UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA
DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Informática de Sistemas y Computadores

Solución para mejorar la resiliencia de enjambres de drones de gran dimensión mediante comunicaciones inalámbricas multi-salto

Trabajo Fin de Máster

Máster Universitario en Ingeniería de Computadores y Redes

AUTOR/A: Clérigues Ferrer, David

Tutor/a: Tavares de Araujo Cesariny Calafate, Carlos Miguel

Director/a Experimental: WUBBEN, JAMIE

CURSO ACADÉMICO: 2021/2022

Reconocimientos

En un primer lugar, me gustaría agradecer a mis tutores, Carlos Tavares Calatafe y Jamie Wubben, por depositar su voto de confianza en mí, por su magnifica compresión y, sobretodo, por haberme ayudado y guiado en los momentos más cruciales. La confianza y el esfuerzo que han depositado en mí en estos años me ha ayudado a desarrollarme como persona llevándome a completar distintas metas. Además, sé que con todo la experiencia obtenida sabré afrontar retos futuros. Además, quiero agradecer al Grupo de Investigación de Redes y Computadores (GRC) por haberme aceptado en su equipo durante estos meses. Gracias a ellos he podido trabajar en un clima agradable y familiar donde cada día he podido aprender cosas nuevas. Por otra parte, quiero agradecer a mis amigos Cristián Stratu y Srdjan Zec por haber estado ahí en todo momento para apoyarme, motivarme y aconsejarme frente a mis dudas. Por último, pero no menos importante, quiero agradecer a mis padres y mis abuelos por haberme permitido crecer como persona durante todos estos años y por haber estado ahí pase lo que pase.

David Clérigues Ferrer
Valencia, 13 de agosto de 2022

Abstract

Drone swarms are a booming IT sector that every day awakens more and more the interest of the most knowledgeable. Within this field there are hundreds of open researches that seek to cover the best possible solution in different sectors such as natural disasters, rescue of people, military equipment, etc... In this report we are looking for a multi-hop solution to improve the resilience of the swarms. For this purpose, the Ardusim simulator developed by the GRC group has been used. This simulator is intended to create a communications channel capable of communicating a swarm with hidden nodes. Because of this, a general network construction has been implemented using graphs in order to allow communication between two drones out of range. This communication makes use of different filtering and message accumulation mechanisms or, better said, piggybacking to speed up and improve the performance of the channel. This report covers these sections, concluding with an experimentation, on a protocol of the simulator, where the correct functioning of the network is corroborated, both in its construction and in its communication.

Palabras Clave: Simulation, Ardusim, Protocol, Piggybacking, Graph, Swarm, Drones, Hidden Nodes

Resumen

Los enjambres de drones son un sector informático en auge que cada día despierta más y más el interés de los más entendidos. Dentro de este ámbito hay cientos de investigaciones abiertas que buscan abarcar la mejor solución posible en distintos sectores como el de las catástrofes naturales, rescate de personas, equipos militarizados, etc... En este memoria se busca abarcar una solución multisalto que permite mejorar la resilencia de los enjambres. Para ello, se ha hecho uso del simulador Ardusim desarrollado por el grupo GRC. Con este simulador se pretende crear un canal de comunicaciones capaz de comunicar un enjambre con nodos ocultos. Debido a esto, se ha implementado una construcción de la red general mediante el uso de grafos para poder permitir la comunicación entre dos drones fuera de rango. Dicha comunicación hace uso de distintos mecanismos de filtrado y de acumulación de mensajes o, mejor dicho, piggybacking para agilizar y mejorar las prestaciones del canal. Esta memoria abarca dichos apartados concluyendo con una experimentación, sobre un protocolo propio del simulador, donde se corrobora el correcto funcionamiento de la red tanto su construcción como su comunicación.
Palabras Clave: Simulación, Ardusim, Protocolo, Piggybacking, Grafo, Enjambre, Drones, Nodos ocultos

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

El ser humano es curioso por naturaleza. El paso del tiempo ha demostrado que nuestra especie ha sido capaz de evolucionar y adaptarse a nuevas eras. Si nos fijamos, hace unas décadas los computadores eran dispositivos gigantescos destinados a centros de calculo y operaciones militares. Hoy en día la gran mayoría de ciudadanos disponen de uno de estos en formato ultra-compacto cuando hace un par de años era inimaginable.

En la actualidad, expertos del sector tecnológico se enfrentan en su día a día a distintos desafíos para hacer la vida más cómoda a nuestra sociedad. Sin embargo, aún hay mucho por descubrir y por hacer.

Hay un sector clave, con muchos horizontes abiertos, que nos va a permitir aumentar nuestra comodidad en un futuro. Dicha rama emergente es conocida por trabajar con drones, conocidos como *Unmanned aerial vehicles* (UAVs). Hoy en día, estos dispositivos son completamente polivalentes ya que su uso es aplicable a un gran abanico de entornos como el uso militar, la monitorización de áreas críticas, misiones de búsqueda y rescate, entrega de paquetes, análisis de superficies afectadas por calamidades, etc.

Uno de los problemas principales de realizar nuevos desarrollos en este área es el coste para poder evaluar y poner en práctica las distintas hipótesis. Para ello se han desarrollado simuladores que permiten crear entornos virtuales que se asemejen al mundo real.

En del Departamento de Informática de Sistemas y Computadores (DISCA), situado en la Universidad Politécnica de Valencia, se desarrolló una aplicación capaz de abarcar todo este ámbito gracias al arduo trabajo del estudiante Francisco Fabra en su tesis doctoral, y a la supervisión del tutor Carlos T. Calafate. Dicho simulador, llamado ArduSim [1], es capaz de ejecutar distintas pruebas de vuelo virtuales y físicas mediante el empleo de distintos protocolos, donde cada uno se encarga de abarcar un ámbito operacional distinto. El enfoque principal de esta memoria se basa en la extensión

de un protocolo ya implementado, el protocolo MUSCOP, capaz de gestionar un enjambre de drones que tenga que seguir una misión planificada.

1.1. Motivación

Innovación y progreso son dos palabras que van ligadas de cara a obtener un futuro mejor. El campo de trabajo de los drones cada día busca innovar y buscar soluciones a problemas reales, aunque siendo un sector tan heterogéneo hay muchas ramas por cubrir. Los enjambres constituyen uno de los focos principales de esta memoria ya que son capaces, mediante su uso, de satisfacer un sinfín de misiones desde la monitorización de ciertas áreas hasta la búsqueda y el rescate de personas.

Los enjambres son un conjunto de UAVs que deben estar coordinados en todo momento. Para ello se establece un rango de comunicación mínimo que permita a los artilugios comunicarse. No obstante, ¿qué sucede si uno de ellos sale de dicho rango? Hoy en día, con las propuestas existentes, el UAV simplemente se sale de la formación. Por lo tanto, con esta memoria se busca alcanzar una solución que permita suplir dicha carencia, es decir, se busca tener nodos más allá del rango de cobertura que puedan comunicarse con el nodo central, permitiendo así tener una mayor flexibilidad y adaptabilidad ante las adversidades inesperadas.

1.2. Objetivos

En este trabajo se busca desarrollar una solución eficiente que permita a los distintos nodos, que están fuera del rango de comunicaciones, comunicarse con el nodo central. Para ello, se busca desarrollar un nuevo protocolo llamado Multihop que sea capaz de ampliar las funcionalidades de comunicación actuales. De esta forma, misiones con un enjambre que incluyan nodos más allá del rango de comunicaciones del nodo central podrán igualmente llevarse a cabo de forma óptima.

El desarrollo del protocolo permite hacer uso de una capa de comunicaciones que puede ser extraída y adoptada por otros protocolos. Por tanto, el protocolo Multihop queda relevado a ser una librería genérica de comunicaciones donde el verdadero foco de la investigación y aplicación queda relegado en lograr comunicaciones multi-salto para entornos inalámbricos específicos, como el caso de enjambres de UAVs. Como objetivos parciales a lograr destaca:

- Análisis y comprensión del funcionamiento actual de MUSCOP sobre ArduSim.
- Extracción y modificación de los conceptos multi-salto aplicados sobre MUSCOP.
- Optimización de los pasos implementados sobre el protocolo MUSCOP Extendido.
- Aplicación de la solución sobre un canal de comunicaciones versátil que permita aplicar el multi-salto sobre distintos protocolos.
- Observación y realización de distintos experimentos para corroborar el correcto funcionamiento del proyecto junto con un análisis e interpretación de los resultados.

1.3. Estructura de la Tesis

La estructura empleada para el desarrollo de la memoria sigue un modelo detallado donde se cubren los aspectos más relevantes con un total de ocho capítulos.

- El primer capítulo tiene como enfoque mostrar los aspectos generales de la memoria, siendo estos la introducción del contexto, las motivaciones que me llevan a trabajar en este área, los objetivos a desarrollar, y la estructura de la propia memoria.
- A continuación, el capítulo 2 contiene información sobre el estado del arte repasando así los trabajos más ilustres de la actualidad
- Seguidamente, el capitulo 3 tiene como objetivo principal establecer las bases la simulación de drones sobre el propio simulador.
- Por otra parte, en el capitulo 4 se muestra parte de la solución alcanzada. En este caso, se expone y explica el desarrollo de los grafos y como esto afecta a la nueva capa de comunicaciones
- El capítulo 5 se centra en concretar como funciona el envío y la recepción de mensajes sobre la nueva capa de comunicaciones una vez la red ha sido construida.
- El capítulo 6 explica de forma detallada como funciona MUSCOP en su versión original y como ha sido aplicada la nueva capa de comunicaciones sobre dicho protocolo.

- El capítulo 7 se centra en la experimentación; en él se validan las distintas pruebas realizadas y se evalúa el comportamiento obtenido, llegando así a corroborar el funcionamiento de la idea planteada previamente.
- Por último, la memoria cierra con una conclusión sobre todos los conceptos más importantes extraídos al desarrollar la solución de este enlace multisalto junto con unas guias para los trabajos futuros.

Capítulo 2

Trabajos Relacionados

En este capítulo se centra en los enjambres de drones para comprobar el estado del arte actual respecto al trabajo desarrollado. Para ello, por una parte se hará énfasis en aquellos trabajos relacionados con la coordinación de enjambres, y por otra parte se estudiarán aquellos trabajos que analicen métodos eficientes para descubrimiento de topología y encaminamiento de mensajes en un entorno multi-salto.

2.1. Coordinación de Enjambres

En un primer lugar, la coordinación de los enjambres es crucial para llevar a cabo distintas funcionalidades. Hoy en día existen una gran variedad de métodos para lograr dichas prestaciones, desde hacer uso de misiones previamente planificadas, aplicar patrones de consenso, mantener comunicaciones maestro-esclavo, etc.

Liguo Weng et al. [2] han desarrollado un método de coordinación descentralizado para UAVs de combate basado en el funcionamiento del sistema inmunológico humano. Según se expone en su trabajo, cada dron cuenta con tres estados diferentes, donde en cada uno de estos se envía una señal indicando qué se está haciendo. Inicialmente todo nodo empieza patrullando. Una vez se detecta una amenaza se cambia de estado y se emite una señal de advertencia (más prioritaria) para notificar a los nodos adyacentes. Cuando se recibe dicha señal se prioriza la detección frente a la patrulla; por lo tanto, se procesa y se cambia a un nuevo estado donde se procede a movilizarse hacía la posición donde se ha detectado la amenaza. Los experimentos obtenidos son extremadamente positivos para la escalabilidad pero, por otra parte, se perjudica la adaptabilidad y la robustez, provocando que este experimento no llegue a ser tan práctico y factible fuera de la simulación.

