ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

SIMULACIÓN Y ANÁLISIS DE UNA RED FANET USANDO EL SIMULADOR NS-3

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

WELLINGTON GUSTAVO MOPOSITA PAGUAY

DIRECTOR: Ing. CHRISTIAN JOSÉ TIPANTUÑA TENELEMA, PhD.

Quito, Septiembre 2023

AVAL

Certifico que el presente trabajo fue desarrollado por WELLINGTON GUSTAVO MOPOSITA PAGUAY bajo mi supervisión.

_____

Ing. Christian José Tipantuña Tenelema, PhD.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo WELLINGTON GUSTAVO MOPOSITA PAGUAY, 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.

_____

WELLINGTON GUSTAVO MOPOSITA PAGUAY

DEDICATORIA

A mis padres.

El trabajo duro, paciencia y cariño hacen realidad los sueños.

AGRADECIMIENTO

A mis padres por el tiempo compartido y el apoyo en todas las decisiones tomadas.

Al Dr. Christian José Tipantuña Tenelema por su paciencia y entrega al desarrollar el presente trabajo.

A la Escuela Politécnica Nacional y todos los docentes por el conocimiento compartido.

ÍNDICE DE CONTENIDO

RESUMEN

Las redes FANET (Flying Ad-hoc Network) son redes conformadas por múltiples vehículos aéreos no tripulados (UAV: Unmanned Aerial Vehicles) interconectados entre sí (UAV to UAV) o conectados con una estación base (UAV to Ground Station), estas redes pueden ser desplegadas en situaciones extremas en donde otro tipo de redes podrían ser inviables y, además, son un subgrupo de las redes Ad-hoc.

Los protocolos de enrutamiento que se pueden utilizar en las redes FANET son: OLSR (Optimized Link State Routing), DSDV (Destination-Sequenced Distance Vector), AODV (Ad-hoc On-demand Distance Vector), DSR (Dynamic Source Routing), AOMDV (Ad-hoc On-demand Multipath Distance Vector), HWMP (Hybrid Wireless Mesh Protocol), los cuales se pueden clasificar en tres clases diferentes: protocolos proactivos, protocolos reactivos y protocolos híbridos.

Cada protocolo utilizado en el despliegue de la red FANET tendrá un impacto directo en su rendimiento por lo que, este trabajo de titulación busca simular una red FANET mediante el uso del software NS-3 con el objetivo de determinar el rendimiento de cada protocolo. Para realizar el análisis de rendimiento se considera dos escenarios distintos, el primer escenario toma en cuenta un número de UAVs (por ejemplo, 50) en vuelo con una posición estática, en tanto que en el segundo escenario se considera que los UAVs toman posiciones de vuelo aleatorias. Cada escenario es analizado en base a parámetros de funcionamiento de la red FANET, en específico se consideran: throughput, tasa de entrega de paquetes y retardo promedio.

El presente trabajo inicia con una descripción conceptual de los UAVs, ventajas desventajas, redes Ad-hoc, características y su clasificación, además de conceptos de las redes MANET, VANET y FANET, así como los protocolos de enrutamiento de las redes FANET.

Posteriormente en el capítulo 2, se abordan características, instalación del simulador NS3, para finalmente en el capítulo 3 realizar análisis de los resultados de la simulación.

PALABRAS CLAVE: FANET, NS-3, protocolo, UAV, enrutamiento.

ABSTRACT

FANET networks (Flying Ad-hoc Network) are networks of multiple unmanned aerial vehicles (UAV) interconnected with each other (UAV to UAV) or connected with a base station (UAV to Ground Station), these networks can be deployed in extreme situations where other types of networks couldn´t be an option for deployment and, FANET networks are a subgroup of Ad-hoc networks.

The routing protocols that can be used in FANET networks are: OLSR (Optimized Link State Routing), DSDV (Destination-Sequenced Distance Vector), AODV (Ad-hoc On-demand Distance Vector), DSR (Dynamic Source Routing), AOMDV (Ad-hoc On-demand Multipath Distance Vector), HWMP (Hybrid Wireless Mesh Protocol), which can be classified into three different classes: proactive protocols, reactive protocols, and hybrid protocols.

Each protocol used in the deployment of the FANET network will have a direct impact on its performance, therefore, this degree work seeks to simulate a FANET network by using the NS-3 software in order to determine the performance of each protocol. To carry out the performance analysis, two different scenarios are considered, the first scenario takes into account a number of UAVs (for example, 50) in flight with a static position, while in the second scenario it is considered that the UAVs take positions of random flight. Each scenario is analyzed based on the operating parameters of the FANET network, specifically throughput, packet delivery rate and average delay are considered.

This paper begins with a conceptual description of UAVs, advantages, disadvantages, Ad-hoc networks, characteristics and their classification, concepts of MANET, VANET and FANET networks, as well as FANET network routing protocols.

Subsequently, in chapter 2, characteristics and installation of the NS3 simulator are addressed, to finally analyze the simulation results in chapter 3.

KEYWORDS: FANET, NS-3, protocol, UAV, routing.

1. INTRODUCCIÓN

El uso de vehículos aéreos no tripulados (UAVs: Unmanned Aerial Vehicles) o drones ha aumentado considerablemente con el pasar de los años, esto debido a que las aplicaciones de estos dispositivos son muchas, como es el caso de la agricultura, ingeniería, la industria del transporte de carga, entre otros [1]. Sin embargo, el trabajar con dispositivos UAVs de manera individual tiene ciertas limitaciones, entre ellas, la más notable es la autonomía energética, misma que está relacionada a la altura, tiempo de vuelo del dispositivo y rendimiento de la red. Para poder superar estas limitaciones se tiene como opción principal adquirir un UAV que cumpla con los requerimientos de autonomía necesarios para una aplicación específica, tal como es el sobrevolar una zona de gran tamaño por varias horas, sin embargo, el costo del dispositivo, así como su peso pueden aumentar considerablemente. Por otro lado, el tener un solo dispositivo UAV tiene sus limitantes en cuanto a cobertura para operación del mismo, se requerirían varios receptores interconectados para no perder el control del dispositivo UAV. Para superar estas limitaciones utilizando dispositivos UAVs de costos asequibles, se propone el uso de múltiples dispositivos UAVs de forma simultánea, conformando así una red FANET (Flying Ad-hoc Network) [2].

El presente trabajo, busca simular una red FANET en el software de simulación NS-3, así como los protocolos de enrutamiento que intervienen en la comunicación, esto con el fin de hacer el análisis de su rendimiento considerando dos escenarios:

- Nodos en vuelo en posición estática.
- Nodos en vuelo en posición aleatoria.

Los protocolos de enrutamiento utilizados en las redes FANET no fueron desarrollados específicamente para su aplicación en estos escenarios, por lo que se usan los protocolos implementados en redes MANET (Mobile Ad-hoc Network) y se pueden clasificar de la siguiente manera [3] [4]:

- **Protocolos Proactivos:** La principal característica de estos protocolos es que las tablas de enrutamiento se actualizan y almacenan en cada UAV. Para las redes FANET se tienen los protocolos OLSR (Optimized Link State Routing) y DSDV (Destination-Sequenced Distance Vector).
- **Protocolos Reactivos:** También se los conoce como protocolos bajo demanda o protocolos pasivos, esto debido a que, la tabla de enrutamiento es actualizada únicamente si se tiene información para enviar.

En las redes FANET se tienen los protocolos reactivos AODV (Ad-hoc On-demand Distance Vector), DSR (Dynamic Source Routing) y AOMDV (Ad-hoc On-demand Multipath Distance Vector).

- **Protocolos híbridos:** Estos protocolos son una combinación de características de los protocolos reactivos y proactivos, para las redes FANET se tiene HWMP (Hybrid Wireless Mesh Protocol).

Además, el presente trabajo hará una revisión de los conceptos relacionados a las redes Ad-hoc, características y su clasificación, así como una breve guía del proceso de instalación y características del software de simulación NS-3, integración con el IDE Eclipse y el desarrollo del código de simulación de una red FANET.

1.1 OBJETIVOS

El objetivo general de este Proyecto Técnico es:

- Simular una red FANET utilizando NS-3

Los objetivos específicos del Proyecto Técnico son:

- Revisar los fundamentos teóricos de las redes Ad-hoc incluyendo su características generales y clasificación.
- Describir los aspectos técnicos de redes FANET incluyendo la arquitectura y los protocolos de enrutamiento, así como las características generales de NS-3, Wireshark y el entorno de desarrollo ECLIPSE.
- Describir el diseño de una red FANET
- Implementar la red FANET en el simulador NS-3 considerando dos escenarios diferentes, uno con nodos estáticos y el otro con nodos móviles.
- Analizar el despliegue de protocolos de enrutamiento en la red FANET simulada en NS-3.

1.2 ALCANCE

En el presente trabajo se evalúa el rendimiento de una red FANET simulada por medio del software NS-3 en dos escenarios distintos:

- El primer escenario en donde se tendrá un número de UAVs (por ejemplo, 50) en vuelo en posiciones estáticas.
- El segundo escenario plantea el despliegue de un número de UAVs en vuelo en posiciones aleatorias.

Cada escenario busca implementar los protocolos de enrutamiento de las redes FANET, mismos que se clasifican en:

- Protocolos proactivos: OLSR, DSDV.
- Protocolos reactivos: AODV, DSR, AOMDV.
- Protocolos híbridos: HWMP.

Para cada protocolo y escenario se generan archivos con extensión *pcap*, mismos que pueden ser analizados mediante el uso del software Wireshark para determinar que el protocolo deseado se ha implementado adecuadamente.

El rendimiento de la red se evalúa a través del uso de parámetros como el throughput, tasa de entrega de paquetes y retardo promedio, mismos que se obtienen directamente de la simulación. Adicionalmente, se busca generar un código en donde el usuario pueda modificar el número de UAVs a desplegar, así como el protocolo de enrutamiento a usar en la red.

Como punto de partida, se hará un recorrido por los conceptos relacionados a los UAVs, clasificación de los UAVs, historia relacionada al desarrollo de los UAVs, así como, los conceptos de redes Ad-hoc, características, funcionamiento, clasificación y conceptos de las redes MANET, VANET (Vehicular Ad-hoc Network) y FANET. Posteriormente, se abordan los conceptos relacionados a los protocolos de enrutamiento de las redes FANET. Además, se muestra una guía del proceso de instalación y despliegue del software de simulación NS-3. Como resultado de este trabajo no se tiene un producto final demostrable, debido a que todo el análisis se realiza mediante simulación.

1.3 MARCO TEÓRICO

En este capítulo se abordarán conceptos relacionados al tema de estudio, con el fin de familiarizar al lector con los conceptos relacionados a los UAVs, además de su historia, ventajas y desventajas, de igual manera, las redes Ad-hoc, características y su clasificación en redes MANET, VANET y FANET, siendo las redes FANET objeto de mayor énfasis. Permitiendo conocer el funcionamiento de estas redes, el uso adecuado de ellas, así como, ventajas y desventajas de su aplicación. Finalmente, en este capítulo se revisan los conceptos asociados a los protocolos de enrutamiento de las redes FANET.

1.3.1 UAV

Los UAVs son dispositivos con capacidades de vuelo, los cuales se asocian con el término “no tripulado” debido a que, son controlados sin la intervención directa de un ser humano,

diferenciándose así de los demás dispositivos con capacidades de vuelo similares [5]. Los UAVs pueden ser controlados por medio de un sistema de control remoto en tierra o sistemas de desplazamiento controlados por una computadora a bordo, para lo cual se utilizan sensores y sistemas GPS que ayudan con el posicionamiento y desplazamiento en el aire [6].

A pesar de que su desarrollo inicial fue destinado al uso militar, en la actualidad existen UAVs de carácter comercial destinados a diferentes usos, entre los que destacan los siguientes:

- Operaciones de búsqueda y reconocimiento.
- Video vigilancia.
- Monitoreo de tráfico.
- Fotografía y video.
- Control de tráfico.
- Topografía.
- Fumigación en campo abierto.

Los UAVs pueden clasificarse por sus características físicas generales en dos grupos:

- **Ala fija:** Estos UAVs se caracterizan por ser físicamente similares a los aviones convencionales, como se observa en la Figura 1.1, además de usar los mismos métodos de despegue, vuelo y descenso. A su vez, son controlados por radio control e impulsados usando sistemas de combustión interna o motores eléctricos, tales como, turbinas de propulsión o rotores posicionados en la parte frontal, y pueden mantenerse en vuelo por su forma aerodinámica, siempre y cuando continúen su marcha hacia adelante y tengan velocidad de sustentación [7].

Figura 1.1. UAVs de ala fija.

- **Ala rotatoria:** La característica más notable de estos dispositivos está dada por el uso de al menos 4 rotores posicionados de manera horizontal en los alrededores

del dispositivo como se puede apreciar en la Figura 1.2, los cuales permiten el desplazamiento en el aire, trayectorias definidas, despegues y aterrizajes verticales, además de cambios de dirección más rápidos y agresivos que los dispositivos de ala fija. Debido a estas características este tipo de UAVs es utilizado en diversas aplicaciones, tales como: tareas de búsqueda y rescate, monitoreo de tráfico, video vigilancia, soporte para redes de comunicación inalámbricas, estaciones aéreas de comunicación celular, entre otras [8] [9] [10].

Figura 1.2. UAVs de ala rotatoria.

1.3.2 HISTORIA DE LOS UAVS

El uso de objetos voladores no tripulados se evidencia en la historia desde el año 1849, siendo el uso militar el principal motivo de su desarrollo inicial, en este año, en Venecia, se utilizaron globos con capacidades de vuelo, esto debido a que estaban constituidos de helio o nitrógeno caliente en su interior [11]. En el año 1907 aparece el primer modelo de un helicóptero quadrirrotor mostrado en la Figura 1.3, similar a los drones usados en la actualidad, este modelo nace de la mano de los hermanos Jacques y Louis Bréguet con la ayuda del profesor Charles Richet [12].

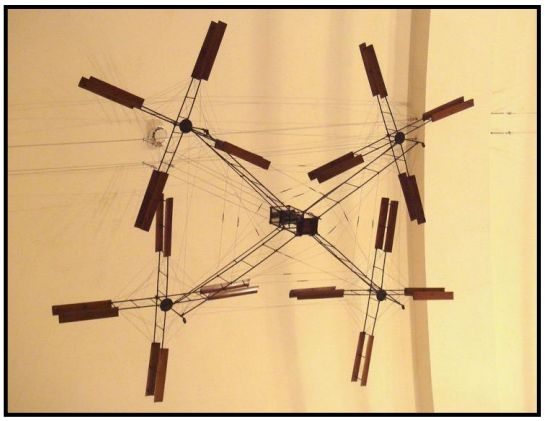

Figura 1.3. Quadrirrotor Bréguet-Richet, tomado de [13].

En 1918 aparece la primera aeronave no tripulada controlada a distancia por medio de un control remoto, misma que se puede ver en la Figura 1.4, dicha aeronave recibió el nombre

de “Kettering Bug” y fue desarrollada por la armada de los Estados Unidos, sin embargo, no fue usada en combate [14].

Figura 1.4. Kettering Aerial Torpedo “Bug”, tomado de [14].

El término UAV fue utilizado por primera vez en el año 1935 para denominar a la aeronave radio controlada “Queen Bee”, el cual fue un dispositivo de ala fija usado por primera vez en misiones de entrenamiento de puntería. Se produjeron alrededor de 380 de estos dispositivos dando por finalizada su producción en el año 1947 [15].

En la Figura 1.5 se puede apreciar a “Queen Bee” en vuelo.

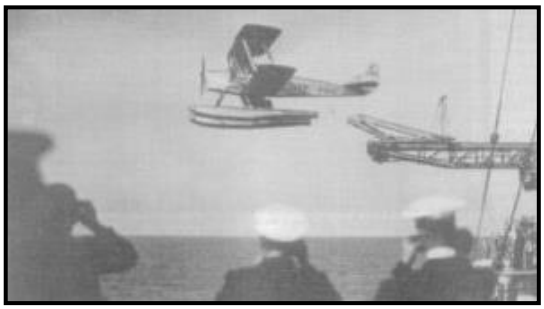

Figura 1.5. Queen Bee siendo lanzado desde el HMAS Australia, tomado de [16].

Los años posteriores se basaron en el desarrollo de esta tecnología para su uso en intervenciones militares, para el año 2006 se introducen en la FAA (Federal Aviation Administration) los términos para el uso comercial de los UAVs como dispositivo civil y para fines de entretenimiento, delimitando así su uso para actividades no-militares [17].

En el CES (Consumer Electronics Show) del año 2010, aparece el primer UAV destinado a la venta comercial, denominado “Parrot AR Drone”, basado en un diseño quadrirrotor, con estructura de carbono o polipropileno expandido, además de una batería de litio de 1000 mAh, conocido por ser un dispositivo controlado completamente por Wi-Fi, a través del uso de un Smartphone como mando de control [18] [19].

El “Parrot AR Drone” fue el inicio del desarrollo de drones para uso y aplicaciones comerciales. Para el año 2013, Amazon anunciaría su servicio de entrega de paquetes usando UAVs, mismo que fue denominado “Prime Air”, sin embargo, no fue hasta el año

2016 en donde se realizó la primera entrega mediante este servicio. El principal retraso en la implementación del mismo fue debido a que, por disposición de la FAA no se permitió sino hasta ese año la operación de UAVs de carácter comercial fuera de la línea de vista del operador [20]. En la Figura 1.6 se puede apreciar uno de los drones utilizados en el servicio “Prime Air”

Figura 1.6. Amazon Prime Air, tomado de [21].

En este mismo año aparece uno de los mejores UAVs considerados hasta la fecha, el “Phantom 4” de la empresa china DJI (Dà-Jiāng Innovations), galardonado por sus características que incluían baterías de alta duración, control de estabilidad, cámara con ángulos de visión de 94°, además, el hecho de ser compatible con algoritmos de machine learning permitió la investigación y desarrollo de nuevas tecnologías a partir del uso de UAVs [18] [22] [23].

Con el pasar de los años los UAVs han adoptado nuevas características, tales como mejores cámaras, mejoras en la autonomía, mayor velocidad de desplazamiento, mayor capacidad de carga, entre otros, incluso en la actualidad se pueden tener un sin número de opciones en cuanto a precios, características y fabricantes, en la Figura 1.7 se hace un resumen de los eventos importantes en el desarrollo de los UAVs.

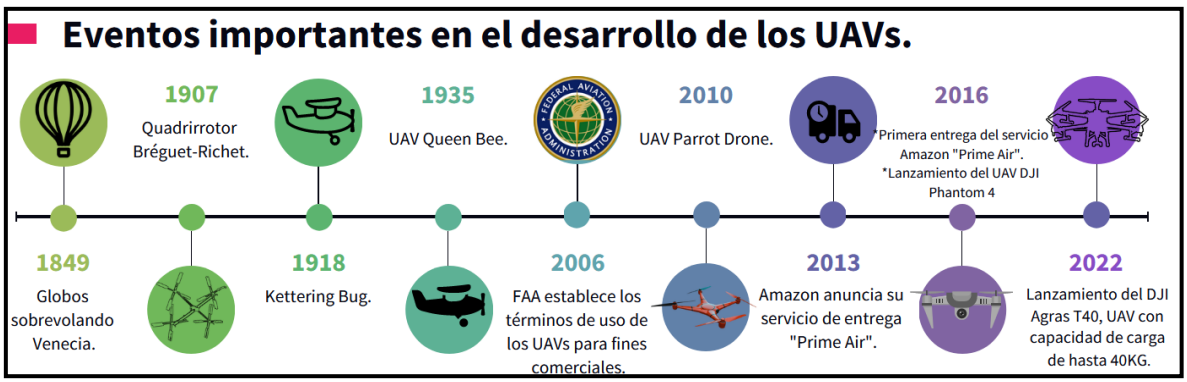

Figura 1.7. Eventos importantes en el desarrollo de los UAVs.

1.3.3 VENTAJAS Y DESVENTAJAS DE LOS UAVS

Los UAVs han ganado popularidad no solo por su capacidad de despliegue rápido sin necesidad de una infraestructura existente de gran tamaño, las múltiples aplicaciones en

investigación debido a la compatibilidad con plataformas Open Source, sino que, además, se ha considerado su uso en la industria por sus ventajas claras ante otros dispositivos con capacidades de vuelo, de las cuales se destacan las siguientes [24] [25]:

- **Reducción del riesgo humano:** El ser dispositivos controlados remotamente por un operador en tierra reduce considerablemente la capacidad de accidentes o incidentes relacionados con el piloto, los cuales podrían existir en casos como, por ejemplo: al sobrevolar en áreas geográficamente inaccesibles, zonas de incendios o desastres naturales.
- **Reducción de tiempos de operación:** En caso de que se requiera realizar una determinada tarea que involucre monitoreo desde el aire, el uso de UAVs reduce considerablemente el tiempo en el que se realizan las operaciones, puesto que su despliegue es rápido y sencillo.
- **Menor contaminación:** Al ser dispositivos que cuentan con baterías, la emisión de gases producidos por combustión es nula, lo cual ayuda a la preservación del ambiente, además, el tiempo de vida útil de las baterías podría ser prolongado dependiendo del tipo de UAV.
- **Precisión:** Estos dispositivos pueden llegar a tener movimientos muy precisos, llegando a realizar tareas en vuelo que requieren alta precisión, tales como el cambio de luminarias o grabación de zonas específicas en espacios de difícil acceso en donde se requiere que el dispositivo pueda moverse incluso milimétricamente.
- **Transporte de carga:** Algunos modelos de UAVs permiten el transporte de carga, lo cual es una característica en desarrollo puesto que los modelos que admiten esta funcionalidad cuentan con una baja capacidad de carga y autonomía. Sin embargo, existen modelos destinados principalmente a la agricultura o entrega de paquetes que permiten el desplazamiento con carga variable.
- **Acceso por aire:** Estos dispositivos permiten sobrevolar zonas de difícil acceso o inaccesibles por otros medios.
- **Integración con sensores a bordo:** El uso de sensores a bordo es una característica aprovechada por el tamaño y la versatilidad de los dispositivos, tales como, GPS, altímetros, temperatura, distancia, niveles de combustible, así como cámaras y tarjetas de almacenamiento otorgando la posibilidad de poder capturar imágenes, video y datos de los sensores a bordo [26].

- **Transferencia de información:** Permiten la comunicación con estaciones en tierra para poder transmitir información en tiempo real mientras estén dentro del rango de cobertura.

A pesar de las ventajas mencionadas anteriormente, existen desventajas en el uso de UAVs de manera individual, entre ellas se tienen las siguientes [27]:

- **Autonomía relativamente baja:** Para reducir el peso de los dispositivos y permitir la maniobrabilidad en el aire no es posible usar baterías de gran capacidad, lo cual limita el tiempo de operación y cobertura para el control en el aire, así como, la capacidad de transferencia de datos.
- **Distancia limitada de operación y control desde tierra:** Relacionado directamente a la capacidad del hardware, en algunos casos puede existir una limitación importante en cuanto a distancia de operación remota.
- **Susceptible a interferencias**: Dependiendo de la tecnología de comunicación con el dispositivo que se utilice, se puede tener más o menos interferencias de las señales existentes en la zona de operación.
- **Alto costo**: Acorde a la aplicación, es posible que el hardware requerido pueda no ser suficiente, en especial si se adquieren UAVs de gamas bajas, por lo que, si se requieren UAVs de alta gama con funcionalidades o sensores específicos como cámaras con infrarrojos o espacios para carga, el costo del dispositivo puede ascender rápidamente.
- **Operación a baja altitud**: Para los dispositivos de gama inicial el vuelo en alturas considerables, como las mayores a 50 metros, pueden ser un reto, en especial en zonas con fuertes vientos. Por otro lado, la altura a la que puede volar un UAV puede estar regulada por las leyes del país en donde se despliega el dispositivo, como es el caso de la FAA que menciona que la altura máxima permitida es de 600 pies (182 metros) [28] [29].

1.3.4 REDES MÓVILES AD-HOC

Las redes Ad-hoc son también conocidas como redes descentralizadas, esto debido a que no requieren de un nodo central o una infraestructura existente para poder establecer una comunicación entre dos o más dispositivos. En estas redes, la comunicación se da entre los dispositivos que la conforman, es decir, cada dispositivo puede reenviar datos a otro dispositivo cuando considere necesario, además cada dispositivo de la red puede actuar como transmisor o receptor, conformando así, una red temporal de alta movilidad [30].

Los dispositivos más comunes que se pueden integrar a estas redes Ad-hoc son las computadoras portátiles, y, uno de los requerimientos para la comunicación entre estos dispositivos es que deben usar el mismo SSID (Service Set Identifier) y el mismo canal, además, el dispositivo debe estar en modo Ad-hoc. Cabe mencionar que, para enviar datos de un nodo a otro se pueden utilizar nodos intermedios que sean parte de la red Ad-hoc [31].

1.3.5 CARACTERÍSTICAS DE LAS REDES AD-HOC

La característica más notable de estas redes es su movilidad, ya que pueden ser desplegadas en cualquier escenario en donde se dispongan al menos 2 dispositivos para ser configurados dentro de esta red. Además, los dispositivos como los computadores de escritorio que dispongan de una tarjeta de red inalámbrica también se pueden incluir en estas redes, como se ve en la Figura 1.8, debido a que, sí disponen de una fuente de energía móvil pueden cambiar de posición libremente, formando parte de la red Ad-hoc siempre y cuando formen parte de la misma red inalámbrica [31].

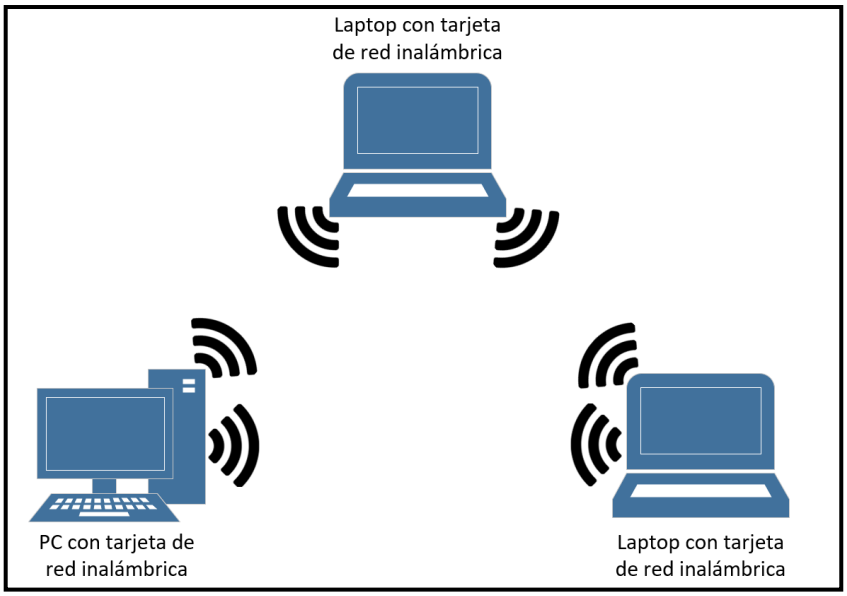

Figura 1.8. Representación de una red Ad-hoc en donde se incluye una computadora de escritorio.

Entre otras características generales, destacan las siguientes [30]:

- **Topología Variable:** Los nodos al moverse libremente, incluso con velocidades variables y depender únicamente de un enlace con un nodo vecino generan una topología variable, puesto que, este puede salir de cobertura e incluso formar un nuevo enlace con otro nodo para ampliar el área de cobertura o a su vez generar una nueva red Ad-hoc.

- **Rutas variables:** Al tener libre movilidad, los nodos pueden llegar a desconectarse, lo que obliga al tráfico a buscar nuevas rutas para poder llegar al destino, esto puede ilustrarse de mejor manera en la Figura 1.9, en donde, el nodo B sale de cobertura y ya no es ruta entre A y C, por esto el tráfico entre estos dos dispositivos deberá utilizar otra ruta como A-E-F-C.

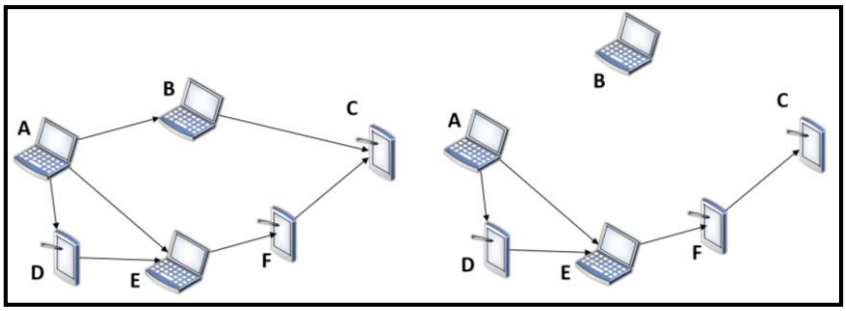

Figura 1.9. Rutas variables en redes Ad-hoc, basado [30].

- **Capacidad limitada de los dispositivos:** Al ser dispositivos portátiles, requieren de baterías, las cuales tienen un periodo de utilidad mientras dispongan de carga. En algunos casos estos periodos pueden ser de minutos u horas, por otro lado, el hardware puede verse limitado por cuestiones de diseño del propio dispositivo para hacerlo más ligero y portable.
- **Limitaciones de los enlaces inalámbricos:** Los enlaces inalámbricos por su naturaleza pueden presentar anchos de banda limitados, además de efectos de desvanecimiento, interferencias del medio y ruidos. Por otro lado, estos enlaces dependen directamente del hardware para su funcionamiento, por ejemplo, al funcionar con baterías de bajo nivel de carga, la transmisión del enlace suele ser a baja potencia por lo que el rendimiento en esta tarea puede verse reducido con el fin de alargar el tiempo de funcionamiento con baterías.
- **Arquitectura descentralizada:** En estas redes, cualquier dispositivo que la integre puede ser dispositivo final y enrutador a la vez, es decir, puede recibir y enviar información a los dispositivos que considere necesario, dejando de lado la necesidad de un dispositivo central que enrute el tráfico por la ruta preferida.

1.3.6 CLASIFICACIÓN DE LAS REDES AD-HOC

Las redes Ad-hoc se clasifican en 3 subgrupos, redes MANET, redes VANET y redes FANET, siendo estas últimas un subgrupo de las redes VANET [5], esto se puede apreciar en la Figura 1.10.

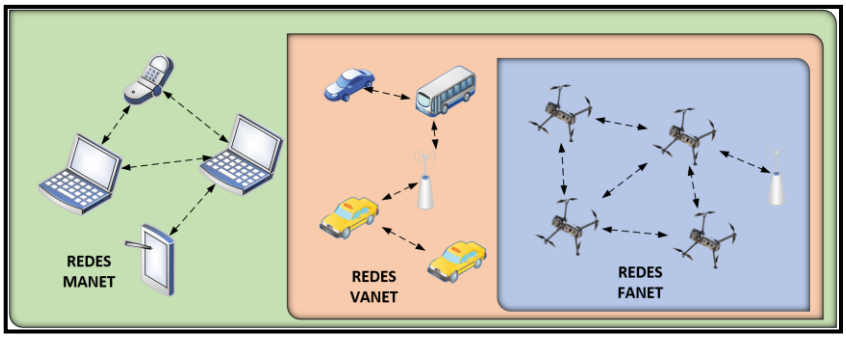

Figura 1.10. Clasificación de las redes Ad-hoc, basado [5].

- **Redes MANET:** Las redes móviles Ad-hoc o MANET (Mobile Ad-hoc Network), como se muestra en la Figura 1.11, son redes en donde los dispositivos que la conforman están conectados en modo Ad-hoc y además se encuentran en movimiento constante, dichos dispositivos no se limitan únicamente a computadores portátiles, sino también smartphones, tablets, PDA y demás dispositivos autónomos en movimiento constante que dispongan de capacidades de comunicación inalámbricas [32] [33].

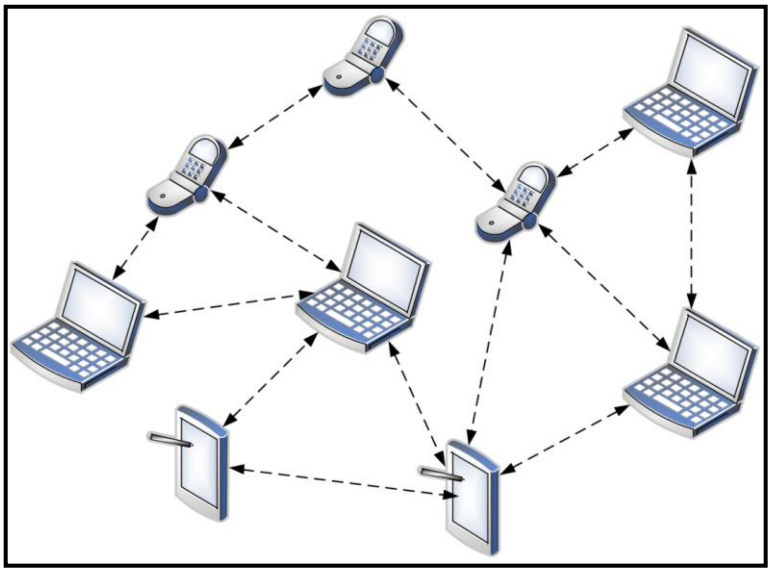

Figura 1.11. Redes Manet [33].

- **Redes VANET:** Las redes móviles vehiculares o VANET (Vehicular Ad-hoc Network) son redes Ad-hoc conformadas por equipos de carretera y vehículos, estando éstos últimos en posición fija o móvil, además, dichos dispositivos deben contar con capacidades de comunicación inalámbricas, convirtiéndose así en nodos de la red VANET [34].

  Si el vehículo se encuentra como un elemento fijo en una carretera recibe el nombre de dispositivo RSU (Road-Side Unit) y su función es la de enviar, recibir y retransmitir paquetes para poder aumentar la cobertura de la red, por otro lado, si

el vehículo en encuentra en movimiento se denomina OBU (On Board Unit) y puede comunicarse con dispositivos RSU o con dispositivos OBU [35].

Estas redes fueron desarrolladas con el objetivo de tener un sistema de transporte eficiente, seguro y libre de congestión, considerándose así parte fundamental de las Smart Cities, ya que los sistemas inteligentes de transporte (ITS) hacen uso de estas.

