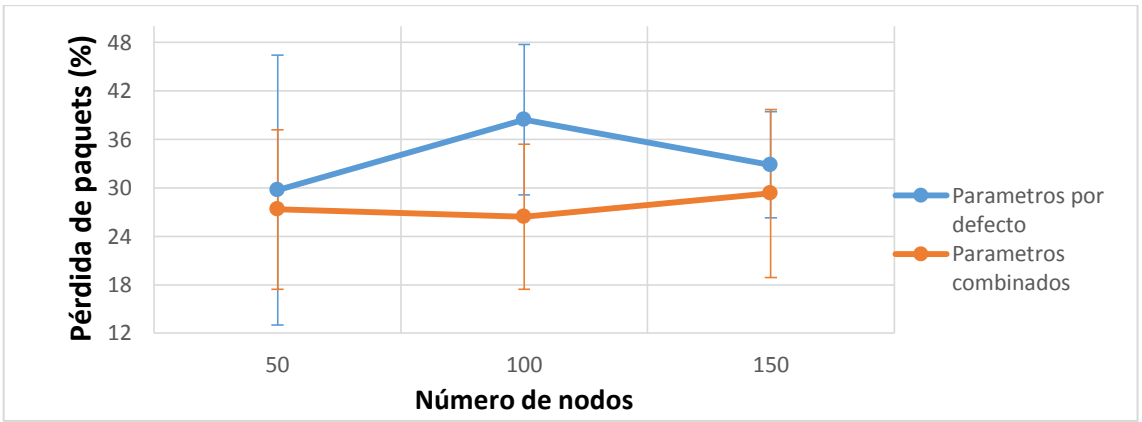
En la figura 3.12 se observa que se obtiene un incremento de throughput y las pérdidas de paquetes disminuyen; como en el análisis anterior a una velocidad de 1.5 m/s, esta combinación de parámetros aumenta la eficiencia del protocolo.



a) Throughput



b) Retardo promedio



c) Pérdida de paquetes

Figura 3.13. Métricas con parámetros combinados y por defecto, para tres densidades de nodos a una velocidad de 5 m/s**.**

En la figura 3.13 se observa el desempeño del protocolo OLSR con sus parámetros por defecto vs los parámetros combinados, se observa que al igual que el caso expuesto en la figura 3.11 si se opta por una densidad de nodos mucho mayor, el desempeño del protocolo decae, aunque se seguirá obteniendo una mayor eficiencia que la obtenida con los valores por defecto.

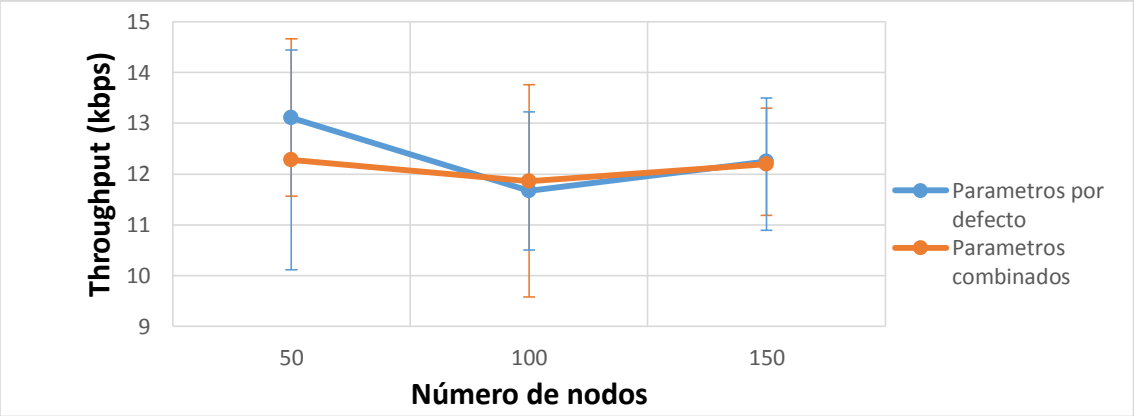
3.3.2 SEGUNDA VARIACIÓN COMBINADA

Esta combinación es similar a la presentada anteriormente, con la diferencia que se añaden dos parámetros para realizar esta combinación los cuales son: Mid_Interval y HNA_Interval. Esta combinación presenta un valor por debajo (1 segundo) del de defecto del Mid_Interval, el cual reducirá el envío de los mensajes MID; los mensajes MID sirven para declarar múltiples interfaces (si las hay) del nodo, es decir; la lista de direcciones de interfaz que están asociadas a su dominio principal. El análisis de todos los mensajes MID debe completarse dentro de un cierto período de actualización (Mid_Interval). Dicho esto, un valor menor de este intervalo permitirá al protocolo declarar de manera más rápida las interfaces de los nodos; además, la información utilizada en la red por los mensajes MID ayudará a cada nodo a calcular su tabla de enrutamiento de manera más rápida.

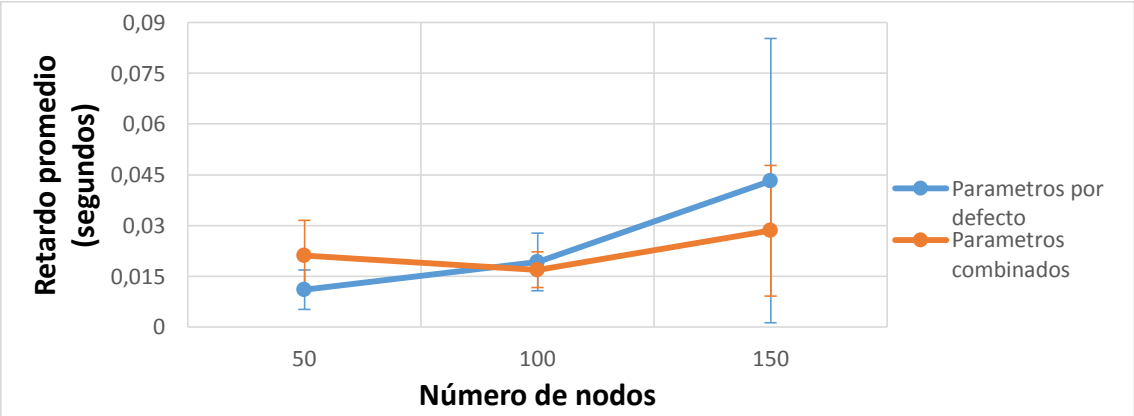
También se analiza esta combinación junto a un valor de 1.25 segundos para el HNA_Interval, un nodo con hosts y/o redes asociadas debe generar periódicamente un mensaje de Host and Network Association (HNA), que contenga pares de (dirección de red, máscara de red) correspondientes a los hosts y redes conectados. Los mensajes HNA deben transmitirse periódicamente cada HNA_Interval. Por esta razón se propone un tiempo menor al de por defecto, esto permitirá que los nodos pueden intercambiar información sobre redes externas de manera más rápida y así ayudar a calcular la tabla de enrutamiento.