Por otra parte, Francisco Fabra et al. [3] han propuesto MUSCOP, una solución que permite coordinar los drones de un enjambre para llevar a cabo misiones planificadas independientemente de su formación. Para ello, durante la ejecución de la misión se mantiene la sincronización entre los drones esclavos y el maestro. Durante el transcurso de la misión se mantiene un control del enjambre de forma dinámica, actualizando el enjambre ante cualquier pérdida. La solución planteada demuestra una gran escalabilidad y resiliencia ante la pérdida de paquetes en el canal de comunicaciones siendo capaz de mantener la cohesión del enjambre. Por otra parte, la complejidad de la misión es el factor principal responsable por los retardos, perjudicando a las prestaciones.

Carlos Sampedro et al. [4] han centrado su investigación en desarrollar una arquitectura autónoma, robusta y flexible capaz de afrontar misiones de alto nivel bajo circunstancias irregulares, como pueden ser escenarios donde los UAV no disponen de una señal GPS para coordinarse. Los resultados obtenidos durante la simulación demuestran la escalabilidad y flexibilidad de la arquitectura de planificación de misión propuesta para un número variable de UAVs y diferentes tipos de escenarios, con un número variable de obstáculos.

Por último, Liang Hong et al. [5] han propuesto una solución centrada en la optimización del encaminamiento de mensajes según la topología empleada para mantener una cohesión dentro del enjambre. Principalmente, los drones tienen que tener un ID único, mantenerse en un rango de comunicaciones seguro, trabajar en modo full-duplex, y moverse en dos dimensiones. Mediante el intercambio de mensajes HELLO (un tipo de mensaje único diseñado para esta solución) se consigue reconfigurar la formación de forma dinámica ante una adversidad, ya sea debido a un obstáculo, o por problemas medioambientales. Según se ha concluido, mediante la experimentación, emplear dicha solución aumenta considerablemente las prestaciones de la red.

2.2. Aproximaciones multisalto para enjambres

Gran parte del estado del arte muestra soluciones donde los enjambres trabajan en conjunto dentro de un mismo rango de comunicaciones. Uno de los mayores problemas existentes hoy en día es la falta de soluciones que permitan trabajar con drones fuera del rango de comunicaciones del nodo coordinador.

Jiansong Miao et al. [6] se adentran en un sector un poco más especifico, proponiendo una solución capaz de obtener una mayor eficiencia energética

para lograr comunicaciones seguras en un escenario multi-salto. Se plantea un escenario donde dos usuarios (origen/destino) buscan comunicarse dentro de una distancia considerable. Para ello, se ha hecho uso de un tipo de UAV especial llamado RUAV que será el encargado de encaminar el mensaje confidencial que estos dos usuarios quieren intercambiar. Además, se hace uso de JUAV, un derivado de los UAV, donde su misión principal es confundir y molestar a los atacantes que pretendan espiar la comunicación entre usuarios. Según los resultados, este escenario, aplicando sus propios algoritmos para la comunicación y convergencia, ha demostrado aumentar la seguridad y privacidad de las comunicaciones, incluyendo una optimización energética que permite reducir el consumo total.

Por otra parte, Davide Caputo et al. [7] enfrentan las dificultades provocadas por las comunicaciones multi-salto, como pueden ser la escalabilidad, comunicación, adaptabilidad, etc. Para ello aplican el algoritmo GSO en su solución, el cual es un híbrido de los algoritmos GA y PSO. Mediante el uso de este comprueba que su aplicación consigue aumentar las prestaciones del encaminamiento de mensajes dentro de un enfoque más centralizado, logrando un ahorro de energía entre los drones y la base de operaciones. Su enfoque busca preservar la funcionalidad de la red y evitar la muerte súbdita de los drones; a pesar de los resultados, se remarca que se requieren ciertas mejoras para optimizar la técnica propuesta.

Para finalizar, Alexis Pospischil et al. [8] se centran en el estudio del enrutamiento multi-salto de datos telemétricos sobre los enjambres. Dentro del estudio, se analizan las prestaciones obtenidas al hacer uso de protocolos como OLSR, DSDV, AODV y Epidemic. Los mecanismos son comparados mediante una serie de pruebas donde se alternan los valores del ratio de paquetes entregados y sus sobrecargas. Con ello, se hace un enfoque sobre como se comportan las prestaciones según los valores esperados. Cada protocolo empleado es óptimo para ciertos escenarios dentro de la simulación, pero el mayor problema es que ninguno ha conseguido alcanzar las prestaciones deseadas, concluyendo así que procede buscar soluciones más eficientes en el ámbito de los enjambres de drones, lo cual es justamente el objetivo del presente trabajo.

Capítulo 3

Simulación de drones utilizando ArduSim

Dentro del sector de los UAV se suele trabajar, en una primera instancia, con simuladores, ya que operar directamente con los dispositivos físicos puede suponer una extensa pérdida de bienes debido al alto coste de estos, y a la elevada probabilidad de que ocurran accidentes. Además, las simulaciones permiten a los usuarios refinar las prestaciones obtenidas, llevando así al desarrollo de un producto más óptimo.

Cada simulador es capaz de llevar a cabo distintas operaciones. Por lo tanto, es importante utilizar el que más se adapte a la investigación en curso. Por otra parte, los investigadores suelen tender a desarrollar su propio simulador, ya que este suele ir estrictamente ligado al producto que se busca obtener.

Para el desarrollo de la idea planteada en este trabajo se ha hecho uso del simulador ArduSim. Dicha aplicación permite realizar distintos tipos de simulaciones en tiempo real, siendo una innovación dentro del sector. Además, las comunicaciones entre drones vienen establecidas por el uso de un canal inalámbrico virtual que emula las comunicaciones en entornos inalámbricos compartidos, incluyendo pérdidas y posibles colisiones, igualmente en tiempo real.

El simulador permite una amplia versatilidad a la hora de ejecutar distintas misiones donde el usuario es capaz de definir la misión a seguir, así como la cantidad de UAVs que van a intervenir en la misma. Además, al ser código abierto, permite a cada usuario diseñar sus propias soluciones como pueden ser protocolos, misiones, configuraciones, etc. La figura 3.1 muestra la interfaz principal del simulador.

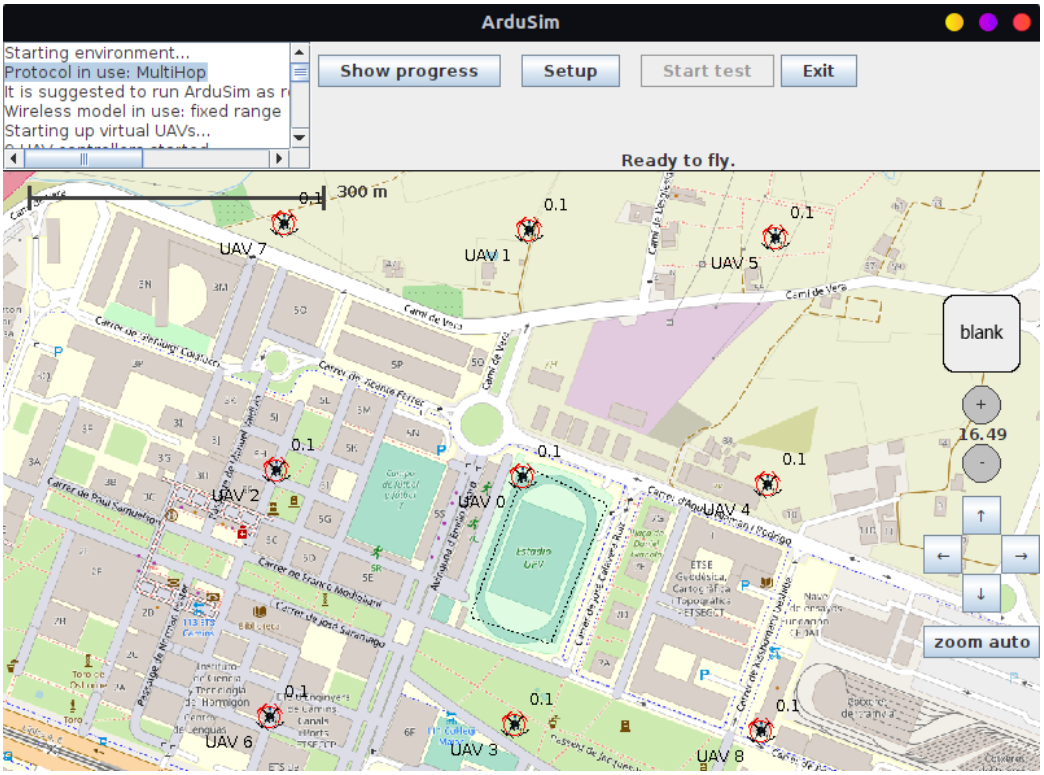


Figura 3.1: ArduSim.

3.1. Arquitectura

Con el paso del tiempo, la arquitectura de ArduSim ha ido siendo actualizada para optimizar los procesos y obtener mayores prestaciones. En la actualidad, el simulador dispone de varios mecanismos que permiten llevar a cabo distintas pruebas. Para ello, hace uso de envíos por difusión (Simulated Broadcast), y también contiene un detector de colisiones (UAV Collision detector) que permite a los distintos UAV coordinar su vuelo para no chocar entre ellos. Además, se dispone de una interfaz gráfica (GUIControl) para configurar los parámetros de los experimentos, y así poder visualizar el desarrollo de la misión aunque, por otra parte, se pueden ejecutar las misiones mediante línea de comandos para agilizar los experimentos. Los drones disponen de sus propios hilos para ejecutar las distintas ordenes adscritas siguiendo una lógica determinada.

Existe una gran variedad de mecanismos adicionales en cada capa aunque, al ser conceptos más complejos y abstractos, quedan fuera del alcance de esta memoria, salvo aquellos elementos relacionados con el canal físico y la

capa MAC. En la figura 3.2 se puede apreciar la arquitectura interna de la aplicación.

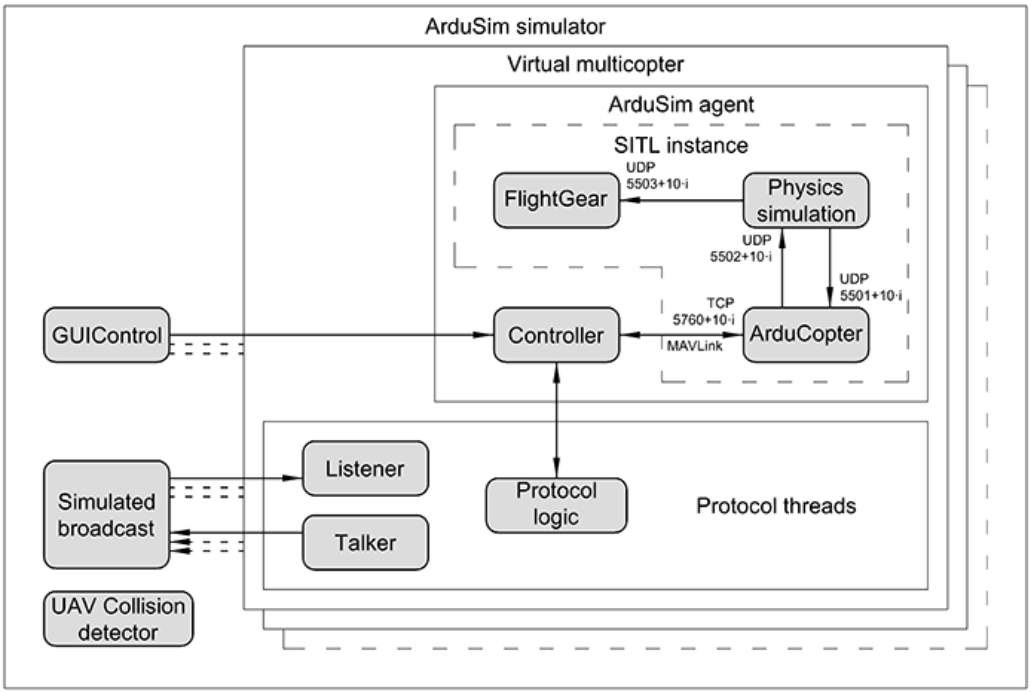


Figura 3.2: Arquitectura de ArduSim.