La comunicación en estas redes se muestra en la Figura 1.12 y puede ser entre vehículos (Vehicle-to-Vehicle "V2V"), vehículos e infraestructura (Vehicle-to-Infrastructure "V2I"), así como, infraestructura e infraestructura (Infrastructure-to-Infrastructure "I2I") [36].

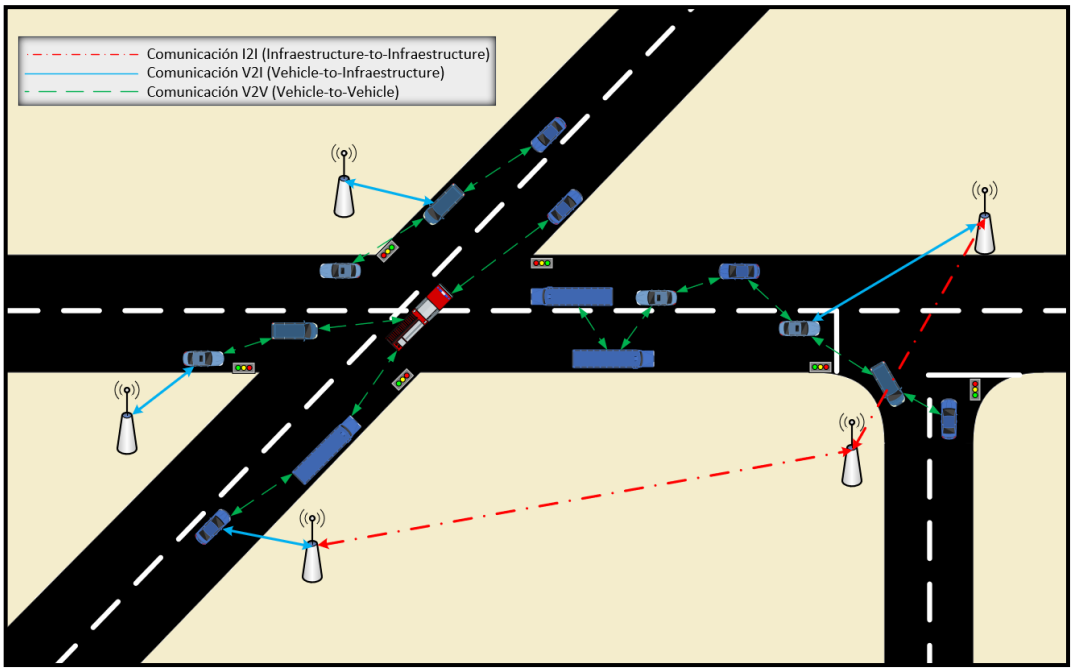

Figura 1.12. Comunicación en las redes VANET, basado [36].

- **Redes FANET:** Las redes FANET (Flying Ad-hoc Network) son redes Ad-hoc conformadas por dispositivos en vuelo, en posición estática o en libre movimiento, entre los dispositivos que conforman dichas redes están los UAVs y se analizarán a fondo más adelante, puesto que es el objeto de estudio del presente trabajo de titulación.

1.3.7 REDES FANET

Las redes FANET están conformadas por grupos de UAVs, en donde, cada uno de estos dispositivos es considerado un nodo entre los que se puede retransmitir la información, además, tiene que enviar y recibir la información de la posición de los demás UAVs, y, en caso de existir una estación en tierra también es necesario conocer la posición de esta [5].

Como se revisó en el apartado 1.3.3, los dispositivos UAVs tienen ciertas limitaciones al trabajar por separado, sin embargo, para poder superar esas limitaciones es necesario crear redes de varios UAVs, con el fin de aprovechar al máximo el potencial de estos dispositivos.

Estas redes permiten el despliegue de UAVs a escala masiva, así como a mayor altitud, ya que cada uno de los UAVs en vuelo se comunica con los dispositivos vecinos, permitiendo así la comunicación con UAVs más distantes, eliminando en cierta forma la restricción de cobertura, sin embargo, la altitud del dispositivo puede estar ligada directamente al hardware, por otro lado, en el caso de que uno de los dispositivos salga de operación, el tráfico puede redirigirse a través de otra ruta, tomando otro UAV el lugar del dispositivo que salió de operación [37].

Puesto que estas redes pueden ser desplegadas en situaciones en donde se requiera conectividad y otro tipo de redes sean inviables son de interés para su desarrollo, entre otras aplicaciones se destacan las siguientes [37]:

- Monitoreo del entorno en situaciones de desastres naturales como deslaves, inundaciones, terremotos, incendios forestales, entre otros, en donde las redes fijas estén inhabilitadas y el acceso por tierra sea limitado o inaccesible, las redes FANET pueden actuar como enlaces de comunicación para las zonas afectadas.
- Búsqueda y rescate en zonas o áreas extensas de difícil acceso, inaccesibles o de limitada cobertura celular.
- Servicios de emergencia que requieran la intervención por aire.
- Tareas como la video vigilancia y monitoreo de zonas extensas o peligrosas, mismas que puedan representar un riesgo para el operador e inviables por vía terrestre.

Además, estas redes pueden ser utilizadas en diferentes ámbitos como la retransmisión de información o inclusive pueden interconectar redes existentes, por otro lado, se puede desplegar una red FANET para realizar la recolección de datos en zonas donde existan sensores y estaciones de telemetría, las cuales, se puedan encontrar fuera del alcance de las centrales de gestión.

Pueden usarse como dispositivos de procesamiento y almacenamiento por aire, esto debido a que, los dispositivos cercanos a las estaciones base que requieren la información serán los primeros en distribuir la misma hacia su destino, sin embargo, los dispositivos lejanos pueden destinar parte de sus recursos propios al almacenamiento y procesamiento

de información, compartiendo resultados con los dispositivos vecinos cuando el canal se encuentre disponible [38].

1.3.8 CARACTERÍSTICAS DE LAS REDES FANET

Las redes FANET al estar conformadas por dispositivos en vuelo tienen como característica principal la alta movilidad y debido a esto, cambios en su topología en mayor grado que otras redes Ad-hoc, además, destacan las siguientes características adicionales [5]:

- **Movilidad en los nodos:** Al ser dispositivos desplegados en el aire, cada uno de estos puede volar libremente, cambiando su ubicación y rumbo más rápidamente en comparación con los dispositivos de otras redes Ad-hoc, por otro lado, la movilidad está ligada con el hardware disponible en el dispositivo, pudiendo desplazarse a mayor o menor altura con velocidad variable.
- **Localización de los UAVs:** Estos dispositivos al presentar características de alta movilidad, su localización debe ser actualizada de manera más rápida y constante puesto que la pérdida o demora de esta información puede derivar en choques entre dispositivos en vuelo o a su vez la pérdida completa de altitud, además, es necesario conocer la ubicación de los UAVs con el fin de evitar impactos directos con objetos presentes en la zona de despliegue de la red.
- **Cobertura de los nodos:** Cada dispositivo puede abarcar una cobertura extensa dependiendo de las capacidades del hardware a bordo, sin embargo, por estar en el aire tienen mayores ventajas que los dispositivos que transmiten en tierra, puesto que la señal puede alcanzar mayor distancia mientras no tenga obstáculos y el hardware lo permita.
- **Autonomía energética:** En las redes FANET, el consumo de energía es alto, esto debido a que el hardware disponible en los UAVs hasta cierto punto puede ser una limitante, ya que, para tener una mayor autonomía energética se requieren baterías de mayor capacidad, mismas que pueden llegar a tener un gran tamaño, sin embargo, esto haría al dispositivo pesado y de difícil maniobrabilidad, lo cual a su vez requeriría mayor energía para poder mantenerse en el aire, además, este consumo está también ligado a la transferencia de datos, por lo que, esta característica es crítica en el despliegue de la red y puede reducir el rendimiento de las comunicaciones para poder garantizar la supervivencia del dispositivo.
- **Línea de vista (LOS):** La comunicación para control y transferencia de información con los dispositivos es posible, siempre y cuando, se puedan divisar entre ellos, es decir tener línea de vista entre los dispositivos y los elementos de control en tierra.

En algunos casos, ciertos dispositivos tienen preconfigurado el retorno a cobertura de la estación en tierra o en su defecto regresar al lugar de despegue, lo cual es una medida de contrarrestar la pérdida de visión con la estación de control.

1.3.9 ARQUITECTURAS DE COMUNICACIÓN EN LAS REDES FANET

Para la transmisión de información dentro de estas redes, se definen 2 tipos de arquitecturas de comunicación, las cuales son:

- **UAV to UAV:** En esta arquitectura, la comunicación se realiza entre dispositivos UAVs, es decir cada UAV está conectado directamente con el UAV vecino, como se muestra en la Figura 1.13, permitiendo así la transferencia de datos y posicionamiento entre ellos. Esta comunicación puede darse a través de varios UAVs usando múltiples saltos entre estos, en donde, la ruta a seguir estará definida por el protocolo de enrutamiento, lo cual a su vez definirá también la tasa de transferencia de datos y el rendimiento de la red.

  Se requiere de una estación base, tal como un radio control o una tarjeta de desarrollo, que no necesariamente puede estar conectada directamente con todos los UAVs desplegados, sino que, puede tener la función de torre de control y estar ligada únicamente a un grupo pequeño de UAVs o a un determinado UAV, los cuales permitan actividades de control y monitoreo de la red desplegada [39].

  Este tipo de comunicación entre los dispositivos puede ser difícil de mantener, debido a la alta movilidad de los UAVs.

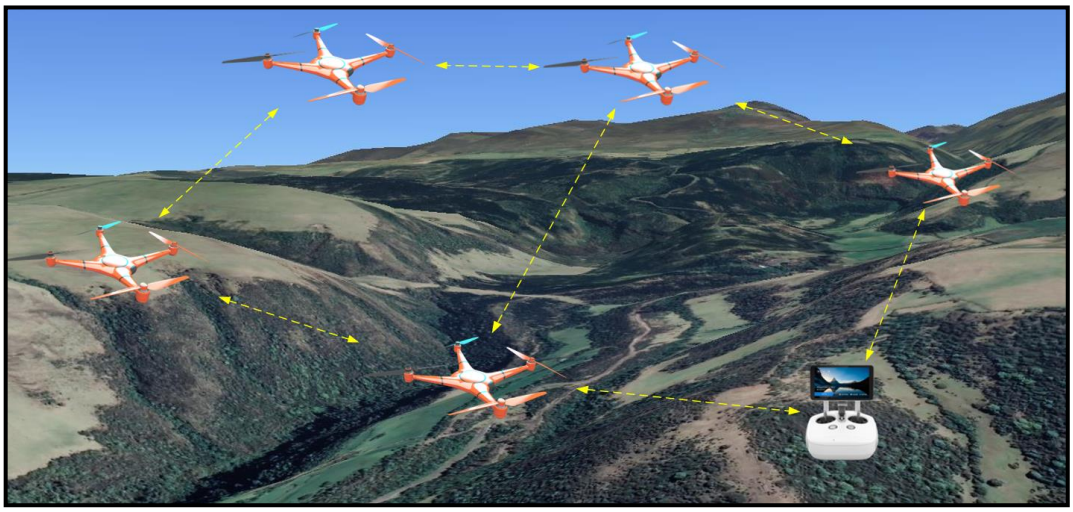

Figura 1.13. Arquitectura de comunicación FANET UAV to UAV [3].

- **UAV to Ground Station:** También conocida como UAV to Infrastructure, para este caso, como se muestra en la Figura 1.14, es necesaria la existencia de una estación base, por ejemplo: red celular o satelital, con la cual se debe conectar la red FANET

desplegada, para transmitir y recibir información de posicionamiento, servicios o datos que pueda proveer dicha red. Dentro de estas redes se puede sustituir la estación base que figura como estación de transferencia de datos en tierra por una estación ya sea celular o satélite, aumentando considerablemente la cobertura de operación. Sin embargo, el hardware a implementar puede ser mucho más costoso, en especial para la comunicación con enlaces satelitales [5].

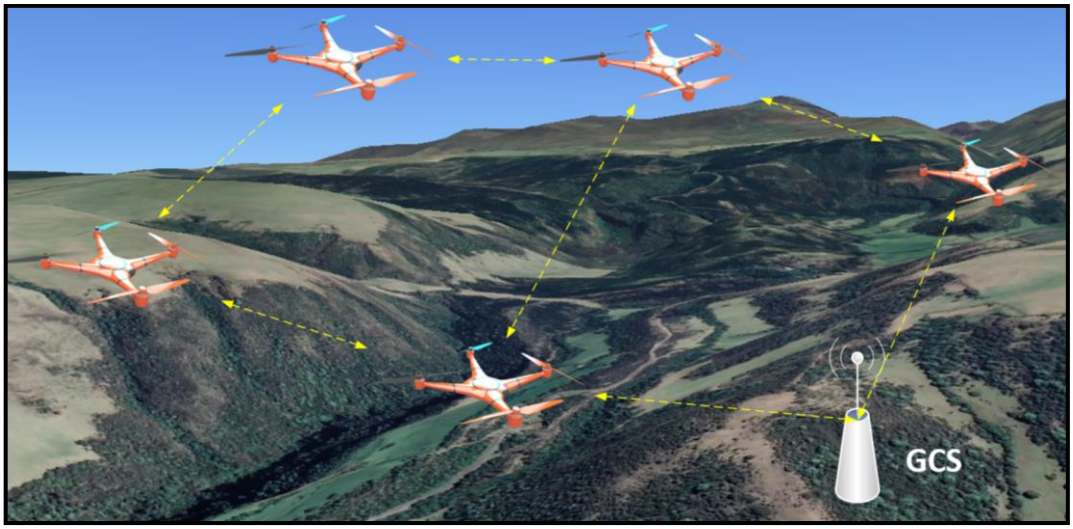

Figura 1.14. Arquitectura de comunicación FANET UAV to Ground Station [3].

En ambos casos es necesaria la presencia de una estación de control de posición fija o variable en tierra (GCS: Ground Control Station) hacia la cual van a converger los datos desde y hacia los UAVs desplegados. Sin embargo, es necesario recalcar que para que una red de múltiples UAVs sea considerada FANET, debe tener una comunicación Ad-hoc, por lo que la comunicación no se realiza directamente desde el GCS con todos los dispositivos UAV desplegados, si no, solamente con un grupo de UAVs que puede cambiar su posición debido a la característica de alta movilidad de dichas redes. Por lo que, en caso de falla del dispositivo, puede ser reemplazado por otro UAV disponible para la comunicación con tierra, cumpliendo así la característica principal de estas redes al no tener una infraestructura centralizada [3].

1.3.10 MEDIOS DE COMUNICACIÓN EN LAS REDES FANET

Los medios de comunicación usados en las redes FANET, deben adaptarse a la característica de alta movilidad de las mismas, así como, al tipo de arquitectura desplegada, sea esta UAV-to-UAV o UAV-to-Infraestructure, por lo que, dichos medios se pueden clasificar acorde a su rango de cobertura en medios de comunicación de corto o largo alcance [40].

- **Medios de comunicación de corto alcance:** En estos medios se encuentran las tecnologías que permiten la comunicación inalámbrica por medio del uso de bandas libres, permitiendo la transferencia de información desde algunos centímetros de distancia hasta algunos cientos de metros, como es el caso de la tecnología Wi-Fi, ZigBee y Bluetooth.
    - **Wi-Fi:** Los estándares IEEE 802.11, con sus variantes IEEE 802.11 a/b/g/n/ac y frecuencias de operación de 2.4 GHz, 3.6 GHz, 5 GHz y 60 GHz, son la tecnología de comunicación inalámbrica preferida para el despliegue de redes WLAN, ya que, el hardware compatible con esta tecnología, se encuentra presente en la mayoría de dispositivos UAVs disponibles en el mercado. Además, inicialmente su rango de cobertura puede ser limitado a unos cuantos metros, aumentando el rango por medio de la red FANET hasta varios kilómetros dependiendo de los dispositivos que la conformen [41].
    - **Bluetooth:** El estándar IEEE 802.15.1, opera en la banda libre de 2.4GHz, con cobertura de 10 a 100 metros y tasas de transferencia de datos de 1 Mbps a 3 Mbps, siendo esta última, la tasa obtenida con la tecnología Bluetooth 5.2 [42].
    - **ZigBee:** Con su estándar IEEE 802.15.4, es una tecnología de comunicación inalámbrica de banda libre de corto alcance (10-100m), bajo consumo energético y una baja tasa de transferencia de información (250kbps a 2.4GHz, 40kbps a 915MHz y 20kbps a 868MHz), por lo que la aplicación de esta tecnología, puede darse en escenarios críticos en donde la autonomía este por encima de la tasa de transferencia de información [43].
- **Medios de comunicación de largo alcance:** Pueden ser redes celulares, satelitales o WiMax, las cuales son utilizadas en la arquitectura de comunicación UAV-to-Infrastructure, permitiendo a los dispositivos que conforman la red FANET conectarse directamente con la estación base disponible en el sector de despliegue, estableciendo así una comunicación desde unos cientos de metros hasta algunos kilómetros.
    - **WiMax (WorldWide Interoperability for Microwave Access):** Este tipo de tecnología es parte del estándar 802.16, permite la comunicación a largas distancias con tasas de transferencia de datos de hasta 75 Mbps, para ello se hace uso de bandas licenciadas y no licenciadas desde 2.3 GHz a 5.8 GHz (excluyendo la banda libre de 2.4 GHz), dichas bandas pueden estar o

no disponibles para su uso dependiendo de las regulaciones del país en donde se aplique [44] [45].

- **Redes celulares:** Estas redes permiten la transmisión de voz y datos, haciendo uso de ciertos estándares para la transferencia de datos, tales como: GPRS (General Packet Radio Service), UMTS (Universal Mobile Telecommunications System), LTE (Long Term Evolution) y 5G-NR (Fifth Generation-New Ran). En la actualidad es la tecnología de comunicación más utilizada, alcanzando velocidades teóricas de hasta 10 Gbps en las redes 5G, pudiendo ser implementadas como medio de comunicación en las redes FANET, siempre y cuando, el hardware del UAV pueda establecer un enlace de comunicación con las mismas, además de aprovechar todas las características disponibles de la red. Por otro lado, para la comunicación con los dispositivos, en las áreas de cobertura se despliegan estaciones base, conformando lo que se denominan células, siendo dichas estaciones las que van a transmitir la información desde los UAVs hacia los centros de control. Para el uso de esta tecnología es necesario usar la infraestructura de operadores celulares, lo cual podría generar un costo significativo en el despliegue de la solución [46].
- **Enlaces satelitales:** Para este tipo de comunicación se utiliza la infraestructura satelital conformada por satélites en órbita y estaciones terrenas, en donde la tasa de transferencia de datos está dada por el transpondedor presente en el satélite en órbita. Este tipo de comunicación es la de mayor cobertura en especial en áreas remotas, a su vez, es la de mayor costo en cuanto a implementación.

El medio de comunicación a implementarse en la red FANET es crucial para la selección de los UAVs, ya que el hardware a bordo del dispositivo puede variar y debe ser compatible con la tecnología a implementar, además que la selección de esta puede estar ligada directamente al consumo de energía por los dispositivos.

1.4 PROTOCOLOS DE ENRUTAMIENTO

Los protocolos de enrutamiento son parte esencial de la comunicación, puesto que son el conjunto de reglas que se encargan de encaminar el tráfico desde un dispositivo origen a un dispositivo destino. El enrutamiento tiene como objetivo el hallar el mejor camino posible, o en caso de requerir, direccionar el tráfico por la ruta más óptima si la ruta

conocida no se encuentre disponible. Esto se logra cuando los dispositivos comparten la información de enrutamiento conocida para poder construir las tablas de enrutamiento.

Se pueden distinguir 2 clases de enrutamiento:

- **Enrutamiento estático:** Las tablas de enrutamiento no cambian una vez que el dispositivo la conoce, es decir, existirá una sola ruta para llegar desde un dispositivo A hacia un dispositivo B, por otro lado, estas rutas pueden ser agregadas manualmente [47].
- **Enrutamiento dinámico:** En este caso, las tablas de enrutamiento son actualizadas constantemente por los dispositivos, en base a los protocolos de enrutamiento sobre los cuales estén configurados para operar. De existir un cambio en la red, dicho cambio es actualizado en las tablas de enrutamiento, por lo que pueden existir diferentes caminos para poder llegar desde un dispositivo A hacia un dispositivo B [48].

Para determinar la mejor ruta a seguir, es necesario el uso de los algoritmos de enrutamiento, los cuales se encargan de decidir el mejor camino a seguir para poder entregar el paquete de información al dispositivo destino, estos algoritmos son utilizados a nivel de red y los más utilizados son:

- **Algoritmo vector distancia:** Para la selección de una ruta en la comunicación entre dos dispositivos, se utiliza como métrica la cantidad de saltos desde el nodo origen hasta el nodo destino, es decir, el número de dispositivos por los que debe viajar el paquete de datos hasta llegar a su destino. Para ello, un nodo envía la tabla de enrutamiento al nodo vecino hasta completar la comunicación, la red que se considerará óptima es aquella que tenga el menor número de saltos [49].
- **Algoritmo de estado de enlace:** Este algoritmo es también conocido como SPF (Short Path First, en español, primero la ruta más corta), en donde para la comunicación entre dos dispositivos no basta con solo elegir el camino más corto, si no que busca el camino de menor costo, esto se logra mediante el intercambio de paquetes LSP (Link State Packet, Paquete de estado de enlace), los cuales contienen información acerca de los nodos vecinos, tal como el ID, tipo de enlace y vector distancia, mediante la comparación de dicha información se obtiene la mejor ruta a utilizar en la comunicación [50] [47].

1.4.1 TIPOS DE PROTOCOLOS DE ENRUTAMIENTO EN LAS REDES FANET

Las redes FANET al tener la característica de alta movilidad, la selección adecuada del protocolo de enrutamiento a implementarse es crítica, esto con el fin de mantener estable el enlace de comunicación entre dispositivos. Por otro lado, no se encuentran definidos protocolos de enrutamiento específicos desarrollados únicamente para las redes FANET, por lo que se utilizan protocolos de las redes MANET [4], debido a esto, se pueden clasificar los protocolos de enrutamiento en los siguientes [38]:

- Protocolos estáticos.
- Protocolos reactivos.
- Protocolos proactivos.
- Protocolos híbridos.
- Protocolos basados en la posición geográfica.
- Protocolos jerárquicos.

Sin embargo, los principales protocolos de estudio asociados a la investigación de las redes FANET son los protocolos reactivos, proactivos e híbridos, razón por la cual será el objeto de estudio en el presente trabajo.

- **Protocolos Reactivos:** También se los conoce como protocolos bajo demanda o protocolos pasivos, esto debido a que, la tabla de enrutamiento es actualizada solo si se tiene información para enviar. Para las redes FANET se tienen los protocolos reactivos AODV (Ad-hoc On-demand Distance Vector), DSR (Dynamic Source Routing) y AOMDV (Ad-hoc On-demand Multipath Distance Vector). Sin embargo, se puede generar una sobrecarga en la red por el incremento de tamaño de las cabeceras en los paquetes transmitidos, por otro lado, los tiempos de establecimiento de la comunicación pueden aumentar debido al descubrimiento de la ruta y posterior actualización de las tablas de enrutamiento [3].
- **Protocolos Proactivos:** La principal característica de estos protocolos es que las tablas de enrutamiento se actualizan y almacenan en cada UAV. Para las redes FANET se tienen los protocolos OLSR (Optimized Link State Routing) y DSDV (Destination-Sequenced Distance Vector). Se debe considerar que, tras la implementación de este tipo de protocolos, puede presentarse un alto consumo en ancho de banda debido a la transmisión de las tablas de enrutamiento almacenadas por cada uno de los dispositivos [3].

- **Protocolos híbridos:** Estos protocolos son una combinación de características de los protocolos reactivos y proactivos, como es el caso de HWMP (Hybrid Wireless Mesh Protocol) [4].

1.4.2 PROTOCOLOS DE ENRUTAMIENTO EN LAS REDES FANET

Como se mencionó anteriormente, la elección de un protocolo de enrutamiento es crucial para mantener la adecuada comunicación dentro de la red FANET. Sin embargo, se utilizan ciertos protocolos de las redes MANET, en esta sección se describirán los protocolos de enrutamiento aplicables a las redes FANET.

1.4.2.1 Ad Hoc On-Demand Distance Vector (AODV)

Como su nombre lo indica, este protocolo del tipo reactivo, utiliza el algoritmo vector distancia para su funcionamiento, es decir, se genera una tabla de enrutamiento hacia un destino únicamente cuando es necesaria la comunicación, estableciéndola solo entre los nodos que la requieren. Dicha tabla de enrutamiento mantiene únicamente la información del nodo destino y el último nodo por el cual han pasado los paquetes de información.

Para establecer una ruta, se utilizan los siguientes mensajes de control [51]:

- **Route Request (RREQ):** Este mensaje es transmitido a través de broadcast y es generado por un nodo origen que requiere comunicarse con otro nodo destino. Este mensaje será enviado a través de nodos intermedios hasta alcanzar dicho nodo destino, en caso de que un nodo reciba dos mensajes RREQ solo mantendrá aquel que tenga menor número de saltos, descartando el otro.
- **Route Reply (RREP):** Es transmitido por el nodo destino que ha recibido un mensaje RREQ, por lo que la transmisión se realizará de manera unicast hacia el nodo origen que generó el paquete RREQ, generando así una ruta definida.
- **Route Error (RERR):** Este tipo de mensajes puede ser generado por los nodos que son parte de una ruta para informar de una desconexión.

1.4.2.2 Dynamic Source Routing (DSR)

Este protocolo de tipo reactivo, busca una ruta desde un nodo origen hacia un nodo destino de manera periódica. Utiliza los mismos paquetes de control que AODV, sin embargo, en este caso, la información de todos los nodos intermedios en la ruta hacia un nodo destino y a su vez, la información del nodo origen, es almacenada en las tablas de enrutamiento enviadas al nodo vecino, conociendo cada nodo intermedio toda la ruta atravesada [30].

DSR consta de dos fases identificadas para establecer una ruta, las cuales son las siguientes [52]:

- **Fase de descubrimiento:** Esta fase es solicitada bajo demanda y se transmite un paquete RREQ desde el nodo origen utilizando broadcast, el cual contiene el identificador del nodo destino y este a su vez registrará todos los nodos por los que atraviese hasta llegar al nodo destino. Si un nodo intermedio recibe el paquete, este lo pasa a su nodo vecino con toda la información incluida su propio lugar en la ruta. Una vez que un paquete RREQ ha llegado hacia un nodo destino, este determina la mejor ruta por medio del algoritmo vector distancia y transmite un paquete RREP de manera unicast hacia el nodo origen por la ruta definida.
- **Fase de mantenimiento de ruta:** Al transmitirse paquetes RREQ de manera periódica se actualiza constantemente el estado de la ruta, en caso de haber una desconexión se generará un paquete RERR con lo que se retornará a la fase de descubrimiento.

1.4.2.3 Ad-Hoc On-Demand Multipath Distance Vector (AOMDV)

Este protocolo es una variante del protocolo AODV, el cual mantiene múltiples rutas durante la fase de descubrimiento, superando así problemas de latencia, además de que dichas rutas pueden servir para el transporte de información o como respaldo en caso de una falla en la ruta principal.

Para el descubrimiento de múltiples rutas, el nodo origen envía paquetes RREQ en la red por medio de broadcast a través de los nodos intermedios hacia un nodo destino, en caso de que un nodo, sea intermedio o destino, reciba dos mensajes RREQ de nodos vecinos diferentes, no lo descarta como en el caso del protocolo AODV si no que lo mantiene, sin embargo, si un nodo intermedio recibe un mensaje RREQ que lo señale como nodo predecesor, es decir, que haya sido utilizado como nodo intermedio previamente, lo descartará con el fin de evitar lazos en la red [53].

Una vez que el nodo destino reciba los paquetes RREQ de distintos nodos, enviará paquetes RREP por medio de las múltiples rutas descubiertas previamente, estableciendo la comunicación entre nodo origen y destino, en donde la ruta de mayor prioridad será aquella que tenga menor número de saltos y, en caso de una desconexión de esta ruta, se utilizarán las rutas siguientes en orden ascendente del número de saltos [54].

1.4.2.4 Optimized Link State Routing (OLSR)

Basado en el algoritmo de estado de enlace, este protocolo proactivo utiliza nodos de tipo MPR (Multi Point Relay), los cuales son los únicos que pueden realizar broadcast dentro de la red con el fin de enviar mensajes de control para el establecimiento de una ruta de comunicación.

Los mensajes de control propagados dentro de la red son [51]:

- **TC (Topology Control):** Este tipo de paquete es enviado periódicamente en la red para actualizar constantemente las tablas de enrutamiento. En un protocolo de enrutamiento LSR (Link State Routing), este tipo de paquete es enviado por todos los dispositivos que conforman la red, sin embargo, para el caso del protocolo OLSR es enviado únicamente por los nodos MPR, con el fin de reducir la cantidad de paquetes enviados a la red y evitando la sobrecarga.
- **HELLO:** Este mensaje es generado por todos los nodos de la red hacia los nodos vecinos ubicados a 1 o 2 saltos de distancia y es usado para la selección de MPR, descubrimiento de vecinos y verificación del estado del enlace.

1.4.2.5 Destination-Sequenced Distance Vector (DSDV)

Este protocolo de enrutamiento de tipo proactivo utiliza el algoritmo de Bellman-Ford, en donde cada nodo de la red actualiza su tabla de enrutamiento por medio de actualizaciones periódicas cada 15 segundos, además, se tienen dos formas de actualización de la tabla de enrutamiento [55]:

- **Actualización completa:** Se presenta cuando existe un cambio considerable en la topología desplegada y se envía toda la información de las tablas de enrutamiento hacia los nodos con el fin de conocer el estado real de la red. Este escenario se puede presentar con mayor frecuencia con nodos en movimiento constante.
- **Actualización incremental:** En caso de existir un cambio menor en la red, se actualizan únicamente los datos requeridos para informar de dicho cambio, esto se realiza en las tablas obtenidas a partir de la última actualización completa.

En las tablas de enrutamiento se tienen datos como la dirección destino y el número de saltos, además, se utiliza una versión modificada del enrutamiento por vector distancia para establecer la mejor ruta. Esto se logra, agregando un número de secuencia el cual es generado por el nodo destino.

Dicho número de secuencia puede representar la existencia de un enlace entre nodos, es decir, si este número es par, la comunicación entre dos nodos se ha establecido correctamente, mientras que, si es impar, no existe comunicación entre los nodos. Por ende, en caso de ser un nodo intermedio de una ruta establecida, el protocolo deberá buscar una nueva ruta para llegar al destino requerido.

Por otro lado, el número de secuencia, es considerado el mejor si este es el más reciente, en caso de existir dos números de secuencia similares, la selección de la mejor ruta se establecerá acorde al número de saltos, el cual mientras sea menor es mejor [51] [55].

1.4.2.6 Hybrid Wireless Mesh Protocol (HWMP)

Inicialmente definido en el estándar IEEE 802.11s y utilizado para redes inalámbricas de tipo mesh, este protocolo recibe la clasificación de tipo híbrido, debido a que presenta características tanto reactivas como proactivas para la selección de una ruta [4].

La parte reactiva se produce cuando el descubrimiento de la ruta se realiza bajo demanda siguiendo los lineamientos del protocolo AODV revisado previamente en la Sección 1.4.2.1 y que es aplicado a redes con alta movilidad. Por otro lado, para la parte proactiva se utiliza el protocolo TBRP (Tree Based Routing Protocol), siendo este último, un protocolo aplicado a las redes inalámbricas de gran alcance, mismo que permite el descubrimiento de nodos vecinos. Esto se logra, ya que se define la comunicación entre los nodos utilizando redes jerárquicas de tipo padre-hijo, conformando una red en forma de árbol, en donde todos los nodos hijos van a estar conectados hacia un nodo raíz por medio de nodos intermedios denominados padres [56].

Para evitar bucles en la topología, se utilizan números de secuencia basados en el nodo destino, de manera similar al protocolo DSDV. Además, se tienen los siguientes mensajes de control: RANN (root announcement), PREQ (path request), PREP (path reply), PERR (path error).

Para el descubrimiento de los nodos vecinos es necesaria la implementación de protocolos de gestión entre nodos adyacentes (Peer Link Management Protocol). Para ello se utilizan

los métodos RANN o PREQ, en donde, se creará una ruta mediante la creación de “árboles” estableciendo la comunicación desde el nodo raíz, el cual permite la comunicación de toda la red con el exterior, hacia el nodo destino y con cada uno de los nodos intermedios [57].

- **Proactive Root Announcement (RANN):** En este método el nodo raíz emite periódicamente mediante broadcast mensajes RANN con el objetivo de actualizar el número de secuencia y la métrica a través de los nodos. Este mensaje es recibido por un nodo denominado MP (Mesh Point), el cual, en caso de tener información para enviarlo al nodo raíz, actualiza los datos y envía un mensaje PREQ de manera unicast, generando una generando una ruta en dirección desde el nodo intermedio hacia el nodo raíz. Una vez el nodo raíz ha recibido el mensaje PREQ, responde hacia el nodo MP con un mensaje PREP estableciendo de esta manera una ruta bidireccional.
- **Proactive Path Request (PREQ):** Para este método, el nodo raíz emite mensajes PREQ por medio de broadcast, el cual contiene un número de secuencia único, cuando un nodo MP recibe este mensaje actualiza la información contenida como la métrica. Almacena la tabla de enrutamiento generada y retorna el mensaje hacia el nodo raíz, generando así una ruta unidireccional desde el nodo MP hacia su nodo raíz. Recibido el paquete, el nodo raíz determinará la necesidad de comunicación bidireccional con el nodo MP, en caso de requerirse, se envía un mensaje PREP desde el nodo raíz hacia el nodo MP. La comunicación bidireccional se establecerá únicamente cuando existan datos a transmitir desde un nodo MP.

El resumen de los protocolos de enrutamiento descritos en esta sección, se detallan en la Tabla 1.1.

Tabla 1.1. Resumen de los tipos de protocolos de enrutamiento de las redes FANET.