Este conjunto de valores funciona mejor a velocidades altas (3 y 5 m/s) y no degradan (no perjudican) en velocidades bajas (1.5 m/s); por esta razón, se analizó esta variación con esas tres velocidades y con las tres densidades de nodos (50, 100 y 150).

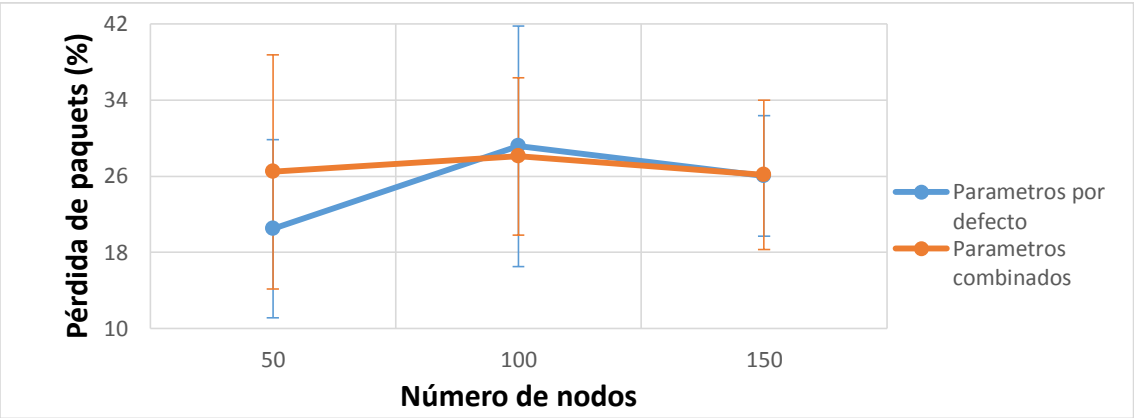
Las Figuras 3.14 – 3.16 indican el desempeño de las simulaciones realizadas considerando estas variaciones con sus respectivos análisis.



a) Throughput



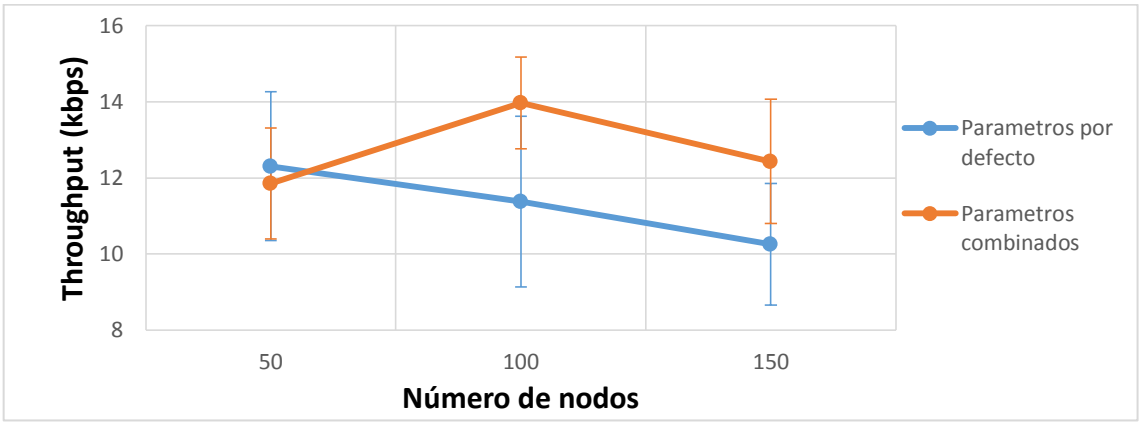
b) Retardo promedio



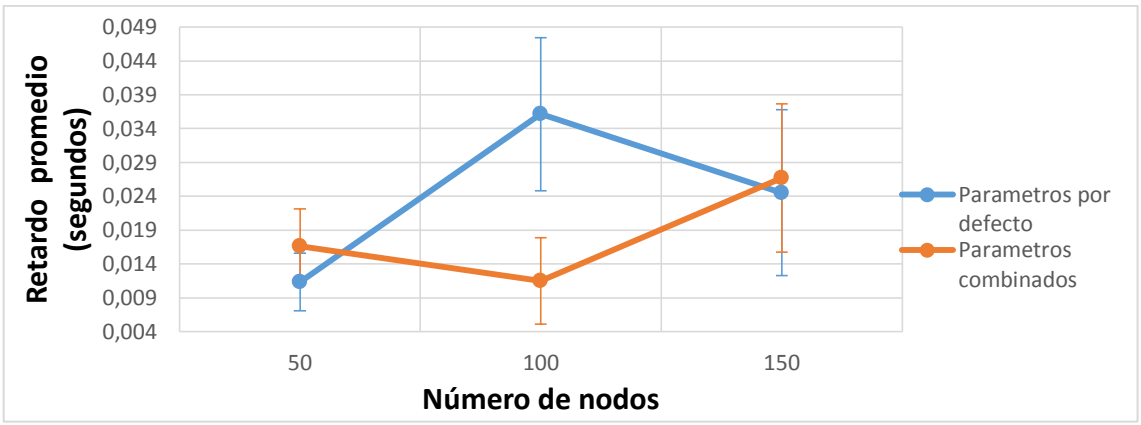
c) Pérdida de paquetes

Figura 3.14. Métricas con parámetros combinados y por defecto, para tres densidades de nodos a una velocidad de 1.5 m/s.

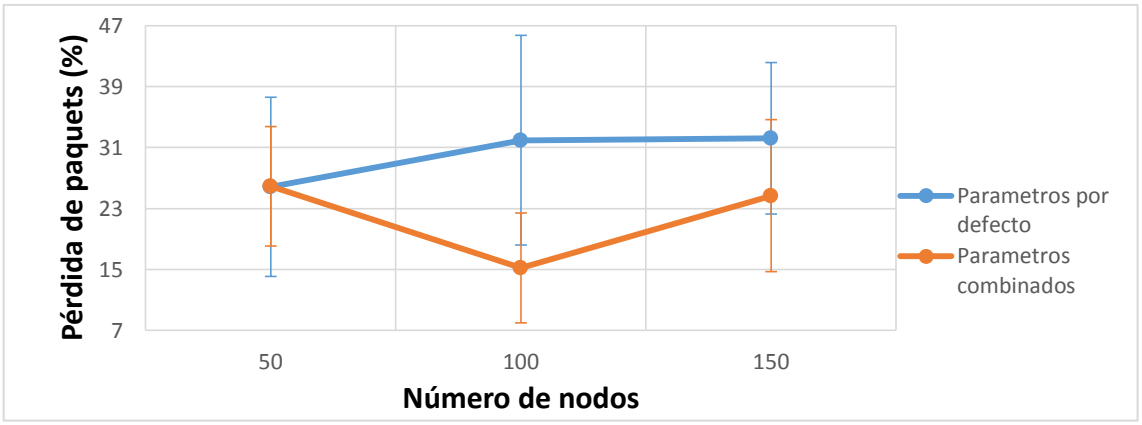
Al observar la figura 3.14 se puede apreciar que no existe un cambio significativo en los resultados con respecto a los obtenidos con sus valores por defecto, de hecho, el retardo obtenido con los parámetros por defecto aumenta en comparación al obtenido con los parámetros combinados al igual que la pérdida de paquetes. Esto se puede dar debido a que al reducir el tiempo de HNA_Interval no permite que los nodos envíen toda la información sobre las redes externas que pueden existir en sus interfaces.