3.2. Canales de Comunicaciones

Ardusim hace uso de distintos canales de comunicación desde una perspectiva más básica, como el nivel físico, hasta un enfoque de alto nivel. A continuación, se explican de forma ascendente:

- En primer lugar, LowLevelCommLink es el encargado de crear los enlaces de comunicación asignando un identificador único a cada UAV junto con un puerto. El propio enlace permite recibir los mensajes dentro del puerto asignado. Además, es el encargado de hacer uso del mecanismo de envío de mensajes por broadcast. Por otra parte, el propio canal tiene un tamaño de datagrama máximo de 1472 bytes ya que se basa en el MTU de Ethernet, y asume que los protocolos IP y UDP son utilizados; por lo tanto, el MTU resultante, considerando las cabeceras de cada protocolo, viene dado por los valores 1500 (MTU) - 20 (IP) - 8 (UDP). Mediante el uso de dicho valor se establece el tamaño máximo del buffer en bytes.

- En segundo lugar, HighLevelCommLink es una capa de más alto nivel ya que, en vez de trabajar con los datos a un bajo nivel (bytes), se encarga de tratar con objetos JSON y, así, permitir una mayor compresión para el desarrollador. Dentro de este, se envían y se reciben los datos haciendo uso del LowLevelCommLink. A diferencia de su nivel inferior, que solo permitía hacer broadcast y recibir los mensajes en el puerto asignado, en este canal se permite hacer uso de reconocimientos, filtrado de mensajes mediante el uso de campos obligatorios, y envío de mensajes bloqueantes a la espera de recibir una confirmación de llegada.

- Por último, el canal multi-salto permite conocer la red, es decir, se hace un descubrimiento de los nodos adyacentes para posteriormente ir construyendo la red. Dentro de este canal se hace uso del HighLevelCommLink para el envío/recepción de datos, pero el procesamiento de estos es realizado por el propio canal multi-salto. No se busca entrar más en detalle ya que dicho canal es el foco principal de esta memoria y, por tanto, se explicará con detenimiento en capítulos posteriores.

3.3. Modelo de la capa física

El simulador hace uso del estándar 802.11a para mantener un modelo de comunicación realista gracias a la extracción de los datos experimentales mediante el uso de UAVs reales. Cabe destacar que los paquetes, dentro del modelo realista, serán propensos a fallar según la distancia empleada: a más distancia, más probabilidad de perder el mensaje. Este modelo fue desarrollado mediante el estudio de los canales de comunicaciones establecidos al interconectar dos multicópteros reales durante su vuelo midiendo, de forma precisa, la fluctuación en las prestaciones de la red producidas cuando usamos en canal de 5GHz. Dentro de la propia arquitectura se encuentra un módulo especifico que permite detectar las colisiones entre los paquetes de distintos drones; este complemento resulta extremadamente útil cuando se quiere añadir un comportamiento más realista a la simulación. Además, este detector de colisiones tiene un comportamiento muy similar al de CSMA/CA, algoritmo usado en WiFi.

Para la simulación existen otros tipos de modelos que permiten modelar comportamientos más cerca de un caso ideal, como podría ser la de rango fijo, la cual es una aproximación más simple al no permitir al usuario variar el nivel de pérdidas de paquetes según la distancia. Por otro lado, también se puede alejar los nodos la máxima distancia posible para comprobar los peores casos

presentes. Además, existe el modelo básico donde no hay ninguna limitación de rango, y en el que los paquetes siempre van a llegar sin pérdidas.

Posteriormente, se explicarán los experimentos realizados con los modelos de comunicaciones más adecuados para este proyecto.

3.4. Protocolos

Hoy en día, el simulador es capaz de ejecutar una amplia variedad de protocolos donde cada uno tiene un enfoque distinto. Para ello, en la siguiente lista se muestra cuales son los protocolos estables, y la funcionalidad que ofrecen:

- Durante el desarrollo del simulador se creó Mission, un protocolo base para desarrollar misiones preestablecidas. Con este protocolo se definió como debían ser las misiones, y como tenían que actuar los drones frente a estas.

- Antes de formalizar el detector de colisiones se realizaron distintas pruebas para observar las prestaciones obtenidas, y como se debían corregir los errores. Para ello, el protocolo MBCAP [9] ha sido desarrollado con la finalidad de perfeccionar dicho mecanismo. En este, durante la ejecución de la misión, se establece un orden de prioridad al detectar que dos drones van a interferir entre sí. Una vez establecida dicha prioridad, el dron menos prioritario da paso al más prioritario para que así no haya una colisión, y que ambos puedan continuar con el transcurso de su misión.

- Cuando hablamos de aterrizaje muchas veces tenemos que tener en cuenta que este no va a ser preciso. Debido a esto, Vision [10] ha sido desarrollado para mejorar el aterrizaje haciendo uso de una cámara junto con marcadores AruCo. El funcionamiento se encuentra explicado en más detalle en [10].

- Follow Me [11] es un protocolo un tanto distinto a los demás, ya que este requiere de un piloto real que sea capaz de manejar al nodo maestro. Dentro de este, los esclavos obedecen en todo momento las ordenes del nodo líder, y deben seguirle en todo momento.

- En algunas ocasiones pueden aparecer obstáculos que dificulten o impidan la ejecución de la misión. Para solventar dicha ocurrencia se desarrolló el protocolo Shake Up, capaz de modificar y reconfigurar

dinámicamente los UAVs del enjambre, permitiendo así la reducción de riesgos de colisión.

- A continuación, el protocolo MUSCOP [3] fue desarrollado para mantener la cohesión del enjambre durante el transcurso de la misión, siendo así tolerante a fallos. Durante el transcurso de la misión se mantiene un seguimiento de los nodos activos, y si alguno de ellos llega a fallar se descarta del enjambre, y se procede a la reconfiguración de este.

3.5. Formaciones

Seguidamente, al realizar una misión, el usuario debe seleccionar un tipo de formación para llevar a cabo el experimento. La siguiente lista expone los distintos tipos de formaciones disponibles:

- Para empezar, la formación Matricial sitúa al nodo maestro en el centro, donde este se verá rodeado por los esclavos de manera a formar cuadrados de tamaño creciente.

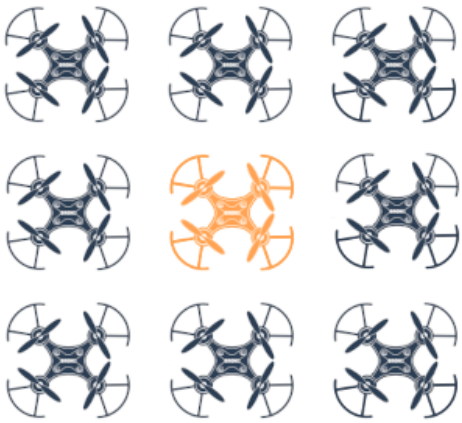


Figura 3.3: Formación Matricial.

- La formación menos compleja es la Linear. Los UAV se posicionan en una línea donde se van colocando de izquierda a derecha, estando el nodo maestro en el centro.

Figura 3.4: Formación Linear.

- Otra formación ofrecida por el simulador es la Circular. En esta, el dron maestro se sitúa en el centro, y los esclavos se van generando alrededor suyo en el sentido a las agujas del reloj, llegando a formar un circulo.

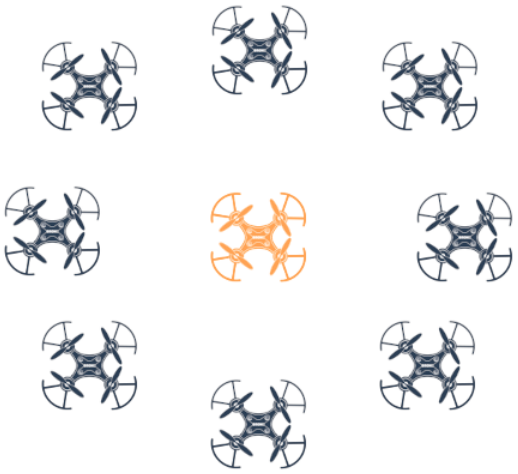


Figura 3.5: Formación Circular.

- La formación aleatoria ofrece un despliegue para realizar ciertos tipos de pruebas. El nodo maestro se centra en el mapa y, seguidamente, se crean distintos nodos alrededor en distintas posiciones completamente aleatorias.

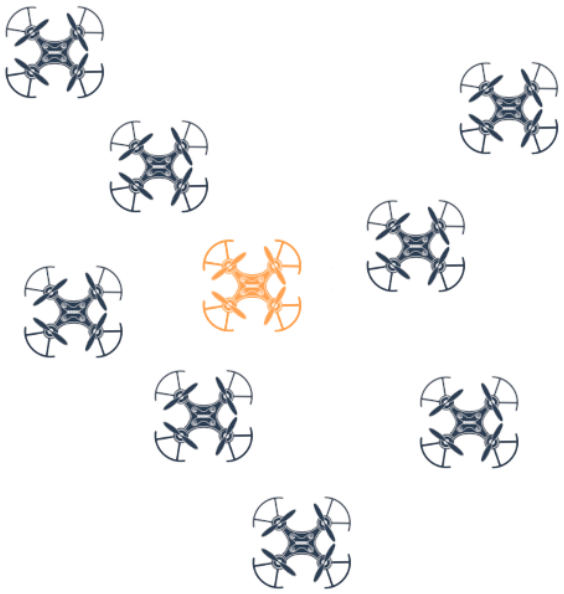


Figura 3.6: Formación Aleatoria.

Para finalizar, existen más conceptos sobre Ardusim que pueden ser explicados, como la integración de la GUI, el control y gestión de los UAV o, la propia asignación y distribución de los nodos. No obstante, al ser recursos más complejos que se encuentran fuera del alcance del trabajo, se ha decidido no entrar en detalles.

Capítulo 4

Caracterización de la topología de red mediante grafos

Al emplear un enjambre con una comunicación completa, es decir, con todos los drones pudiendo comunicarse entre si, y por ende libre del problema de los nodos ocultos, tener una noción de la topología de la red no es estrictamente necesario ya que hay comunicación directa entre nodos. Por otra parte, cuando surge el problema de los nodos ocultos, es necesario tener una visión general de como está compuesta la topología, ya que es necesaria dicha visión para poder llevar a cabo el encaminamiento de mensajes. Además, conocer la propia red da lugar a poder operar con aquellos nodos que se consideren activos, y así descartar aquellos que no son requeridos.

Las topologías pueden depender del tipo de formación seleccionada. Cabe destacar que la cantidad de drones del enjambre, así como la distancia entre ellos, son factores que pueden afectar al desarrollo de la misión; por ello, es estrictamente crucial emplear una buena estructura de datos que pueda almacenar todos los valores pertinentes para llevar a cabo dichas funciones.

Dentro de Java existen distintos tipos de estructuras de datos, pero hay que destacar que no todas sirven (o son poco óptimas) cuando se trata de almacenar los valores para dar paso a la construcción de la red. En las siguientes secciones se explica como se ha llevado a cabo el proceso de la construcción de la topología.