Protocolo	**Tipo de protocolo**	**Algoritmo**	**Mensajes de control**	**Características**
AODV	Reactivo	Vector distancia	• Route Request (RREQ) • Route Reply (RREP) • Route Error (RERR)	• La tabla de enrutamiento contiene información del nodo destino y el último nodo tomándolo como origen. • Una única ruta establecida para la comunicación entre el nodo origen y destino
DSR	Reactivo	Vector distancia	• Route Request (RREQ) • Route Reply (RREP)	• La tabla de enrutamiento mantiene la información de todos los nodos de la ruta hacia el nodo destino.

			• Route Error (RERR)	• Una única ruta establecida para la comunicación entre el nodo origen y destino
AOMDV	Reactivo	Vector distancia	• Route Request (RREQ) • Route Reply (RREP) • Route Error (RERR)	• La tabla de enrutamiento contiene información del nodo destino y el último nodo por el cual el paquete atravesó, tomándolo como nodo origen. • Múltiples rutas establecidas hacia el nodo destino.
OLSR	Proactivo	Estado de enlace	• Topology Control (TC) • Hello	• Las tablas de enrutamiento son actualizadas periódicamente por medio de la transmisión de mensajes TC a través de broadcast. • Una única ruta establecida para la comunicación entre el nodo origen y destino
DSDV	Proactivo	Vector distancia	• Hello	• Las tablas de enrutamiento pueden actualizarse completa o parcialmente. • Utiliza un número de secuencia basado en el nodo origen para la selección de la mejor ruta. • Una única ruta establecida para la comunicación entre el nodo origen y destino
HWMP	Híbrido	Vector distancia	• Root Announcement (RANN) • Path Request (PREQ) • Path Reply (PREP) • Path Error (PRERR)	• Se basa en las características del protocolo AODV para su parte reactiva, el protocolo TBRP para la parte proactiva, además de características del protocolo DSDV para evitar bucles. • Generación de redes jerárquicas de tipo padre-hijo en donde las tablas de enrutamiento son actualizadas usando los métodos RANN o PREQ. • Una única ruta establecida para la comunicación entre el nodo origen y destino.

2. METODOLOGÍA

En la presente sección se detalla el diseño e implementación de la red FANET, partiendo desde la introducción del software de simulación, su implementación y la codificación del escenario de pruebas, así como los protocolos de enrutamiento.

Para el desarrollo del presente trabajo se ha obtenido la información a partir de la investigación en base a artículos publicados en revistas de renombre de la comunidad científica, además de, los manuales de cada uno de los programas utilizados, por otro lado, se realizará un análisis experimental de los resultados obtenidos en las simulaciones de los diferentes escenarios, tanto de UAVs en posición estática como de UAVs en movimiento.

Inicialmente se da una introducción del software NS-3 (Network Simulator 3), su instalación, complementos como NetAnim e integración con Eclipse, posteriormente se detallan los escenarios de simulación y su implementación.

El software de simulación se implementará dentro de una máquina virtual utilizando el software de simulación VMware en su versión 17.0, como sistema huésped se utilizará la distribución de Linux denominada Ubuntu en su versión 22.04, el uso de una distribución de Linux es mandatorio puesto que es el sistema operativo requerido para la instalación y despliegue del software de simulación NS-3 [58].

Como sistema host se tiene Windows 10 ejecutándose sobre una computadora portátil.

Los recursos de software necesarios para este trabajo son:

- Ubuntu 22.04
- NS-3 3.34
- Wireshark
- ECLIPSE IDE for C/C++ Developers

Por otro lado, cabe mencionar que se ha utilizado el siguiente recurso de hardware.

- Laptop Core i7 8va generación, 16 GB de RAM, 1 TB HDD.

2.1 SOFTWARE DE SIMULACIÓN NS-3

Software de uso libre destinado principalmente a la investigación y uso educativo, permite la simulación de diferentes ambientes de red y arquitecturas con múltiples dispositivos, nodos e interfaces, así como varias tecnologías entre las cuales se pueden destacar las

redes celulares, redes PTP, redes Wi-Fi, etc. [59]. Al ser un dispositivo licenciado bajo GNU GPLv2 (General Public License) permite el uso de múltiples bibliotecas, además de la vinculación con programas externos usados en simulaciones en tiempo real o como complemento, tal es el caso de ECLIPSE que puede integrarse como IDE (Integrated Development Environment) externo.

Para la implementación de scripts para ambientes de simulación se pueden usar los lenguajes de programación C++ y Python, en donde al momento de la escritura es necesario considerar que NS-3 se basa en la simulación de eventos discretos, por lo que, va a seguir las instrucciones del script en base a una línea de tiempo determinada [60].

2.2 INSTALACIÓN DE NS-3

Como se mencionó anteriormente, para la instalación de NS-3 es necesario utilizar Linux, por lo que para el presente trabajo se hace uso de una máquina virtual en VMware.

Previo a la instalación y acorde al manual oficial [61], se mencionan algunos prerrequisitos dependiendo del sistema operativo sobre el que se instale el simulador, sin embargo, al usar la versión 22.04 de Ubuntu, muchos de los módulos ya se encuentran preinstalados por lo que se consideran únicamente qmake y el compilador g++ con los siguientes comandos:

```
sudo apt install qtbase5-dev qt5-qmake -y
sudo apt install build-essential -y
```

Comando 1: Instalación qmake y compilador g++ en Ubuntu.

Una vez instalados los prerrequisitos, se siguen los siguientes pasos:

1. Utilizando la herramienta Terminal, se crea un directorio en cualquier carpeta preferida, en este caso se ha creado una en el escritorio de Ubuntu.

   ```
   cd Desktop
   mkdir NS3
   cd NS3
   ```

   Comando 2: Cambio de directorios para la instalación de NS-3.

2. Descargar manualmente el archivo comprimido para instalación desde la página web oficial de NS-3 y colocarlo en la carpeta creada en el paso 1, también se puede descargar utilizando el comando:

   ```
   wget https://www.nsnam.org/releases/ns-allinone-3.34.tar.bz2
   ```

   Comando 3: Descarga del archivo para la instalación de NS-3.

En donde, la versión del instalador puede variar acorde se requiera.

3. Descomprimir el archivo descargado manualmente o con el comando:

```
tar xfj ns-allinone-3.34.tar.bz2
```

Comando 4: Descompresión del archivo descargado.

4. Una vez descomprimido el archivo, se recomienda renombrar la carpeta obtenida por 3.34, para acortar el nombre del directorio, para ello se utiliza el siguiente comando:

```
mv ns-allinone-3.34 3.34
cd 3.34
```

Comando 5: Renombre de la carpeta creada y cambio de directorio

5. Dentro de la carpeta renombrada previamente creada, se debe utilizar el siguiente comando para la instalación del comprimido:

```
./build.py --enable-examples --enable-tests
```

Comando 6: Instalación de NS-3

6. Ingresar a la carpeta netanim-3.108 e iniciar la instalación de NetAnim.

```
cd netanim-3.108
qmake NetAnim.pro
```

Comando 7: Cambio de directorio e instalación de NetAnim.

7. Validar la instalación utilizando el archivo *test.py* ubicado en la carpeta ns-3.34 del directorio de instalación de NS-3 con el siguiente comando:

```
cd ..
cd ns-3.34
./test.py
```

Comando 8: Cambio de directorio y prueba de instalación.

Posterior a la instalación, para la ejecución de los scripts, es necesario compilarlos previamente, para ello, se deben almacenar dentro de la carpeta ns-3.34, la cual puede cambiar de nombre acorde a la versión utilizada, y, en dicha carpeta se encuentra el archivo *waf*, el cual es necesario para la compilación. Como ejemplo se considera al script denominado *scratch-simulator* dentro de la carpeta *scratch* y para su compilación se utilizan los siguientes comandos:

```
./waf --run scratch/scratch-simulator
```

Comando 9: Formato de comando para ejecución de scripts en NS-3.

2.2.1 INSTALACIÓN ECLIPSE IDE

La integración con herramientas como Eclipse IDE permiten un ambiente de programación más amigable con el usuario debido a la posibilidad de tener interfaz gráfica.

Para la instalación de Eclipse es necesario seguir los siguientes pasos [62]:

1. Descargar el instalador de Eclipse IDE a través del siguiente enlace: https://www.eclipse.org/downloads .
2. Instalar el paquete descargado con los siguientes comandos:

```
cd Downloads
tar -xvzf eclipse-inst-jre-linux64.tar.gz
cd eclipse-installer
./eclipse-inst
```

Comando 10: Instalación de Eclipse.

3. En la ventana de instalación que aparecerá seleccionar e instalar la opción: “Eclipse IDE for C/C++ Developers” como se muestra en la Figura 2.1.

Figura 2.1. Instalador de Eclipse IDE.

4. Abrir Eclipse y seleccionar la ruta por defecto como espacio de trabajo acorde a la Figura 2.2, esta ruta va a ser modificada más adelante para poder trabajar en conjunto con NS-3.

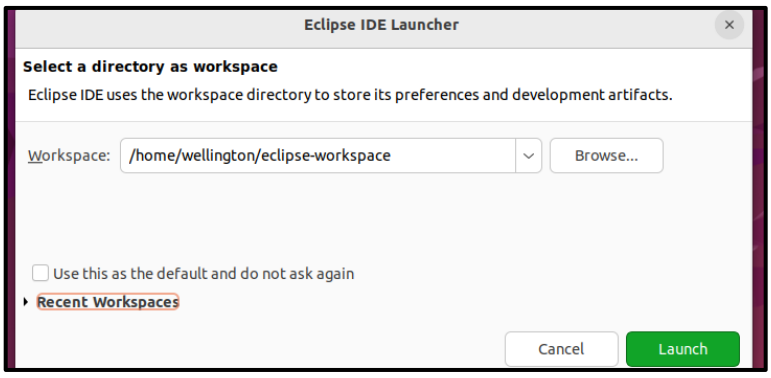

Figura 2.2. Inicio de Eclipse IDE.

5. Para la adecuada integración con NS-3 es necesario instalar Mercurial con el siguiente comando:

```
sudo apt-get install mercurial -y
```

Comando 11: Instalación de mercurial.

6. Instalar el plugin de Mercurial Eclipse en Eclipse IDE, acorde a los siguientes pasos [63]:
    a. Una vez abierto Eclipse, ir a la pestaña *Help*.
    b. Seleccionar *Install New Software*.
    c. Click en *Add*.
    d. Agregar el link: https://foss.heptapod.net/mercurial/mercurialeclipse-updatesite/-/raw/branch/default/p2 como repositorio para descargar el plugin, si se ha seleccionado adecuadamente la pantalla será como se observa en la Figura 2.3.

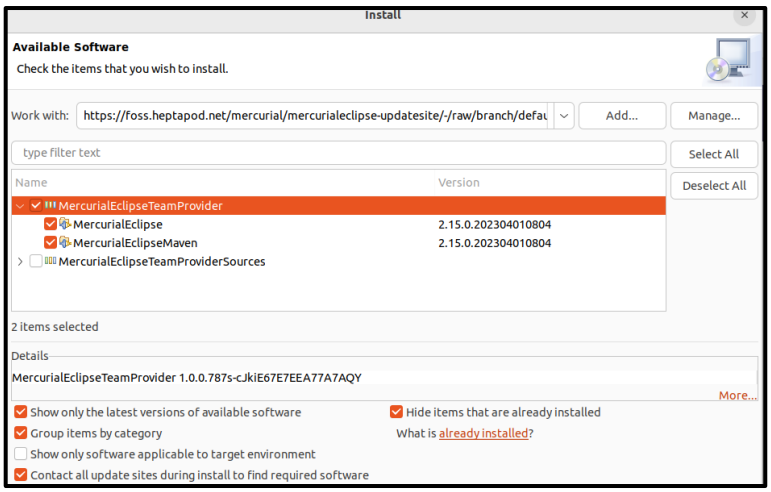

Figura 2.3. Instalación Mercurial Eclipse.

    e. Instalar el plugin aceptando términos y condiciones, posterior a esto, Eclipse IDE se reiniciará.

7. Crear un nuevo proyecto C++
    a. Seleccionar la opción *C Managed Build*.
    b. Colocar un nombre de proyecto, por ejemplo: NS-3.
    c. Seleccionar la ubicación de la carpeta de instalación de NS-3, misma que fue creada en el apartado 2.2.
    d. Seleccionar Cross GCC como se muestra en la Figura 2.4.

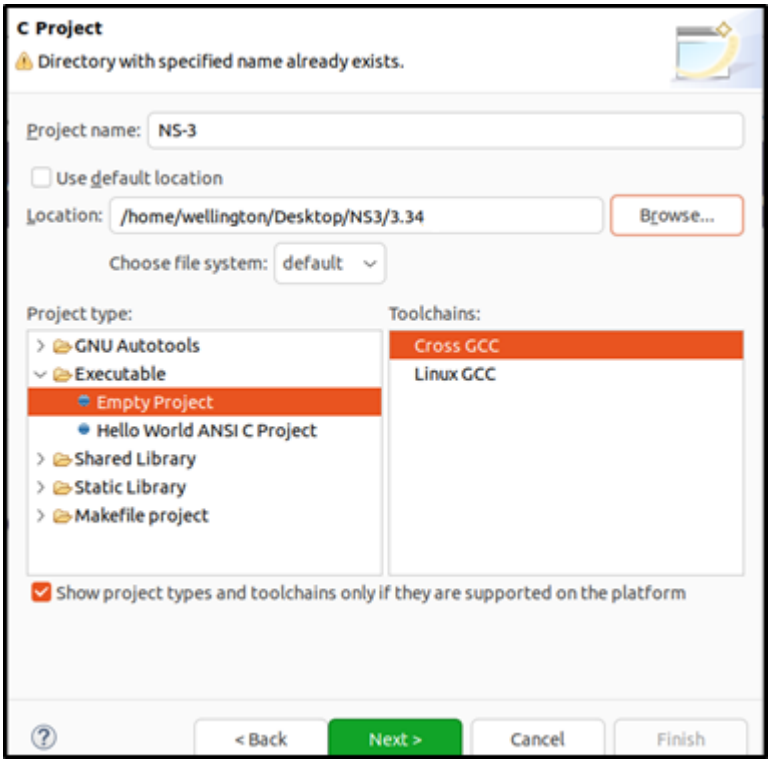

Figura 2.4. Creación de proyecto C++ en Eclipse.

e. Click en Next y no modificar las opciones por defecto.

f. En la siguiente ventana Cross GCC Command, hacer click en *Finish* y el proyecto será creado.

8. En el proyecto creado, hacer click derecho y seleccionar *Team*.
   a. Click en *Share Project*.
   b. Seleccionar Mercurial.
   c. Y mantener la ruta por defecto.
9. Para la ejecución de los scripts es necesario configurar el *WAF Builder*, para ello se debe hacer lo siguiente:
   a. Al igual que en el paso anterior, hacer click en el proyecto creado, seleccionar *Properties*.
   b. Seleccionar la opción *C/C++ Build*.
   c. Para la opción *Build Command* se debe utilizar la ubicación del ejecutable *waf* instalado por defecto en la carpeta raíz de NS-3, en este trabajo la ruta es: “/home/wellington/Desktop/NS3/3.34/ns-3.34/waf”
   d. Desmarcar la opción *Generate Makefiles automatically* y usar la dirección: “/home/wellington/Desktop/NS3/3.34/ns-3.34/build”, así como se muestra en la Figura 2.5.

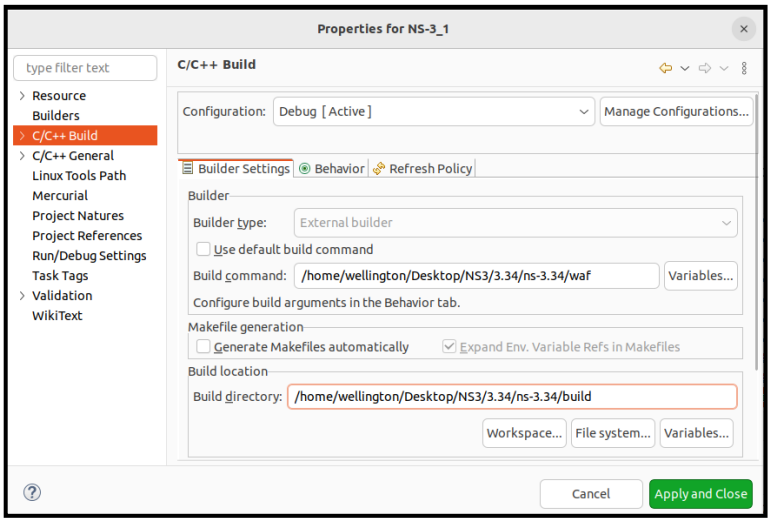

Figura 2.5. Configuración *WAF Builder*.

10. Para configurar un depurador, en la ventana principal de Eclipse, seleccionar la opción *Debug Configurations* ubicada en la pestaña *Run*.
    a. Seleccionar la opción *C/C++ Application.*
    b. En el apartado proyecto seleccionar el nombre del proyecto creado anteriormente.
    c. En el apartado *C/C++ Application*, click en *Search Project*, escribir y seleccionar *scratch-simulator*.

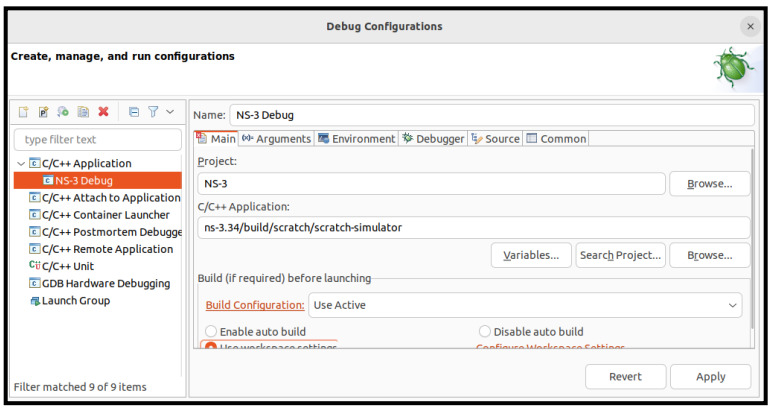

Figura 2.6. Configuración de depurador.

    d. En la ventana mostrada en la Figura 2.6, seleccionar la pestaña *Environment* y crear una nueva variable haciendo click en *Add*, como nombre de esta variable colocar: *LD_LIBRARY_PATH*.
    e. En el apartado *Value* colocar la ruta: “/home/wellington/Desktop/NS3/3.34/ns-3.34/build/lib”, finalmente click en *Apply* y *Close*, este proceso se muestra en la Figura 2.7.

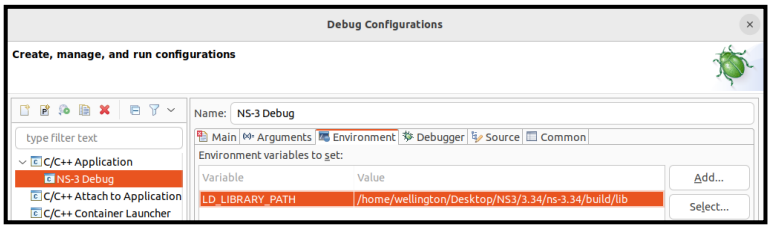

Figura 2.7. Configuración de nueva variable.

11. Configurar ejecutable externo, en este caso se configura WAF, para ello:
    a. En la ventana principal de Eclipse IDE en la pestaña *Run*, seleccionar la opción *External Tools* y después *External Tools Configurations*.
    b. Crear un nuevo programa y cambiar el nombre, por ejemplo: Run.
    c. En el apartado ubicación seleccionar la ruta de WAF, en este caso: "/home/wellington/Desktop/NS3/3.34/ns-3.34/waf".
    d. Como directorio de trabajo seleccionar la carpeta raíz de la instalación de NS-3, en este caso: "/home/wellington/Desktop/NS3/3.34/ns-3.34".
    e. Como argumento colocar: –-run ${string_prompt} –-vis.
    f. Si los pasos anteriores se realizaron correctamente se tiene una ventana como la Figura 2.8, para finalizar click en *Apply*.

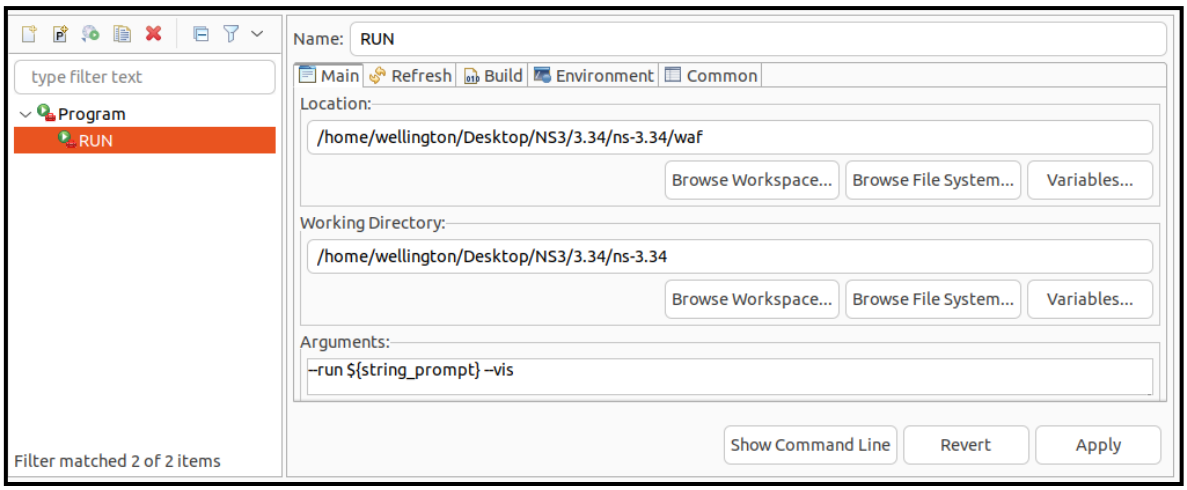

Figura 2.8. *External Tools Configurations*.

Si los pasos mencionados anteriormente se han seguido adecuadamente, será posible la escritura y ejecución de scripts a través de la interfaz de Eclipse.

En el presente trabajo se ha utilizado únicamente para escritura de scripts, puesto que herramientas como NetAnim se pueden ejecutar directamente desde la línea de comandos usando la herramienta Terminal.

2.2.2 INSTALACIÓN WIRESHARK

Wireshark es un analizador de paquetes de red, tiene como principal objetivo analizar los paquetes transmitidos en una red y mostrar la información relacionada al tipo de paquetes capturados, pudiendo capturar tráfico de red en tiempo real o por medio de archivos, además de, aplicar filtros para obtener únicamente la información de interés [64].

La información adquirida puede ser tratada para la resolución de problemas en la comunicación, identificación de problemas relacionados a la seguridad de la comunicación en la red, identificar la adecuada implementación de parámetros en la red, entre otros.

Wireshark admite el uso de archivos de otros capturadores de paquetes, tales como archivos *pcap*, *libpcap*, *pcap-ng*, entre otros. Para el presente trabajo se hace uso de archivo *pcap* generados directamente del código implementado en NS-3 [64].

Para la instalación de Wireshark se recomienda hacerlo a través de la PPA (Personal Package Archive) oficial con el fin de obtener actualizaciones estables del software. Se deben seguir los siguientes pasos:

1. Agregar el repositorio usando el siguiente comando:

```
sudo add-apt-repository ppa:wireshark-dev/stable
```

Comando 12: Agregar repositorio para la instalación de Wireshark.

2. Actualizar el repositorio y obtener actualizaciones de software:

```
sudo apt update && upgrade -y
```

Comando 13: Obtener actualizaciones de software.

3. Instalar Wireshark.

```
sudo apt install wireshark -y
```

Comando 14: Instalación de Wireshark

4. Ejecutar la aplicación con derechos de superusuario (comando sudo), especialmente en el caso de que se requiera realizar una captura de paquetes en tiempo real, esto es necesario para poder usar las interfaces del sistema operativo.

```
sudo wireshark
```

Comando 15: Ejecución de Wireshark.

En el caso de no iniciar la aplicación en modo superusuario, se podrá hacer uso de la misma solo con la carga de archivos generados previamente, pero no permitirá la captura de datos en tiempo real.

2.3 CARACTERÍSTICAS DE NS-3

NS-3 posee características especiales para poder vincular el simulador con aplicaciones en tiempo real o para poder analizar los resultados obtenidos a partir de una simulación, entre dichas características se pueden destacar las siguientes [65]:

- **NetAnim. -** Módulo integrado dentro de la instalación de NS-3, el cual permite la visualización del resultado de una simulación, así como información adicional de la red implementada, para ello, en la simulación se generan archivos XML que pueden ser cargados posteriormente en el módulo NetAnim [66]. En la Figura 2.9 se muestra el módulo en operación.

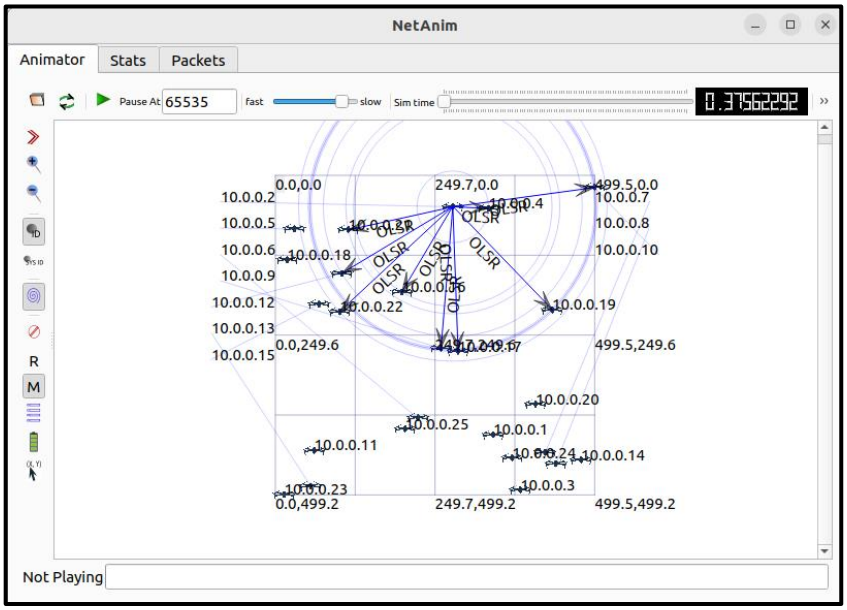

Figura 2.9. NetAnim [66].

- **Archivos *pcap*.** - Estos archivos generados en NS-3 contienen la información de los paquetes transmitidos entre nodos, permitiendo analizar información tal como: el nombre del protocolo, número de secuencia, IP origen y destino, etc. Dichos archivos pueden ser analizados utilizando Wireshark.
- **Rastreo de nodos.** - El simulador permite la generación de ficheros que pueden ayudar a determinar la ruta de los nodos, además de información clave como los datos enviados y recibidos.

2.4 CONCEPTOS PARA LA SIMULACIÓN EN NS-3

Para trabajar en la creación de scripts en NS-3 es necesario tener en cuenta los siguientes conceptos [67]:

- **Nodo. -** Denominación que reciben los dispositivos finales, tales como computadores, celulares, UAVs, etc. En el lenguaje C++ un nodo está representado por la clase *Node*, estos dispositivos pueden obtener funciones determinadas, dependiendo de los parámetros de operación detallados en el script, entre ellos se tienen: protocolos de red, direcciones IP, ID, etc.
- **NodeContainer. -** Clase a la que se refiere como un arreglo de nodos dentro del lenguaje C++.
- **Aplicación. -** Denominación que recibe una instrucción que se despliega sobre un nodo, similar a un programa de computadora que busca cumplir ciertas funciones o tareas a nivel de usuario, dentro del lenguaje C++ está representado por la clase *Application*, entre las instancias más utilizadas dentro del simulador NS-3 para representar la comunicación cliente/servidor se tiene: *UdpEchoClientApplication* y *UdpEchoServerApplication.*
- **Canal. -** Representado por la clase *Channel*, refiriéndose a la conexión física entre nodos, es decir, un enlace de comunicación entre dos o más dispositivos, por el cual se intercambia información, dichos enlaces pueden ser cableados o inalámbricos.
- **Dispositivos de red. -** Denominación que recibe el dispositivo que permite la conexión de un nodo hacia un canal, permitiéndole la comunicación dentro del medio representado en dicho canal, se encuentra asociado a la clase *NetDevice* dentro del lenguaje C++ y cumple las funciones de hardware y software de manera similar a una tarjeta de red en un computador.

  Además, se debe tener en cuenta que para cada canal es necesario un tipo específico de dispositivo de red, por ejemplo: para un canal de tipo Wi-Fi *WifiChannel,* es necesario un dispositivo de red representado por *WifiNetDevice.*
- **Topology helper. –** Requerido en la gestión de múltiples conexiones entre dispositivos, atribuye las características de la red implementada a cada nodo, así como a los periféricos que conformen dicho nodo, tal como una tarjeta de red.

2.5 ESTRUCTURA DE NS-3

NS-3 está compuesto de múltiples librerías y clases distribuidos en diferentes módulos, los cuales presentan una jerarquía de capas como lo muestra la Figura 2.10, por lo que los objetos definidos en las capas superiores pueden verse afectados por cambios en las capas inferiores [68].

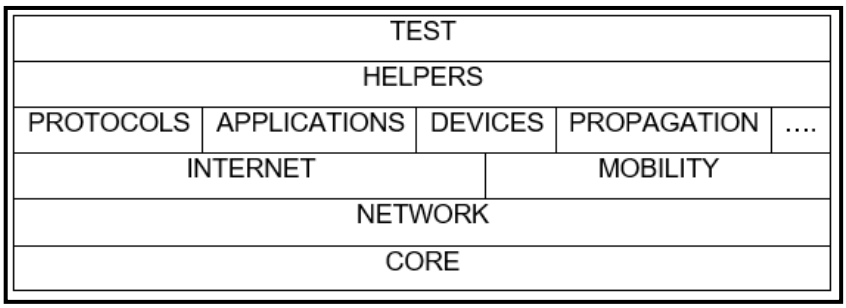

Figura 2.10. Estructura de capas en NS-3 [68].

Se pueden describir los siguientes módulos [68] [69]:

- **Core. -** Como su nombre lo indica, es la parte fundamental del simulador, en donde, se almacenan instrucciones básicas para la ejecución de las simulaciones, esto debido a que contiene componentes de todas las arquitecturas, protocolos y hardware, permitiendo así el manejo de los elementos mostrados en la Figura 2.11.

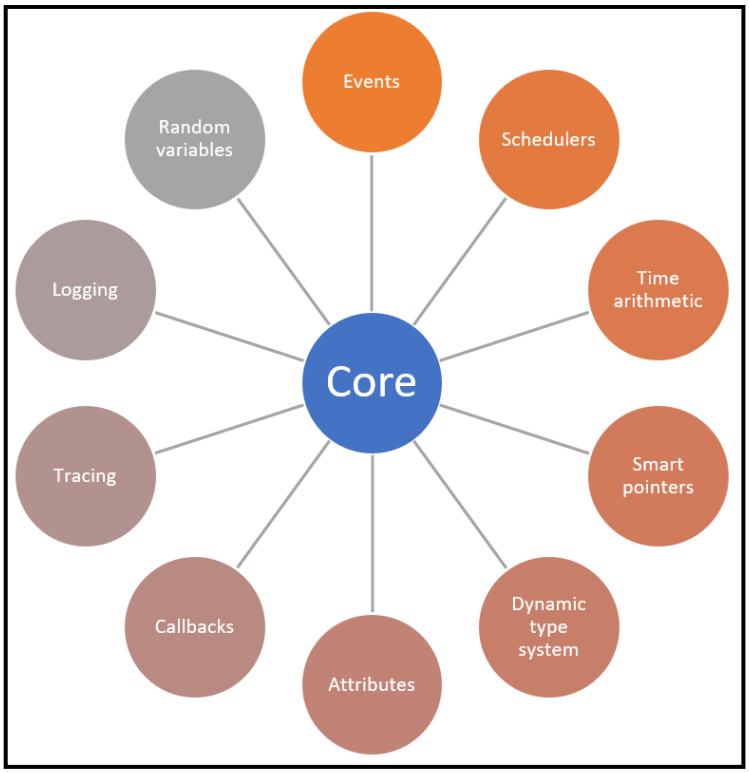

Figura 2.11. Componentes Core NS-3.

- **Variables aleatorias. -** La generación de estas variables es fundamental para los procesos de simulación, esto debido a que de no existir esta función se obtendrían siempre los mismos valores sin importar el escenario, dejando de lado la posibilidad de simular eventos cercanos a la realidad, como es el caso de la posición aleatoria de un nodo. Para el uso de esta función, el simulador NS-3 integra un generador pseudo aleatorio de números (PRNG), mismo que depende de la capacidad de procesamiento de la máquina host en donde está alojado el simulador [70].
- **Funciones hash.** – NS-3 contiene una función básica de generación de números hash, el cual es un algoritmo matemático que permite la codificación de un bloque de datos en una serie de caracteres de longitud fija. Se pueden implementar funciones hash de 32 y 64 bits de un bloque de datos o líneas de caracteres, para ello se hace uso de la biblioteca ns3/hash [71].
- **Eventos discretos. -** Como se mencionó en el apartado 2.1, NS-3 ejecuta instrucciones secuenciales acorde a una línea de tiempo determinada, para ello se requiere que el simulador sea capaz de generar una cola de eventos en donde una vez ejecutado un evento en un tiempo determinado, salte al siguiente evento en cola, siempre y cuando sea siguiente en la línea de tiempo, continuando de manera sucesiva con los demás hasta finalizar con los eventos en cola [72].
- **Simulador. -** Módulo responsable de iniciar el periodo de simulación de un script implementado, trabaja en conjunto con el módulo de eventos discretos realizando la función de planificador, para ello es necesario el uso de la variable *Simulator::Run* para ejecutar instancias secuenciales en lazo, hasta que se reciba la variable *Simulator::Stop* deteniendo así la simulación, también se pueden programar funciones de planificación (*Simulator::Scheduler*) y ejecución en un tiempo determinado (*Simulator::Time*) [72].
- **Callback. -** Módulo en donde su función es llamar instancias mientras se realiza una simulación, es decir, si se tienen instancias definidas previamente con todos sus parámetros, por ejemplo, A y B, se las puede llamar a ejecución para obtener resultados de estas mientras se realiza una simulación, sin necesidad de volver a declarar por completo las instancias, si no que solamente se podrá utilizar un identificador para insertar datos entre ellas para obtener dichos resultados [73].
- **Objeto. -** Acorde al manual de uso, NS-3 es un sistema de objetos C++, en donde, un objeto puede almacenar datos o ser usado en el proceso de obtención de un resultado, siendo este un elemento que cuenta con las capacidades del diseño clásico orientado a objetos, las cuales son: abstracción, encapsulación, herencia y polimorfismo [74].