a) Throughput

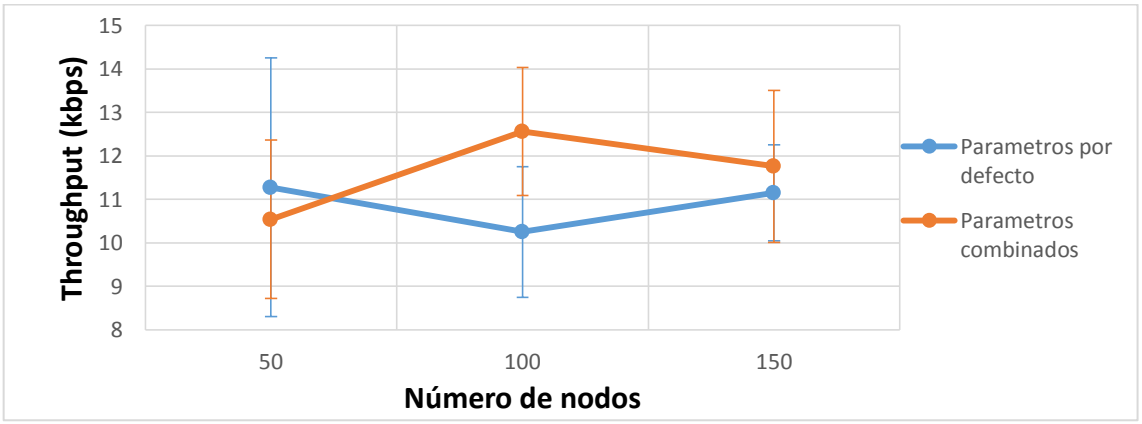


b) Retardo promedio

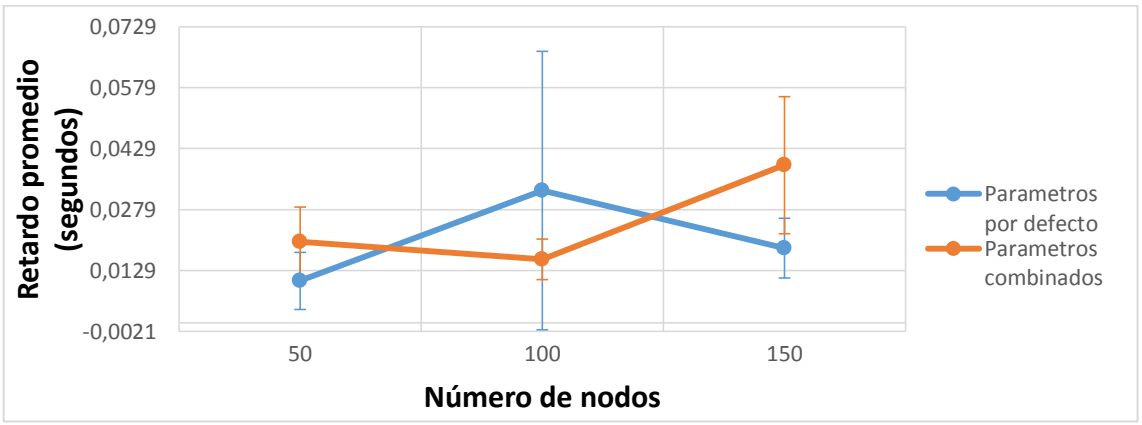


c) Pérdida de paquetes

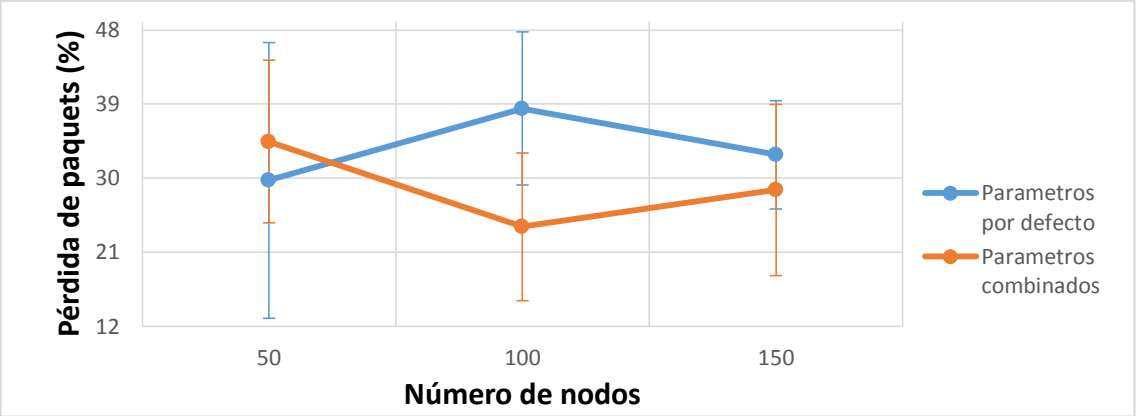
Figura 3.15. Métricas con parámetros combinados y por defecto, para tres densidades de nodos a una velocidad de 3 m/s.



a) Throughput



b) Retardo promedio



c) Pérdida de paquetes

Figura 3.16. Métricas con parámetros combinados y por defecto, para tres densidades de nodos a una velocidad de 5 m/s**.**

En este análisis se decidió hacerlo en conjunto con las gráficas de las figuras 3.15 y 3.16 ya que los resultados obtenidos en dichas graficas son muy similares. Los resultados obtenidos con los parámetros combinados son mejores a los obtenidos con los de defecto. El throughput es sumamente mayor al obtenido, las pérdidas de paquetes y el retardo son significativamente menores, los resultados demuestran que la eficiencia del protocolo mejora al usar esta combinación a altas velocidades (3 y 5 m/s).

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

El simulador NS-3 es una herramienta muy difundida en el campo de la investigación debido a su amplio API (Application Programming Interface). El simulador NS- 3 al poseer la característica de código abierto permite la implementación de estándares de alta disponibilidad como los son: modelos de movilidad, protocolo de enrutamiento (OLSR, AODV, DSR, etc.) entre otros; esto permite comprender el beneficio y comprobar la funcionalidad del protocolo que se esté analizando con el propósito de lograr un mejor rendimiento y abrir la posibilidad del desarrollo de nuevos protocolos de enrutamiento.