4.1. Descubrimiento de vecinos

Cuando un enjambre es desplegado en un ambiente de entorno real, a priori, no se conocen los otros nodos que van a pertenecer al enjambre. Por tanto, es muy conveniente establecer una fase previa a la construcción de la

red donde se descubran los vecinos de cada nodo para así tener una noción básica de la red. Cabe destacar que esta fase puede devolver directamente la totalidad de la información de la red si todos los drones del enjambre se encuentra en un mismo rango de comunicaciones (todos comunican con todos, o al menos con el dron líder del enjambre).

Dentro de esta fase, llamada Neighbor Discovery, cada nodo busca conocer a los otros nodos que se encuentran dentro de su rango de comunicaciones. En dicho estado, cada dron va a realizar dos operaciones en bucle durante un periodo de tiempo limitado. El propio usuario es el encargado de establecer dicho margen temporal aunque, como aproximación, se han establecido 8 segundos ya que, para las pruebas realizadas en apartados posteriores, se ha comprobado que este tiempo es óptimo en la mayor parte de configuraciones estudiadas. Antes de entrar en detalle, los mensajes enviados y recibidos dentro de esta fase contienen un formato especifico. El formato del mensaje viene dado por el identificador del UAV junto al identificador de la fase, tal y como se muestra en la tabla 4.1 (También hay una etiqueta donde se indica el receptor, pero como en este caso es una difusión, por lo que el valor establecido contiene un valor de -1).

Cuadro 4.1: Formato del mensaje en la fase Neighbor Discovery.

senderID: numUAV	**msgID: Nieghbor Discovery**	**receiverID: -1**

En una primera instancia, cada dron, envía su mensaje. Seguidamente, va a escuchar en el canal buscando recibir un mensaje ajeno (de otro dron) de esta misma fase. Si no se recibe nada no hay ningún procesamiento pero, si se consigue recibir un mensaje, se procesa el identificador del nodo que ha enviado el mensaje, el cual se almacena como un vecino.

Algorithm 1 Neighbor Discovery Phase

```
startTime ← CurrentTime
while CurrentTime − startTime < 8000 do      ▷ Los 8 segundos en ms
    Envío Mensaje
    msg ← Recibo Mensaje
    if msg ≠ null then
        Vecinos ← Identificador recibido
    end if
end while
```

De forma más visual, en la tabla 4.2 se muestra el resultado esperado en un hipotético caso donde el enjambre está compuesto por 9 drones en una

formación matricial, con un rango de separación entre drones de 100 metros, y con un rango de comunicaciones de 125 metros.

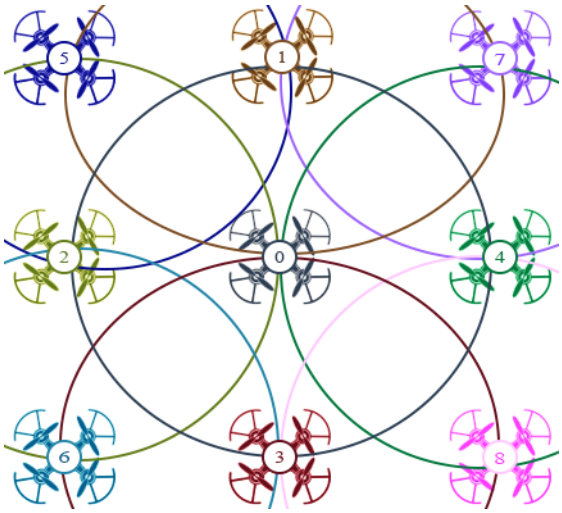


Figura 4.1: Ejemplo visual de los valores obtenidos en la tabla 4.2

Cuadro 4.2: Lista de los vecinos de cada nodo.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Vecinos	1,2,3,4	0,5,7	0,5,6	0,6,8	0,7,8	1,2	2,3	1,4	3,4

4.2. Construcción de la red mediante grafos

Previamente se mencionó que existen varias estructuras de Java que son capaces de almacenar los datos. El mayor problema es saber cual es la más adecuada para la construcción de la red. Entre todas las estructuras posibles, Java trae de forma nativa los famosos diccionarios, las pilas, los vectores, los árboles, etc. Un gran inconveniente de estos es que operar con ellos con las previsiones establecidas puede llegar a ser muy tedioso y confuso. Por tanto, se ha optado por hacer uso de una librería externa que introduzca un tipo de estructura de datos más fácil de usar, y que aporte mayores prestaciones.

De entre todas las librerías, se ha optado por hacer uso de JgraphT[12] ya que permite hacer uso de una gran variedad de grafos. Además, la propia librería trae una gran variedad de herramientas adicionales que favorecen a la construcción del grafo, al encaminamiento de mensajes, así como otros tipos de funcionalidades que darán lugar a la impresión e ilustración de estos.

Los propios grafos pueden contener como valores una gran variedad de datos desde los más primitivos, como los valores enteros, dobles, cadenas de caracteres, etc., hasta tipos de datos más complejos como objetos (pueden ser propios, es decir, creados por el usuario), estructuras de datos, etc., siempre y cuando dicho tipo de valor esté definido correctamente. Además, los enlaces empleados también ofrecen una gran ventana de posibilidades desde enlaces unidireccionales, con ciclos, sin ciclos, y bidireccionales, hasta enlaces que contengan peso propio (esto ayudará cuando se busque encontrar un camino óptimo desde dos puntos).

Seguidamente, el propio usuario tiene la libertad de poder definir sus propios grafos y enlaces. Tan solo se debe consultar la documentación pertinente para poder ir adaptando el objeto a las necesidades del programador.

Con todo el abanico de posibilidades que ofrece la librería se ha decidido trabajar con dos tipos de grafos distintos llamados SimpleWeightedGraph y DirectedAcyclicGraph, donde ambos harán usos de enlaces DefaultWeightedEdge (aristas con peso). Para la simulación, al tener una distancia fija, se ha decidido dejar el peso de los enlaces con un valor igual a 1 pero, si se quiere llevar a un ámbito más realista, este peso puede ser cambiado por la distancia que existe entre los nodos (de esta forma el encaminamiento variará según la distancia, que a su vez está relacionada con la potencia de la señal radio recibida).

A continuación, se van a explicar los grafos escogidos, así como sus diferencias y prestaciones.

- SimpleWeightedGraph: Es un grafo no dirigido, es decir, los enlaces establecidos entre dos nodos son bidireccionales y no restringen la dirección del enlace. Además, no permite bucles hacía un mismo nodo, ni múltiples enlaces hacía un mismo destino. Este grafo es muy útil para construir la topología de forma rápida y eficaz. Además, al tener enlaces bidireccionales, es más sencillo encontrar los caminos más cortos entre dos drones mediante el uso de los algoritmos de Dijkstra, Bellman-Ford, Astar, o FloydWarshall [13].

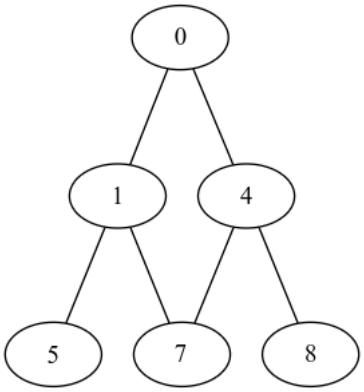


Figura 4.2: Ejemplo de SimpleWeightedGraph.

- DirectedAcyclicGraph: Este grafo es similar al anterior, salvo por que los enlaces son estrictamente dirigidos, es decir, son unidireccionales. Aunque sean similares hay que tener en cuenta que emplear este grafo, a priori, puede ser muy problemático si no se controla bien como se construye la topología. La única ventaja aparente es que, al ser más restrictivo, permite al usuario saber qué drones están más alejados, es decir, cuales son hojas (concepto importante que se explicará posteriormente).

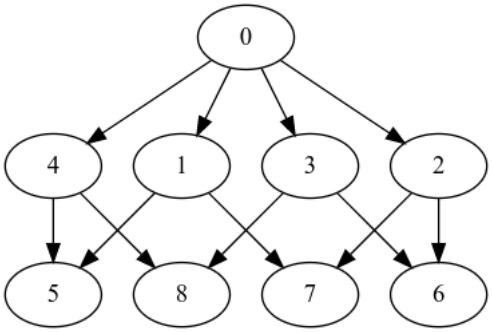


Figura 4.3: Ejemplo de DirectedAcyclicGraph.

Para el desarrollo de la capa multisalto se han empleado ambos grafos. El grafo simple se encarga del encaminamiento y de construir la red, mientras que, por otra parte, el grafo dirigido se construye, en base al simple, para saber si el nodo es una hoja o no. Es importante que se sepa si un nodo es una hoja o no ya que afectará al rendimiento de la capa de red.

Aquellos drones definidos como hojas son los que están más alejados, ubicándose en la periferia de la red (en nuestro caso drones el la periferia de

la formación). Se encargarán de no transmitir los mensajes recibidos (se disminuye el envío de mensajes, favoreciendo a reducir la congestión), y en cada fase de la misiones serán los que empezarán a enviar el mensaje pertinente al maestro (nos aseguramos de que el mensaje llegue al maestro para continuar con la misión). Mediante esta implementación se favorece al piggybacking, concepto que es explicado en el siguiente capitulo.

Una vez explicados los conceptos previos, se da paso a la explicación de la fase posterior a Neighbor Discovery. Esta nueva fase esta dividida en dos partes. En un primer lugar, se inicializa la red con los datos obtenidos previamente. Para ello, se añade al grafo el identificador del nodo junto con los identificadores de los vecinos para establecer así las interconexiones entre ellos. Posteriormente, se da paso a la construcción de la red en base a los mensajes recibidos. En esta segunda etapa los mensajes enviados contienen el identificador del nodo, el identificador del mensaje, el receptor (siendo una multidifusión) y, por último, la red actual. Este mensaje se ve representado en la tabla 4.3 y; cabe destacar que se va actualizando en cada ciclo del bucle.

Cuadro 4.3: Estructura del mensaje en la fase Build Network.

senderID: numUAV	**msgID: Build Network**
receiverID: -1	**network: currentNetwork**

El funcionamiento detallado de esta segunda parte es similar al Neighbor Discovery. Durante un periodo de tiempo establecido por el usuario (se han establecido 15 segundos, ya que es el mínimo valor óptimo para que funcione) se va a enviar la vista de red de un determinado nodo, y de manera similar se espera recibir la vista de red del vecino. Si se ha recibido un mensaje correcto se actualizará la red con los valores recibidos, esto se detalla en el algoritmo 3. Sobre cada iteración el mensaje se irá actualizando junto con la nueva red para que así todo el enjambre pueda tener una topología consistente. En el algoritmo 2 se muestra el comportamiento razonado.

Algorithm 2 Build Network

```
startTime ← CurrentTime
while CurrentTime − startTime < 15000 do   ▷ Los 15 segundos en ms
    Envio Mensaje
    msg ← Recibo Mensaje
    if msg ≠ null then
        Update Network
    end if
end while
```

El proceso para actualizar la red es un poco más complejo de explicar. Cuando se recibe el mensaje se extrae la topología adjunta al mensaje. Los valores extraídos tienen un formato único para poder facilitar su inserción en la red del propio nodo. Estos valores son un mapa donde la llave es el valor del vértice, y su contenido los vecinos a los que enlaza. Por tanto, por cada llave que hay en el mapa, se debe iterar hasta finalizar. En cada iteración se añadirá el vértice a la red si no está contenido y, a continuación, se accederá a la lista de vecinos de dicho identificador sobre el cual estamos iterando. Por cada vecino existente se procesará el enlace para crear la conexión entre la llave y su valor de vecino asociado.