- **Logging. -** Módulo que permite la recolección de datos de una simulación, con el objetivo de monitorear o depurar la misma [75].
- **Tracing. -** En caso de nuevas implementaciones es necesario el uso de un sistema de verificación que nos permita validar que los paquetes llegan desde el origen hacia un destino determinado, además de la composición de los mismos, para ello el simulador cuenta con capacidades de tracing, por ejemplo, con ayuda de este módulo se pueden generar archivos *pcap* analizables con wireshark [76].
- **Movilidad.** - Las capacidades de movilidad de cada dispositivo son gestionadas en este módulo (mobility), para ello, se basa en la posición cartesiana y velocidad del objeto, pudiendo generarse un movimiento en dos o tres dimensiones, entre los modelos de movilidad más relevantes se tienen los siguientes [77]:
    - Constant. - Modelo que puede ser usado para dar a un objeto una posición estática (ConstantPosition), así como de velocidad (ConstantVelocity) y aceleraciones constantes (ConstantAcceleration).
    - GaussMarkov. - Este modelo de movilidad es el implementado en el presente trabajo, ya que es usado en su mayoría en simulaciones de vuelo [78] [79], debido a que define variables como la velocidad, dirección y ángulo de paso, en donde este último, es aquel que se forma entre el eje longitudinal de la aeronave y el suelo u horizonte, además, se encuentra representado en la Figura 2.12. En NS-3 se modela una variante en 3D del modelo Gauss Markov original, el cual tiene la característica de poseer memoria y aleatoriedad, dichas características pueden configurarse con un parámetro denominado *alpha* y, cada dispositivo empieza con una velocidad, dirección y ángulo de paso definidos, con el paso del tiempo, estos valores son actualizados en base a valores promedios anteriores, así como de una variable gaussiana.

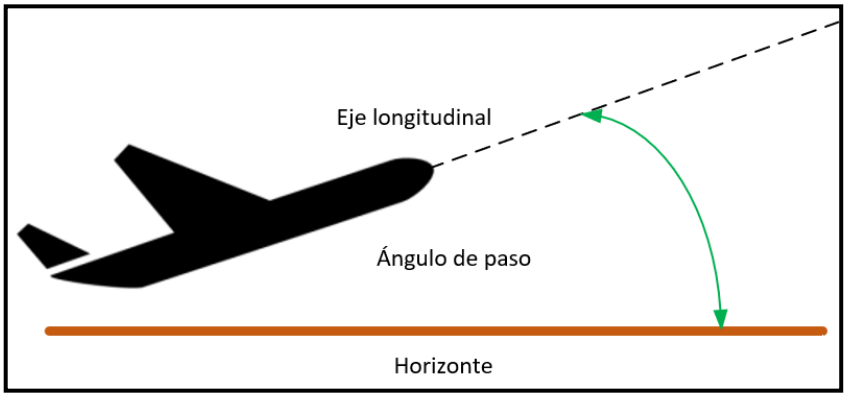

Figura 2.12. Representación del ángulo de paso.

- Hierarchical. - Este modelo se basa en la combinación de representaciones denominadas padre e hijo, en donde, la posición está dada por la suma de los vectores padres con las posiciones de los hijos. Este modelo es especialmente usado para simulaciones de vehículos en carretera.
- RandomDirection2D.- Modelo que propone el movimiento de un nodo considerando direcciones aleatorias.
- RandomWalk2D.- Modelamiento de movilidad en 2 dimensiones basado en dirección y velocidad aleatorias.
- RandomWaypoint. - En este modelo la movilidad del nodo está dada por un punto de referencia y velocidad aleatorios.

- **Network.** - En este módulo se definen todos los componentes requeridos para la conectividad entre nodos, tales como: direcciones IP, protocolos, puertos, cabeceras, entre otros. Siendo imprescindible su uso en cualquier ambiente de simulación de redes. En la Figura 2.13 se muestran los submódulos más representativos de este módulo.

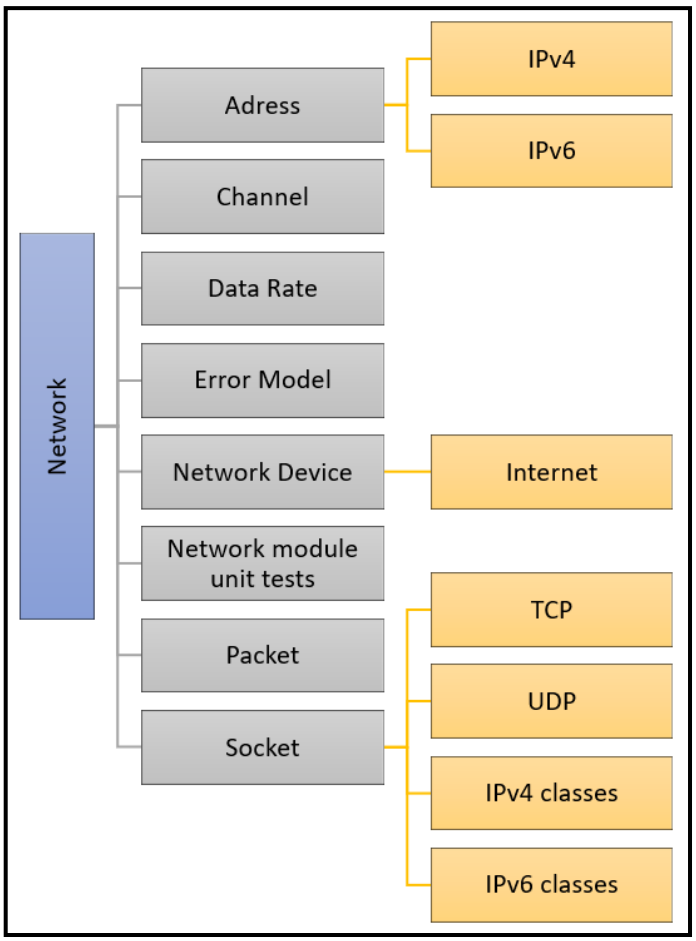

Figura 2.13. Submódulos del módulo Network.

- **Internet. -** Este módulo permite la implementación de los componentes relacionados con la arquitectura TCP/IPv4 e IPv6, entre las que se encuentran protocolos ARP, UDP, TCP, Neighbor Discovery, entre otros. Otorgando estas funciones a los nodos.
- **Mesh. -** Módulo que tiene como función principal proveer enrutamiento en la capa MAC, requerido para la implementación de comunicación tipo mesh entre nodos acorde al estándar 802.11s, trabaja en conjunto con el módulo Wi-Fi y es indispensable para el despliegue del protocolo HWMP, el cual fue descrito en la sección 1.4.2.6.
- **Wifi.** - Provee funcionalidades de conectividad inalámbrica para los nodos, en donde la conectividad está basada en la infraestructura del estándar 802.11 y en redes Ad-hoc, la arquitectura de este módulo se puede ver en la Figura 28, en donde se definen las siguientes capas:
    - **WifiNetDevice. -** Modela una interfaz de red inalámbrica basada en el estándar 802.11, en donde se pueden definir aspectos como: el estándar a implementar sea 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac u 802.11ax, características específicas como DL OFDMA o UL OFDMA, QoS, 802.11s, 802,11p, entre otros.
    - **MAC High.** - Define el tipo de comunicación inalámbrica, considerando 3 elementos de una red Wi-Fi, definidos como: Access Point (*ns3::ApWifiMac*), Estación (ns3::StaWifiMac) y Ad-hoc (ns3::AdhocWifiMac). El uso de estos debe ser acorde al escenario requerido a implementar.
    - **MAC Low.** - Esta capa ofrece funciones de control de acceso al medio como DCF y EDCAF [80], manejo de tramas (*FrameExchangeManager*), sea agregación, retransmisión, protección y QoS.
    - **WifiPHY.** - Se encarga del modelamiento de la recepción de paquetes y monitorear el consumo de energía, se tienen dos modelos de capa física: El primer modelo que hace referencia la clase *YansWifiPhy*, implementa un modelo de capa física básico, por otro lado, la clase SpectrumWifiPhy que permite la implementación de múltiples tecnologías coexistiendo en el mismo canal.
    - **WifiChannel. -** Se encarga de la creación del canal utilizando la clase *Ns3::WifiChannel*, siendo la clase *ns3::YansWifiChannel* el único canal disponible, con parámetros de pérdidas y retardo disponibles en los módulos de propagación disponibles en NS-3.

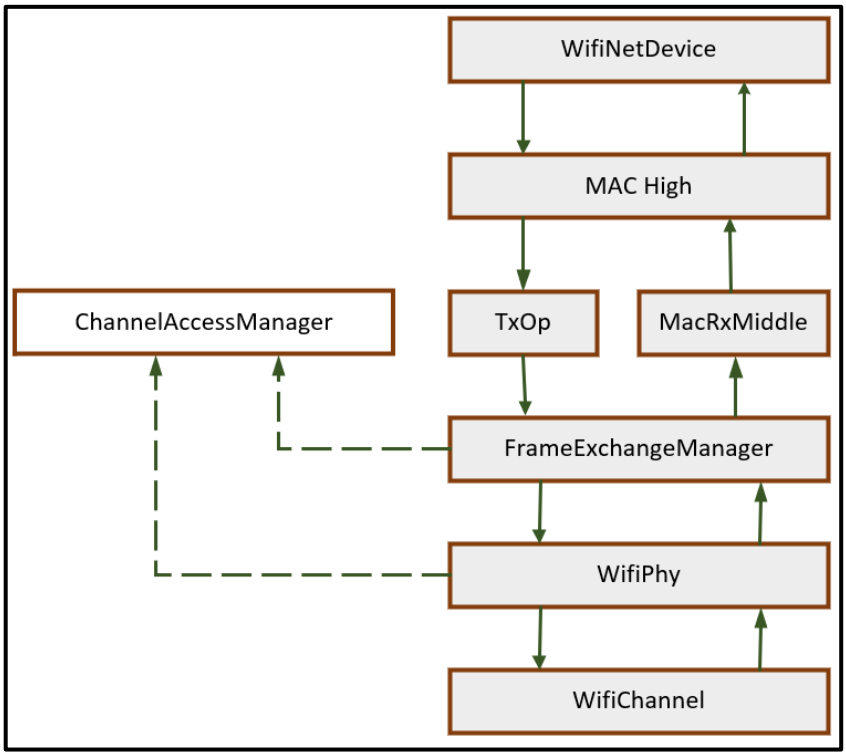

Figura 2.14. Arquitectura del módulo Wi-Fi, basado [69].

- **OLSR.** - Este módulo implementa las especificaciones del protocolo OLSR, y está basado en el RFC 3626 [81], permite la implementación de dicho protocolo únicamente en IPv4, trabaja en conjunto con la clase *ns3::OlsrHelper*.
- **AODV. -** Módulo responsable de la implementación de las especificaciones del protocolo AODV, la cual está basada en el RFC 3561 [82], puede ser implementado únicamente en IPv4, además, trabaja en conjunto con la clase *ns3::AodvHelper*
- **DSDV. -** Este módulo permite la implementación del protocolo proactivo DSDV en IPv4 y trabaja en conjunto con la clase *ns3::DsdvHelper*.
- **DSR. -** Permite la implementación del protocolo DSR en escenarios de redes inalámbricas multisaltos y redes Ad-hoc, las especificaciones de este módulo están basadas en el RFC 4728 [83].
- **Aplicaciones. -** Este módulo permite agregar funciones a los nodos definidos en una red, entre estas funcionalidades se puede destacar, como ejemplo el generar tráfico UDP por medio de un modelo cliente/servidor, con el uso de la clase ns3::UdpClientHelper para definir el nodo cliente y, para el nodo servidor se utiliza la clase ns3::UdpServerHelper [84].
- **Flow Monitor. -** Este módulo permite medir el rendimiento de una red, para ello se utilizan punteros instalados en los nodos de la red y hacer un seguimiento de los paquetes transmitidos en los nodos, clasificando paquetes enviados,

retransmitidos, recibidos y perdidos. Este módulo permite el análisis en redes IPv4 e IPv6.

2.6 REQUERIMIENTOS DE DISEÑO EN REDES FANET

Para la implementación de una red FANET tanto en escenarios reales como para propósitos de simulación, es necesario tener en cuenta ciertos parámetros, en conjunto con las características de las redes FANET que se discutieron en la sección 1.3.8, dichos parámetros a considerar son los siguientes [37] [5]:

- **Latencia:** El retraso en la transferencia de datos está presente en todas las redes de comunicación, este parámetro es crítico para el diseño de redes FANET debido a la característica de alta movilidad de dichas redes, ya que acorde a la aplicación de estas redes se requerirá latencia baja como en el caso de operaciones de búsqueda y rescate con video, en donde la información debe ser transmitida prácticamente en tiempo real para la toma de decisiones. Por otro lado, en aplicaciones como topografía o fumigación, la latencia puede ser tolerable.
- **Ancho de banda:** El tipo de tráfico que se transmita por la red FANET puede requerir cierto ancho de banda, es decir no es lo mismo transmitir información captada por sensores a bordo, en donde el ancho de banda puede ser bajo por el tipo de información transmitida que es en el orden de los Kb. Sin embargo, en escenarios en donde se debe transmitir datos como video o incluso destinar cierto ancho de banda para que la red FANET sea soporte de otras redes como redes celulares o Wi-Fi, el ancho de banda a requerir puede llegar a ser alto, por lo que es recomendable definir adecuadamente el escenario en donde se requiere implementar dichas redes FANET.
- **Restricciones del Hardware:** Dependiendo del tipo de aplicación se pueden requerir ciertas características de los UAVs, sean éstas: alta capacidad de carga, múltiples sensores a bordo, compatibilidad con múltiples plataformas de comunicación, larga tiempo de autonomía, etc. Sin embargo, no todas las características pueden estar disponibles en muchos UAVs, ya que, por ejemplo, para poder tener alta capacidad de carga, el tamaño de los UAVs puede aumentar significativamente, lo que podría conllevar a que no tenga una alta maniobrabilidad y si el escenario de aplicación considera esto como una característica crítica, dicho UAV no será el adecuado para la implementación. Por otro lado, puede que el hardware de comunicación no esté destinado si no únicamente a la transferencia

de datos relacionados a la carga. Considerando esto es necesario escoger adecuadamente el tipo de UAV a desplegar dependiendo de la aplicación.

- **Capacidad computacional:** Considerando que una red FANET es un subgrupo de las redes Ad-hoc, se debe considerar que cada UAV puede ser un nodo que debe manejar tablas de enrutamiento para mantener la comunicación en la red, debido a esto, un parámetro a considerar es la capacidad de manejar información entrante y saliente, en especial en el caso de que se tengan sensores a bordo, esto debido a que en caso de sobrepasar la capacidad de procesamiento de esta información pueden ocurrir retrasos que pueden generar problemas en la operaciones de estas redes.
- **Hardware de control adicional:** Algunos UAVs del mercado no están listos para la operación en conjunto, si no únicamente de manera individual, por lo que es necesario implementar hardware adicional para poder desplegar redes FANET, tal es el caso del uso de tarjetas de desarrollo adicionales como Raspberry en conjunto con las APIs del fabricante para permitir la comunicación Ad-hoc [85].

El considerar los puntos descritos anteriormente puede significar ahorro de tiempo en el despliegue de una red FANET.

2.7 SIMULACIÓN DE REDES FANET EN NS-3

En esta sección se detallan las partes fundamentales de los códigos implementados en la simulación de redes FANET, así como la implementación de los protocolos de enrutamiento, para ello es necesario tomar en cuenta los parámetros discutidos a lo largo del presente trabajo.

Cabe destacar que se han desarrollado scripts independientes para los protocolos de enrutamiento, siendo los siguientes:

- **Archivo uav.cc:** simula una red Fanet, en donde el usuario puede definir el número de UAVs a desplegar, la velocidad de estos dispositivos y el protocolo a implementar, siendo en este caso OLSR, AODV, DSDV, DSR y AOMDV.
- **Archivo hwmp.cc:** simula una red Fanet, en donde el usuario puede definir el número de UAVs a desplegar, la velocidad de estos dispositivos y el protocolo a implementar es el protocolo HWMP.

2.7.1 DEFINICIÓN DE MÓDULOS

Los módulos y su definición se abordaron en la sección 2.5, se ubican en la cabecera del script a implementar, permiten el uso de las clases que son parte de NS-3 con el fin de usar todas las dependencias y sus funciones como parte del script, los módulos disponibles en el simulador se detallan en la Tabla 2.1 [69].

Tabla 2.1. Módulos disponibles en NS-3.

Antenna	AODV Routing	3GPP HTTP Applications
Bridge NetDevice	BRITE Integration	Buildings Module
Click Modular Router Integration	CSMA Network Device	CSMA Layout
Core	DSDV Routing	DSR Routing
Energy Models	File Descriptor Network Device	Flow Monitor
Internet Models (IP, TCP, Routing, UDP)	Internet Applications	LR-WPAN models
LTE Models	MPI Distributed Simulation	Mesh Device
Mobility	Network	Network Animation
Nix-Vector Routing	OLSR Routing	OpenFlow Switch Device
Point-To-Point Network Device	Point-to-Point Layout Helpers	Propagation Models
Spectrum Models	Statistics	Tap Bridge Network Device
Topology Input Readers	Traffic-control	UAN Models
Virtual Device	Visualizer	WAVE Module
WiMAX Models	Wifi Models	6LoWPAN

Los módulos son declarados de la siguiente manera:

```
#include <ns3/nombre-module.h>
```

Segmento de código 1. Formato a usar para declarar un módulo en NS-3.

Es importante mencionar que para la simulación se han utilizado módulos ya desarrollados e incluidos en NS-3, puesto que el objetivo de este trabajo no es implementar nuevos módulos para los escenarios de simulación.

1. Script de implementación de una red Fanet con los protocolos AODV, OLSR, DSDV, DSR y AOMDV denominado UAV.cc

```
#include <fstream>
#include <iostream>
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/wifi-module.h"
#include "ns3/aodv-module.h"
#include "ns3/olsr-module.h"
#include "ns3/dsdv-module.h"
#include "ns3/dsr-module.h"
#include "ns3/applications-module.h"
#include "ns3/netanim-module.h"
#include "ns3/flow-monitor-module.h"
#include "myapp.h"
```

Segmento de código 2. Módulos usados en la implementación del script UAV.cc

2. Script de implementación de una red Fanet con el protocolo HWMP, denominado hwmp.cc

```
#include <fstream>
#include <iostream>
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/wifi-module.h"
#include "ns3/olsr-module.h"
#include "ns3/applications-module.h"
#include "ns3/netanim-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/mesh-module.h"
#include "ns3/mesh-helper.h"
```

Segmento de código 3. Módulos usados en la implementación del script hwmp.cc.

2.7.2 DEFINICIÓN DE VARIABLES

A continuación, se presentan las variables definidas en los códigos implementados en cada escenario, con el fin de permitir la libre modificación del mismo.

1. Script de implementación de una red Fanet con los protocolos AODV, OLSR, DSDV, DSR y AOMDV denominado UAV.cc

```
double txp = 30;
int nUav;
int protocol;
int vel;
int nSinks = 2;
int iUav = (nUav/2);
int sender = iUav + nSinks;
double TotalTime = 10.0;
uint32_t packetSize = 512;
std::string m_protocolName;
m_protocolName = "protocol";
```

Segmento de código 4. Variables declaradas en script UAV.cc.

En donde las variables se utilizan de la siguiente manera:

- ***txp*:** Define la potencia de transmisión en dBm.
- ***nUav*:** Define el número de UAVs a desplegar en la red FANET.
- ***protocol*:** Variable usada para la selección del protocolo a implementar (OLSR, AODV, DSDV, DSR).
- ***nSinks*:** Variable utilizada para definir el número de nodos que van a funcionar como receptores en la comunicación.
- ***iUav*:** Variable utilizada para definir el número de nodos intermedios en la comunicación.
- ***sender*:** Define el número de nodos emisores en la comunicación.
- ***TotalTime*:** Variable que define el tiempo total de la simulación en segundos.
- ***packetSize*:** Variable usada para definir el tamaño del paquete en bytes.
- ***vel*:** Variable usada para definir la velocidad de movimiento máxima de los nodos en el modelo de movilidad.
- ***m_protocolName:*** Variable de tipo cadena de caracteres usada para definir el nombre del protocolo implementado.

2. Script de implementación de una red Fanet con el protocolo HWMP, denominado hwmp.cc

```
double txp = 20;
int nUav;
int vel;
int nSinks = 2;
int iNodes = (nUav/2);
int sender = iNodes + nSinks;
double TotalTime = 5;
uint32_t packetSize = 512;
double m_randomStart;
uint32_t m_nIfaces;
```

Segmento de código 5. Variables declaradas en script hwmp.cc.

En donde algunas variables se han definido previamente en el código UAV.cc, además de las siguientes:

- **m_randomStart:** Variable que se utiliza en el inicio aleatorio de la comunicación inalámbrica tipo mesh.
- **m_nIfaces:** Variable que define el número de interfaces en la comunicación mesh.

2.7.3 DESPLIEGUE DE LA RED

En esta sección se hace un recorrido sobre el despliegue de la red en modo Ad-hoc, así como la definición de la arquitectura para esta red, cabe recalcar que para el presente trabajo se utiliza comunicación Wi-Fi para el despliegue de la red y para esto, se siguen los siguientes pasos:

1. **Definir el número de nodos:** Se solicita al usuario ingresar el número de nodos a requerir, se almacena el valor en la variable *nUav* y luego se muestra el valor seleccionado.

```
std::cout<<"Ingrese el numero de nodos UAV: (N<255) \n";
std::cin>>nUav;
std::cout<<"\n La red FANET contiene "<< nUav <<" nodos UAV \n";
```

Segmento de código 6. Definir el número de nodos.

2. **Creación de nodos:** Se define un grupo de nodos con la ayuda de la clase *NodeContainer*.

```
NodeContainer uav;
uav.Create (nUav);
```

Segmento de código 7. Definición de un grupo de nodos.

3. **Definir el tipo de capa física a implementar:** La capa física que se utiliza en el presente trabajo es *YansWifiPhy*.

```
YansWifiPhyHelper wifiPhy;
```

Segmento de código 8. Definición de la capa física.

4. **Configuración y creación del canal inalámbrico:** Por medio del uso de *YansWifiChannelHelper* se define un canal con modelo de retardo por propagación *ConstantSpeedPropagationDelayModel* y modelo de pérdida de propagación *ConstantSpeedPropagationDelayModel*.

```
YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::RangePropagationLossModel");
wifiPhy.SetChannel (wifiChannel.Create ());
```

Segmento de código 9. Configuración y creación del canal inalámbrico.

5. **Configuración de modo Ad-hoc:** Al iniciar la configuración de la capa MAC se selecciona el modo Ad-hoc en conjunto con el módulo helper.

```
WifiMacHelper wifiMac;
wifiMac.SetType ("ns3::AdhocWifiMac");
```

Segmento de código 10. Configuración de la red Ad-hoc.

6. **Creación y configuración de dispositivos inalámbricos:** Se definen los parámetros de operación de los dispositivos inalámbricos, considerando potencias de transmisión y recepción, así como un container por la gran cantidad de dispositivos que se pueden llegar a implementar.

```
WifiHelper wifi;
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                              "DataMode",StringValue (phyMode),
                              "ControlMode",StringValue (phyMode));
wifiPhy.Set ("TxPowerStart",DoubleValue (txp));
wifiPhy.Set ("TxPowerEnd", DoubleValue (txp));
wifiPhy.Set ("RxGain", DoubleValue (-95));
NetDeviceContainer devices = wifi.Install (wifiPhy, wifiMac, uav);
```

Segmento de código 11. Creación y configuración de dispositivos inalámbricos.

7. **Definición de una red mesh:** Este paso es necesario únicamente para la implementación del protocolo HWMP, ya que, es necesario el despliegue de una red tipo mesh, esto se logra utilizando el helper respectivo, además, del mismo modo que en los pasos anteriores se crean y configuran los dispositivos inalámbricos.

```
mesh.SetMacType ("RandomStart", TimeValue (Seconds (m_randomStart)));
mesh.SetNumberOfInterfaces (m_nIfaces);
meshDevices = mesh.Install (wifiPhy, uav);
```

Segmento de código 12. Definición de una red mesh.

2.7.4 MODELO DE MOVILIDAD DE LA RED

En esta sección se explica la implementación de movilidad en los nodos, para ello se hace uso del modelo de movilidad de Gauss Márkov que se explicó en el apartado 2.5.

Se utiliza el mismo modelo de movilidad en todos los scripts debido a que el presente trabajo es una simulación de UAVs.

1. **Definir la velocidad máxima del nodo:** Se solicita al usuario ingresar la velocidad de movimiento máxima de los nodos, esto modifica el parámetro *MeanVelocity* con la ayuda de la variable *vel* y la cadena de caracteres *ssSpeed*, para implementarlo en el modelo de movilidad y finalmente mostrar el valor seleccionado.

```
  std::cout<<"\n Ingrese la velocidad promedio maxima [m/s] (Esto modifica el parametro
MeanVelocity): \n";
  std::cin>>vel;
  std::stringstream ssSpeed;
  ssSpeed << "ns3::UniformRandomVariable[Min=0.0|Max=" << vel << "]";
```

Segmento de código 13. Definir la velocidad de movimiento máxima de los nodos.

2. **Configurar el modelo de movilidad:** En este caso con ayuda del helper de movilidad *MobilityHelper* y la función *SetMobilityHelper*.

```
MobilityHelper mobility;
mobility.SetMobilityModel ("ns3::GaussMarkovMobilityModel",
  "Bounds", BoxValue (Box (0, 500, 0, 500, 0, 25)),
  "TimeStep", TimeValue (Seconds (0.5)),
  "Alpha", DoubleValue (0.85),
  "MeanVelocity", StringValue (ssSpeed.str ()),
  "MeanDirection", StringValue ("ns3::UniformRandomVariable[Min=0|Max=6.283185307]"),
  "MeanPitch", StringValue ("ns3::UniformRandomVariable[Min=0.05|Max=0.05]"),
  "NormalVelocity", StringValue
("ns3::NormalRandomVariable[Mean=0.0|Variance=0.0|Bound=0.0]"),
  "NormalDirection", StringValue
("ns3::NormalRandomVariable[Mean=0.0|Variance=0.2|Bound=0.4]"),
  "NormalPitch", StringValue
("ns3::NormalRandomVariable[Mean=0.0|Variance=0.02|Bound=0.04]"));
```

Segmento de código 14. Definir y configurar el modelo de movilidad.

Es necesario mencionar, que el modelo de movilidad es configurable y tiene los siguientes parámetros:

- ***Bounds*:** Área de simulación tomando como referencia un cubo en donde se define ancho, largo y alto.
- ***TimeStep*:** Tiempo en segundos en el cual se actualizará la dirección de movimiento del nodo.
- ***Alpha*:** Define la cantidad de memoria y aleatoriedad del movimiento de un nodo.
- ***MeanVelocity*:** Variable que define la velocidad promedio de movimiento del nodo.

- ***MeanDirection*:** Define la dirección promedio de los nodos en radianes.
- ***MeanPitch*:** Define el valor promedio del ángulo de paso del nodo, el cual fue definido previamente en la sección 2.5 del presente trabajo.
- ***NormalVelocity*:** Variable de tipo gaussiana usada para calcular la variación de velocidad del nodo.
- ***NormalDirection*:** Variable de tipo gaussiana usada para calcular la variación de dirección del nodo.
- ***NormalPitch*:** Variable de tipo gaussiana usada para calcular la variación del ángulo de paso.

Para el presente trabajo se han mantenido los valores por defecto del modelo de movilidad a excepción de valores como el área de simulación y la velocidad promedio de los nodos.

3. **Definir la posición inicial de cada nodo:** Se definen posiciones aleatorias para cada nodo con ayuda de la clase *ns3::UniformRandomVariable* y se determinan valores mínimos y máximos.

```
mobility.SetPositionAllocator ("ns3::RandomBoxPositionAllocator",
  "X", StringValue ("ns3::UniformRandomVariable[Min=0|Max=500]"),
  "Y", StringValue ("ns3::UniformRandomVariable[Min=0|Max=500]"),
  "Z", StringValue ("ns3::UniformRandomVariable[Min=0|Max=25]"));
```

Segmento de código 15. Definición de la posición inicial de cada nodo.

4. Asignar el modelo de movilidad a cada nodo.

```
mobility.Install (uav);
```

Segmento de código 16. Instalación del modelo de movilidad en cada nodo.

2.7.5 INTEGRACIÓN DE LOS PROTOCOLOS DE ENRUTAMIENTO

En esta sección se explica la integración de los diferentes protocolos de enrutamiento en la red creada previamente.

1. Se solicita al usuario seleccionar el tipo de protocolo a implementar.

```
std::cout<<"\nIngrese el numero del protocolo (1=OLSR;2=AODV;3=DSDV;4=AOMDV;5=DSR) ";
 std::cin>>protocol;
```

Segmento de código 17. Selección del tipo de protocolo a implementar.

2. Declarar los helpers requeridos de cada uno de los protocolos.

AODV:

```
AodvHelper aodv;
```

Segmento de código 18. Helper del protocolo AODV.

OLSR:

```
OlsrHelper olsr;
```

Segmento de código 19. Helper del protocolo OLSR.

DSDV:

```
DsdvHelper dsdv;
```

Segmento de código 20. Helper del protocolo DSDV.

DSR:

```
DsrHelper dsr;
DsrMainHelper dsrMain;
```

Segmento de código 21. Helper del protocolo DSR.

En el caso del protocolo DSR, es necesario declarar el helper *DsrMainHelper* para agregar el enrutamiento a cada nodo [86].

Debido a que no existe un módulo definido para el protocolo AOMDV en NS-3 se utiliza como punto de partida el módulo existente del protocolo AODV, además de módulos adicionales que se explicarán más adelante en el presente trabajo.

3. Habilitar el enrutamiento con los protocolos AODV, OLSR, DSDV y AOMDV.

```
internet.SetRoutingHelper (list)
internet.Install (uav);
```

Segmento de código 22. Habilitar los protocolos AODV, OLSR, DSDV.

4. Habilitar el enrutamiento con el protocolo DSR.

```
internet.Install (uav);
dsrMain.Install (dsr, uav);
```

Segmento de código 23. Habilitar el protocolo DSR.

5. Selección del tipo de enrutamiento a implementar (Solo para el archivo uav.cc).

```
 if(protocol==1)
 {
    list.Add (olsr, 100);
    m_protocolName = "OLSR";
 }
 else if(protocol==2)
{
    list.Add (aodv, 100);
    m_protocolName = "AODV";
}
 else if(protocol==3)
{
    list.Add (dsdv, 100);
    m_protocolName = "DSDV";
}
 else if (protocol==4)
 {
    list.Add (aodv, 100);
    m_protocolName = "AOMDV";
 }
 else if (protocol==5)
 {
    m_protocolName = "DSR";
 }
else
{
 NS_FATAL_ERROR ("Seleccione el protocolo adecuado:" << protocol);
}
```

Segmento de código 24. Selección de tipo de protocolo en archivo uav.cc.

6. Habilitar el enrutamiento con el módulo mesh requerido en el protocolo HWMP.

```
mesh = MeshHelper::Default ();
 if (!Mac48Address (m_root.c_str ()).IsBroadcast ())
   {
    mesh.SetStackInstaller (m_stack, "Root", Mac48AddressValue (Mac48Address (m_root.c_str
()())));
   }
 else
   {
    mesh.SetStackInstaller (m_stack);
   }
 if (m_chan)
   {
    mesh.SetSpreadInterfaceChannels (MeshHelper::SPREAD_CHANNELS);
   }
 else
   {
    mesh.SetSpreadInterfaceChannels (MeshHelper::ZERO_CHANNEL);
   }
```

Segmento de código 25. Habilitar el módulo para comunicación mesh.

2.7.6 DIRECCIONAMIENTO IP

Para esta sección es necesario iniciar el stack de protocolos IPv4 y dar un direccionamiento IP base para los nodos.

```
Ipv4AddressHelper ipv4;
  NS_LOG_INFO ("Assign IP Addresses.");
  ipv4.SetBase ("10.0.0.0", "255.255.252.0");
  Ipv4InterfaceContainer ifcont = ipv4.Assign (devices);
```

Segmento de código 26. Habilitar el direccionamiento IP.

2.7.7 GENERACIÓN DE TRÁFICO EN LA RED

En el presente trabajo, para los protocolos AODV, OLSR, DSR y DSDV, se hace uso de la generación de tráfico UDP por un puerto específico definido previamente, además, es necesario generar tráfico a cada uno de los nodos por medio del uso de la clase *ns3::UdpSocketFactory* en conjunto con la clase *ns3::OnOffApplication*, la cual es usada en la generación de tráfico hacia un único nodo, por esto, es necesario el uso de un lazo *for* para generar tráfico a todos los nodos, como se muestra a continuación:

```
uint16_t port = 9;
for(int i = 0; i<nSinks; i++)
{

        OnOffHelper onoff("ns3::UdpSocketFactory",InetSocketAddress(ifcont.GetAddress(i), port));
        onoff.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Constant=1]"));
        onoff.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=1]"));
        onoff.SetAttribute("PacketSize", UintegerValue(packetSize));
        onoff.SetAttribute ("DataRate", DataRateValue (rate));
        ApplicationContainer apps = onoff.Install(uav.Get(sender+i));
        apps.Start(Seconds(1.0));
        apps.Stop(Seconds(TotalTime));
}
```

Segmento de código 27. Generación de tráfico en la red para los protocolos AODV, OLSR, DSR y DSDV.