El análisis del protocolo OLSR se lo realiza utilizando tres métricas: throughput, tasa de entrega de paquetes y retardo promedio extremo a extremo. Estas métricas son analizadas en diferentes escenarios, variando: la densidad de los nodos en la red (50, 100 y 150 nodos) y la velocidad de desplazamiento de los nodos (1.5, 3 y 5 m/s).

La eficiencia del protocolo OLSR puede variar por diversos factores, entre ellos: la movilidad de los nodos, los cuales al realizarse simulaciones a diferentes velocidades el protocolo actúa de manera diferente en cada una de ellas; la densidad de la red, el número de nodos también juega un papel importante para comprobar la eficiencia del protocolo, ya que a mayor número de nodos mayor es el procesamiento al momento de realizar el enrutamiento en toda la red.

Los parámetros del protocolo OLSR que viene establecidos en el simulador NS-3 dan como resultado un throughput máximo de 13.11 kbps, una pérdida de paquetes del 32.87% y un retardo promedio de 0.04 segundos para este tipo de redes. Se destaca que la velocidad de los nodos influye en el resultado de las métricas mencionadas anteriormente, al aumentar la velocidad de estos, los valores de las métricas mencionadas disminuyen; por lo tanto, no se está aprovechando toda la eficiencia que el protocolo presenta frente a redes de tipo MANET.

La eficiencia del protocolo OLSR en las redes MANET se incrementa mediante la variación de los parámetros más relevantes del protocolo, *Hello_Interval, Tc_Interval, Mid_Interval y HNA_Interval* estos permiten tener una mejor eficiencia del protocolo OLSR; por lo tanto, el estudio realizado cumple con los objetivos presentados en este proyecto de titulación.

Se comprobó a través de simulaciones, que se mejora el rendimiento del protocolo OLSR al variar de forma conjunta los parámetros Hello_Interval y Tc_Interval, debido a que estos dos parámetros son muy importantes al momento de anunciar los vecinos de la topología de red y para armar la tabla de enrutamiento; la combinación propuesta entre estos dos parámetros da mejores resultados en todas las métricas comparándolas con los resultados obtenidos con los parámetros por defecto. Lo cual cumple con el objetivo de este estudio técnico.

Se planteó otra combinación adicional a la anterior, se incluyen a los parámetros mencionados anteriormente los parámetros Mid_Interval y HNA_Interval , los cuales son de suma importancia para realizar el enrutamiento hacia redes externas. Estos dos parámetros sirven para anunciar las interfaces no-OLSR de los nodos y para agregar rutas externas a las tablas de enrutamiento de los nodos. La combinación propuesta entre estos cuatro parámetros da mejores resultados en todas las métricas comparándolas con los resultados obtenidos con los parámetros por defecto. Lo cual cumple con el objetivo de este estudio técnico.

Se determina que el mejor valor para el parámetro Hello_Interval es de 1 segundo; esto quiere decir que, los mensajes HELLO se enviarán de manera más rápida, esto permite reducir la cantidad de tráfico de enrutamiento y también minimizar el período de tiempo en el que se abandona el rango de transmisión del nodo vecino actual y está registrado en la lista de vecinos de otro nodo móvil. El mejor valor para el parámetro Tc_Interval es de 2,5 segundos, los mensajes TC se envían más rápido, estos se generan y retransmiten para inundar de información topológica en toda la red solo a través de nodos MRP. Por lo tanto, al reducir el tiempo de envío de estos mensajes la información que permite el cálculo de la ruta es proporciona a cada nodo en la red de forma más rápida y eficiente.

El mejor valor para el parámetro Mid_Interval es de 1 segundo, esto permite que los mensajes MID se distribuyan más rápidamente en aquellos nodos que tienen más de una interfaz. El valor que da mejores resultados para el parámetro HNA_Interval es de 1,25 segundos, esto permite que los nodos intercambien información de routing externa de forma más rápida y eficiente.

Los mensajes HNA se consideran como una versión generalizada de los mensajes TC, la diferencia radica en que los mensajes TC pueden dar información sobre la cancelación de la ruta, mientras que lo información proporcionada por los mensajes HNA solo se elimina después del tiempo de expiración.

4.2. RECOMENDACIONES

NS-3 es un simulador muy difícil de entender, por lo cual se debe realizar una revisión previa antes de manejar dicho simulador. Un buen inicio es leer la API del simulador; también se puede obtener ayuda de la comunidad que utiliza este simulador contactándolos a través de foros web en los cuales pueden proporcionar ayuda y/o tutoriales para resolver cualquier duda o problema.

Otros parámetros que pueden ser objetivo de estudio son: MPR_COVERAGE, que permite definir la cantidad de MRPs que debe cubrir un nodo; TC_REDUNDANCY, que ajusta el conjunto de enlaces anunciados en un mensaje TC, y enlaza los parámetros de histéresis que afectan el establecimiento del enlace y los procedimientos de falla. Estos parámetros no son considerados para este estudio técnico, pero quedan remarcados para futuros estudios.

Al momento de exportar las librerías al código fuente, también se lo debe hacer en el archivo .waf correspondiente, ya que al no realizar esto tendrá errores de referencia hacia dichas librerías

5. REFERENCIAS BIBLIOGRÁFICAS

[1] T. O. L. Q. W. B. G. Lucy Coya Rey, «SELECCIÓN DEL PROTOCOLO DE ENRUTAMIENTO PARA LA IMPLEMENTACIÓN DE UNA MANET,» de VIII Congreso Internacional de Telemática y Telecomunicaciones, CITTEL'2014, La Habana, 2014.

[2] M. Leonardo, «disi unitn,» 28 Mayo 2012. [En línea]. Available: http://disi.unitn.it/locigno/didattica/NC/11-12/08_OLSR_Leonardi.pdf. [Último acceso: 02 Octubre 2019].

[3] T. Clausen, «Optimized Link State Routing Protocol (OLSR)». Patente RFC 3626, Octubre 2003.

[4] O. O. K. Diaa Eldein Mustafa Ahmed, «An Overview of MANETs: Applications, Characteristics, Challenges and Recent Issues,» International Journal of Engineering and Advanced Technology (IJEAT), vol. 6, nº 4, pp. 128-133, Abril 2017.

[5] S. K. Sandeep Kumar, «Study of MANET: Characteristics, Challenges, Application, Routing Protocol and Security Attacks,» International Journal of R&D in Engineering, Science and Management, vol. 2, nº 5, pp. 266-274, 05 Julio 2015.

[6] A. Khattri, «geeks for geeks,» 08 Marzo 2018. [En línea]. Available: https://www.geeksforgeeks.org/introduction-of-mobile-ad-hoc-network-manet/. [Último acceso: 03 Octubre 2019].

[7] D. S. S. Tyagi, «Semantic Schoolar,» 05 Mayo 2013. [En línea]. Available: https://pdfs.semanticscholar.org/168f/d4d05d16ed88aafc9767e796d485aab7f757.pdf. [Último acceso: 03 Octubre 2019].