Algorithm 3 Update Network

```
mapEdges ← receivedNetwork
for key : mapEdges do
    if network no contiene key then
        Añade valor a la topología
    end if
    enlaces ← enlacesDelVértice
    for i = 0; i < enlaces; i++ do
        Procesar Eje      ▷ Se vincula el vértice con el valor extraído de los vecinos
    end for
end for
```

Por lo general, todos los nodos del enjambre tendrán la visión de una misma topología. Previamente, se han mostrado dos tipos de grafos resultantes de esta operación.

Capítulo 5

Procesamiento de mensajes sobre capa multisalto

En el capítulo previo se ha detallado como se construye la topología red del canal multisalto, pero no se ha explicado como se procesan los mensajes, ni como se envían o reciben. En este capitulo se busca abarcar dichas características para poder dar una visión completa a esta nueva capa innovadora. Cabe destacar que sobre esta capa también se va a trabajar con mensajes en formato JSON.

5.1. Envío de mensajes multisalto

En primer lugar, el envío de mensajes es procesado de dos formas distintas o, mejor dicho, cuando se envía un mensaje hay dos procesos a seguir. Antes de enviar el mensaje se comprueba que el módulo del multisalto está activo; en caso afirmativo el mensaje es adaptado para posteriormente ser enviado. En caso contrario, simplemente se envía sin adaptar. Se ha hablado de adaptar pero ¿Qué significa esto?

Cuando se trabaja con el módulo multisalto hay que tener en cuenta que van a existir dos tipos de mensajes; aquellos que serán considerados dirigidos, es decir, que tienen un destino y, por otra parte, aquellos que serán tratados como difusión.

Cuando un mensaje tiene un destino concreto antes de enviarse se comprueba cual es el destino para poder construir, mediante los algoritmos previamente mencionados (Dijkstra, Bellman-Ford, etc.), un camino absoluto con la mínima distancia posible o, mejor dicho, el más corto desde el origen hasta el destino. Dicho camino se añade, con su etiqueta correspondiente, al mensaje JSON para que aquellos nodos que estén en rango puedan recibirlo

siempre y cuando estén contenidos en el camino.

Por otra parte, los mensajes por difusión son modificados para añadir un identificador (una cadena de caracteres alfanumérica aleatoria). Dicho identificador es anotado en una lista de identificadores para los mensajes de difusión antes de enviarse; en el siguiente punto se detalla esta característica. El algoritmo 4 se muestra el comportamiento previamente explicado sobre el código.

Algorithm 4 Adapt Message to Multihop

Input: JSONObject
Output: JSONObject
  if mensaje tiene destino **then**
    mensaje ← ["PathList", crearCamino]
  else ▷ Si no hay destino, se asume que es una difusión
    identificador ← crearIdentificador
    mensaje ← ["Broadcast", identificador]
    msgIDs ← identificador
  end if

5.2. Recepción de mensajes multisalto

A continuación, la recepción de mensajes tiene ciertas semblanzas al envío de mensajes. Al invocar a esta función se hace una llamada a la capa previa, HighLevelCommLink; cuando esta capa responde se almacena su valor en el mensaje (llegando a tener contenido o no). Si durante la llamada se ha recibido un mensaje vacío simplemente se devuelve ese valor. Por otro lado, si el mensaje tiene algún contenido, antes de devolverlo tendremos que comprobar si el módulo multisalto está activado. Si no lo está simplemente se devuelve el contenido aunque, por otra parte, si está activado se hará una serie de comprobaciones para ver si se descarta o no. Cuando se confirma el uso de multisalto el mensaje pasa a ser procesado. Si el mensaje contiene una etiqueta de difusión se extrae el identificador para posteriormente comprobar si dicho valor no está almacenado en la lista de identificadores de difusión. Si la secuencia está contenida se devuelve un mensaje vacío (para que no sea procesado) pero si no lo está, la cadena de caracteres es almacenada en la lista antes de pasarse el mensaje a otras capas. Cabe destacar, que si el mensaje no contiene la etiqueta de difusión se comprueba que sí contenga una etiqueta de encaminamiento; si dicha etiqueta tampoco existe, se pasa un mensaje vacío. En contra parte, si existe dicha lista, primero se extrae. Tras la extracción se hacen dos comprobaciones a la vez: (i) la lista debe

contener al menos dos valores, y (ii) el segundo valor debe corresponder al nodo que ha recibido el mensaje. Si estas condiciones se satisfacen el mensaje es devuelto, pero si no ha podido cumplirse simplemente se devuelve una referencia vacía. El algoritmo 5 expone el comportamiento explicado de una forma más concisa.

Algorithm 5 Filter Multihop Message

Input: JSONObject
Output: JSONObject o Null
  if mensaje tiene identificador de difusión **then**
    identificador ← difusión
    if lista de identificadores no contiene el identificador recibido **then**
      msgIDs ← identificador
      Devuelvo el mensaje
    end if
  else if mensaje tiene encaminamiento **then**
    camino ← lista al destino
    if lista con al menos dos nodos y soy el segundo nodo **then**
      Devuelvo el mensaje
    end if
  end if
  Devuelvo referencia vacía

Las referencias vacías permiten filtrar mejor los mensajes, es decir, descartarlos en caso de no tener que procesarse por dicho nodo para así agilizar las prestaciones. En la figura 5.1 se muestra como actúa la recepción de mensajes frente a las difusiones mientras que la figura 5.2 ante mensajes dirigidos en una formación linear con multisalto.

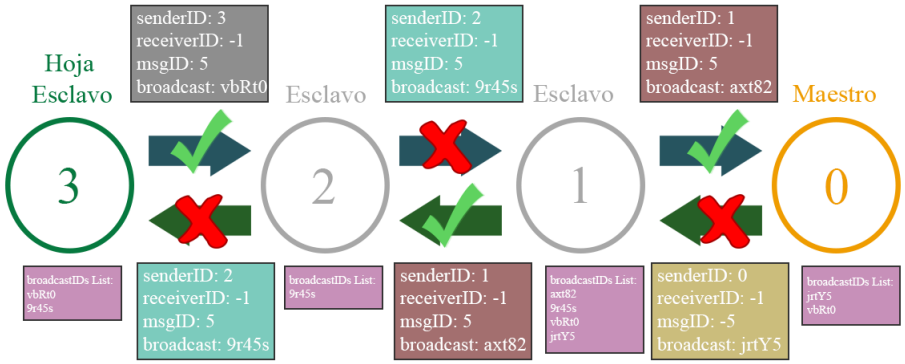


Figura 5.1: Procesamiento de mensajes por difusión.

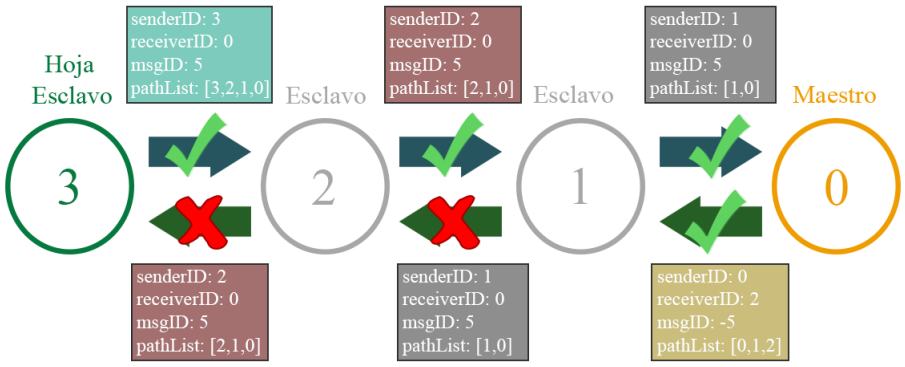


Figura 5.2: Procesamiento de mensajes dirigidos.

5.3. Piggybacking sobre los mensajes

Se ha explicado tanto el envío como la recepción de mensajes dentro de la capa pero ¿Qué pasa cuando un nodo ha de enviar un mensaje que ha recibido? ¿Lo envía así sin más, o hace algún tipo de comprobación extra? La respuesta a dicha pregunta es: Realiza una serie de comprobaciones para ver si hace uso de piggybacking. Pero, ¿Qué significa esto? El piggybacking es una técnica empleada para acumular varios reconocimientos enviados por distintos nodos en uno solo. Mediante dicha implementación se consigue re-

ducir el número de mensajes enviados y, además, se consigue acumular todo en un único mensaje, garantizando así que lleguen todos los mensajes al destino. Dicho mecanismo es aplicado para los mensajes que tienen un destino común, siendo esto aplicable tanto a mensajes normales como reconocimientos. Cuando el mensaje es recibido pasa por una serie de comprobaciones para ver si el mensaje obtenido se debe acumular o no. Para ello hay dos posibilidades; En primer lugar, el mensaje es dirigido. Cuando ambos mensajes tienen un destino establecido (tanto el mensaje recibido como el mensaje que envía el nodo actual periódicamente) se comprueba que ambos destinos sean el mismo y que los identificadores del mensaje sean de la misma fase. Si es así se realiza una acumulación, en caso contrario, simplemente se envía. En segundo lugar, si el mensaje es una difusión en vez de estar dirigido. Se comprueba que ambos mensajes sean difusiones (tanto el recibido como el enviado). Si es así, además, se comprueba que los identificadores en la etiqueta msgID coinciden en ambos mensajes (para asegurarse de que ambos mensajes están en el mismo estado y así no se acumulen mensajes de diferentes fases). Si todo se satisface se realiza una acumulación. En caso contrario, se envía el mensaje recibido sin acumular. Hay que remarcar que cuando se realiza piggybacking el identificador de difusión es modificado, ya que el contenido del mensaje original ha sido modificado. Cabe destacar que, para ambos mensajes, también se comprueba que no estén actualmente acumulados, ya que si lo están no se realizará ninguna operación de las previamente mencionadas. En el algoritmo 6 se muestra el código explicado en los párrafos previos.

Algorithm 6 Piggybacking

Input: JSONObject
Output: JSONObject
  if mensaje es difusión **then**
    mismoIdentificador ← idMensajeRecibido == idMensajeEnviado
    if mismoIdentificador y no estoy acumulado **then**
      mensajeEnviado ← mensajeRecibido
      mensajeEnviado ← identificadorDifusión
    end if
  else if mensaje es dirigido **then**
    mismoIdentificador ← idMensajeRecibido == idMensajeEnviado
    mismoDestino ← destMensajeRecibido == destMensajeEnviado
    if mismoDestino y mismoIdentificador y no estoy acumulado **then**
      mensajeEnviado ← mensajeRecibido
    end if
  end if

Por último, en la figura 5.3 se muestra un escenario donde se realiza piggybacking sobre una formación linear con mensajes dirigidos.