Para el caso de protocolo AOMDV, como se explicó previamente en la sección 2.7.5, no existe el módulo de implementación para este protocolo, por lo que se parte del módulo existente para el protocolo AODV, en conjunto con el módulo “myapp.h” disponible en [87], esto con el fin de generar y mantener múltiples rutas de manera similar a como lo hace el protocolo AOMDV, esto se logra al modificar el parámetro *SetAllowBroadcast* como se puede apreciar en el Segmento de código 27:

```
if (protocol==4)
{
  for(int i=0;i<nSinks;i++)
    {
   Address sinkAddress1 (InetSocketAddress (ifcont.GetAddress (i), port));
   PacketSinkHelper packetSinkHelper1 ("ns3::UdpSocketFactory", InetSocketAddress
(Ipv4Address::GetAny (), port));
   ApplicationContainer sinkApps1 = packetSinkHelper1.Install (uav.Get (i));
   sinkApps1.Start (Seconds (0.));
   sinkApps1.Stop (Seconds (TotalTime));

    Ptr<Socket> ns3UdpSocket1 = Socket::CreateSocket (uav.Get
(sender+i),UdpSocketFactory::GetTypeId ());
   Ptr<MyApp> app1 = CreateObject<MyApp> ();
   app1->Setup (ns3UdpSocket1, sinkAddress1, packetSize, numPackets, DataRate (rate));
   uav.Get (sender+i)->AddApplication (app1);
   ns3UdpSocket1->SetAllowBroadcast (true);
   app1->SetStartTime (Seconds (0.));
   app1->SetStopTime (Seconds (TotalTime));
  }
}
```

Segmento de código 28. Generación de tráfico en la red para el protocolo AODMV.

Para el protocolo HWMP, la generación de tráfico se logra por medio del uso de la clase *ns3::UdpEchoServerHelper*, esto debido a que, al ser una red de tipo mesh la clase *ns3::OnOffApplication* no permite que todos los nodos se comuniquen adecuadamente, esto se puede apreciar en el código a continuación:

```
UdpEchoServerHelper echoServer (port);
ApplicationContainer serverApps = echoServer.Install (uav.Get (i));
serverApps.Start (Seconds (0.0));
serverApps.Stop (Seconds (TotalTime));
UdpEchoClientHelper echoClient (ifcont.GetAddress (i), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue ((uint32_t)(TotalTime*(1/1))));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1)));
echoClient.SetAttribute ("PacketSize", UintegerValue (packetSize));
ApplicationContainer clientApps = echoClient.Install (uav.Get (sender+i));
clientApps.Start (Seconds (0.0));
clientApps.Stop (Seconds (TotalTime));
```

Segmento de código 29. Generación de tráfico en la red para el protocolo HWMP.

2.7.8 GENERACIÓN DE ARCHIVO NETANIM

Para el uso del módulo NetAnim definido en la sección 2.3 es necesario generar archivos XML en el formato apropiado, esto se logra con el uso del siguiente código:

```
pAnim= new AnimationInterface ("uav.xml");
pAnim->EnablePacketMetadata(true);
uint32_t uavImg = pAnim->AddResource("/home/wellington/Downloads/uav.png");
```

Segmento de código 30. Generación de archivos XML a usarse en NetAnim.

En donde se habilita la generación de información relacionada a los mensajes en la comunicación, además del nombre del protocolo implementado, esto se puede visualizar de mejor manera en las siguientes figuras:

OLSR:

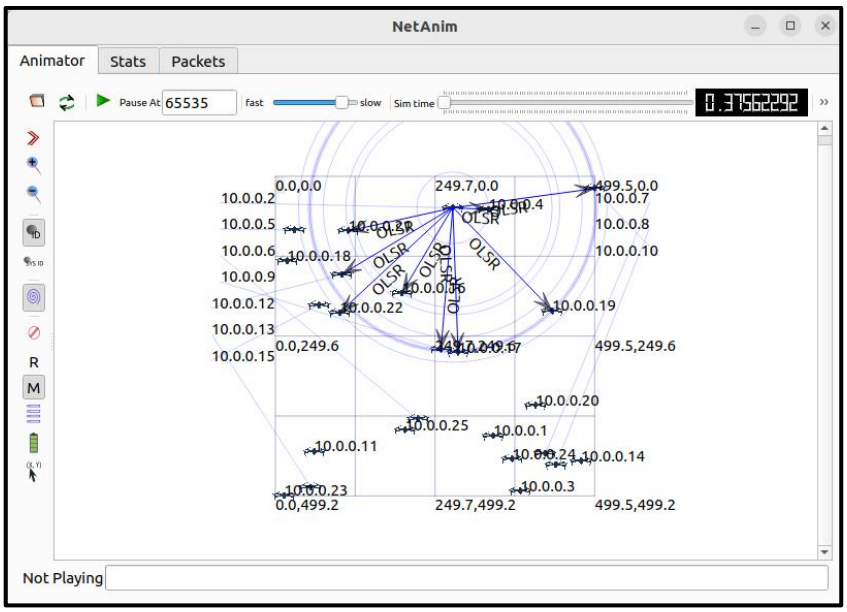

Figura 2.15. Simulación de una red Fanet con el protocolo OLSR.

AODV:

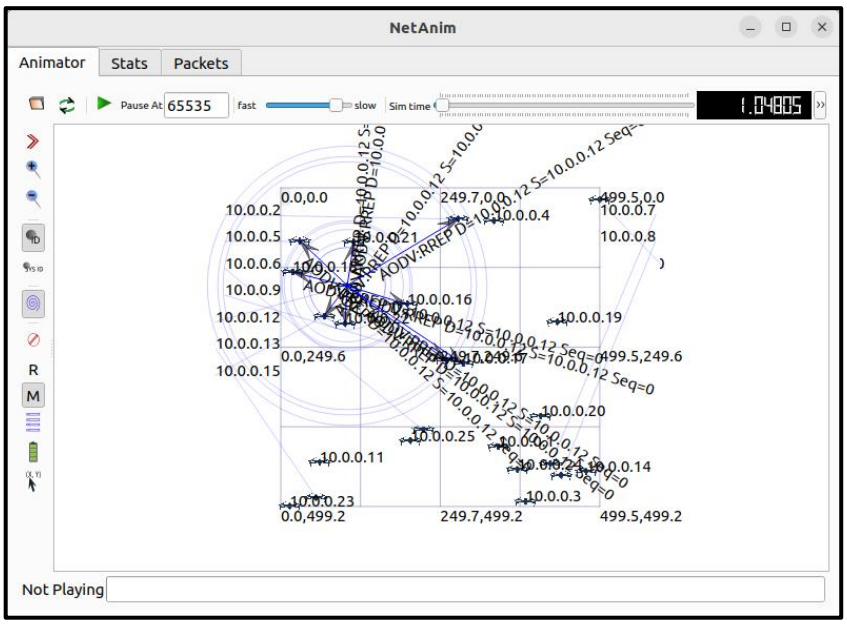

Figura 2.16. Simulación de una red Fanet con el protocolo AODV.

DSDV:

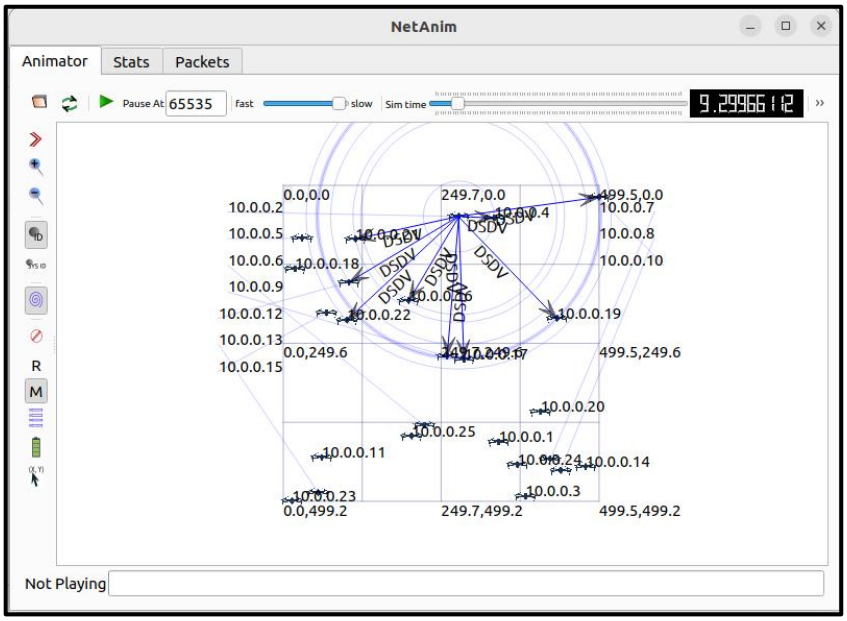

Figura 2.17. Simulación de una red Fanet con el protocolo DSDV.

DSR:

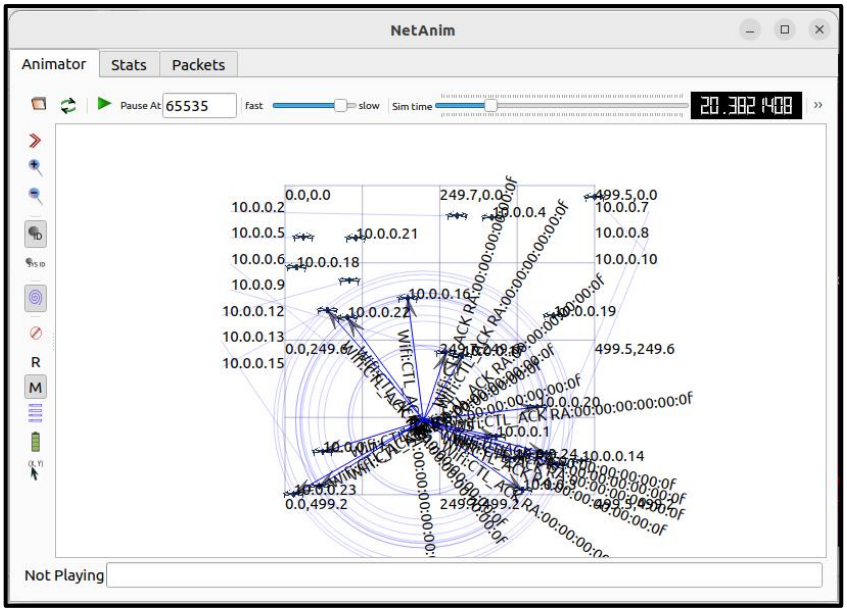

Figura 2.18. Simulación de una red Fanet con el protocolo DSR.

HWMP:

Este protocolo es conocido que puede tener problemas en la implementación de información adicional en el módulo NetAnim [88].

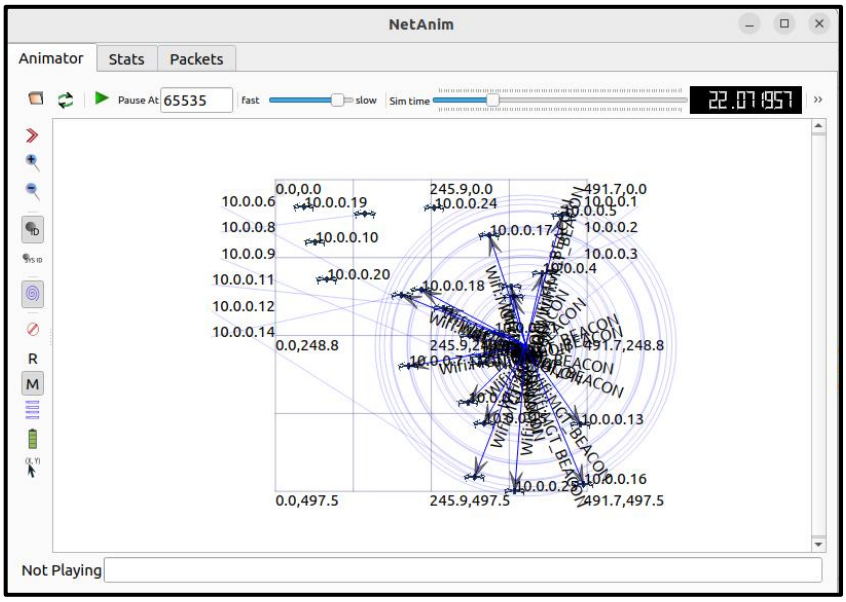

Figura 2.19. Simulación de una red Fanet con el protocolo HWMP.

AOMDV:

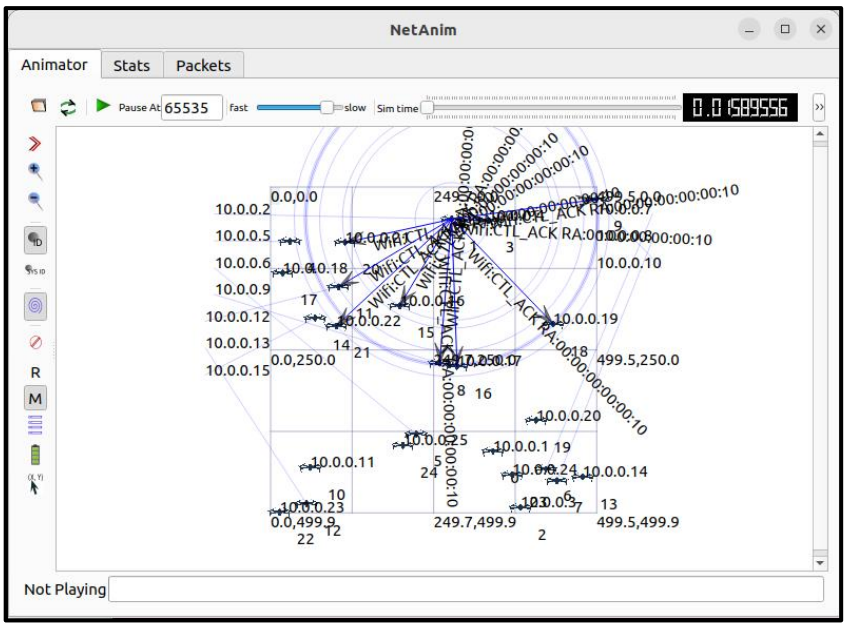

Figura 2.20. Simulación de una red Fanet con el protocolo AOMDV.

2.7.9 GENERACIÓN DE ARCHIVOS PCAP

Para la generación de estos archivos se utiliza la variable *m_protocolName*, misma que se utiliza en el apartado 2.7.5, para generar archivos diferentes con el nombre del tipo de protocolo seleccionado previamente, con el fin de llevar un orden en los archivos generados, esto se logra con la siguiente línea de código:

```
wifiPhy.EnablePcap (m_protocolName, devices);
```

Segmento de código 31. Generación de archivos PCAP.

OLSR:

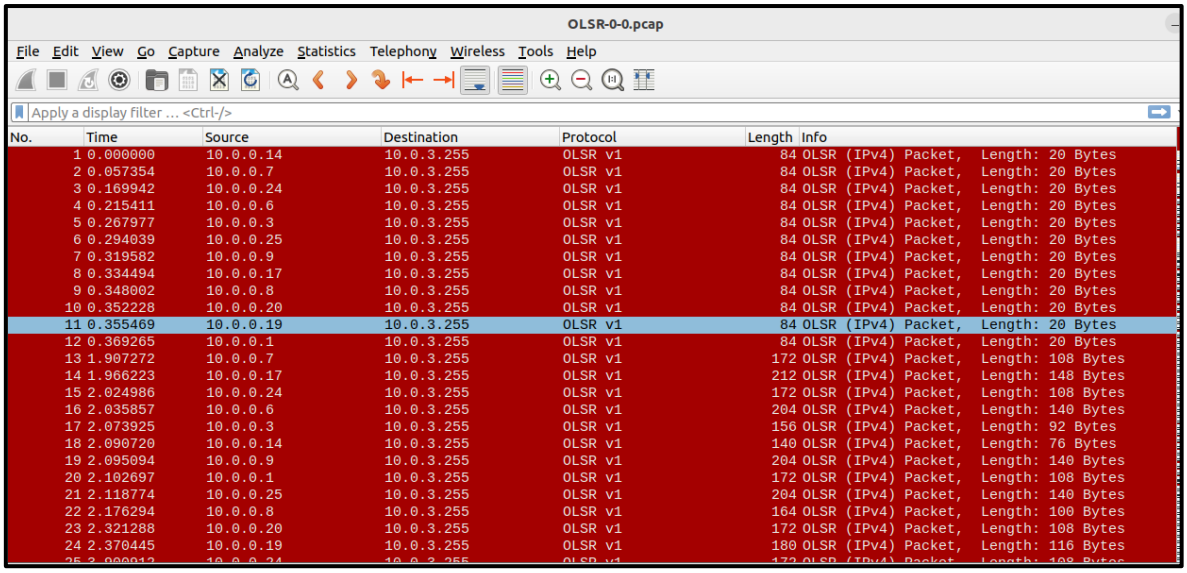

Figura 2.21. Archivo pcap de la red Fanet con protocolo OLSR.

AODV:

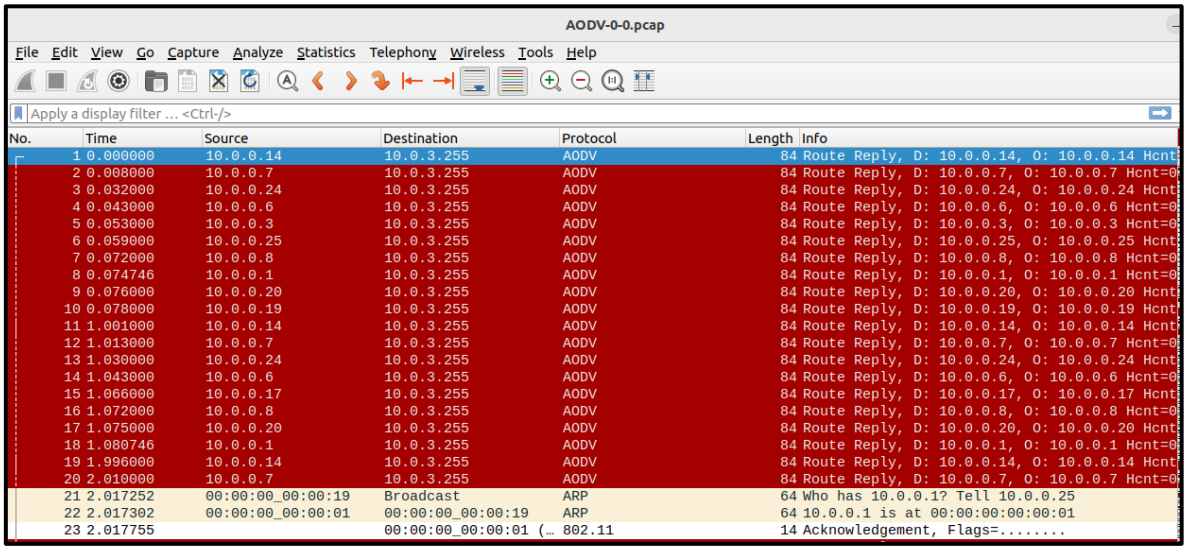

Figura 2.22. Archivo pcap de la red Fanet con protocolo AODV.

DSDV:

El nombre del protocolo en el archivo *.pcap* generado en la simulación, al ser analizado en Wireshark no aparece de manera adecuada, sin embargo, en el apartado 2.7.8 se observa que se ha implementado este protocolo.

DSR:

Acorde a [89] y como se observa en la Figura 2.23, este protocolo presenta problemas al ser identificado por Wireshark, debido a que utiliza una cabecera diferente a la implementada en el RFC 4728, por lo que se tiene un mensaje de error.

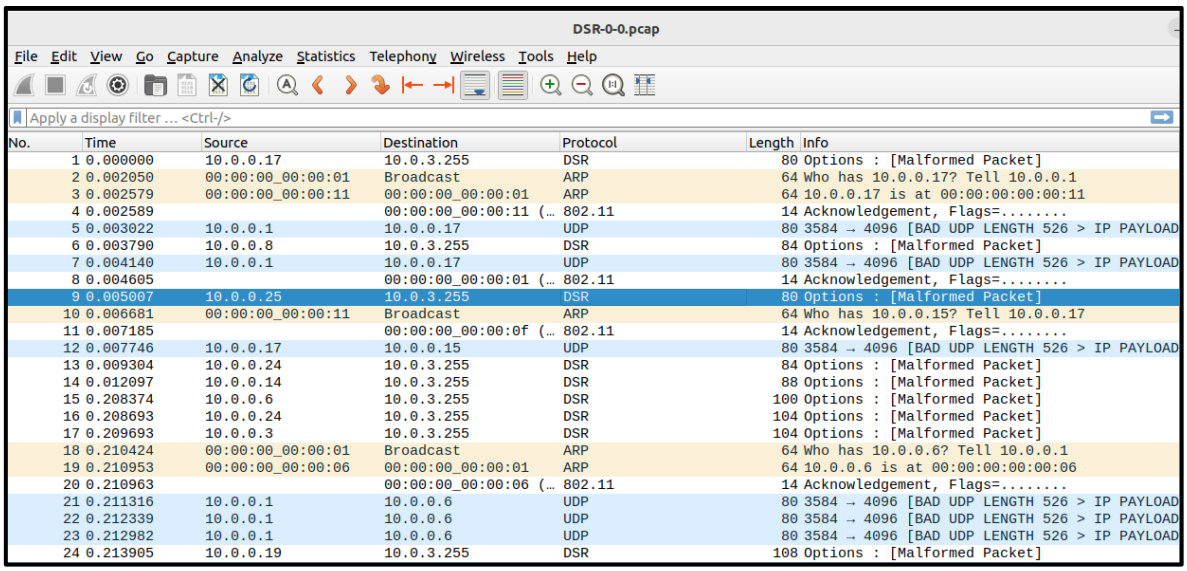

Figura 2.23. Archivo pcap de la red Fanet con protocolo DSR.

HWMP:

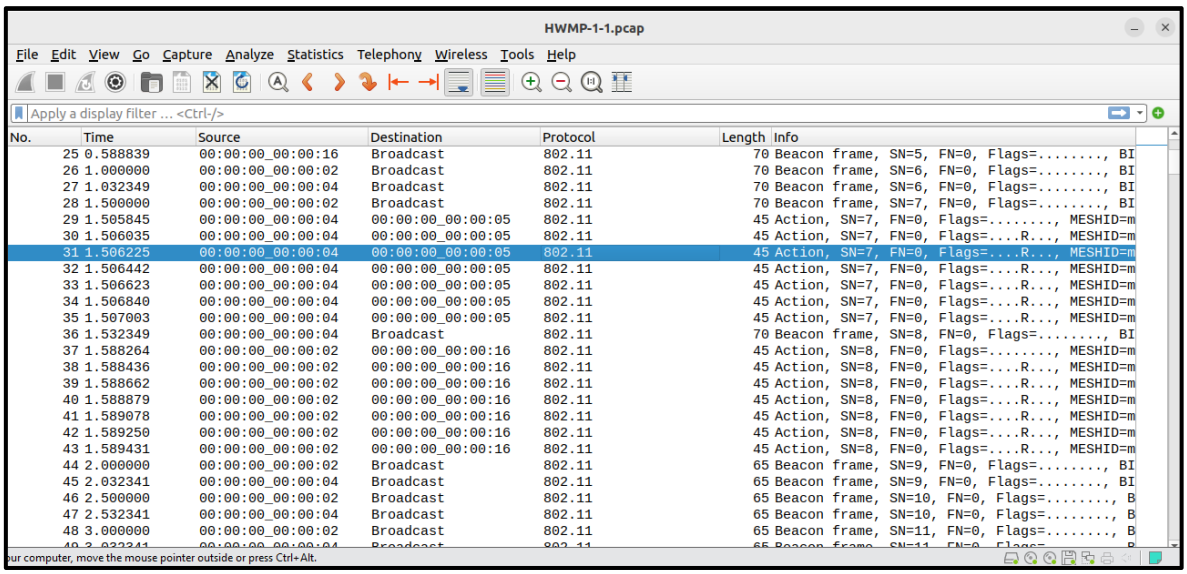

Figura 2.24. Archivo pcap de la red Fanet con protocolo HWMP.

AOMDV:

En este protocolo, a diferencia del protocolo AODV, se puede apreciar en la Figura 2.25 que se generan paquetes por medio de broadcast, además de paquetes 802.11

característicos de las redes tipo malla, esto debido a que en el apartado 2.7.7 se genera tráfico por múltiples rutas.

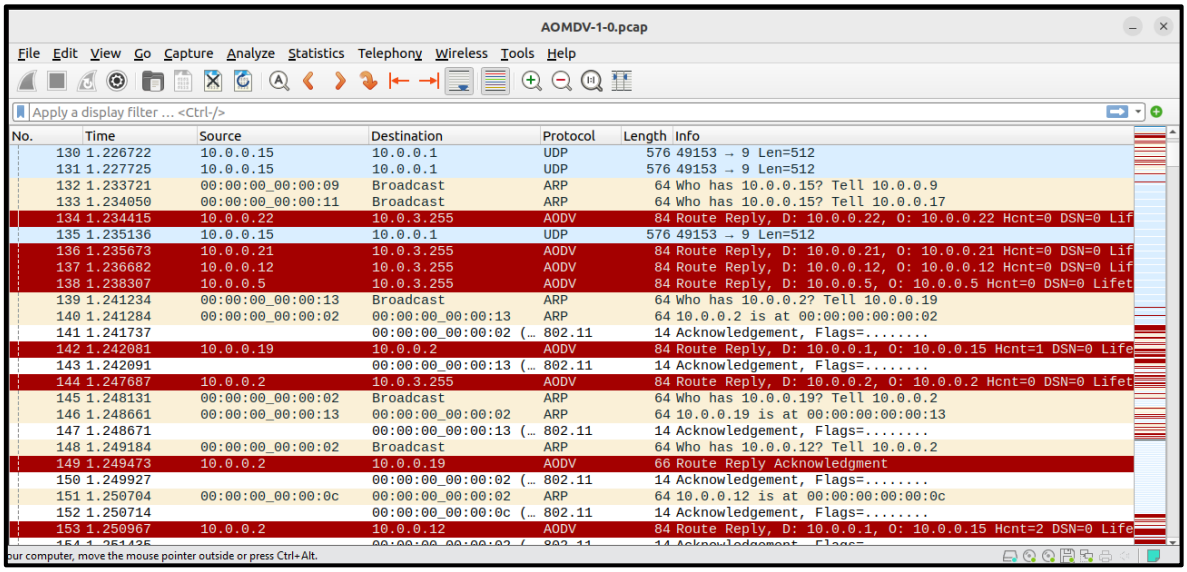

Figura 2.25. Archivo pcap de la red Fanet con protocolo AOMDV.

2.7.10 OBTENCIÓN DE RESULTADOS

Para el análisis del rendimiento de cada uno de los protocolos implementados se utiliza el módulo *Flow Monitor* definido en la sección 2.5 y se utilizan parámetros como el throughput, tasa de entrega de paquetes y retardo promedio.

Throughput:

Determina el rendimiento de la comunicación, esto en base al número total de paquetes entregados al destino en un periodo de tiempo y es medido en bits por segundo, se obtiene de la ecuación 2.1.

$$Throughput = \frac{bytes*8}{T_{transmision}} \quad (2.1)$$

En donde:

$bytes$ = es el tamaño total en bytes de paquetes entregados exitosamente.

Tasa de entrega de paquetes:

Porcentaje que representa la cantidad de paquetes enviados exitosamente al nodo destino con respecto a la cantidad de paquetes generados por un nodo origen [90], la ecuación 2.2 permite visualizar la obtención de este valor.

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

Retardo promedio:

Este parámetro está representado por la suma promedio de la diferencia de los retardos de cada paquete recibido por un nodo destino y el tiempo en que un paquete es enviado por un nodo origen [90], esto se obtiene en base a la ecuación 2.3.

$$Retardo\ promedio = \frac{\sum_{i=1}^{Paquete_{i\ recibido}} (T_{i\ recibido} - T_{i\ enviado})}{Paquetes_{recibidos}} \qquad (2.3)$$

Para la obtención de estos parámetros se utiliza la siguiente línea de código:

```
Ptr < Ipv4FlowClassifier > classifier = DynamicCast < Ipv4FlowClassifier >(flowmon.GetClassifier());
          std::map < FlowId, FlowMonitor::FlowStats > stats = monitor->GetFlowStats();

double Delaysum = 0;
uint64_t txPacketsum = 0;
uint64_t rxPacketsum = 0;
uint32_t txPacket = 0;
uint32_t rxPacket = 0;
uint32_t PacketLoss = 0;
uint64_t txBytessum = 0;
uint64_t rxBytessum = 0;
double delay;
double throughput = 0;
int flowId = 0;

for (std::map < FlowId, FlowMonitor::FlowStats > ::const_iterator iter = stats.begin(); iter != stats.end(); ++iter) {
                  Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow(iter->first);
txPacket = iter->second.txPackets;
rxPacket = iter->second.rxPackets;
PacketLoss = txPacket - rxPacket;
delay = iter->second.delaySum.GetMilliSeconds();
        std::cout << "  Tx Packets: " << iter->second.txPackets << "\n";
        std::cout << "  Rx Packets: " << iter->second.rxPackets << "\n";
        std::cout << "  Packet Loss: " << PacketLoss << "\n";
             std::cout << "  Tx Bytes:   " << iter->second.txBytes << "\n";
        std::cout << "  Rx Bytes:   " << iter->second.rxBytes << "\n";
        std::cout << "  Throughput: " << iter->second.rxBytes * 8.0 / 9.0 / 1000 / 1000  << " Mbps\n";
        NS_LOG_UNCOND("  Per Node Delay: " << delay / txPacket << " ms");
        std::cout << "   PDR for current flow ID : " << ((rxPacket *100) / txPacket) << "%" << "\n";
        flowId ++;
                  txPacketsum += iter->second.txPackets;
                  rxPacketsum += iter->second.rxPackets;
                  txBytessum += iter->second.txBytes;
                  rxBytessum += iter->second.rxBytes;
                  Delaysum += iter->second.delaySum.GetMilliSeconds();
            throughput += iter->second.rxBytes * 8.0 / 9.0 / 1000 / 1000;
           }
NS_LOG_UNCOND("**********Resultados promedio*************");
NS_LOG_UNCOND("Paquetes enviados = " << txPacketsum);
NS_LOG_UNCOND("Paquetes recibidos = " << rxPacketsum);
NS_LOG_UNCOND("Total Paquetes Perdidos = " << (txPacketsum-rxPacketsum));
NS_LOG_UNCOND("Total Bytes Enviados = "<<txBytessum);
NS_LOG_UNCOND("Total Bytes Recibidos = "<<rxBytessum);
NS_LOG_UNCOND("Delay: " << Delaysum / txPacketsum << " ms");
std::cout << "Throughput Promedio= "<<throughput/flowId << " Mbit/s" << std::endl;
std::cout << "Packet Delivery Ratio: " << ((rxPacketsum *100) / txPacketsum) << "%" << "\n";
```

Segmento de código 32. Obtención de resultados.

Los resultados de la simulación de cada escenario se obtienen como se muestra en la Figura 2.26:

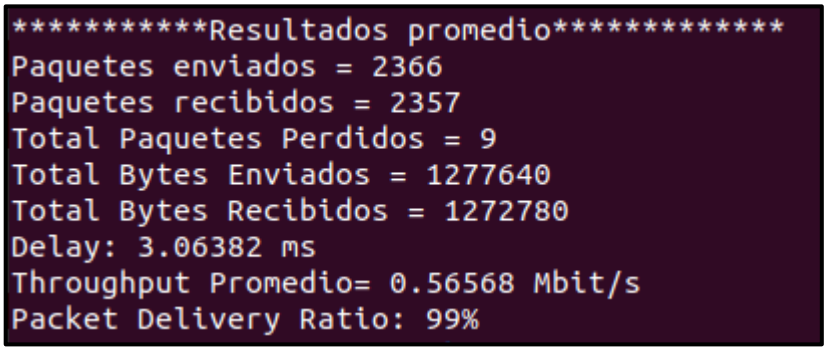

Figura 2.26. Resultados de simulación obtenidos del módulo Flow Monitor.

Acorde a [91], el protocolo DSR no es compatible con el módulo *Flow Monitor*, por lo que no es posible obtener información del rendimiento de este protocolo para su análisis.

3. RESULTADOS Y DISCUSIÓN

En este capítulo se explican los escenarios simulados, así como los resultados de las simulaciones ejecutadas de los cuales se hará un análisis de los mismos para determinar los protocolos de mejor rendimiento. Para ello se realizan pruebas con cada protocolo de enrutamiento en dos escenarios, con el fin de establecer una comparativa de rendimiento entre los protocolos implementados.

Cabe mencionar que acorde se mencionó en el apartado 2.7.10, no es posible obtener resultados de la implementación del protocolo DSR por lo que no se tienen datos para su comparación, debido a esto solo se realiza la comparación de los protocolos OLSR, AODV, DSDV, AOMDV, HWMP.

En el análisis se hará referencia a la tasa de entrega de paquetes con su abreviación en inglés: PDR (Packet Delivery Ratio), de igual manera que para determinar si un protocolo tiene mejor rendimiento se considera lo siguiente:

- Throughput: Mientras mayor sea este valor, es mejor.
- PDR: Mientras mayor sea este valor, es mejor.
- Retardo promedio: Mientras menor sea este valor, es mejor.

3.1 ESCENARIOS A IMPLEMENTAR

Como se indicó en la sección 1.2, se implementan tres escenarios:

1. **Escenario 1**: UAVs en vuelo en posición estática, en donde se realizan pruebas con 25 y 50 UAVs.
2. **Escenario 2:** UAVs en vuelo a velocidad máxima de 10 [m/s] en posiciones aleatorias, en donde se realizan pruebas con 25 y 50 UAVs.
3. **Escenario 3:** UAVs en vuelo a velocidad máxima de 20 [m/s] en posiciones aleatorias, en donde se realizan pruebas con 25 y 50 UAVs

La variación de la velocidad, así como la posición son clave para la obtención de datos más cercanos a la realidad, puesto que en un despliegue de una red FANET la alta movilidad es una característica clave, por otro lado, la variación del número de dispositivos permite verificar la variación en el rendimiento de estos protocolos en estos escenarios.

Como se explica en la sección 2.7.4, el parámetro *MeanVelocity* es el que permite tener nodos en vuelo en posición estática o con velocidad variable, en donde al establecer el valor máximo como 0 se obtiene un nodo estático en vuelo, mientras que al colocar otro valor como máximo el nodo se moverá a esa velocidad máxima promedio en m/s.

Los parámetros utilizados en la simulación se describen en la Tabla 3.1:

Tabla 3.1. Parámetros utilizados en la simulación.