[8] K. M. K. Muralidhar, «An investigation into the operational limitations of mobile ad hoc networks,» de International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET),, Chennai, 2017.

[9] N. U. Meenakshi Yadav, «"Survey on MANET: Routing Protocols, Advantages, Problems and Security",» INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTER SCIENCE & ENGINEERING, vol. 1, nº 2, 30 Octubre 2014.

[10] S. A. ISHU VARSHNEY, «STUDY ON MANET: CONCEPTS,,» ELK ASIA PACIFIC JOURNAL OF COMPUTER SCIENCE AND INFORMATION SYSTEM, vol. 3, nº 2, 2017.

[11] A. Medina Santos, «Universidad Politecnica de Catalunya,» 23 Febrero 2006. [En línea]. Available: https://upcommons.upc.edu/bitstream/handle/2099.1/3556/53839-1.pdf. [Último acceso: 08 Octubre 2019].

[12] V. Paul, «SlideShare,» 13 Noviembre 2011. [En línea]. Available: https://es.slideshare.net/victerpaul/28-routing-in-mobile-adhoc-networks. [Último acceso: 07 Otubre 2019].

[13] S. Pardeep Kumar, «Implementation of modified OLSR protocol in AANETs for UDP and TCP,» Journal of King Saud University - Computer and Information Sciences, vol. 31, nº 3, 24 Febrero 2019.

[14] S. M.Gunasekar, «Automatic Tuning Of OLSR Routing Protocol,» de Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14), Tirupur, Tamilnadu, 2014.

[15] D. G. J. P. C. Gomez, «uter,» 2005. [En línea]. Available: https://dx-dev.computer.org/csdl/pds/api/csdl/proceedings/download-article/crawler-bypass/23730016/pdf. [Último acceso: 11 Octubre 2019].

[16] D. S. D. S. Dharam Vir, «International Journal of Modern Engineering Research (IJMER),» 01 Enero 2013. [En línea]. Available: www.ijmer.com. [Último acceso: 04 Abril 2020].

[17] T. Zimmerman, «Lars Strand,» 23 Abril 2004. [En línea]. Available: https://www.tldp.org/HOWTO/OLSR-IPv6-HOWTO/index.html. [Último acceso: 14 Octubre 2019].

[18] «nsnam,» 2011. [En línea]. Available: https://www.nsnam.org/. [Último acceso: 17 Octubre 2019].

[19] A. Jain, «ResearchGate,» 02 Febrero 2014. [En línea]. Available: https://www.researchgate.net/post/What_are_the_differences_between_NS2_NS3. [Último acceso: 17 Octubre 2019].

[20] «Network simulator 3,» 2000. [En línea]. Available: http://networksimulator3.com/network-simulator-3-free-download/. [Último acceso: 17 Octubre 2019].

[21] E. D. L. F. Rubio, «cursos de laf,» 26 Abril 2012. [En línea]. Available: http://cursos.delaf.cl/archivos/cursos/comunicaciones-inalambricas/laboratorios/Laboratorio+02+Modos.pdf. [Último acceso: 17 Octubre 2019].

[22] «GeeksforGeeks,» 05 Febrero 2019. [En línea]. Available: https://www.geeksforgeeks.org/network-simulator-3/. [Último acceso: 17 Octubre 2019].

[23] S. S. R. K. S. G. Rachna Chaudhary, «A study of comparison of Network Simulator - 3 and Network Simulator -2,» International Journal of Computer Science and Information Technologies (IJCSIT), vol. 3, nº 1, pp. 3085 - 3092, 2012.

[24] n.-3. project, «nsnam,» 16 Octubre 2007. [En línea]. Available: https://www.nsnam.org/docs/architecture.pdf. [Último acceso: 22 Octubre 2019].

[25] «NS3 code,» 31 Diciembre 2015. [En línea]. Available: http://ns3-code.com/ns-network-simulator-3/. [Último acceso: 22 Octubre 2019].

[26] P. C. Marqués, «ruinet,» Julio 2017. [En línea]. Available: https://riunet.upv.es/bitstream/handle/10251/88608/CARD%C3%93S-%20Dise%C3%B1o%20e%20implementaci%C3%B3n%20de%20un%20modelo%20del%20protocolo%20AURP%20para%20redes%20de%20sensores%20subacu%C3%A1ticas.pdf?sequence=1. [Último acceso: 22 Octubre 2019].

[27] C. A. JURADO LUCAS, «Universidad Politecnica Salesiana,» Abril 2015. [En línea]. Available: http://dspace.ups.edu.ec/handle/123456789/10345. [Último acceso: 22 Octubre 2019].

[28] J. E. C. A. Cristian Duran-Faundez, 29 Febrero 2016. [En línea]. Available: http://crduran.ubb.cl/old-html-based-site/documents/tutorial-ns3/tutorial-ns-3.pdf. [Último acceso: 23 Octubre 2019].

[29] R. Ammon, «“NS3- Doxygen Core,» 2015. [En línea]. Available: https://www.nsnam.org/doxygen/group__core.html#details. [Último acceso: 23

Octubre 2019].

[30] L. Y. MAYGUA MARCILLO, «repositorio digital EPN,» 14 Diciembre 2017. [En línea]. Available: http://bibdigital.epn.edu.ec/handle/15000/19024. [Último acceso: 23 Octubre 2019].

[31] R. Ammon, «Ns-3 Doxygen,» 2015. [En línea]. Available: https://www.nsnam.org/doxygen/classns3_1_1_simulator.html#a84be982e6d03b62c3dc3303c75a9b909. [Último acceso: 23 Octubre 2019].

[32] R. Ammon, «nsnam,» 2015. [En línea]. Available: https://www.nsnam.org/docs/release/3.27/models/html/index.html. [Último acceso: 23 Octubre 2019].

[33] A. H. Wheeb, «ResearchGate,» 07 Julio 2016. [En línea]. Available: https://www.researchgate.net/post/how_to_calculate_throughput. [Último acceso: 13 Noviembre 2019].

[34] R. P. G. Maribel, «Análisis y Simulación de protocolos de enrutamiento adecuados en diferentes escenarios para redes Adhoc, mediante la herramienta de NS-3,» Universidad de Loja, Loja, 2015.

[35] Microsoft, «Soporte tecnico de Microsoft,» Microsoft, [En línea]. Available: https://support.microsoft.com/es-es/office/funci%C3%B3n-prueba-t-1696ffc1-4811-40fd-9d13-a0eaad83c7ae. [Último acceso: 05 Abril 2020].

[36] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.

[37] S. S. Neeraj Verma, «Efficiency Comparison of Routing Protocols for Locating Nearby Nodes in MANET,» INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE, vol. 3, nº 8, pp. 324-329, Abril 2017.

[38] L. N. y. T. A. Rasa Bruzgiene, «intechopen,» 11 Mayo 2017. [En línea]. Available: https://www.intechopen.com/books/ad-hoc-networks/manet-network-in-internet-of-things-system. [Último acceso: 03 Octubre 2019].