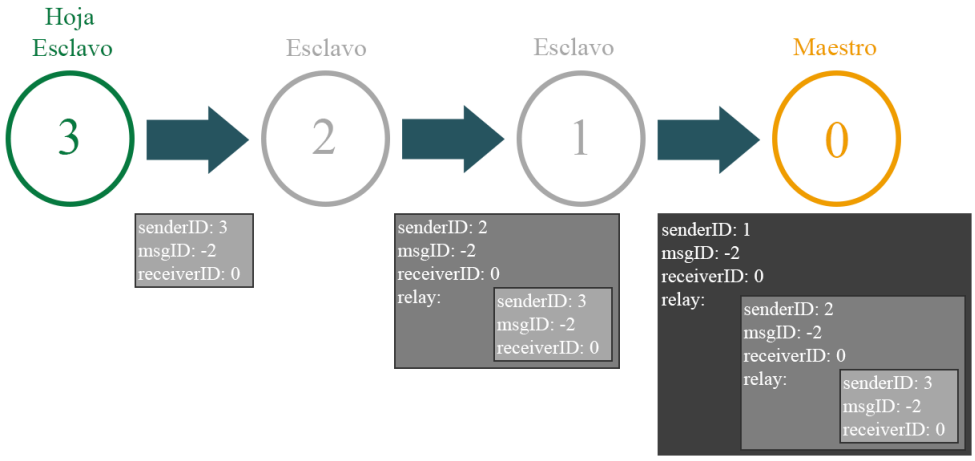
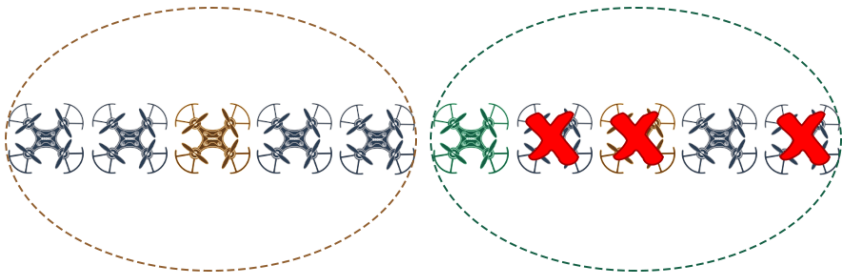


Figura 5.3: Ejemplo de Piggybacking.

Capítulo 6

MUSCOP al detalle

Antes de pasar al último capítulo de la memoria, la experimentación, se busca explicar el protocolo MUSCOP. Este capítulo va dirigido al protocolo sobre el cual se va a realizar toda la experimentación. MUSCOP, es uno de los protocolos más sofisticados que tiene el simulador. Este protocolo tiene la capacidad de ser resiliente ante la pérdida de nodos durante la misión. Otros protocolos no son capaces de operar si en algún punto de la misión se pierde algún componente; por otra parte, MUSCOP es capaz de adaptar y configurar un nuevo enjambre de forma dinámica durante el vuelo. La figura 6.1 muestra como un enjambre normal (figura izquierda) sufre varias pérdidas, en este caso 3 y, por tanto, debe adaptarse a esta situación anómala dejando como resultado un nuevo enjambre (figura derecha) con tan solo dos componentes, siendo uno de ellos el maestro (verde), y el otro un esclavo.



(a) Enjambre Inicial (b) Enjambre Actualizado

Figura 6.1: Tolerancia a fallos de MUSCOP.

Como todo protocolo, para llegar a finalizar una misión, primero ha de pasar por una serie de configuraciones iniciales para que pueda ejecutar la tarea encomendada y así poder finalizarla con éxito. Estos pasos son llamados estados y, a continuación, se va a hacer una visita guiada por todos y

cada uno de ellos para no perder ningún detalle. En total, son ejecutados 4 estados diferentes, aunque la gran mayoría de ellos dependen de una serie de subestados. Para empezar, los dos estados iniciales, que establecen toda la configuración necesaria para el vuelo (la configuración se hace antes del despegue), son llamados Setup y Take Off. Estos estados son ejecutados cuando el usuario empieza la simulación. En un primer lugar, el estado Setup está dividido en dos subestados. Para empezar, se inicializa el subestado Discover; durante este proceso cada elemento del enjambre envía mensajes periódicamente para que los otros nodos sean conscientes de su existencia. Una vez los nodos tiene una noción de la magnitud del enjambre, se pasa al siguiente subestado llamado Obtaining Waypoints. A continuación, el maestro envía un mensaje indicando los datos de la misión, siendo estos su identificador, la cantidad de drones que componen el enjambre, la cantidad de Waypoints que tiene la misión, y la localización de éstos (en tres dimensiones, contemplando los tres ejes). Al finalizar el Setup se pasa al estado del Take Off. Dentro de este estado se encuentran 3 subestados distintos, siendo estos 'obtain target location' seguido de 'move to target' y, finalizando con 'wait until everybody has reached target'. Inicialmente, 'obtain target location' es empleado para que el maestro envíe a los esclavos sus respectivas localizaciones en el aire. Cuando un esclavo recibe su destino, lo guarda y contesta con un reconocimiento. Seguidamente, una vez se han enviado y recibido todas las localizaciones, el maestro empieza a ordenar a los esclavos que se muevan a dichas posiciones; cuando un esclavo recibe esa orden vuelve a contestar con un reconocimiento, llegando así a finalizar el segundo subestado 'move to target'. Por último, pero no menos importante, en el último subestado el maestro estará a la espera de recibir un mensaje de cada dron que confirme que han llegado a su destino. Una vez haya recibido todos los mensajes, procederá a indicar a los esclavos que finalicen el estado del Take Off y pasen al siguiente. Mientras el maestro espera a dichos mensajes, por otra parte, los esclavos están constantemente esperando un mensaje que indique el cambio de estado mientras envían un mensaje indicando que están en su posición asignada. Con todo esto, se finalizan ambos estados, tanto el Setup como el Take off; de esta forma el enjambre ha pasado de estar en el suelo a estar en el aire, y con sus elementos listos para ejecutar la misión. Para mejor entendimiento, en la figura 6.2 se muestran los pasos previamente explicados sobre una máquina de estados.

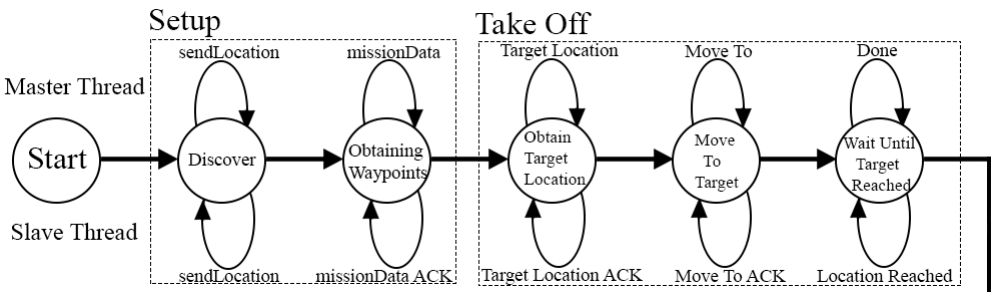


Figura 6.2: Estados Setup y Take off.

Una vez en el aire, llegan los dos últimos estados del protocolo llamados Fly y Land. Como bien indica su nombre, en resumidas cuentas se tratan de aquellos estados que llevan a cabo la misión en el aire, y que permiten a los nodos aterrizar aunque, al igual que los anteriores estados, Fly está compuesto por dos subestados. Estos procesos que se ejecutan en Fly son llamados Waypoint Reached y Move to Waypoint. Además, serán ejecutados de forma secuencial tantas veces como waypoints haya. Para empezar, el maestro tiene una lista donde tiene anotados qué drones están activos; debido a esto, cuando esté situado sobre un waypoint, esperará a recibir un mensaje de cada nodo que esté apuntado en la lista. Por otra parte, los esclavos irán enviado mensajes notificando que se encuentran en dicho waypoint. A continuación, cuando el maestro decide avanzar al siguiente punto, envía un mensaje a los esclavos para que avancen. Estos, al recibir dicho mensaje, avanzarán e irán transmitiendo mensajes hasta llegar al siguiente punto, para que así los otros nodos se den cuenta de que dichos emisores siguen activos. Es importante remarcar que todo nodo contiene una lista con los drones activos, aunque se irá actualizando con el paso de la misión si así se requiere, es decir, dicha lista lleva el conteo de los drones activos si alguno de ellos no ha dado señales de vida durante un periodo de tiempo (igual al tiempo de vida establecido, por defecto de 15 segundos) se considera que ha fallado y es eliminado. De esta forma se mantiene la cohesión del enjambre en todo momento, adaptándose de forma dinámica a los acontecimientos. Para finalizar, una vez se ha llegado al último waypoint, cada dron procede a aterrizar; esto se ejecuta durante el estado Land. Para dar un detalle más visual de lo previamente mencionado, se adjunta la figura 6.3 con el comportamiento de dichos estados.

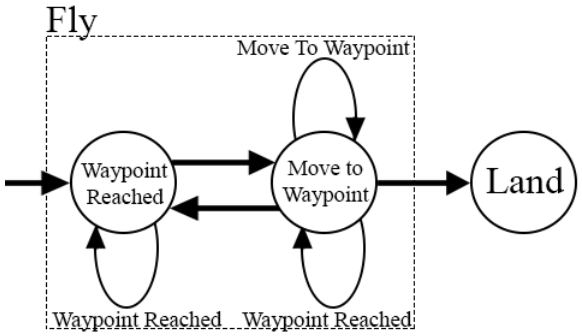


Figura 6.3: Estados Fly y Land.

Todos los estados explicados provienen de una misma máquina de estados, aunque en las figuras anteriores se ha desglosado para mayor comprensión; en la figura 6.4 se puede observar la máquina de estados al completo.

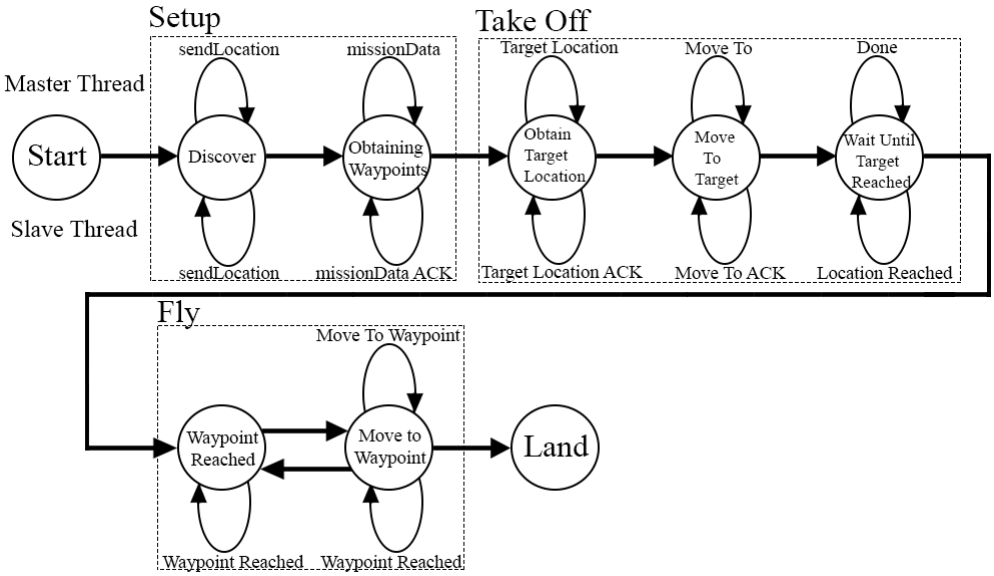


Figura 6.4: Máquina de estados de MUSCOP.

Por último, cuando se hace uso del multisalto, se añade un estado adicional a la máquina previamente mostrada llamado Init Link. Normalmente este estado suele incluirse entre los estados Take Off y Fly, ya que es cuando interesa añadir el multisalto. Para ello, en la figura 6.5 se muestra como resulta la máquina de estados final añadiendo la nueva idea desarrollada.