Protocolos de enrutamiento	OLSR, AODV, DSDV, AOMDV, HWMP
Tiempo de simulación	100 segundos
Área de simulación	500x500 [m]
Altitud	10 a 50 [m]
Número de nodos UAV	25, 50
Protocolo de comunicación	802.11g
Tamaño del paquete	512 bytes
Modelo de movilidad	Gauss Markov
Velocidad del nodo	0,10,20 [m/s]

3.1.1 ESCENARIO 1 – 25 UAVS FIJOS

Este escenario plantea 25 nodos en vuelo en posición fija, en donde, la simulación es ejecutada con el parámetro *MeanVelocity* con valor máximo 0 como se muestra en la Tabla 3.2.

Tabla 3.2. Número y velocidad de los nodos para el Escenario 1 – Prueba 1.

Escenario 1 – Prueba 1	
Número de nodos	25
Velocidad de los nodos [m/s]	0

De las simulaciones realizadas considerando los parámetros de la tabla 4, se obtienen los resultados de la Tabla 3.3.

Tabla 3.3. Resultados del Escenario 1 – Prueba 1.

	OLSR	**AODV**	**DSDV**	**HWMP**	**AOMDV**
Throughput [Mbps]	0.56568	0.0135497	0.57408	1.15968	0.0366941
PDR [%]	99	95	100	100	50
Retardo [ms]	3.06382	15.0968	2.5209	2.60348	2.38542

- **Throughput.**

De los resultados obtenidos, se puede observar en la Figura 3.1 que los protocolos con mejor rendimiento para el escenario propuesto son OLSR, DSDV y HWMP, en donde OLSR y DSDV tienen rendimiento similar.

Por otro lado, los protocolos AODV y AOMDV son los que menor rendimiento presentan, sin embargo, AOMDV prácticamente duplica el rendimiento que AODV por la característica de múltiples rutas.

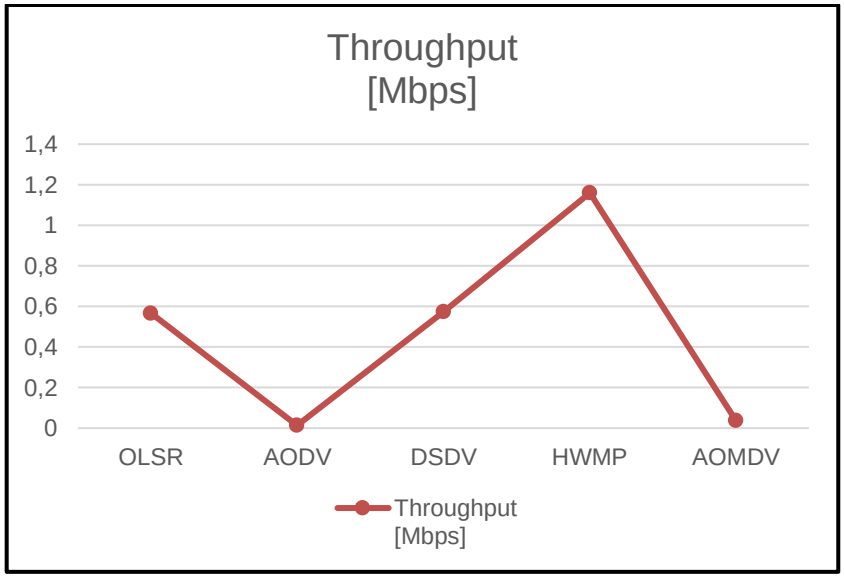

Figura 3.1. Gráfico comparativo Throughput, Escenario 1 – Prueba 1.

- **PDR.**

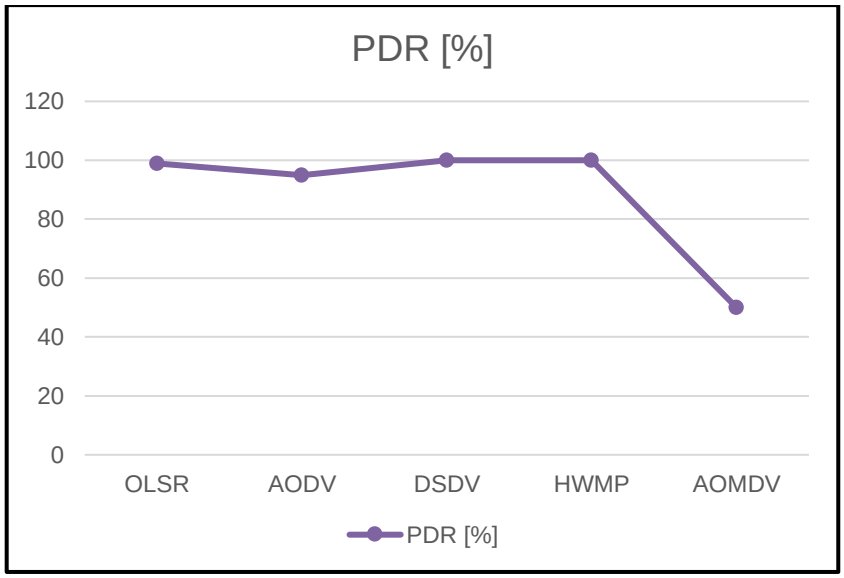

Figura 3.2. Gráfico comparativo PDR, Escenario 1 – Prueba 1.

En la Figura 3.2, se puede observar el gráfico comparativo de los valores obtenidos de las simulaciones de este escenario, en donde este parámetro en casi todos los protocolos se mantiene con la misma tendencia, a excepción del protocolo AOMDV.

- **Retardo promedio.**

Como se observa en la Figura 3.3, los protocolos con mejores resultados son DSDV y AOMDV, seguidos por el protocolo HWMP y OLSR, siendo el protocolo AODV el de menor rendimiento.

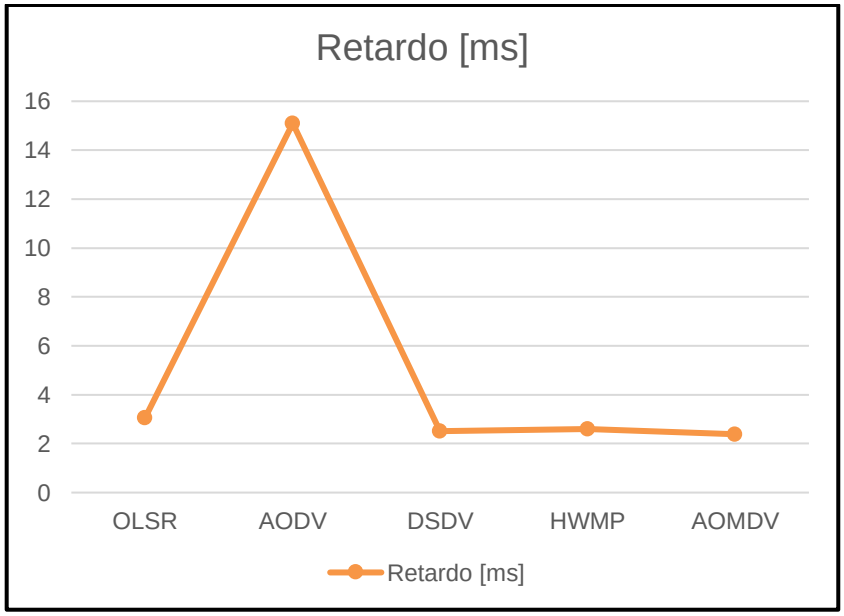

Figura 3.3. Gráfico comparativo retardo, Escenario 1 – Prueba 1.

De este primer escenario se puede concluir que los protocolos con mejor rendimiento son HWMP y DSDV ya que tienen valores cercanos en los tres parámetros, en donde el protocolo HWMP puede ser desplegado en aplicaciones que requieran mayor throughput, este valor alto de throughput se puede asociar a la característica propia del protocolo, la cual es establecer una red de tipo malla para la comunicación, además de su comportamiento de tipo híbrido.

Por otro lado, se puede considerar el despliegue de la red con el protocolo DSDV en caso de que el hardware a bordo no sea compatible con el protocolo HWMP.

3.1.2 ESCENARIO 1 – 50 UAVS FIJOS

Este escenario plantea 50 nodos en vuelo con posición fija, en donde, el parámetro *MeanVelocity* debe tener como valor máximo 0 como se muestra en la Tabla 3.4.

Tabla 3.4. Número y velocidad de los nodos para el Escenario 1 – Prueba 2.

Escenario 1 – Prueba 2	
Número de nodos	50
Velocidad de los nodos [m/s]	0

De las simulaciones realizadas considerando los parámetros de la tabla 6, se obtienen los resultados de la Tabla 3.5.

Tabla 3.5. Resultados del Escenario 1 – Prueba 2.

	OLSR	**AODV**	**DSDV**	**HWMP**	**AOMDV**
Throughput [Mbps]	0.57144	0.00134039	0.57384	1.15998	0.0630773
PDR [%]	100	83	99	99	99
Retardo [ms]	1.8446	91.9817	2.3056	2.65148	4.19201

- **Throughput.**

De los resultados obtenidos, se puede observar en la Figura 3.4 que se mantiene la misma tendencia que el escenario anterior, en donde, los protocolos con mejor rendimiento para el escenario propuesto siguen siendo OLSR, DSDV y HWMP, en donde HWMP es el mejor protocolo en este escenario, seguido de OLSR y DSDV.

Además, los protocolos AODV y AOMDV son los que mantienen un bajo rendimiento.

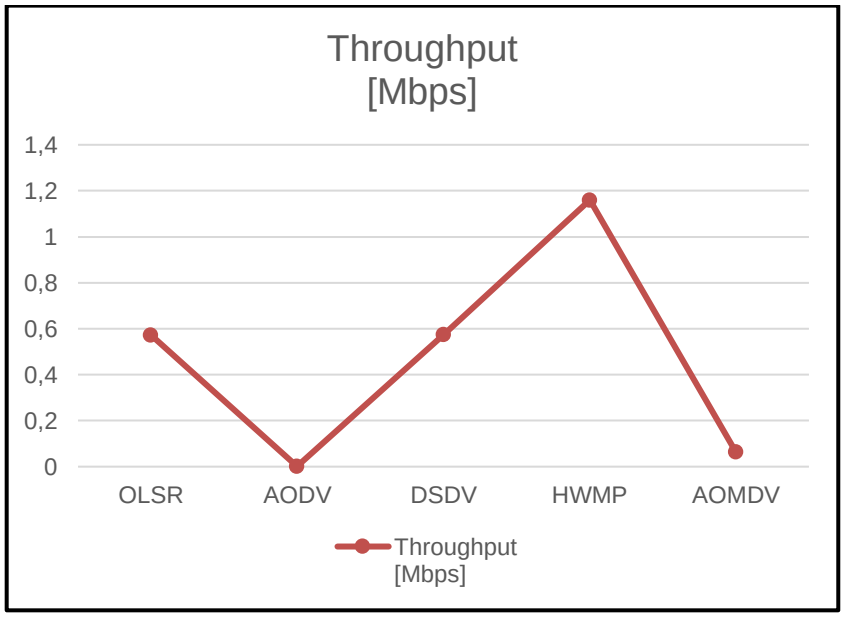

Figura 3.4. Gráfico comparativo Throughput, Escenario 1 – Prueba 2.

- **PDR.**

En la Figura 3.5, se puede observar que este parámetro en casi todos los protocolos se mantiene con la misma tendencia, a excepción del protocolo AODV en donde tiene el menor rendimiento de los protocolos implementados.

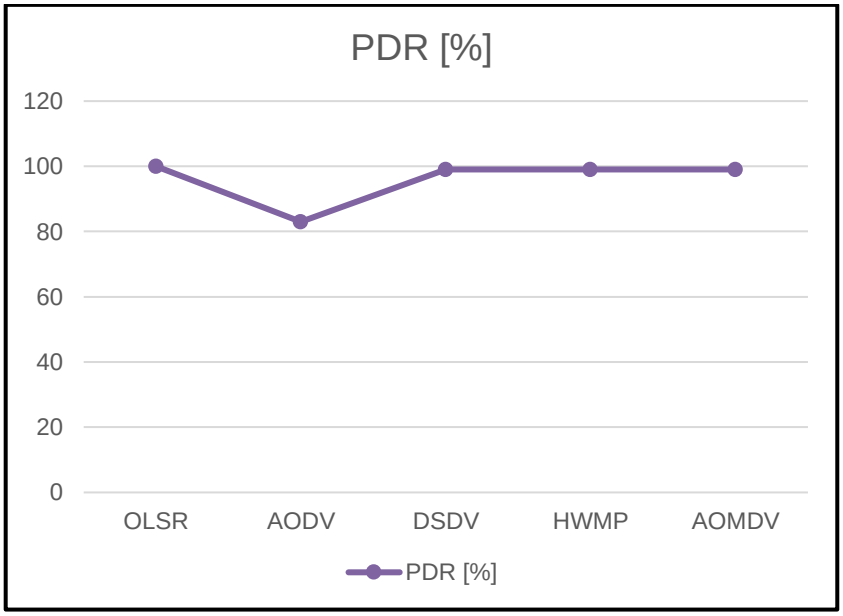

Figura 3.5. Gráfico comparativo PDR, Escenario 1 – Prueba 2.

- **Retardo promedio.**

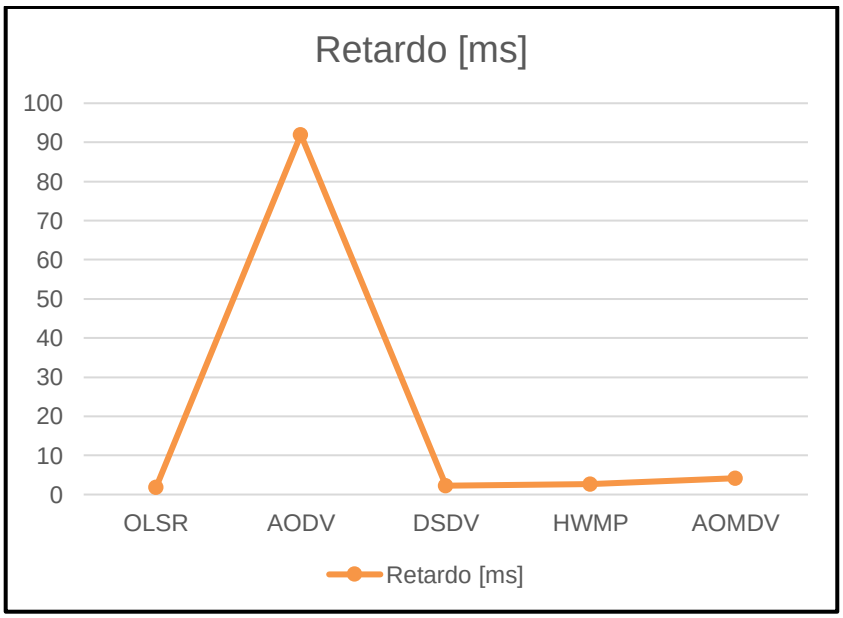

Figura 3.6. Gráfico comparativo retardo, Escenario 1 – Prueba 2.

Como se observa en la Figura 3.6, los protocolos con mejores resultados son OLSR y DSDV, seguidos por el protocolo HWMP y AOMDV, manteniéndose el protocolo AODV con el menor rendimiento similar al escenario anterior.

De este escenario con mayor número de UAVs, se puede concluir que los protocolos con mejor rendimiento considerando que el retardo es un valor crítico son OLSR, DSDV y HWMP, siendo el protocolo OLSR con el menor retardo con throughput similar al protocolo DSDV, sin embargo, en aplicaciones en las cuales, se requiera mayor throughput y bajo retardo, el protocolo HWMP puede ser utilizado en el despliegue de la red Fanet.

3.1.3 COMPARACIÓN DE RENDIMIENTO PARA EL ESCENARIO 1

Este escenario considero nodos en vuelo en posición estática y se obtienen los resultados que se presentan a continuación.

- **Throughput.**

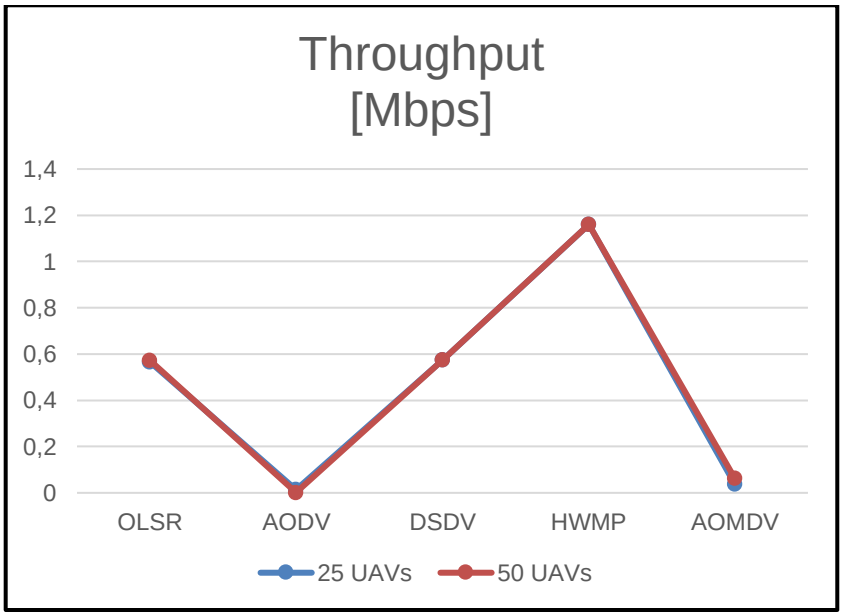

Figura 3.7. Gráfico comparativo Throughput, Escenario 1 – Pruebas 1 y 2.

Se observa en la Figura 3.7, que los protocolos presentan rendimiento similar y el cambio de número de UAVs no genera mayor variación en los resultados.

- **PDR**

En la Figura 3.8 se puede observar que el aumento de número de UAVs solo afecta al rendimiento de los protocolos AODV y AOMDV, en el caso del protocolo AOMDV mejora el rendimiento y en el caso del protocolo AODV lo reduce.

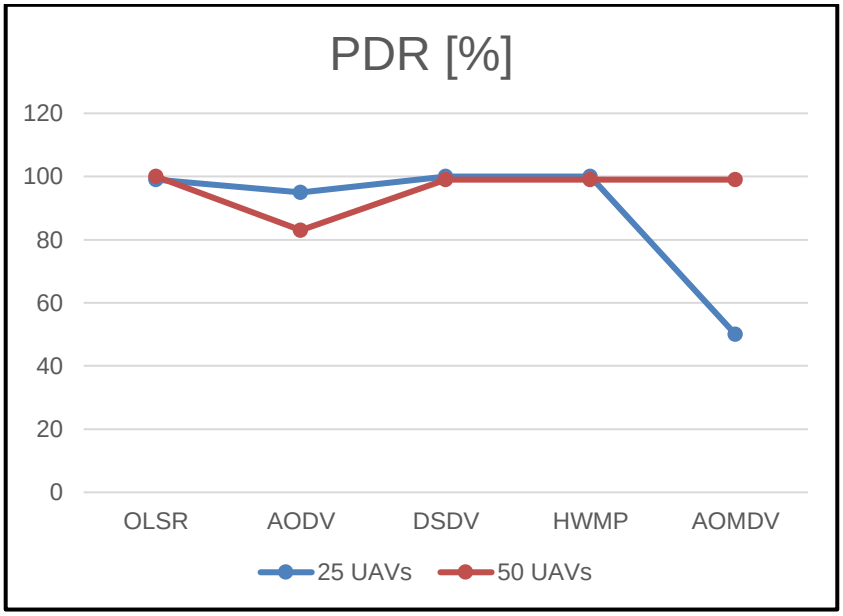

Figura 3.8. Gráfico comparativo PDR, Escenario 1 – Pruebas 1 y 2.

- **Retardo**

Se puede apreciar en la Figura 3.9 que el rendimiento del protocolo AODV es significativamente menor al aumentar el número de dispositivos UAVs, en los otros protocolos el rendimiento se mantiene.

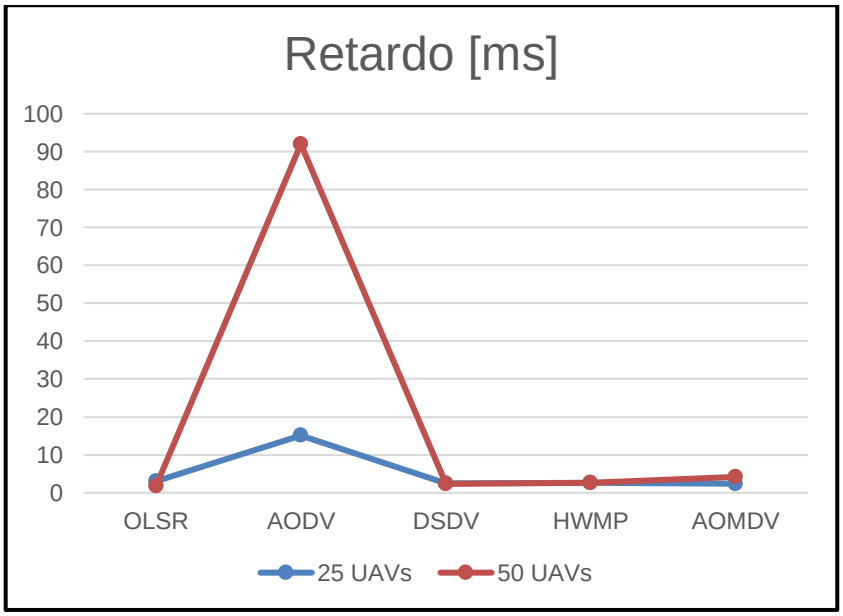

Figura 3.9. Gráfico comparativo retardo, Escenario 1 – Pruebas 1 y 2.

De estas pruebas se puede concluir que el mejor protocolo para el despliegue de una red Fanet con nodos en vuelo en posición fija es el protocolo HWMP ya que presenta valores similares inclusive después de la variación del número de UAVs.

3.1.4 ESCENARIO 2 – 25 UAVS EN VUELO ALEATORIO CON VELOCIDAD 10 [m/s]

A partir de este escenario se plantea que los nodos estén en vuelo en posición aleatoria con velocidad variable, en donde, el parámetro *MeanVelocity* debe tener como valor máximo 10 como se muestra en la Tabla 3.6.

Tabla 3.6. Número y velocidad de los nodos para el Escenario 2 – Prueba 1.

Escenario 2 – Prueba 1	
Número de nodos	25
Velocidad de los nodos [m/s]	10

Una vez ejecutada la simulación de este escenario, se obtienen los resultados de la Tabla 3.7.

Tabla 3.7. Resultados del Escenario 2 – Prueba 1.

	OLSR	**AODV**	**DSDV**	**HWMP**	**AOMDV**
Throughput [Mbps]	0.43464	0.0027197	0.42816	0.0156	0.00825765
PDR [%]	76	83	74	49	82
Retardo [ms]	3.7929	42.6142	6.28888	0.94717	24.565

- **Throughput.**

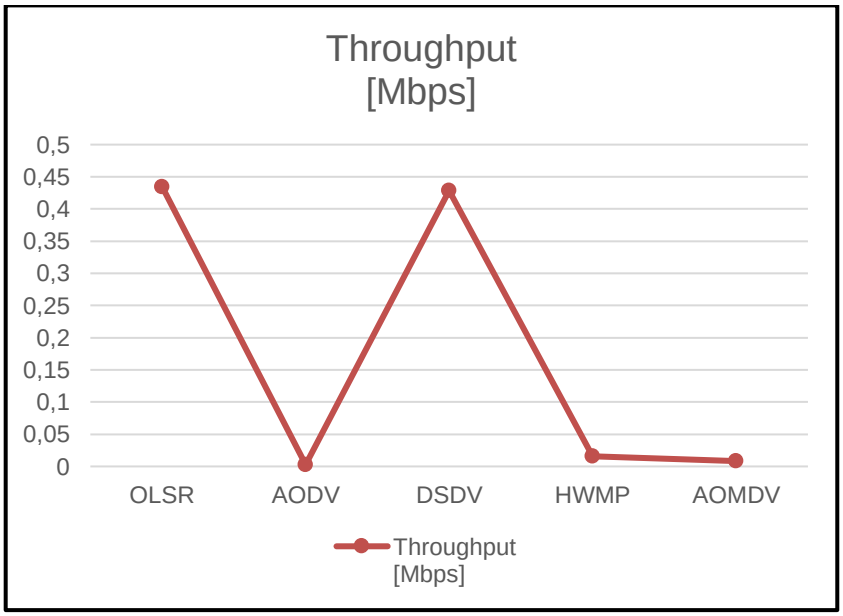

Figura 3.10. Gráfico comparativo Throughput, Escenario 2 – Prueba 1.

De los resultados obtenidos, se puede observar en la Figura 3.10 que los protocolos con mejor rendimiento para el escenario propuesto siguen siendo OLSR y DSDV con valores similares como se tienen en la Tabla 9.

Los protocolos AODV y AOMDV son los que mantienen un bajo rendimiento, de igual manera se puede observar que el protocolo HWMP tiene un descenso considerable en rendimiento, esto debido a que existe alta movilidad entre los nodos haciendo que algunos lleguen a perder conectividad y procedan a buscar una nueva ruta para la transferencia de datos.

- **PDR.**

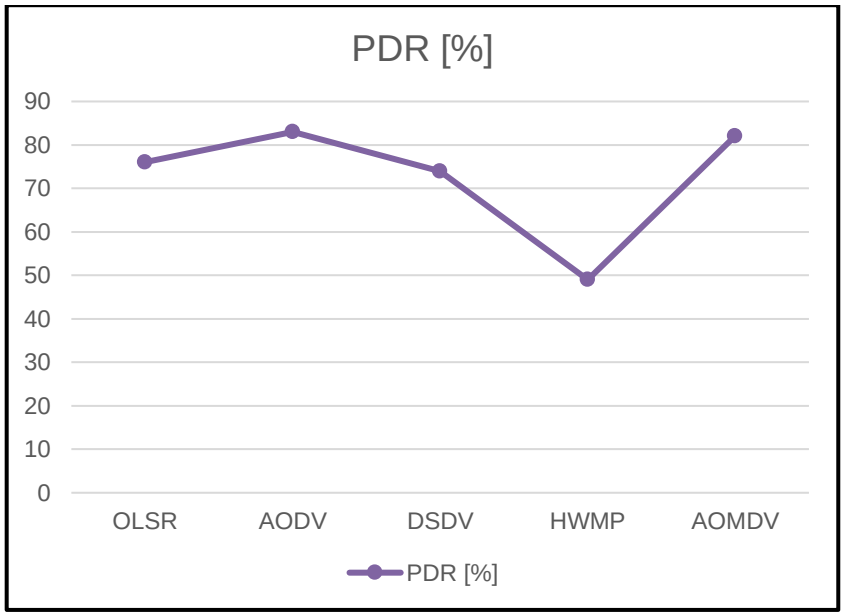

Figura 3.11. Gráfico comparativo PDR, Escenario 2 – Prueba 1.

En la Figura 3.11, se puede observar que este parámetro presenta variaciones considerables en todos los protocolos, siendo los protocolos AODV y AOMDV los de mejor rendimiento, seguidos por los protocolos OLSR y DSDV. Por otro lado, el protocolo HWMP presenta el menor rendimiento, esto coincide con el hecho de tener mayor movilidad y presentar posibles desconexiones entre nodos.

- **Retardo promedio.**

Como se observa en la Figura 3.12, el protocolo HWMP es el que mayor rendimiento presenta, seguido de los protocolos OLSR y DSDV.

Por otro lado, los protocolos AODV y AOMDV son los que menor rendimiento presentan.

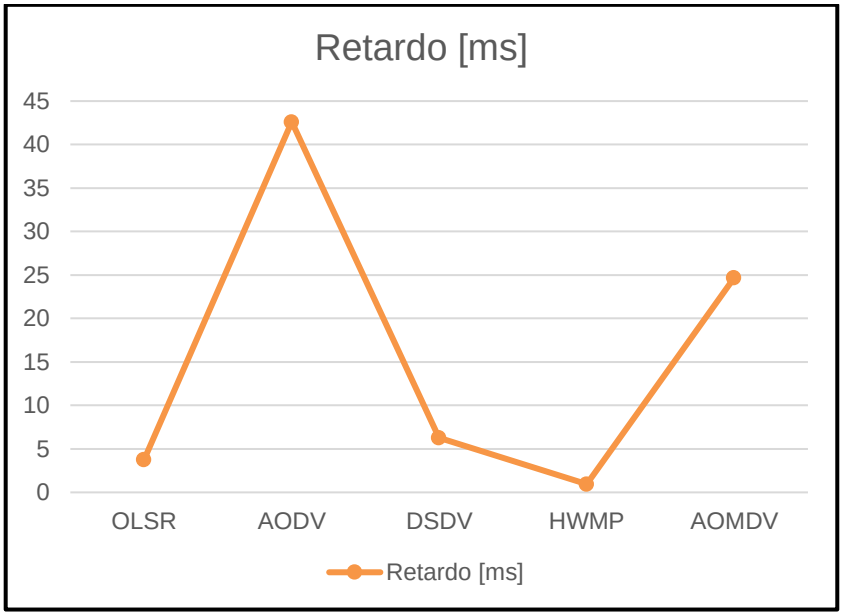

Figura 3.12. Gráfico comparativo retardo, Escenario 2 – Prueba 1.

De este escenario con movilidad aleatoria con una velocidad máxima definida, se puede concluir que los protocolos con mejor rendimiento son OLSR y DSDV, siendo el protocolo OLSR con el menor retardo.

3.1.5 ESCENARIO 2 – 50 UAVS EN VUELO ALEATORIO CON VELOCIDAD 10 [m/s]

Este escenario plantea 50 nodos en vuelo con posición aleatoria y velocidad variable, en donde, el parámetro MeanVelocity debe tener como valor máximo 10, como se muestra en la Tabla 3.8.

Tabla 3.8. Número y velocidad de los nodos para el Escenario 2 – Prueba 2.

Escenario 2 – Prueba 2	
Número de nodos	50
Velocidad de los nodos [m/s]	10

Considerando esto, para cada protocolo se obtienen los resultados de la Tabla 3,9.

Tabla 3.9. Resultados del Escenario 2 – Prueba 2.

	OLSR	**AODV**	**DSDV**	**HWMP**	**AOMDV**
Throughput [Mbps]	0.52728	0.000605836	0.4656	0.0113	0.001315
PDR [%]	92	74	81	41	80
Retardo [ms]	3.33221	115.527	3.29097	1.94717	84.9287

- **Throughput.**

De los resultados obtenidos, se puede observar en la Figura 3.13 que los protocolos se mantienen con un rendimiento similar al escenario anterior, en donde, OLSR y DSDV son los de mejor rendimiento con valores similares.

Los protocolos AODV, AOMDV y HWMP son los que mantienen un bajo rendimiento.

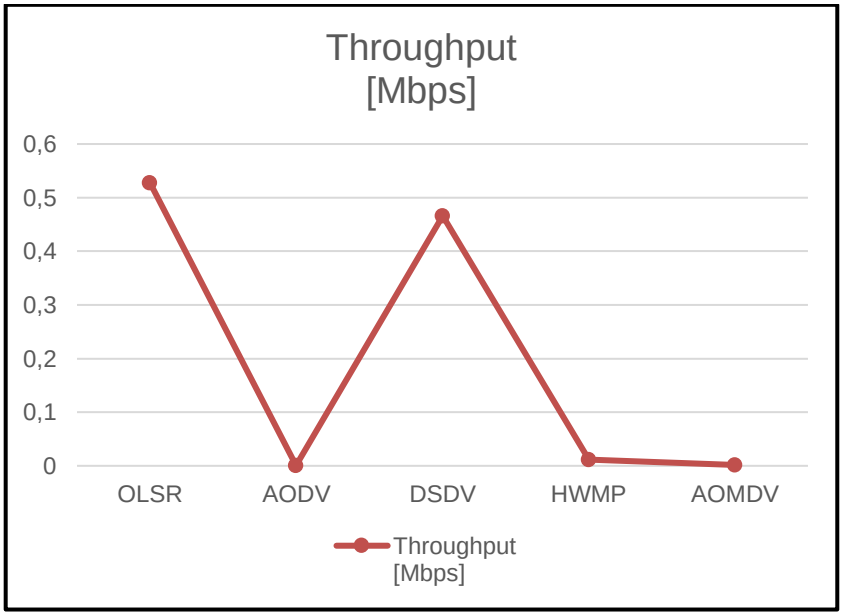

Figura 3.13. Gráfico comparativo Throughput, Escenario 2 – Prueba 2.

- **PDR.**

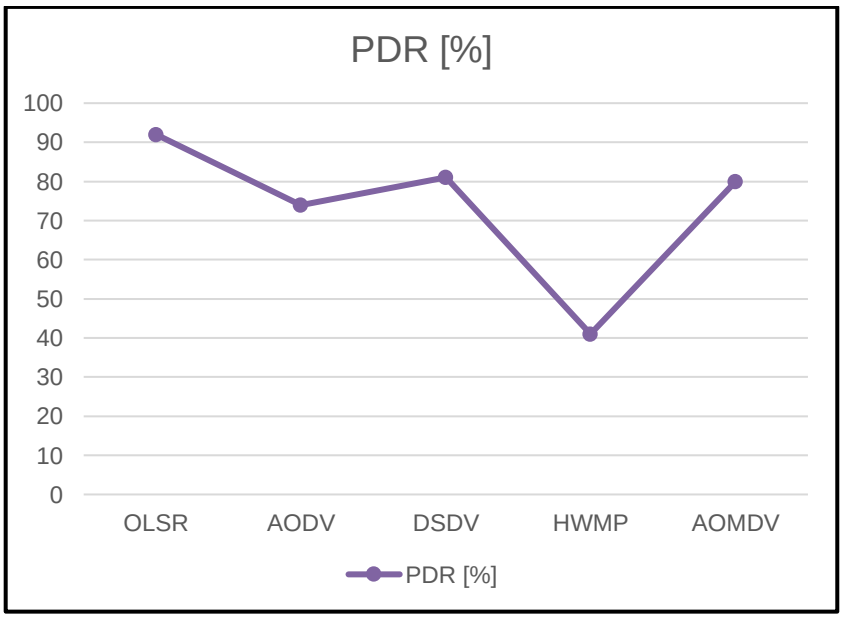

Figura 3.14. Gráfico comparativo PDR, Escenario 2 – Prueba 2.