ANEXOS

ANEXO A. Manual de instalación de NS-3.

ANEXO B. Código de simulación.

ANEXO C. Script para realizar las simulaciones.

ANEXO A

Manual de instalación de NS-3

Antes de realizar la instalación del simulador NS-3 se deben instalar algunos prerrequisitos.

Antes que nada, se debe actualizar la versión de Ubuntu 16.04 la cual tiene los componentes para realizar las simulaciones requeridas. Para ejecutar los comandos como un usuario *root* se utiliza la palabra **sudo.**

- sudo apt-get Update
- sudo apt-get pgrade
- sudo apt-get clean

El núcleo de NS-3 requiere una lista de paquetes los cuales son necesarios para dar soporte a diferentes opciones de NS-3, los cuales son detallados a continuación:

- Instalar C++

  sudo apt-get install build-essential

- Instalar python

  sudo apt-get install python python-dev

- Instalar Mercurial:

  sudo apt-get install mercurial

- Instalar bzr

  sudo apt-get install bzr

- Instalar flex, bison

  sudo apt-get install flex bison

- Instalar tcpdump para el soporte de lectura de archivos de rastreo pcap

  sudo apt-get install tcpdump

- soporte XML

  sudo apt-get install libxml2 libxml2-dev

Los códigos de NS-3 están disponibles en los repositorios de Mercurial. Antes de descargas el simulador, se creó una carpeta para guardar el código fuente, ejemplos, la API y otros recursos de NS-3. Se crea la carpeta llamada ns-3 bajo la carpeta ***/home/username/*** con el comando *mkdir ns-3*; luego se dirige a la carpeta creada con el comando *cd ns-3* y se ejecuta el siguiente comando:

- hg clone http://code.nsnam.org/ns-allinone-3.27

Los últimos números hacen referencia a la versión de NS-3 que se desea instalar (para este estudio técnico se eligió la versión 3.27).

Luego nos dirigimos al directorio que se creó y descargamos el NS-3:

- cd ns-allinone-3.27
- ./download.py

También una alternativa para descargar NS-3 es descargar todo el archivo zip (tar.bz2) usando el siguiente comando:

- wget http://www.nsnam.org/releases/ns-allinone-3.27.tar.bz2
- tar xjf ns-allinone 3.27.tar.bz2

Ahora, se debe construir NS-3. Esto se hace con el programa build.py (descargado anteriormente con el comando hg) haciendo:

La numeración de los Anexos debe realizarse con números en formato romano.

- ./build.py

Con esto comienza la construcción de la herramienta, con lo que se vería una serie de mensajes mostrando el estado del proceso. Esto puede tomar algunos minutos.

Luego se procede a habilitar los ejemplos que viene incluidos en el simulador:

- cd ns-3.27
- ./waf configure --enable-examples
- ./waf

Con esto, la herramienta NS-3 está lista para operar.

Otra alternativa para construir NS-3 es utilizando la herramienta Waf. Esta permite, entre otras cosas, construir una versión optimizada del código ya que, según el tutorial, NS-3 viene por defecto configurado para construir versiones de depuración.

Para trabajar con estas versiones optimizadas de los ejemplos incluidos en NS-3, se realiza lo siguiente:

Lo primero es ubicarse en la carpeta ns-3-dev que se encuentra en el directorio ns-3-allinone. Luego, ejecutar las siguientes líneas:

- ./waf clean
- ./waf −doptimized −−enable−examples −−enable− tests configure

Varias verificaciones se realizarán en ese momento.

Para realizar la comporbacion se procede a ejecutar un ejemplo:

- ./waf --run hello-simulator

Si la salida muestra “Hello Simulator” la instalación es correcta.