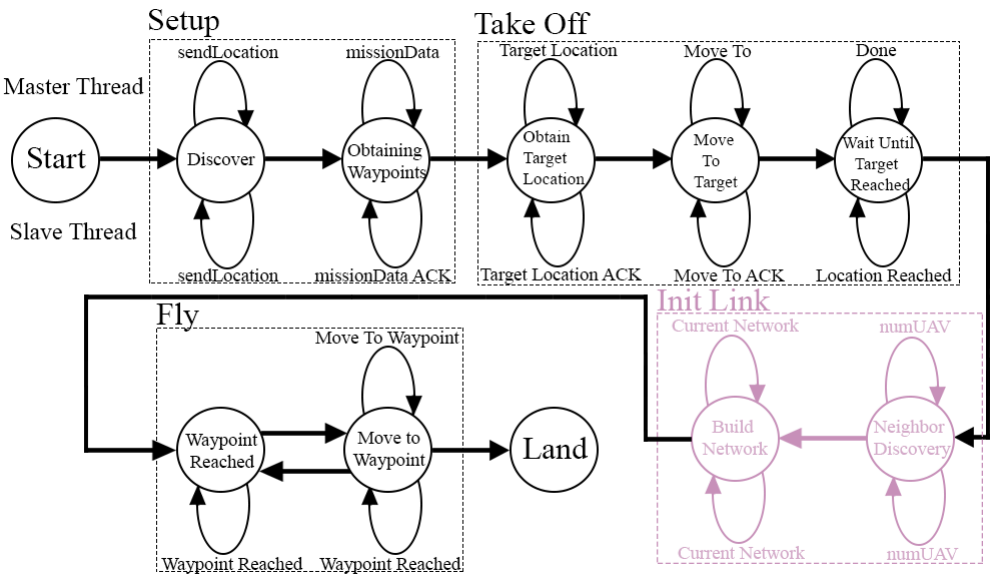
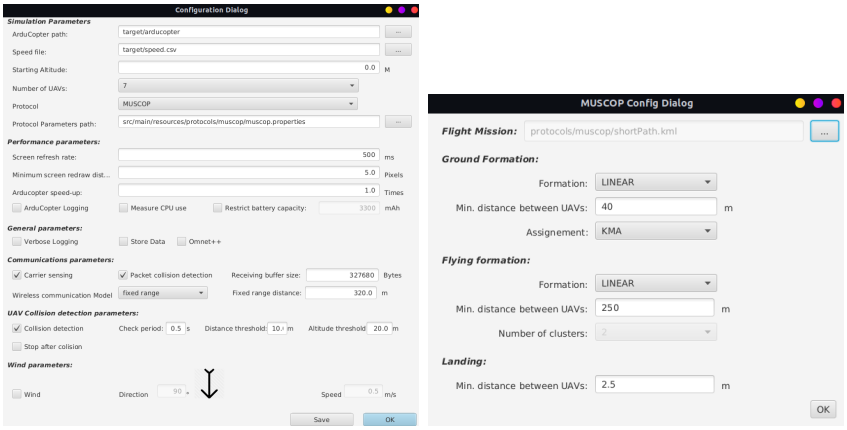


Figura 6.5: Máquina de estados modificada para multisalto.

Capítulo 7

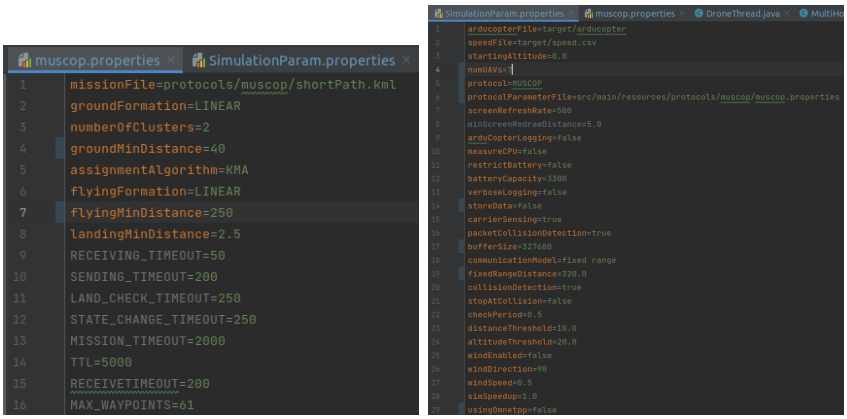
Experimentación

En este capitulo de la memoria se busca mostrar los resultados obtenidos mediante la implementación de las ideas previamente planteadas. Para los experimentos se ha trabajado sobre las formaciones linear y matricial principalmente ya que las otras dos formaciones no están adaptadas para poder usarse fuera de rango, y su modificación supone un problema general para el simulador y sus otros protocolos. Por tanto, se ha querido comprobar mediante experimentos hasta donde se puede llegar con el multisalto. Por ello, se han hecho distintas pruebas desde los dos saltos hasta los cinco. El problema principal que se ha localizado es el envío de los mensajes: al emplear UDP gran parte de los mensajes no llegan a recibirse cuando el número de saltos es elevado. En este caso, a base de pruebas, se ha comprobado que el número de saltos óptimo para la formación linear son de 2 o 3, mientras que para la formación matricial se han logrado buenos resultados con dos saltos. Al intentar aumentar el número de saltos en ambas formaciones se han detectado los mismos problemas derivados de la pérdida de paquetes. Para la experimentación se ha trabajado con un rango fijo, ya que es la configuración más adecuada para hacer distintas pruebas con el multisalto. El rango fijo se ha establecido a unos 340 metros, y la separación entre los drones se ha decidido fijar en unos 250 metros. De esta forma, se asegura que hay multisalto ya que limitamos a que cada dron conozca el mínimo número de vecinos para poder funcionar. Por otra parte, la configuración de la simulación se puede hacer mediante interfaz gráfica, tal y como se muestra en la figura 7.1 o, en contraposición, mediante la modificación de los archivos SimulationParam.properties y muscop.properties mostrados en la figura 7.2.



(a) Enjambre Inicial (b) Enjambre Actualizado

Figura 7.1: Interfaz Gráfica de Ardusim.



(a) Enjambre Inicial (b) Enjambre Actualizado

Figura 7.2: Ficheros SimulationParam.properties y muscop.properties.

Asimismo, se ha comparado ambas versiones de MUSCOP: su versión original, y la modificada con multisalto. Con esta comparación se busca obtener cuales han sido las mejoras de las prestaciones, y sobre que ámbitos supone un inconveniente.

7.1. Formación Linear

Para la formación linear se han realizado pruebas sobre dos tipos de escenarios diferentes. En un primer lugar, se considera el escenario donde hay 5 drones que forman un enjambre fiable con 2 saltos. Seguidamente, se ha

experimentado satisfactoriamente sobre un enjambre compuesto por 7 drones donde se aporta un comportamiento de 3 saltos.

7.1.1. Enjambre con 5 drones

En primer lugar, el envío y recepción de mensajes refleja unos resultados bastante interesantes, como se puede ver en la figura 7.3. El nodo maestro tiene una menor tasa de mensajes enviados pero, por otra parte, una mayor tasa de mensajes recibidos. Esto concuerda ya que, durante la ejecución de la misión, el papel principal del maestro es distinto al de los esclavos ya que, durante el estado Waypoint Reached, solo los esclavos envían mensajes. De esta forma, se puede observar como los esclavos tienen una mayor tasa de envío respecto al maestro, ya que estos si que envían mensajes en dicho estado. Respecto a la recepción también tiene sentido que el nodo maestro reciba muchos más mensajes; hay que tener en cuenta que, al no tener que enviar mensajes en un estado, tiene mucho más tiempo para leer más mensajes y procesarlos. Por tanto, viendo los resultados, se observa un comportamiento bastante coherente entre la relación maestro-esclavo y sus mensajes. Para mayor detalle, se ha obtenido que hay una tasa media de envío de 130,8 mensajes, mientras que para la recepción esta tasa asciende a 199,4. Los valores mostrados en la figura se pueden comprobar al detalle en la tabla 7.1.

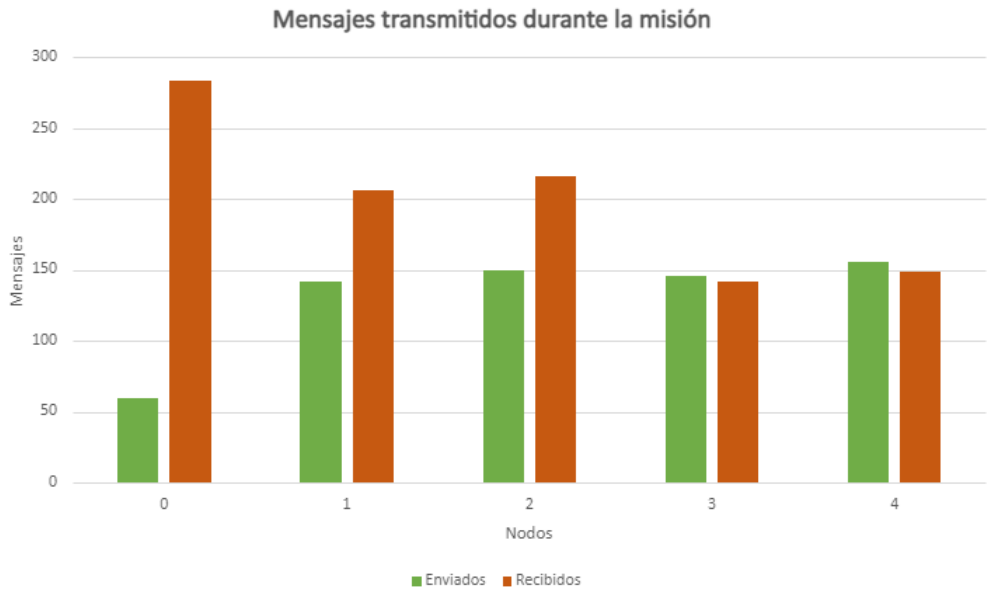


Figura 7.3: Envío/Recepción de mensajes en enjambre linear de 5 nodos.

Cuadro 7.1: Cantidad de mensajes transmitidos y recibidos durante la misión.

Identificador de nodo					
	0	**1**	**2**	**3**	**4**
Enviados	60	142	150	146	156
Media de Mensajes Enviados				130,8	
Recibidos	284	206	216	142	149
Media de Mensajes Recibidos				199,4	

Seguidamente, el tiempo que transcurre durante la ejecución de la misión es bastante estable, es decir, a pesar de tener que emplear multisalto, los drones más alejados solo tardan 6 segundos más en completar la misión. En un primer lugar, el maestro es el primero en finalizar, como cabe esperar y, a continuación, los siguientes nodos tardan unos pocos segundos más. Los más cercanos solo tardan 4 segundos más, mientras que los más alejados 6. Con esto se puede concluir que, para este escenario, hay un tiempo de vuelo medio de 145,056 segundos o, lo que es lo mismo, 2 minutos y 41 segundos. Los valores han sido extraídos de la figura 7.4 y se han vertido sobre la tabla 7.2 para una mayor comprensión.

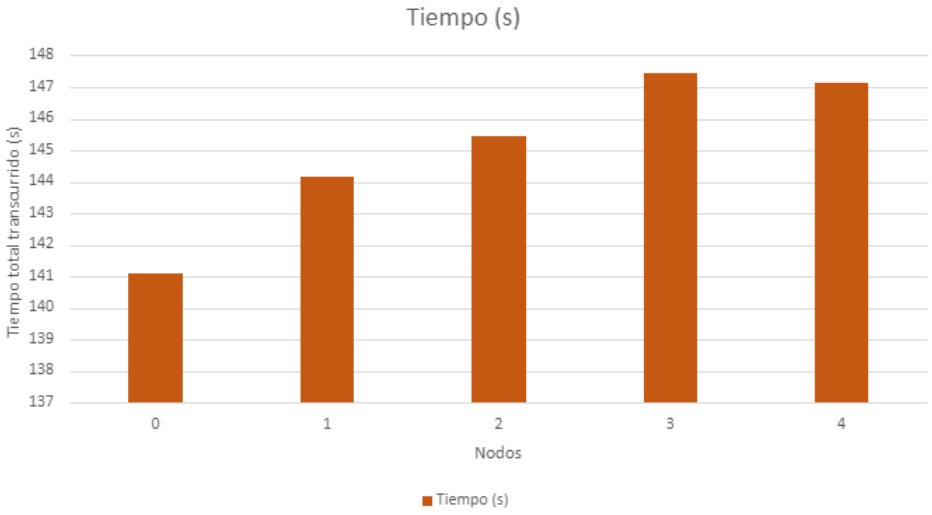


Figura 7.4: Tiempo transcurrido para finalizar la misión en un enjambre linear de 5 drones.