En la Figura 3.14, se puede observar que este parámetro presenta variaciones considerables en todos los protocolos, siendo el protocolo OLSR con mejor rendimiento, además, los protocolos DSDV y AOMDV presentan rendimiento similar, seguidos por el protocolo AODV y el protocolo HWMP es el que menor rendimiento presenta.

- **Retardo promedio.**

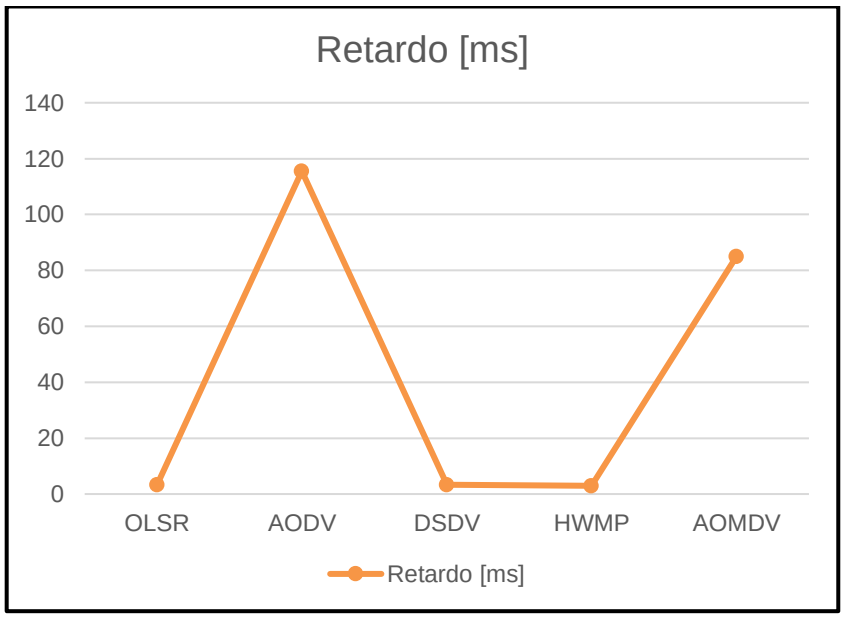

Figura 3.15. Gráfico comparativo retardo, Escenario 2 – Prueba 2.

Como se observa en la Figura 3.15, el protocolo HWMP es el que mayor rendimiento presenta, seguido de los protocolos OLSR y DSDV.

Por otro lado, los protocolos AODV y AOMDV son los que menor rendimiento presentan.

De este escenario con movilidad aleatoria, con velocidad máxima definida y mayor número de UAVs, se puede concluir que los protocolos con mejor rendimiento son OLSR y DSDV, sin embargo, el protocolo HWMP si bien no presenta throughput similar a OLSR y DSDV presenta menor retardo que ambos, lo cual indica que puede ser utilizado en escenarios en donde se requiera un retardo y throughput bajos.

3.1.6 COMPARACIÓN DE RENDIMIENTO PARA EL ESCENARIO 2

Este escenario consideró nodos en vuelo en posición aleatoria con velocidad de movimiento máxima de 10 [m/s]

- **Throughput.**

Se observa en la Figura 3.16, que los protocolos presentan rendimiento similar con variaciones mínimas en los valores por el cambio del número de UAVs, en el caso de los

protocolos OLSR y DSDV tiene una influencia positiva aumentando el valor de este parámetro.

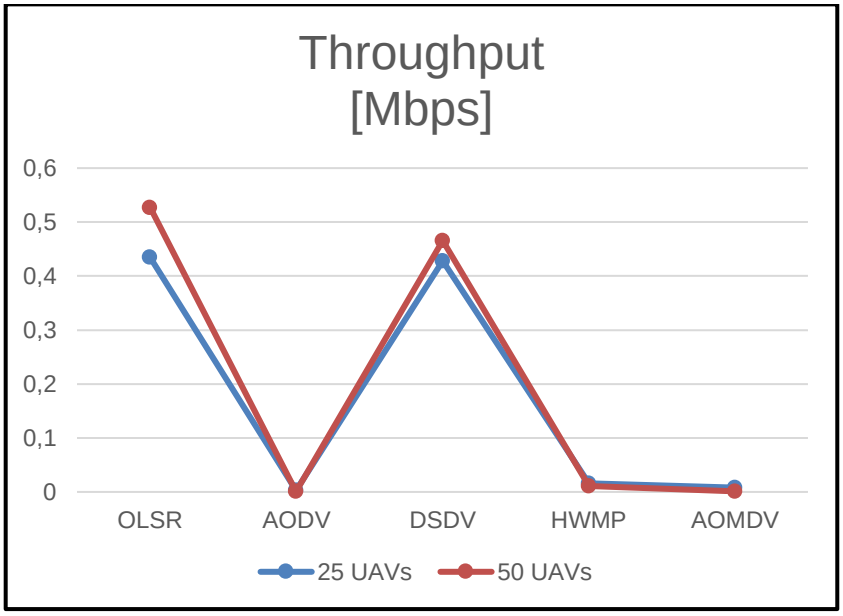

Figura 3.16. Gráfico comparativo Throughput, Escenario 1 – Pruebas 1 y 2.

- **PDR**

En la Figura 3.17 se puede observar que el aumento de número de UAVs influye en el rendimiento de todos los protocolos, teniendo un impacto negativo en los protocolos AODV, HWMP y AOMDV, por otro lado, se tiene un mejor rendimiento en los protocolos OLSR y DSDV.

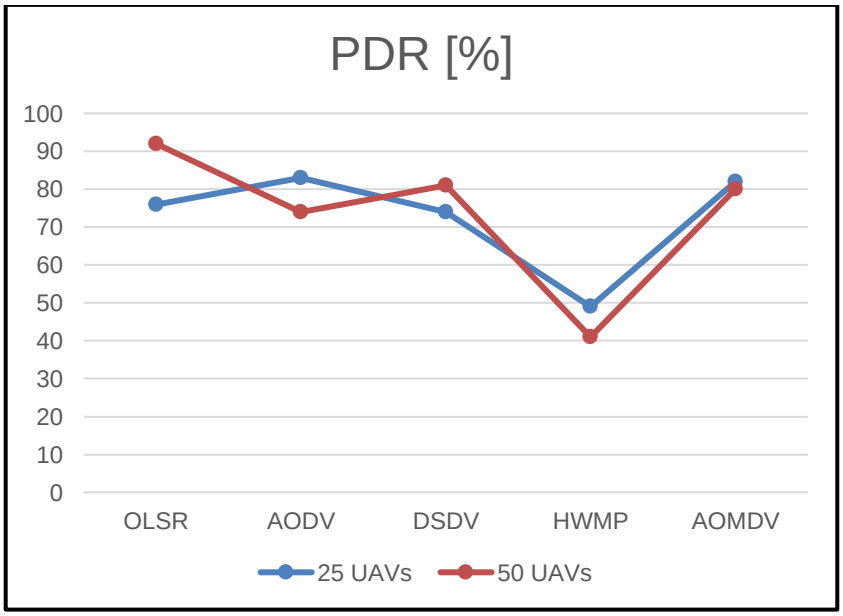

Figura 3.17. Gráfico comparativo PDR, Escenario 1 – Pruebas 1 y 2.

- **Retardo**

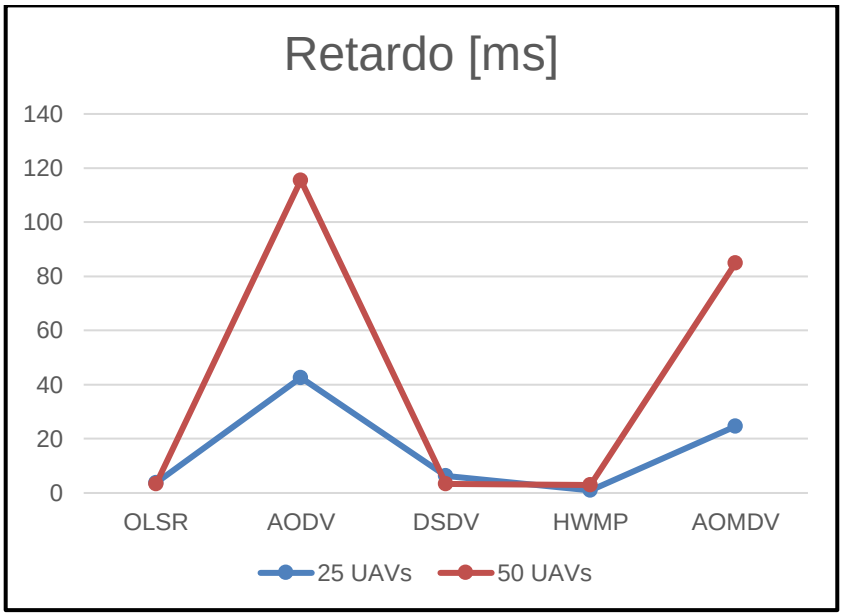

Figura 3.18. Gráfico comparativo retardo, Escenario 1 – Pruebas 1 y 2.

Se puede apreciar en la Figura 3.18 que el rendimiento de los protocolos AODV y AOMDV es significativamente menor al aumentar el número de dispositivos UAVs, en los otros protocolos el rendimiento se mantiene con variaciones mínimas.

De estas pruebas se puede concluir que el mejor protocolo para el despliegue de una red Fanet con nodos en vuelo en posición aleatoria con velocidad máxima de 10 [m/s] es el protocolo OLSR ya que presenta valores similares inclusive después de la variación del número de UAVs, con un mejor valor de throughput al aumentar el número de UAVs.

3.1.7 ESCENARIO 3 – 25 UAVS EN VUELO ALEATORIO CON VELOCIDAD 20 [m/s]

Este escenario plantea 25 nodos en vuelo aleatorio con velocidad variable, en donde, el parámetro *MeanVelocity* debe tener como valor máximo 20, como se muestra en la Tabla 3.10.

Tabla 3.10. Número y velocidad de los nodos para el Escenario 3 – Prueba 1.

Escenario 3 – Prueba 1	
Número de nodos	25
Velocidad de los nodos [m/s]	20

Considerando esto se ejecuta la simulación de cada protocolo y se obtienen los resultados de la Tabla 3.11.

Tabla 3.11. Resultados del Escenario 3 – Prueba 1.

	OLSR	AODV	DSDV	HWMP	AOMDV
Throughput [Mbps]	0.3408	0.001640026	0.3504	0.0204	0.00265459
PDR [%]	60	58	61	59	39
Retardo [ms]	7.62766	37.3525	6.00878	1.01053	19.3706

- **Throughput.**

De los resultados obtenidos, se puede observar en la Figura 3.19 que los protocolos se mantienen con un rendimiento similar al escenario 2, en donde, OLSR y DSDV son los de mejor rendimiento con valores similares.

Los protocolos AODV, AOMDV y HWMP son los protocolos que mantienen un bajo rendimiento.

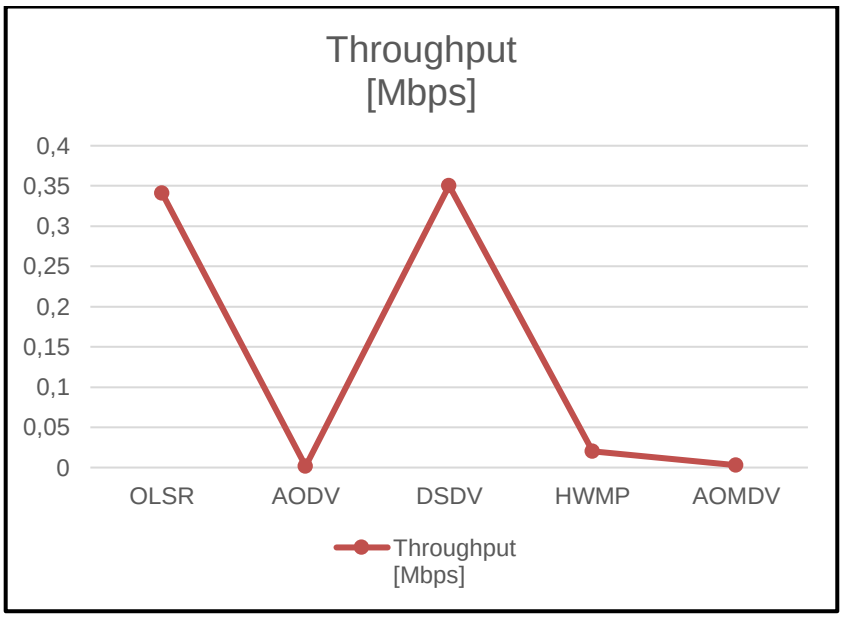

Figura 3.19. Gráfico comparativo Throughput, Escenario 3 – Prueba 1.

- **PDR.**

En la Figura 3.20, se puede observar que este parámetro presenta valores similares en los protocolos, a excepción del protocolo AOMDV siendo el protocolo DSDV con mejor rendimiento.

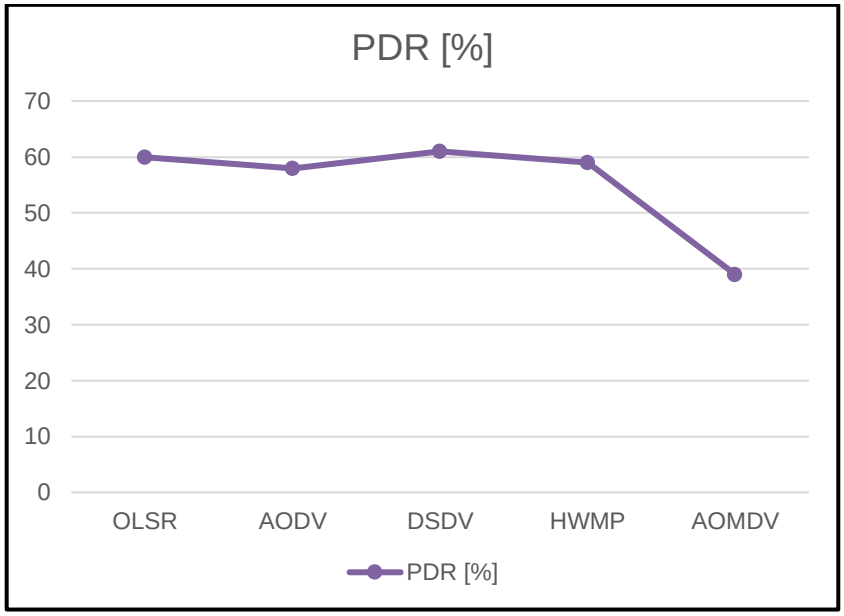

Figura 3.20. Gráfico comparativo PDR, Escenario 3 – Prueba 1.

- **Retardo promedio.**

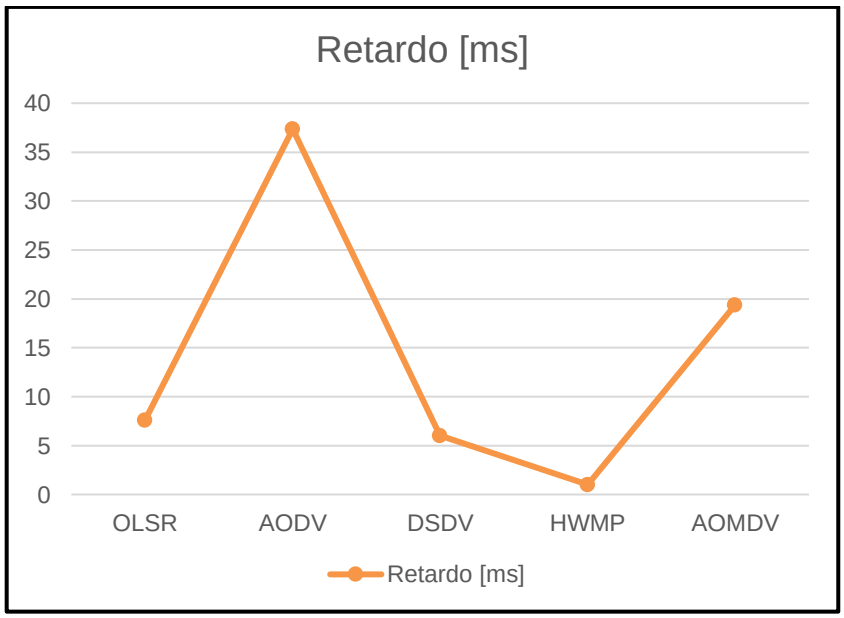

Figura 3.21. Gráfico comparativo retardo, Escenario 3 – Prueba 1.

Como se observa en la Figura 3.21, el protocolo HWMP es el que mayor rendimiento presenta, seguido de los protocolos OLSR y DSDV.

Por otro lado, los protocolos AODV y AOMDV son los que menor rendimiento presentan.

De este escenario con movilidad aleatoria con una velocidad máxima de 20 [m/s], se puede concluir que los protocolos con mejor rendimiento son OLSR y DSDV, siendo el protocolo OLSR con el menor retardo.

3.1.8 ESCENARIO 3 – 50 UAVS EN VUELO ALEATORIO CON VELOCIDAD 20 [m/s]

Este escenario plantea 50 nodos fijos, en donde, el parámetro *MeanVelocity* debe tener como valor máximo 0 como se aprecia en la Tabla 3.12.

Tabla 3.12. Número y velocidad de los nodos para el Escenario 3 – Prueba 2.

Escenario 3 – Prueba 2	
Número de nodos	50
Velocidad de los nodos [m/s]	20

Considerando esto se ejecuta la simulación de cada protocolo y se obtienen los resultados de la Tabla 3.13.

Tabla 3.13. Resultados del Escenario 3 – Prueba 2.

	OLSR	**AODV**	**DSDV**	**HWMP**	**AOMDV**
Throughput [Mbps]	0.42816	0.000524964	0.34272	0.0140	0.00093628
PDR [%]	74	74	59	51	71
Retardo [ms]	8.53381	99.5114	1.55686	1.18745	84.7941

- **Throughput.**

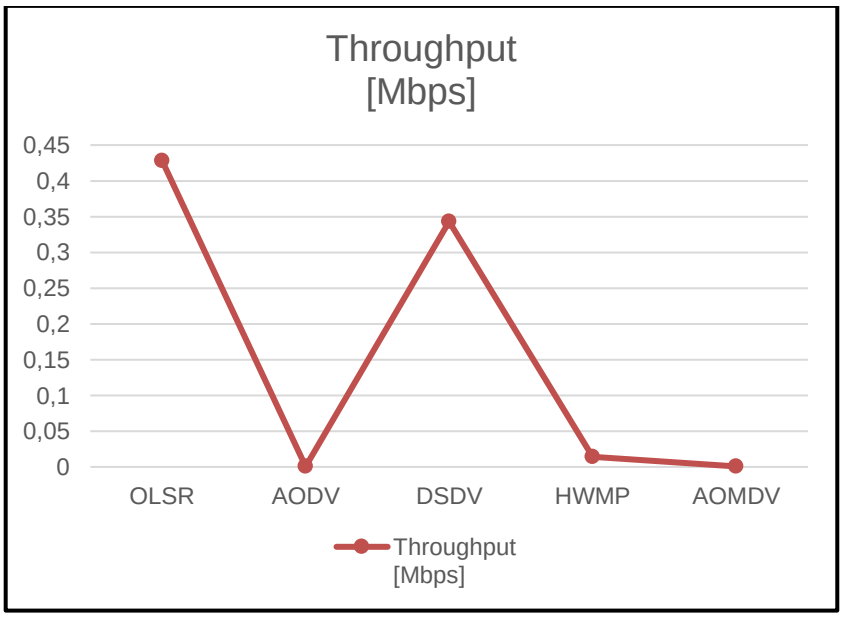

Figura 3.22. Gráfico comparativo Throughput, Escenario 3 – Prueba 2.

De los resultados obtenidos, se puede observar en la Figura 3.22 que el protocolo OLSR es el de mejor rendimiento, seguido por el protocolo DSDV, mientras que los protocolos AODV, AOMDV y HWMP son los que mantienen un bajo rendimiento.

- **PDR.**

En la Figura 3.23, se puede observar que este parámetro presenta valores similares en los protocolos OLSR, AODV y AOMDV. Por otro lado, los protocolos DSDV y HWMP presentan bajo rendimiento.

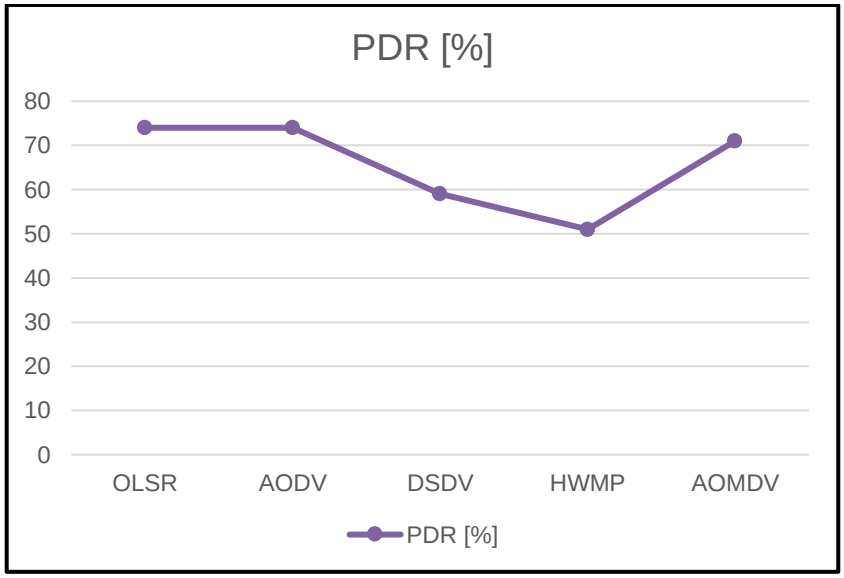

Figura 3.23. Gráfico comparativo PDR, Escenario 3 – Prueba 2.

- **Retardo promedio.**

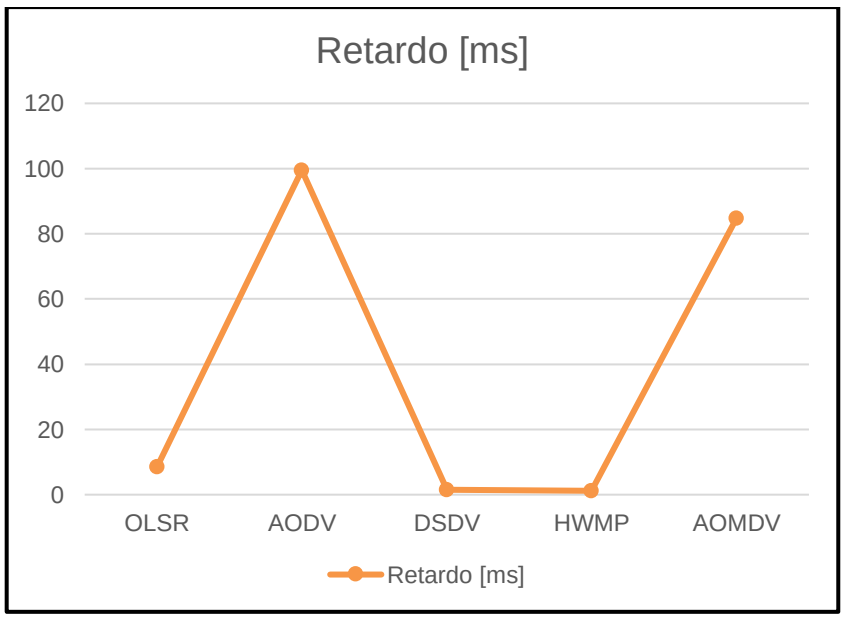

Figura 3.24. Gráfico comparativo retardo, Escenario 3 – Prueba 2.

Como se observa en la Figura 3.24, los protocolos DSDV y HWMP son los protocolos que mejor rendimiento presentan, seguido del protocolo OLSR. Por otro lado, los protocolos AODV y AOMDV son los que menor rendimiento presentan.

De este escenario con movilidad aleatoria con una velocidad máxima de 20 [m/s], se puede concluir que los protocolos con mejor rendimiento son OLSR y DSDV, siendo el protocolo OLSR con el menor retardo.

3.1.9 COMPARACIÓN DE RENDIMIENTO PARA EL ESCENARIO 3

Este escenario consideró nodos en vuelo en posición aleatoria con velocidad de movimiento máxima de 20 [m/s]

- **Throughput.**

Se observa en la Figura 3.25, que los protocolos presentan rendimiento similar al escenario 2, con variaciones mínimas en el rendimiento por el cambio del número de UAVs, en el caso del protocolo OLSR tiene una influencia positiva aumentando el valor de este parámetro.

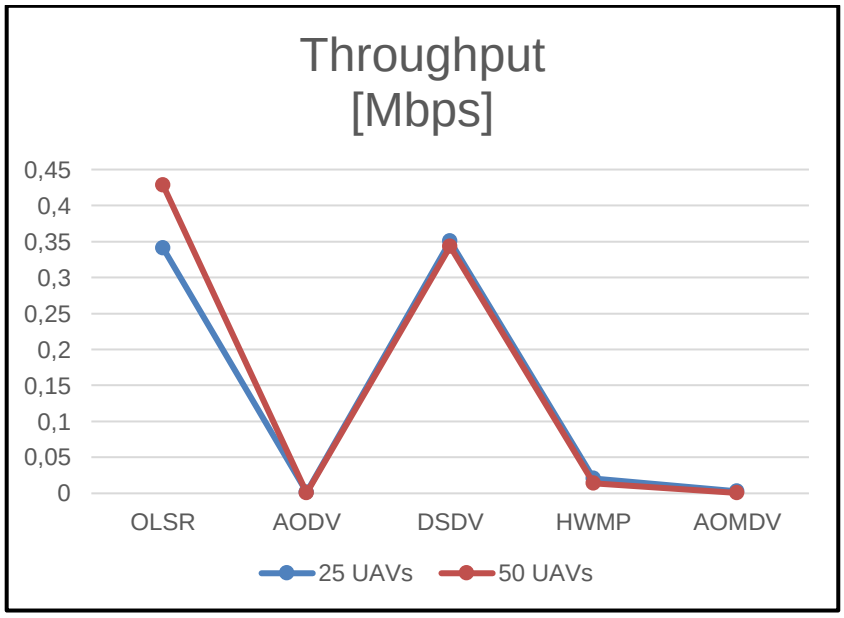

Figura 3.25. Gráfico comparativo Throughput, Escenario 1 – Pruebas 1 y 2.

- **PDR**

En la Figura 3.26 se puede observar que el aumento de número de UAVs influye en el rendimiento de todos los protocolos, teniendo un impacto negativo en los DSDV y HWMP, por otro lado, se tiene un mejor rendimiento en los protocolos OLSR, AODV y AOMDV.

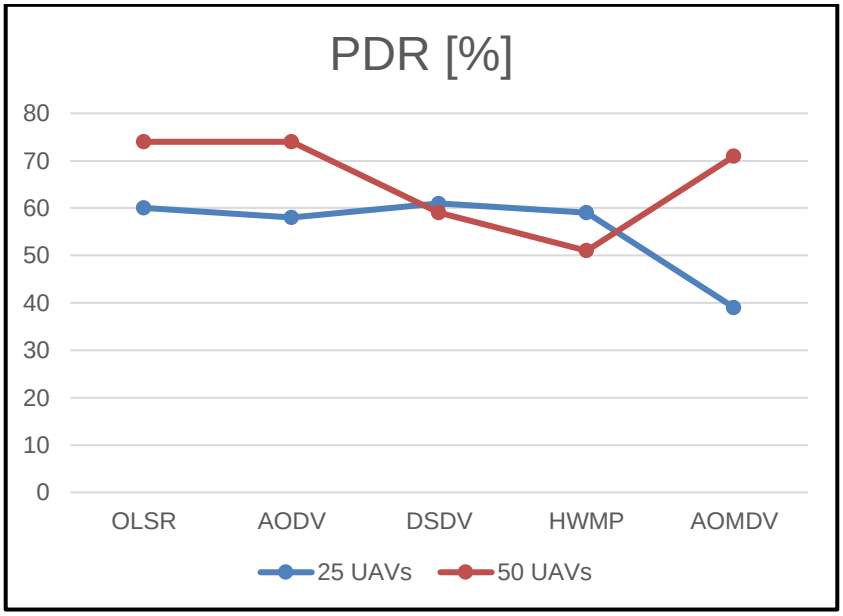

Figura 3.26. Gráfico comparativo PDR, Escenario 1 – Pruebas 1 y 2.

- **Retardo**

Se puede apreciar en la Figura 3.27, que de manera similar al escenario 2, el rendimiento de los protocolos AODV y AOMDV es significativamente menor al aumentar el número de dispositivos UAVs, en los otros protocolos el rendimiento se mantiene con variaciones mínimas.

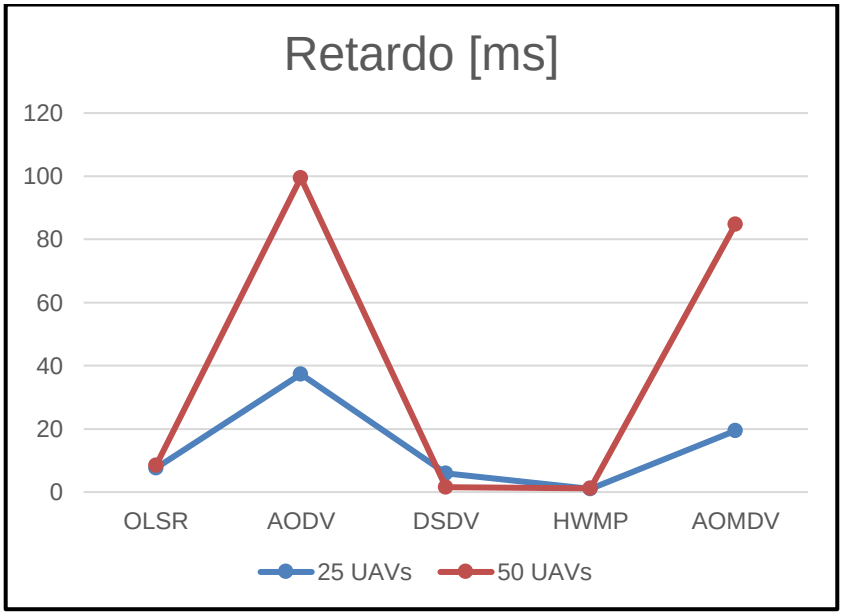

Figura 3.27. Gráfico comparativo retardo, Escenario 1 – Pruebas 1 y 2.

De estas pruebas se puede concluir que el mejor protocolo para el despliegue de una red Fanet con nodos en vuelo en posición aleatoria con velocidad máxima de 20 [m/s] es el protocolo OLSR ya que presenta valores similares inclusive después de la variación del número de UAVs, con un mejor valor de throughput al aumentar el número de UAVs.

3.1.10 COMPARACIÓN DE RENDIMIENTO PARA LOS ESCENARIO 1,2, Y 3 CON 25 UAVS

Esta comparación busca definir el mejor protocolo en los 3 escenarios con 50 UAVs, estableciendo como punto de comparación la velocidad máxima de movimiento.

- **Throughput.**

Se observa en la Figura 3.28, que la variación de velocidad influye mayormente en el protocolo HWMP, en donde al tener velocidad diferente de 0 el protocolo reduce drásticamente el rendimiento.

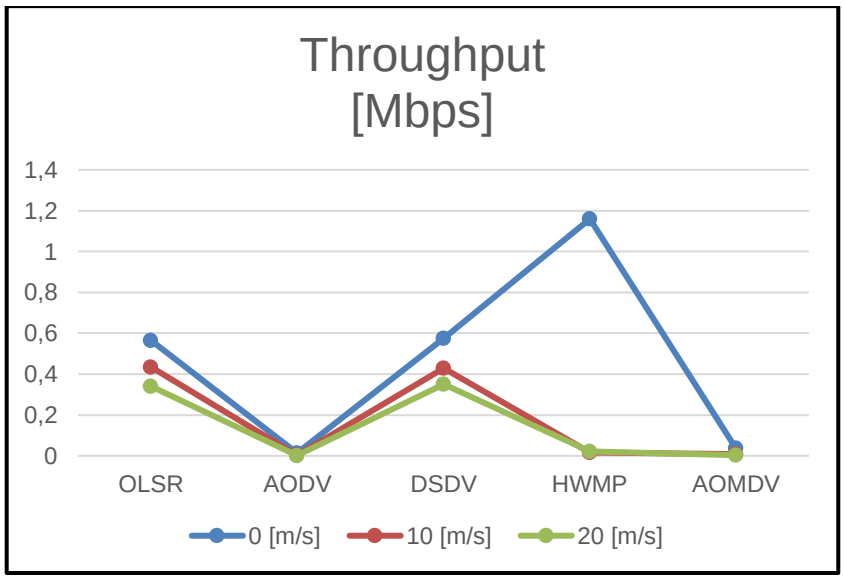

Figura 3.28. Gráfico comparativo del Throughput, Escenario 1 – Pruebas 1 y 2.

- **PDR**

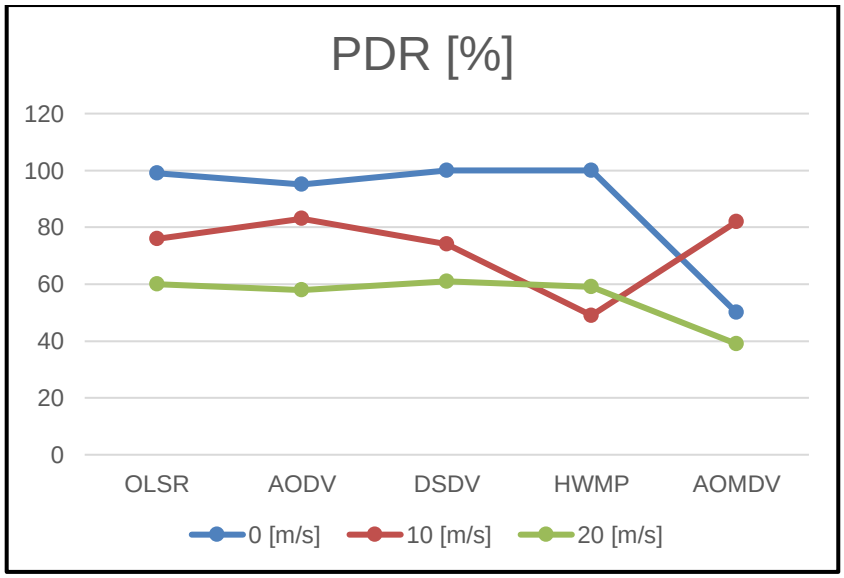

Figura 3.29. Gráfico comparativo del PDR, Escenario 1 – Pruebas 1 y 2.