ANEXO B

Código de simulación

```
#include <ctime>
#include <iostream>
#include <sstream>
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/olsr-module.h"
#include "ns3/olsr-routing-protocol.h"
#include "ns3/olsr-helper.h"
#include "ns3/ipv4-list-routing-helper.h"
#include "ns3/ipv4-static-routing-helper.h"
#include "ns3/applications-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/stats-module.h"
#include "wifi-example-apps.h"
#include "ns3/ipv4.h"

using namespace ns3;
using namespace std;

NS_LOG_COMPONENT_DEFINE ("WiFiDistanceExperiment");

void TxCallback (Ptr<CounterCalculator<uint32_t> > datac,
                 std::string path, Ptr<const Packet> packet) {
  NS_LOG_INFO ("Sent frame counted in " <<
               datac->GetKey ());
  datac->Update ();
  // end TxCallback
}
/*
 * DEFINICION DE VARIABLES
 */
int size = 100; //tamanio de los nodos para la simulacion
int totalTime = 200; //tiempo en segundos que dura la simulacion
int startTime = 0; // tiempo inicio de simulacion
```

```
//intervalos de tiempo (segundos) de espera para el reenvio de mensajes
double helloInterval = 1;
double refreshInterval = 2;
double tcInterval = 2.5;
double dupHoldTime = 30;
bool files = true; // permite generar graficos para el analisis
int port = 4000;
int packetSize = 1024; //tamanio de los paquetes
int numPackets = 400; // numero de paquetes a enviar
int nodeSpeed = 2; // velocidad de los nodos (m/2)
int nodePause = 0; // pausa de los nodos (s)
/*
 * DEFINICION DE VARIABLES
 */

//------------------------------------------------------------------------
//--funcion main
//------------------------------------------------
int main (int argc, char *argv[]) {


  string format ("omnet");
  string experiment ("Optimizacion de parametros de OLSR ");
  string strategy ("wifi-default");
  string input;
  string runID;
  // estadisticas
  {
    stringstream sstr;
    sstr << "run-" << time (NULL);
    runID = sstr.str ();
  }

  // Generacion de aleatoeriada
  srand(time(NULL));
  SeedManager::SetSeed(rand());
  SeedManager::SetRun(rand());

  CommandLine cmd;
  cmd.AddValue("files", "Imprimir archivos analisis", files);
```

```
  cmd.AddValue("size", "Numero de nodos", size);
  cmd.AddValue("time", "Tiempo de simulacion, s", totalTime);
  cmd.AddValue("helloInterval", "Intervalo entre mensajes Hello",
helloInterval);
  cmd.AddValue("refreshInterval", "Intervalo para tener una rapida
conectividad ante los cambios", refreshInterval);
  cmd.AddValue("tcInterval", "Intervalo entre mensajes TC", tcInterval);
  cmd.AddValue("dupHoldTime", "Tiempo de almacenamiento para evitar
reenvio de paquetes duplicados ", dupHoldTime );
  cmd.AddValue("format", "Format to use for data output.", format);
  cmd.AddValue("experiment", "Identifier for experiment.", experiment);
  cmd.AddValue("strategy", "Identifier for strategy.", strategy);
  cmd.AddValue("run", "Identifier for run.", runID);
  cmd.AddValue("nodeSpeed", "Velocidad de los nodos", nodeSpeed);
  cmd.Parse(argc, argv);


  {
    stringstream sstr ("");
    sstr << size;
    input = sstr.str ();
  }

  /*
   * Creacion de los nodos
   */

  std::cout <<"\nNODOS GENERADOS: " << (unsigned)size << "\t"; // mensaje
en pantalla

  //------------------------------------------------------------
  //-- Create nodes and network stacks
  //------------------------------------------
  NS_LOG_INFO ("Creating nodes.");
  NodeContainer nodes;
  nodes.Create (size);

  //------------------------------------------------------------
  //-- Create mobility
  //------------------------------------------
```

```
  MobilityHelper mobilityAdhoc;
  int64_t streamIndex = 0; //usado para tener movilidad consistente a
traves de los escenarios

  ObjectFactory posicion;
  posicion.SetTypeId("ns3::RandomRectanglePositionAllocator");
  posicion.Set("X",
StringValue("ns3::UniformRandomVariable[Min=0.0|Max=500.0]"));
  posicion.Set("Y",
StringValue("ns3::UniformRandomVariable[Min=0.0|Max=500.0]"));
  Ptr<PositionAllocator> taPositionAlloc = posicion.Create() -
>GetObject<PositionAllocator>();
  streamIndex += taPositionAlloc ->AssignStreams(streamIndex);

  std::stringstream ssSpeed;
  ssSpeed << "ns3::UniformRandomVariable[Min=0.0|Max=" << nodeSpeed
<<"]";
  std::stringstream ssPause;
  ssPause << "ns3::ConstantRandomVariable[Constant=" << nodePause <<"]";


mobilityAdhoc.SetMobilityModel("ns3::RandomWaypointMobilityModel","Speed"
, StringValue (ssSpeed.str()), "Pause", StringValue (ssPause.str()),
"PositionAllocator", PointerValue(taPositionAlloc));
  mobilityAdhoc.SetPositionAllocator(taPositionAlloc);
  mobilityAdhoc.Install(nodes);
  streamIndex += mobilityAdhoc.AssignStreams (nodes, streamIndex);
  NS_UNUSED (streamIndex);//no se usa la variable streamIndex a partir de
esta sentencia


  //------------------------------------------------------------
  //-- End mobility
  //---------------------------------------------

  //------------------------------------------------------------
  //-- Create nodes
  //---------------------------------------------
```

```
  /*
   * Creacion Dispositivos
   */

  WifiHelper wifi;
  WifiMacHelper wifiMac;
  wifiMac.SetType ("ns3::AdhocWifiMac");
  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
  YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
  wifiPhy.SetChannel (wifiChannel.Create ());
  wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager",
"DataMode", StringValue("OfdmRate6Mbps"), "RtsCtsThreshold",
UintegerValue (0));
  NetDeviceContainer nodeDevices = wifi.Install (wifiPhy, wifiMac,
nodes);
  /*
   * Creacion Dispositivos
   */

  /*
   * configuracion de los parametros de OLSR
   */
  NS_LOG_INFO ("Enabling OLSR Routing");

  OlsrHelper olsr;
  olsr.Set("HelloInterval", TimeValue(Seconds(helloInterval)));
  olsr.Set("TcInterval", TimeValue(Seconds(tcInterval)));
  olsr.Set("MidInterval", TimeValue(Seconds(tcInterval)));
  olsr.Set("HnaInterval", TimeValue(Seconds(tcInterval)));
  olsr.Set("Willingness", EnumValue(3));//valor por defecto

  /*
   * configuracion de los parametros de OLSR
   */

  //------------------------------------------------------------
  //-- Install internet stack
  //------------------------------------------

  InternetStackHelper internet;
```

```
  internet.SetRoutingHelper(olsr);
  internet.Install (nodes);
  Ipv4AddressHelper ipAddrs;
  ipAddrs.SetBase ("10.0.0.0", "255.0.0.0");
  Ipv4InterfaceContainer interfaces;
  interfaces = ipAddrs.Assign (nodeDevices);

  //------------------------------------------------------------
  //-- Install internet stack
  //------------------------------------------------


  //------------------------------------------------------------
  //-- Install aplications
  //------------------------------------------------

  NS_LOG_INFO ("Create traffic source & sink.");
  Ptr<Node> appSource = NodeList::GetNode (0); // emisor
  Ptr<Sender> sender = CreateObject<Sender>();
  sender ->SetAttribute("Port", UintegerValue(port)); //puerto escucha
  sender ->SetAttribute("PacketSize", UintegerValue(packetSize));
  sender ->SetAttribute("Interval",
StringValue("ns3::ConstantRandomVariable[Constant=0.5]"));
  sender ->SetAttribute("NumPackets", UintegerValue(numPackets));
  appSource->AddApplication (sender);
  sender->SetStartTime (Seconds (1));

  Ptr<Node> appSink = NodeList::GetNode (9); //receptor
  Ptr<Receiver> receiver = CreateObject<Receiver>();
  receiver ->SetAttribute("Port", UintegerValue(port)); //puerto escucha
  appSink->AddApplication (receiver);
  receiver->SetStartTime (Seconds (0));

  Config::Set ("/NodeList/*/ApplicationList/*/$Sender/Destination",
               Ipv4AddressValue ("10.0.0.10")); // se cambia el destino
de los paquetes

  //------------------------------------------------------------
  //-- Install aplications
  //------------------------------------------------
```

```
  //-----------------------------------------------------------
  //-- Analysis files
  //-------------------------------------------

  if (files == true)
  {
        AsciiTraceHelper ascii;
        wifiPhy.EnableAsciiAll(ascii.CreateFileStream("olsr.tr"));
        wifiPhy.EnablePcapAll(std::string("olsr-node"));

        Ptr<OutputStreamWrapper> routingStream =
Create<OutputStreamWrapper> ("olsr.routes", std::ios::out);
        olsr.PrintRoutingTableAllEvery(Seconds (2),routingStream);

        Ptr<OutputStreamWrapper> neighborStream =
Create<OutputStreamWrapper> ("olsr.neighbors", std::ios::out);
        olsr.PrintNeighborCacheAllEvery(Seconds (2), neighborStream);
  }

  //-----------------------------------------------------------
  //-- Analysis files
  //-------------------------------------------




  //-----------------------------------------------------------
  //-- Setup stats and data collection
  //-------------------------------------------

  // Create a DataCollector object to hold information about this run.
  DataCollector data;
  data.DescribeRun (experiment,strategy,input,runID);

  // ****** 1 *****
  // Create a counter to track how many frames are generated.  Updates
  // are triggered by the trace signal generated by the WiFi MAC model
  // object.  Here we connect the counter to the signal via the simple
```

```
  // TxCallback() glue function defined above.
  Ptr<CounterCalculator<uint32_t> > totalTx =
CreateObject<CounterCalculator<uint32_t> >();
  totalTx->SetKey ("tx-frames");
  totalTx->SetContext ("node[0]");
  Config::Connect
("/NodeList/0/DeviceList/*/$ns3::WifiNetDevice/Mac/MacTx",
                   MakeBoundCallback (&TxCallback, totalTx));
  data.AddDataCalculator (totalTx);


  // ****** 2 *****
  // This is similar, but creates a counter to track how many frames
  // are received.  Instead of our own glue function, this uses a
  // method of an adapter class to connect a counter directly to the
  // trace signal generated by the WiFi MAC.
  Ptr<PacketCounterCalculator> totalRx =
CreateObject<PacketCounterCalculator>();
  totalRx->SetKey ("rx-frames");
  totalRx->SetContext ("node[9]");
  Config::Connect
("/NodeList/9/DeviceList/*/$ns3::WifiNetDevice/Mac/MacRx",
                   MakeCallback (&PacketCounterCalculator::PacketUpdate,
                                 totalRx));
  data.AddDataCalculator (totalRx);


  // ****** 3 *****
  // This counter tracks how many packets---as opposed to frames---are
  // generated.  This is connected directly to a trace signal provided
  // by our Sender class.
  Ptr<PacketCounterCalculator> appTx =
CreateObject<PacketCounterCalculator>();
  appTx->SetKey ("sender-tx-packets");
  appTx->SetContext ("node[0]");
  Config::Connect ("/NodeList/0/ApplicationList/*/$Sender/Tx",
                   MakeCallback (&PacketCounterCalculator::PacketUpdate,
                                 appTx));
  data.AddDataCalculator (appTx);
```

```
// ****** 4 *****
// Here a counter for received packets is directly manipulated by
// one of the custom objects in our simulation, the Receiver
// Application.  The Receiver object is given a pointer to the
// counter and calls its Update() method whenever a packet arrives.
Ptr<CounterCalculator<> > appRx = CreateObject<CounterCalculator<> >();
appRx->SetKey ("receiver-rx-packets");
appRx->SetContext ("node[9]");
receiver->SetCounter (appRx);
data.AddDataCalculator (appRx);

// ****** 5 *****
// This DataCalculator connects directly to the transmit trace
// provided by our Sender Application.  It records some basic
// statistics about the sizes of the packets received (min, max,
// avg, total # bytes), although in this scenario they're fixed.
Ptr<PacketSizeMinMaxAvgTotalCalculator> appTxPkts =
  CreateObject<PacketSizeMinMaxAvgTotalCalculator>();
appTxPkts->SetKey ("tx-pkt-size");
appTxPkts->SetContext ("node[0]");
Config::Connect ("/NodeList/0/ApplicationList/*/$Sender/Tx",
                 MakeCallback
                   (&PacketSizeMinMaxAvgTotalCalculator::PacketUpdate,
                   appTxPkts));
data.AddDataCalculator (appTxPkts);


// ****** 6 *****
// Here we directly manipulate another DataCollector tracking min,
// max, total, and average propagation delays.  Check out the Sender
// and Receiver classes to see how packets are tagged with
// timestamps to do this.
Ptr<TimeMinMaxAvgTotalCalculator> delayStat =
  CreateObject<TimeMinMaxAvgTotalCalculator>();
delayStat->SetKey ("delay");
delayStat->SetContext (".");
receiver->SetDelayTracker (delayStat);
data.AddDataCalculator (delayStat);
```

```
Simulator::Stop(Seconds (totalTime+1));

//-------------- Flow Monitor ---------------

Ptr<FlowMonitor> flowMonitor;
FlowMonitorHelper flowHelper;
flowMonitor = flowHelper.InstallAll();

//-------------- Flow Monitor ---------------

Simulator::Run();

 //------------------------------------------------------------
 //-- Generate statistics output.
 //------------------------------------------

 // Pick an output writer based in the requested format.
 Ptr<DataOutputInterface> output = 0;
 if (format == "omnet") {
     NS_LOG_INFO ("Creating omnet formatted data output.");
     output = CreateObject<OmnetDataOutput>();
   } else if (format == "db") {
   #ifdef STATS_HAS_SQLITE3
     NS_LOG_INFO ("Creating sqlite formatted data output.");
     output = CreateObject<SqliteDataOutput>();
   #endif
   } else {
     NS_LOG_ERROR ("Unknown output format " << format);
   }

 // Finally, have that writer interrogate the DataCollector and save
 // the results.
 if (output != 0)
   output->Output (data);

 // Free any memory here at the end of this example.

 //------------- Generate statistics output -------------
 //------------- flow Monitor --------------------
 std::cout<<std::endl<<"Tx\t\tRx\t\tTxPkt\tRxPkt\tThput\t\tPkt_Loss(%)";
```

```
std::cout<<std::endl<<"############################################\n";
  flowMonitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier =
DynamicCast<Ipv4FlowClassifier>(flowHelper.GetClassifier());
  std::map<FlowId, FlowMonitor::FlowStats> stats = flowMonitor-
>GetFlowStats();

  for(std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i =
stats.begin(); i!= stats.end(); ++i)
  {
        Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow(i->first);
        std::cout << t.sourceAddress << "\t";
        std::cout << t.destinationAddress << "\t";
        std::cout << i->second.txPackets << "\t";
        std::cout << i->second.rxPackets << "\t";
        //std::cout << (i-
>second.rxPackets*8.0*1025)/(1024*1024)/(totalTime - start_time)<<
"\t\t"<<std::endl(;)
        std::cout << i->second.rxBytes*8.0/(i-
>second.timeLastRxPacket.GetSeconds()-i-
>second.timeFirstTxPacket.GetSeconds())/1024 << "Kbps" << " \t\t"
<<std::endl;

  }

  //------------- Generate statistics output --------------
    //------------- flow Monitor --------------------

  Simulator::Destroy ();
  return 0;

  // end main
}
```

ANEXO C

Script para realizar las simulaciones.

Se plantea un número de 10 simulaciones por cada velocidad de los nodos que se plantearon para el análisis. Se destaca que cada simulación se la realiza de manera independiente de la anterior. El siguiente script permite realizar lo mencionado anteriormente.

```
#!/bin/sh
NODESPEEDS="1.5 3 5"
SIMULACIONES="1 2 3 4 5 6 7 8 9 10"
for simulacion in $SIMULACIONES
do
for nodeSpeed in $NODESPEEDS
do
echo  Simulación N°:$simulacion, Velocidad de los Nodos:$nodeSpeed[m/s]
./waf --run "wifi-example-sim --nodeSpeed=$nodeSpeed --run=run-
$nodeSpeed-$simulacion"
mv olsr.routes olsr.exp_$simulacion.speed_$nodeSpeed.routes
mv olsr.neighbors olsr.exp_$simulacion.speed_$nodeSpeed.neighbors
mv olsr.tr olsr.exp_$simulacion.speed_$nodeSpeed.tr
mv dat* olsr.exp_$simulacion.speed_$nodeSpeed.stats
done
done
```

ORDEN DE EMPASTADO