En la Figura 3.29 se puede observar que al aumentar la velocidad de movimiento reduce el rendimiento de este parámetro para todos los protocolos, sin embargo, un caso particular es el protocolo HWMP en donde mejora ligeramente el rendimiento al aumentar la velocidad de movimiento.

- **Retardo**

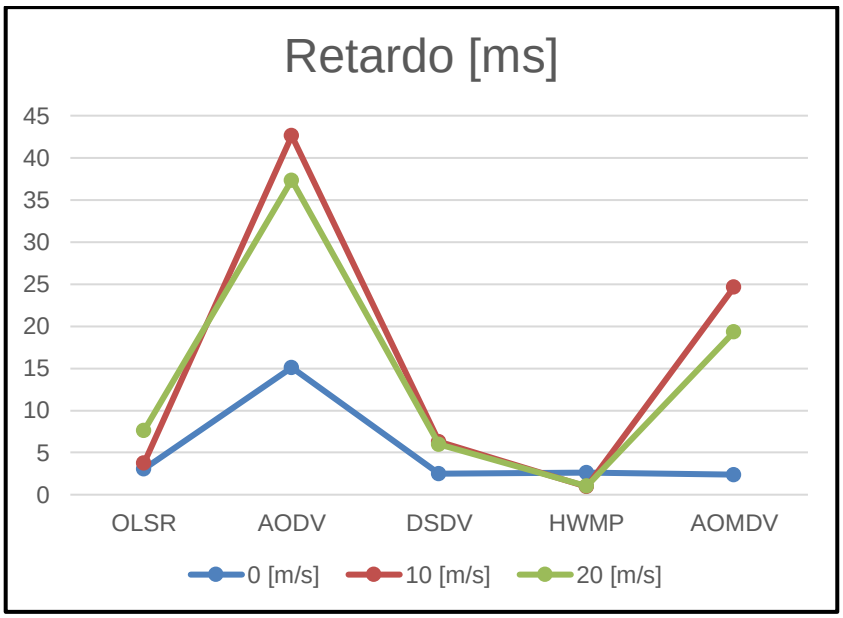

Figura 3.30. Gráfico comparativo del Retardo, Escenario 1 – Pruebas 1 y 2.

Se puede apreciar en la Figura 3.30, que al aumentar la velocidad de movimiento existe un aumento en el retardo de los protocolos AODV y AOMDV, en donde es mínima si la velocidad es diferente de 0.

Para los protocolos OLSR, DSDV y HWMP la variación de la velocidad produce un cambio mínimo en este parámetro,

De estas pruebas se puede concluir que el mejor protocolo para el despliegue de una red Fanet con nodos en vuelo en posición aleatoria con velocidad máxima variable son los protocolos OLSR y DSDV, ya que presentan valores similares inclusive después de la variación de la velocidad, con un valor menor de PDR al aumentar la velocidad de movimiento.

3.1.11 COMPARACIÓN DE RENDIMIENTO PARA LOS ESCENARIO 1,2, Y 3 CON 50 UAVS

Esta comparación busca definir el mejor protocolo en los 3 escenarios con 50 UAVs, estableciendo como punto de comparación la velocidad máxima de movimiento.

- **Throughput.**

Se observa en la Figura 3.31, similar a tener menor número de nodos, la variación de velocidad influye mayormente en el protocolo HWMP, en donde al tener velocidad diferente de 0 el protocolo reduce drásticamente el rendimiento.

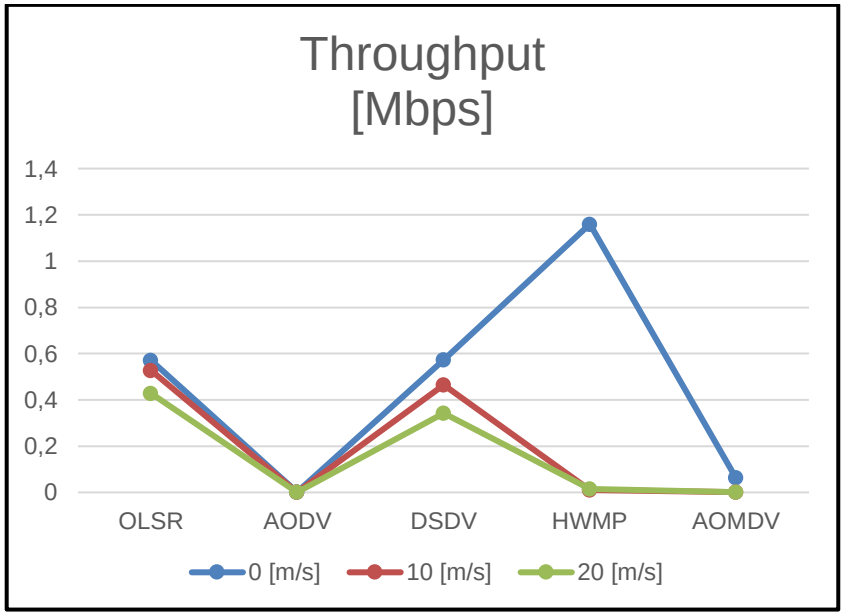

Figura 3.31. Gráfico comparativo del Throughput, Escenario 1 – Pruebas 1 y 2.

- **PDR**

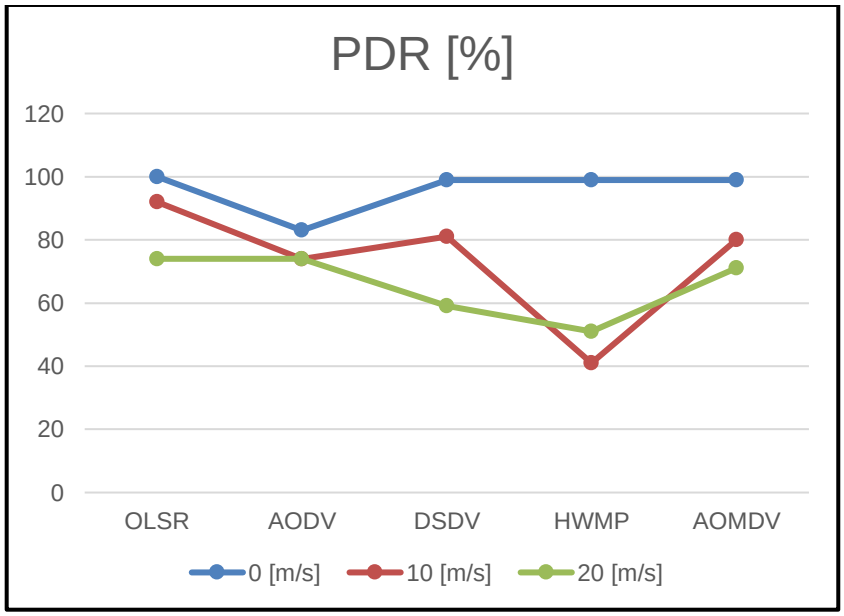

Figura 3.32. Gráfico comparativo del PDR, Escenario 1 – Pruebas 1 y 2.

En la Figura 3.32 se puede observar que al aumentar la velocidad de movimiento reduce ligeramente el rendimiento de todos los protocolos, sin embargo, de igual manera que en

el apartado 3.1.11, el protocolo HWMP mejora ligeramente el rendimiento al aumentar la velocidad de movimiento.

- **Retardo**

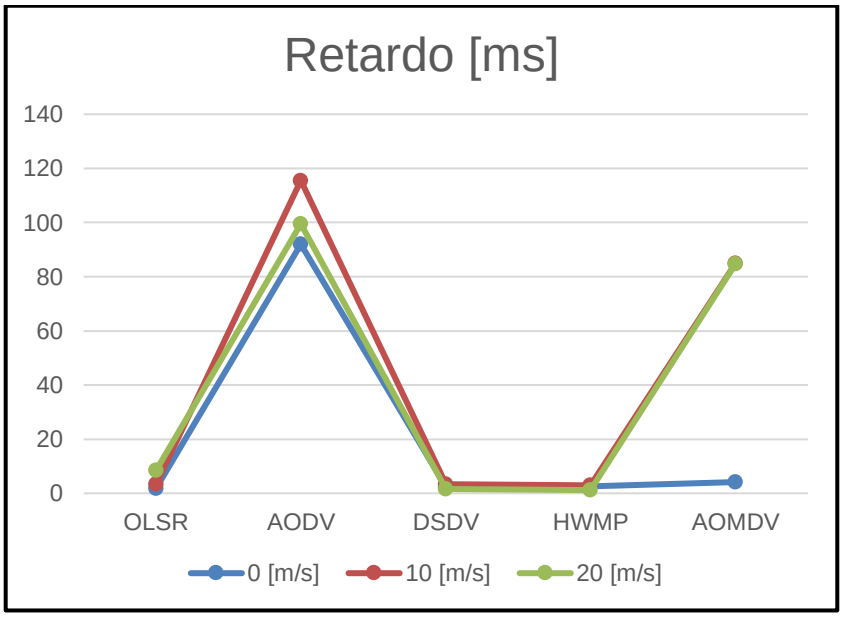

Figura 3.33. Gráfico comparativo del Retardo, Escenario 1 – Pruebas 1 y 2.

Se puede apreciar en la Figura 3.33, que al aumentar la velocidad de movimiento existe una variación mínima en este parámetro, a excepción del protocolo AOMDV en donde el retardo aumenta considerablemente a velocidades diferentes de 0.

En este apartado, de los resultados obtenidos, se puede concluir que el mejor protocolo para el despliegue de una red Fanet con nodos en vuelo en posición aleatoria con velocidad máxima variable son los protocolos OLSR y DSDV, ya que presentan variaciones mínimas en el rendimiento después de la variación de la velocidad, con un valor menor de PDR al aumentar la velocidad de movimiento.

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- El simulador NS-3 es altamente usado en el campo de la investigación ya que permite la simulación de múltiples redes y escenarios, además, al ser de código abierto, da pie a que la comunidad de investigadores pueda desarrollar nuevos módulos, sin embargo, al ser actualizado constantemente se requiere que estos desarrollos independientes se adapten a las nuevas versiones conforme se van liberando, en caso de que un módulo desarrollado independientemente no sea actualizado puede dejar de funcionar tras una actualización del simulador.
- Como se explicó en la sección 2.7.2 parte del código permite la modificación de la potencia de transmisión, lo que podría derivar en futuros desarrollos para el análisis de consumo de energía en redes FANET, además de analizar el comportamiento de dichos protocolos considerando el consumo energético, lo que podría dar un escenario más cercano a una implementación real.
- El uso de *helpers* en NS-3 ayuda en gran medida a la implementación de redes en donde se tenga un número considerable de nodos, ya que no es necesario implementar de manera individual cada línea de código para generar interfaces o tráfico en cada nodo.
- El despliegue de la red FANET utilizando el modelo de movilidad Gauss Markov disponible en el simulador permite obtener resultados más aproximados a una implementación real, debido a que utiliza un modelo en tres dimensiones, a diferencia de otros modelos de movilidad que solo implementan el movimiento en dos dimensiones, por otro lado, el uso de este modelo puede representar bastante carga de procesamiento en el momento de generar resultados, por lo que algunas simulaciones pueden llegar a demorar hasta algunas horas en mostrar resultados, esto está directamente relacionado a los recursos que presente la máquina en la que se trabaje.
- La implementación del protocolo AOMDV se realizó tomando como base el protocolo AODV implementado en NS-3, al aplicar las múltiples rutas se pudo apreciar que el rendimiento del protocolo aumenta considerablemente, como se esperaría en el protocolo AOMDV, sin embargo, al tener alta movilidad se tienen resultados bajos en rendimiento de este protocolo.
- Los valores bajos en rendimiento del protocolo AODV y AOMDV están relacionados a que manejan tráfico por medio de broadcast tanto para el descubrimiento de ruta como es el caso del protocolo AODV, como para mantener una ruta en el protocolo

AOMDV, esto puede presentar problemas al momento de poder extraer la información con el módulo Flow Monitor, además, esta limitación se encuentra definida en el archivo *ipv4-flow-classifier.cc* línea 108, mismo que puede ser encontrado en el repositorio base de NS-3.

- El agregar movilidad a la red, genera cambios significativos en el rendimiento de los protocolos, esto se aprecia de mejor manera con el protocolo HWMP que parte de una red tipo árbol, el tener movilidad puede generar desconexiones entre los dispositivos, lo cual puede asociarse inclusive a choques entre los mismos, como consecuencia, los dispositivos en línea entran en una nueva etapa de descubrimiento de nodos vecinos, destinando tiempo a esta tarea y no a la transferencia de datos.
- Al modificar el número de UAVs de 25 a 50, pero sin modificar los límites del escenario da como resultado una variación mínima en el rendimiento de los protocolos, esto demuestra que los protocolos se pueden implementar en entornos donde exista un número considerable de nodos, sin embargo, en implementaciones reales esto puede presentar problemas de choques entre dispositivos si no se maneja el espacio físico adecuadamente acorde a la cantidad de nodos desplegados.
- De los de las simulaciones realizadas de todos los escenarios en la sección 3, se puede destacar que el protocolo OLSR y DSDV son los que mejor se adaptan al cambio en todos los entornos, tanto de modificación del número de UAVs, como de la modificación de la velocidad de movimiento, esto puede usarse como referencia para la implementación real de redes FANET ya que la alta movilidad es una característica crítica de estas redes.
- Acorde a los resultados obtenidos de las simulaciones realizadas en el escenario 1, se podría considerar la implementación del protocolo HWMP en situaciones en donde el mayor tiempo los nodos se encuentren en vuelo en posición estática, además de requerir una mayor cantidad de throughput.
- La implementación de estas redes, en especial el uso de UAVs podría aumentar con el paso de los años y de igual manera el desarrollo de la tecnología a bordo en los UAVs.
- El protocolo DSR al no poder ser analizado utilizando Flow Monitor no se pueden tener datos del rendimiento del protocolo en ninguno de los escenarios por lo que podría considerarse para futuros trabajos el analizar opciones para obtener información de este protocolo y así realizar el análisis del despliegue en redes de comunicación.

4.2 RECOMENDACIONES

- La selección del tipo de hardware a utilizar para el despliegue de NS-3 es crítico al momento de tener resultados de las simulaciones, ya que algunos módulos pueden llegar a requerir bastante capacidad de procesamiento.
- Es necesario tener en cuenta que el uso de NS-3 puede requerir cierto nivel de conocimiento en lenguaje C++, lo cual puede dificultar las implementaciones a realizar, además que, los manuales de uso son extensos y algunos presentan información limitada, por lo que se recomienda tomar en cuenta que el tiempo de aprendizaje para utilizar el simulador puede llegar a ser bastante amplio.
- Para el uso de NS-3 existen foros y comunidades en internet en donde se pueden hacer consultas acerca de todos los tópicos de este simulador.
- Previo a una implementación, es necesario tener en consideración todas las posibles limitaciones que puedan a llegar a tener los módulos, ya que podría afectar a los resultados esperados, debido a la falta de compatibilidad con ciertas funciones o desarrollos aún no realizados, por otro lado, la implementación de módulos que no sean parte del instalador base de NS-3 podría haber dejado de ser compatible con las versiones a utilizar del simulador.
- Para el presente trabajo se utilizaron módulos desarrollados e implementados en la versión usada del simulador, es necesario mencionar que para el caso de la implementación del protocolo AOMDV se utilizó otra librería de desarrollo independiente, debido a que al momento de realizado este trabajo no existe un módulo de implementación de dicho protocolo en NS-3, por ello, este trabajo se puede tomar como punto de partida para el desarrollo e implementación de este protocolo como parte de versiones futuras del simulador.
- Este trabajo podría ser considerado como punto de partida para el desarrollo de nuevas investigaciones a futuro en el campo de las redes FANET o la implementación de protocolos de enrutamiento.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] G. J. Vachtsevanos y K. P. Valavanis, "Military and Civilian Unmanned Aircraft," *Handbook of Unmanned Aerial Vehicles,* pp. 93-103, 2015.

[2] I. Maza, A. Ollero, E. Casado y D. Scarlatti, "Classification ofMulti-UAV Architectures," *Handbook of Unmanned Aerial Vehicles,* pp. 953-975, 2015.

[3] A. Chriki, H. Touati, H. Snoussi y F. Kamoun, "FANET: Communication, Mobility models and Security issues," *Computer Networks,* vol. 163, 2019.

[4] H. Yang y Z. Liu, "An optimization routing protocol for FANETs," *EURASIP Journal on Wireless Communications and Networking,* vol. 120, 2019.

[5] I. Bekmezci, O. K. Sahingoz y S. Temel, "Flying Ad-Hoc Networks (FANETs): A survey," *Ad Hoc Networks,* vol. 11, nº 3, pp. 1254-1270, 2013.

[6] P. Ekka, "A Review Paper on Unmanned Aerial Vehicle (U.A.V.)," *International Journal of Engineering Research & Technology (IJERT),* vol. 5, nº 23, 2017.

[7] "Fixed Wing UAV Manufacturers Overview," [En línea]. Available: https://www.unmannedsystemstechnology.com/expo/fixed-wing-uav-manufacturers/. [Último acceso: 2022].

[8] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam y M. Debbah, "A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems," *IEEE Communications Surveys & Tutorials,* vol. 21, nº 3, pp. 2334-2360, 2019.

[9] A. Bujari, C. E. Palazzi y D. Ronzani, "FANET Application Scenarios and Mobility Models," *DroNet '17: Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications,* pp. 43-46, 2017.

[10] U. Hakan, Y. Ugur y B. Cuneyt, "A review on applications of rotary-wing unmanned aerial vehicle charging stations," *International Journal of Advanced Robotic Systems,* 2021.

[11] L. Reade, "Bombs over Venice," *History Today,* vol. 8, nº 6, 1958.

[12] J. G. Leishman, "The Bréguet-Richet Quad-Rotor Helicopter of 1907," *Vertiflite,* vol. 47, nº 3, pp. 58-60, 2001.

[13] K. Vyas, "A Brief History of Drones: The Remote Controlled Unmanned Aerial Vehicles (UAVs)," 29 Junio 2020. [En línea]. Available: https://interestingengineering.com/a-brief-history-of-drones-the-remote-controlled-unmanned-aerial-vehicles-uavs.

[14] National Museum of the United States Air Force, "Kettering Aerial Torpedo "Bug"," [En línea]. Available: https://www.nationalmuseum.af.mil/Visit/Museum-Exhibits/Fact-Sheets/Display/Article/198095/kettering-aerial-torpedo-bug/.

[15] L. Krock, “DH.82B Queen Bee (UK),” Noviembre 2002. [En línea]. Available: https://www.pbs.org/wgbh/nova/spiesfly/uavs_05.html.

[16] F. Haynes, Junio 2002. [En línea]. Available: https://www.navyhistory.org.au/queen-bee-radio-controlled-target-aircraft-of-the-1930s/.

[17] Federal Aviation Administration, *Unmanned Aircraft Operations in the National Airspace System,* Docket No. FAA-2006-25714.

[18] L. Dormehl, “The history of drones in 10 milestones,” 11 Septiembre 2018. [En línea]. Available: https://www.digitaltrends.com/cool-tech/history-of-drones/.

[19] P.-J. Bristeau, FrançoisCallou, D. Vissière y NicolasPetit, “The Navigation and Control technology inside the AR.Drone micro UAV,” *IFAC Proceedings Volumes,* vol. 44, nº 1, pp. 1477-1484, 2011.

[20] E. Oswald, “Here’s everything you need to know about Amazon’s drone delivery project, Prime Air,” 3 Mayo 2017. [En línea]. Available: https://www.digitaltrends.com/cool-tech/amazon-prime-air-delivery-drones-history-progress/.

[21] Amazon, “Amazon PrimeAir,” [En línea]. Available: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011.

[22] DJI, “PHANTOM 4,” [En línea]. Available: https://www.dji.com/phantom-4.

[23] D. Prindle, “DJI Phantom 4 Pro review,” 13 Marzo 2018. [En línea]. Available: https://www.digitaltrends.com/drone-reviews/dji-phantom-4-pro-review/. [Último acceso: 2022].

[24] L. Silva, “Ventajas y desventajas de los drones,” [En línea]. Available: https://www.dronesweb.net/ventajas-desventajas-los-drones/. [Último acceso: 2022].

[25] Moldtrans, “Los drones y la logística,” 29 Julio 2020. [En línea]. Available: https://www.moldtrans.com/los-drones-y-la-logistica/. [Último acceso: 2022].

[26] Embention, “Integración de sensores y dispositivos avanzados en UAVs,” 12 Agosto 2016. [En línea]. Available: https://www.embention.com/es/news/sensores-y-dispositivos-en-uavs/. [Último acceso: 2022].

[27] Colombia.com, “Drones: Estas son sus ventajas y desventajas,” 21 Noviembre 2017. [En línea]. Available: https://www.colombia.com/tecnologia/aplicaciones/sdi/168671/drones-estas-son-sus-ventajas-y-desventajas.

[28] Federal Aviation Administration, “Small Unmanned Aircraft Systems (UAS) Regulations (Part 107),” 06 Octubre 2020. [En línea]. Available: https://www.faa.gov/newsroom/small-unmanned-aircraft-systems-uas-regulations-part-107. [Último acceso: 2022].

[29] ComeDroneWithMe, “How High can a Drone Fly? Laws, Safety and Flying Tips.,” [En línea]. Available: https://comedronewithme.com/how-high-can-a-drone-fly/. [Último acceso: 2022].

[30] C. G. Montenegro y J. P. Aspas, “REDES AD-HOC: EL PRÓXIMO RETO,” *Buran,* nº 21, pp. 30-37, 2004.

[31] M. Frodigh, P. Johansson y P. Larsson, “Wireless ad hoc networking - The art of networking without a network,” *Ericsson Review,* nº 4, 2000.

[32] S. Mirza y S. Z. Bakshi, “Introduction to MANET,” *International Research Journal of Engineering and Technology,* vol. 5, nº 1, pp. 17-20, 2018.

[33] M. Kumar y R. Mishra, “An Overview of MANET: History, Challenges and Applications,” *Indian Journal of Computer Science and Engineering ,* vol. 3, nº 1, pp. 121-125, 2012.

[34] C. Campolo, A. Molinaro y R. Scopigno, Vehicular ad hoc Networks, Springer International, 2015.

[35] J. A. A. Silva, M. A. A. Pérez y R. A. Gonzalez, “Redes VANET Vehicular Ad-Hoc Networks, la conectividad de los autos. Primera parte.,” *Revista Visión Politécnica,* 2018.

[36] V. Jindal y P. Bedi, “Vehicular Ad-Hoc Networks- Introduction, Standards, Routing Protocols and Challenges,” *International Journal of Computer Science Issues,* vol. 13, nº 2, pp. 44-55, 2016.

[37] K. Singh y A. K. Verma, “Flying Adhoc Networks: Concept and Challenges,” de *Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics*, vol. 5, IGI Global, 2019.

[38] I. Jawhar, N. Mohamed, J. Al-Jaroodi, D. P. Agrawal y S. Zhang, “Communication and Networking of UAV-Based Systems: Classification and Associated Architectures,” *Journal of Network and Computer Applications,* vol. 84, pp. 93-108, 2017.

[39] D. S. Lakew, U. Sa’ad, N.-N. Dao, W. Na y S. Cho, “Routing in Flying Ad Hoc Networks: A Comprehensive Survey,” *IEEE Communications Surveys & Tutorials,* vol. 22, nº 2, pp. 1071-1120, 2020.

[40] M. A. Khan, I. M. Qureshi y F. Khanzada, “A Hybrid Communication Scheme for Efficient and Low-Cost Deployment of Future Flying Ad-Hoc Network (FANET),” *Drones,* vol. 3, nº 1, p. 16, 2019.

[41] G. R. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. P. Costa y B. Walke, “The IEEE 802.11 universe,” *IEEE Communications Magazine,* vol. 48, nº 1, pp. 62-70, 2010.

[42] Core Specification Working Group, “Bluetooth Core Specification v5.2,” 2019.

[43] S. C. Ergen, “ZigBee/IEEE 802.15.4 Summary,” 2004.

[44] K. Zaka, N. Ahmed, M. Ibrar-ul-Haq, M. I. Anis y B. Faria, “Performance analysis and throughput optimization in IEEE 802.16 WiMax standard,” *First Asian Himalayas International Conference on Internet,* 2009.

[45] S. Banerji y R. S. Chowdhury, “Wi-Fi & WiMAX: A Comparative Study,” *Indian Journal of Engineering,* vol. 2, nº 5, 2013.

[46] R. Shahzadi, M. Ali, H. Z. Khan y M. Naeem, “UAV assisted 5G and beyond wireless networks: A survey,” *Journal of Network and Computer Applications,* vol. 189, 2021.

[47] C. V. Miranda, Redes Telemáticas, Madrid: Paraninfo, 2015.

[48] B. Y. V. Cortez, Análisis comparativo de los protocolos de enrutamiento dinámico de una red de transporte para la universidad de Guayaquil, Universidad de Guayaquil, 2016.

[49] E. E. M. Ruiz y G. D. M. Ruiz, Protocolos de enrutamiento RIP, OSPF y EIGRP, Universidad Tecnológica de Bolívar, 2008.

[50] J. P. C. Olago, Protocolos de enrutamiento de estado de enlace, Cisco Networking Academy, 2007.

[51] K. Singh y A. K. Verma, “Experimental Analysis of AODV, DSDV and OLSR Routing Protocol for Flying Adhoc Networks (FANETs),” *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT),* pp. 1-4, 2015.

[52] D. B. Johnson y D. A. Maltz, “Dynamic Source Routing in Ad Hoc Wireless Networks,” *Mobile Computing,* vol. 353, pp. 153-181, 1996.

[53] B. Mathur y A. Jain, “AOMDV Protocol: A Literature Review,” *International Journal of New Technology and Research (IJNTR),* vol. 4, nº 7, pp. 27-30, 2018.

[54] J. Zhou, H. Xu, Z. Qin, Y. Peng y C. Lei, “Ad Hoc On-Demand Multipath Distance Vector Routing Protocol Based on Node State,” *Communications and Network,* vol. 5, nº 3C, pp. 408-413, 2013.

[55] C. E. Perkins y P. Bhagwat, “Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers,” *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications,* pp. 234-244, 1994.

[56] M. Hashem, S. Barakat y M. A. Alla, “A tree routing protocol for cognitive radio network,” *Egyptian Informatics Journal,* vol. 18, nº 2, pp. 95-103, 2017.

[57] S. M. S. Bari, F. Anwar y M. H. Masud, “Performance Study of Hybrid Wireless Mesh Protocol (HWMP) for IEEE 802.11s WLAN Mesh Networks,” *2012 International Conference on Computer and Communication Engineering (ICCCE),* pp. 712-716, 2012.

[58] “NS-3 Installation,” 8 Marzo 2023. [En línea]. Available: https://www.nsnam.org/wiki/Installation. [Último acceso: 15 Marzo 2023].

[59] “NS-3 Network Simulator,” [En línea]. Available: https://www.nsnam.org/about/. [Último acceso: 2023 Marzo 15].

[60] A. S. Guerrero, “Simulación de eventos,” Centro Cultural Itaca S.C..

[61] “ns-3 Installation Guide,” 09 Abril 2023. [En línea]. Available: https://www.nsnam.org/docs/installation/html/quick-start.html#prerequisites. [Último acceso: 09 Abril 2023].

[62] “NS-3 HOWTO configure Eclipse with ns-3,” [En línea]. Available: https://www.nsnam.org/wiki/HOWTO_configure_Eclipse_with_ns-3. [Último acceso: 21 Marzo 2023].

[63] “Mercurial Eclipse,” [En línea]. Available: https://wiki.mercurial-scm.org/MercurialEclipse. [Último acceso: 1 Abril 2023].

[64] R. S. E. W. Ulf Lamping, “Wireshark User's Guide for Wireshark 1.9,” *NS Computer Software and Services P/L,* 2012.

[65] “GeeksforGeeks,” 8 Agosto 2019. [En línea]. Available: https://www.geeksforgeeks.org/network-simulator-3/. [Último acceso: 15 Marzo 2023].

[66] “NS-3 NetAnim,” 12 Marzo 2023. [En línea]. Available: https://www.nsnam.org/wiki/NetAnim. [Último acceso: 15 Marzo 2023].

[67] “NS-3 Conceptual Overview,” 2 Noviembre 2022. [En línea]. Available: https://www.nsnam.org/docs/release/3.37/tutorial/html/conceptual-overview.html. [Último acceso: 15 Marzo 2023].

[68] “NS-3 Organization,” 4 Octubre 2016. [En línea]. Available: https://www.nsnam.org/docs/release/3.26/manual/html/organization.html. [Último acceso: 15 Marzo 2023].

[69] ns-3 project, “ns-3 Model Library,” 14 Julio 2021. [En línea]. Available: https://www.nsnam.org/docs/release/3.34/models/ns-3-model-library.pdf. [Último acceso: 20 Marzo 2023].

[70] “NS-3 Random Variables,” 4 Octubre 2016. [En línea]. Available: https://www.nsnam.org/docs/release/3.26/manual/html/random-variables.html. [Último acceso: 15 Marzo 2023].

[71] “NS-3 Hash Functions,” 4 Octubre 2016. [En línea]. Available: https://www.nsnam.org/docs/release/3.26/manual/html/hash-functions.html. [Último acceso: 15 Marzo 2023].

[72] “NS-3 Events and Simulator,” 4 Octubre 2016. [En línea]. Available: https://www.nsnam.org/docs/release/3.26/manual/html/events.html. [Último acceso: 15 Marzo 2023].

[73] “NS-3 Callbacks,” 4 Octubre 2016. [En línea]. Available: https://www.nsnam.org/docs/release/3.26/manual/html/callbacks.html. [Último acceso: 15 Marzo 2023].

[74] “NS-3 Object Model,” 4 Octubre 2023. [En línea]. Available: https://www.nsnam.org/docs/release/3.26/manual/html/object-model.html. [Último acceso: 15 Marzo 2023].

[75] “NS-3 Logging,” 4 Octubre 2023. [En línea]. Available: https://www.nsnam.org/docs/release/3.26/manual/html/logging.html. [Último acceso: 15 Marzo 2023].

[76] “NS-3 Tracing,” 04 Octubre 2016. [En línea]. Available: https://www.nsnam.org/docs/release/3.26/manual/html/tracing.html. [Último acceso: 15 Marzo 2023].

[77] “NS-3 Mobility,” [En línea]. Available: https://www.nsnam.org/docs/release/3.28/doxygen/group__mobility.html. [Último acceso: 20 Marzo 2023].

[78] D. Broyles, A. Jabbar y J. P. Sterbenz, “Design and analysis of a 3–D gauss-markov mobility model for highly-dynamic airborne networks,” 2010.

[79] S. K. Maakar, Y. Singh y R. Singh, “An Enhanced Gauss-Markov Mobility Model for Simulation of FANET in 3-D Environment,” *Journal of Advanced Research in Dynamical and Control Systems,* vol. 10, pp. 2004-2013, 2018.

[80] “Wi-Fi Multimedia Extension (IEEE 802.11 EDCA),” [En línea]. Available: https://www.tetcos.com/pdf/v13/Experiments/Wi-Fi-WME-802-11e-QoS-EDCA.pdf. [Último acceso: 28 Marzo 2023].

[81] “Optimized Link State Routing Protocol (OLSR),” Octubre 2003. [En línea]. Available: https://datatracker.ietf.org/doc/html/rfc3626.html. [Último acceso: 28 Marzo 2023].

[82] “Ad hoc On-Demand Distance Vector (AODV) Routing,” Julio 2003. [En línea]. Available: https://datatracker.ietf.org/doc/html/rfc3561.html. [Último acceso: 28 Marzo 2023].

[83] “The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4,” Febrero 2007. [En línea]. Available: https://datatracker.ietf.org/doc/html/rfc4728.html. [Último acceso: 28 Marzo 2023].

[84] “ns-3 Applications,” [En línea]. Available: https://www.nsnam.org/doxygen/d9/dc9/group__applications.html. [Último acceso: 30 Marzo 2023].

[85] I. Bekmezci, I. Sen y E. Erkalkan, “Flying Ad Hoc Networks (FANET) Test Bed Implementation,” *2015 7th International Conference on Recent Advances in Space Technologies (RAST),* pp. 665-668, 2015.

[86] “DsrMainHelper Class Reference,” 19 Abril 2014. [En línea]. Available: https://www.nsnam.org/docs/release/3.19/doxygen/classns3_1_1_dsr_main_helper.html. [Último acceso: 25 Marzo 2023].

[87] F. M. Choudhury, “Github,” [En línea]. Available: https://github.com/tonmoy71/ns3-scratch/blob/master/myapp.h. [Último acceso: 10 Junio 2023].

[88] “ns-3 Bug 1482,” 18 Abril 2017. [En línea]. Available: https://www.nsnam.org/bugzilla/show_bug.cgi?id=1482. [Último acceso: 10 Junio 2023].

[89] “ns-3 DSR Routing,” [En línea]. Available: https://www.nsnam.org/docs/models/html/dsr.html. [Último acceso: 10 Junio 2023].

[90] M. F. Khan, E. A. Felemban, S. Qaisar y S. Ali, “Performance Analysis on Packet Delivery Ratio and End-to-End Delay of Different Network Topologies in Wireless Sensor

Networks (WSNs),” *IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks,* pp. 324-329, 2013.

[91] “Bug 1844 - FlowMonitor fails to capture DSR,” 08 Enero 2015. [En línea]. Available: https://www.nsnam.org/bugzilla/show_bug.cgi?id=1844. [Último acceso: 20 Junio 2023].

ANEXOS

Los scripts generados y utilizados en la ejecución de este trabajo se encuentran cargados en GitHub el cual se puede acceder mediante el siguiente enlace:

https://github.com/wellgmp/Fanet-NS3

ORDEN DE EMPASTADO