Cuadro 7.2: Tiempo de total en completar la misión.

Transcurso total de la misión					
	0	**1**	**2**	**3**	**4**
T (s)	141,112	144,137	145,462	147,431	147,140
Tiempo medio (s)			145,056		
T (m)	2,351	2,402	2,424	2,457	2,452
Tiempo medio (m)			2,417		

Según se puede observar en la tabla 7.3, los vecinos descubiertos por cada dron corresponden con el comportamiento teórico previamente explicado. Los nodos más alejados solo tienen un vecino, mientras que los demás solo tienen 2.

Cuadro 7.3: Tabla de vecinos

Vecinos de cada nodo					
	0	**1**	**2**	**3**	**4**
Vecinos	1,2	0,3	0,4	1	2

Para finalizar, una vez se han obtenido los vecinos, falta saber como ha quedado el grafo resultante. Para ello, se puede observar en la figura 7.5 que el enjambre compuesto por 5 drones ha sido construido correctamente al contrastar la figura con los valores obtenidos en la tabla de vecinos.

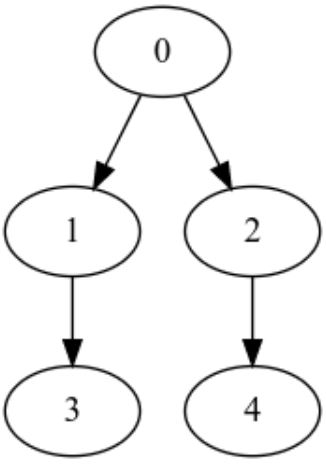


Figura 7.5: Grafo Resultante de un enjambre compuesto por 5 drones

7.1.2. Enjambre con 7 drones

Para completar nuestro estudio, se procede ahora a presentar resultados experimentales con un enjambre de mayor tamaño, concretamente con 7 drones. Dentro de este nuevo escenario, con 3 saltos, se puede observar un comportamiento distinto al de 2 saltos. En un principio el maestro es idéntico, ya que es el que mayor tasa de recepción, tiene junto con la menor tasa de envío. No obstante, el comportamiento de los esclavos varía un poco. Previamente la comunicación era más simple, si hay que enrutar los mensajes solo hay que atravesar un nodo, por lo que es más fácil hacer el piggybacking con un tráfico más limitado, pero cuando hay más nodos intermedios entre las hojas y el maestro, ¿qué sucede? De forma simple, sabemos que el canal se satura más. Para empezar, la tasa de envío de todos los esclavos es prácticamente la misma ya que las hojas empiezan a transmitir los mensajes y, cuando este llega a un nodo destino, empieza a enviar el mensaje. Por lo tanto, es normal que tengan una tasa muy similar. Por otra parte, la recepción de mensajes satura un poco más a los nodos intermedios ya que, a veces, se pierden por el camino y hay que volver a enviarlos hacía el más alejado. La gráfica 7.6 explicada tiene sus datos contrastados en la tabla 7.4 para dar más detalle. Además, se observa que el número medio de mensajes enviados es 138,857, mientras que los recibidos son 207,428, siendo unos valores superiores al del experimento anterior.

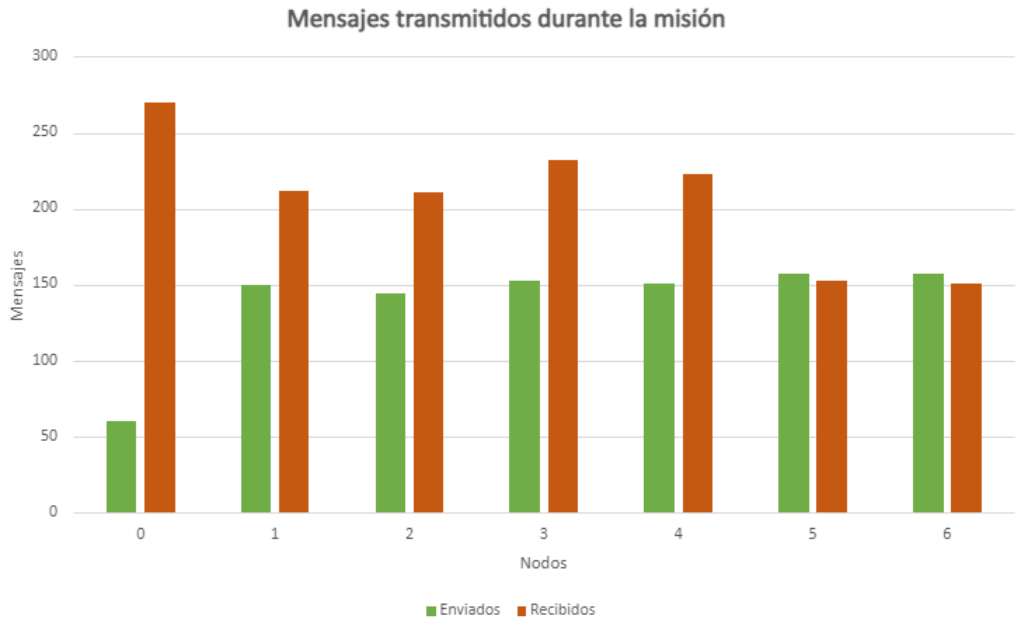


Figura 7.6: Envío/Recepción de mensajes en enjambre linear de 7 nodos.

Cuadro 7.4: Cantidad de mensajes transmitidos y recibidos en la misión.

Identificador de nodo							
	0	**1**	**2**	**3**	**4**	**5**	**6**
Enviados	60	150	144	153	151	157	157
Media de Mensajes Enviados	138,857						
Recibidos	270	212	211	232	223	153	151
Media de Mensajes Recibidos	207,428						

Por otra parte, el tiempo de vuelo cumple un balance esperado, donde cuanto más alejado del maestro está un nodo, más tiempo tarda en finalizar la misión. En este caso, los desfases temporales son crecientes en una franja ínfima de segundos. Los vecinos del maestro tardan un segundo de más en finalizar, mientras que aquellos que están a dos saltos solo tardan dos segundos más respecto al maestro; además, tal y como se aprecia, los extremos pueden llegar a tardar desde cuatro hasta cinco segundos más. El tiempo medio, extraído de la figura 7.7 y contrastado en la tabla 7.5, ronda los 146,771 segundos, lo que corresponde a dos minutos y 44 segundos.

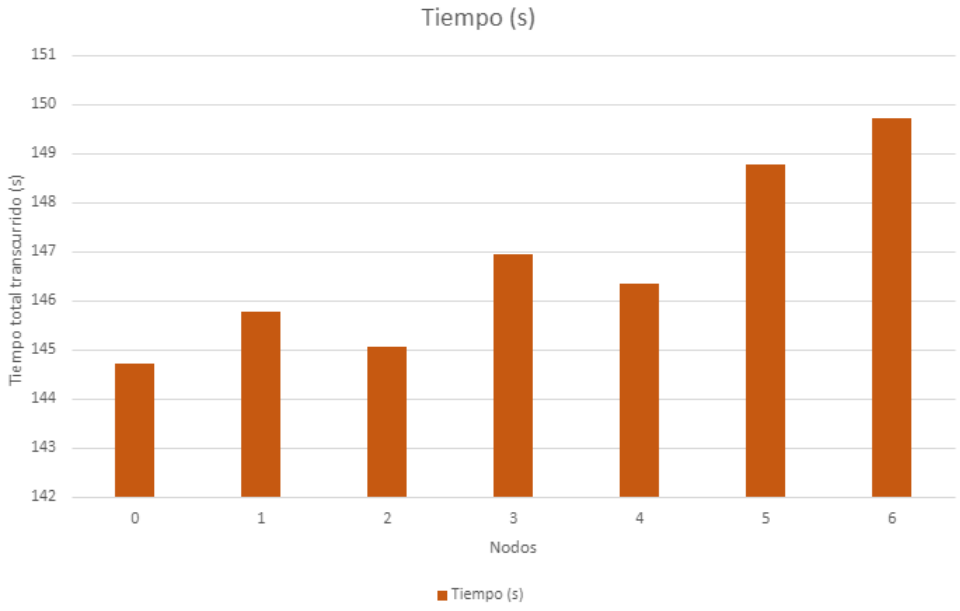


Figura 7.7: Tiempo transcurrido para finalizar la misión en enjambre linear de 7 drones.

Cuadro 7.5: Tiempo de total en completar la misión.

Identificador de nodo							
	0	**1**	**2**	**3**	**4**	**5**	**6**
T (s)	144,715	145,774	145,074	146,962	146,352	148,782	149,740
Tiempo medio (s)				146,771			
T (m)	2,411	2,429	2,417	2,449	2,439	2,479	2,495
Tiempo medio (m)				2,446			

Al seguir en un escenario linear, la tabla de vecinos es prácticamente similar a la del experimento previo. Como se puede observar en la tabla 7.6, los nodos más alejados contienen solo un vecino, mientras que los intermedios conocen a dos vecinos.

Cuadro 7.6: Tabla de vecinos

Vecinos de cada nodo							
	0	**1**	**2**	**3**	**4**	**5**	**6**
Vecinos	1,2	0,3	0,4	1,5	2,6	3	4

Concatenando con la tabla de vecinos se puede observar que el grafo de la figura 7.8 cumple exactamente con los requisitos esperados. Por tanto, se puede confirmar que este experimento ha sido un éxito rotundo.

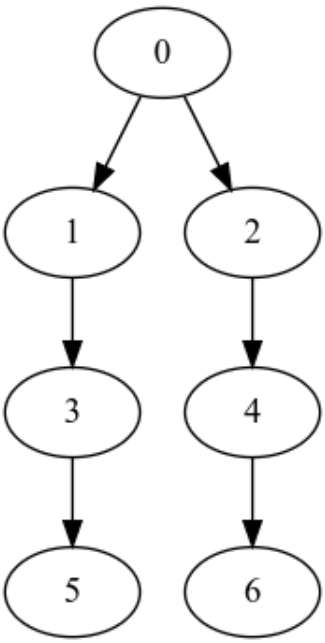


Figura 7.8: Grafo Resultante de un enjambre compuesto por 7 drones.

7.2. Formación Matricial

En esta nueva sección la experimentación es un poco la misma ya que se revisa la tasa de envío/recepción, el tiempo total en ejecutar la misión definida, se analiza la tabla de vecinos, y se finaliza con la construcción del grafo. Sin embargo, en este caso se trabaja sobre un enjambre de 9 drones en formación matricial, dando así un escenario multisalto con dos saltos.

Las tasas de envío y recepción respetan el comportamiento obtenido en los experimentos previos, tanto el del nodo maestro como el de los esclavos. A pesar de ello, se puede observar como la tasa de recepción del maestro se ha disparado. Esto se debe a que, previamente, el maestro siempre tenía dos vecinos; no obstante, en este nuevo escenario, dicha premisa ha cambiado. Para esta prueba, con las prestaciones aportadas, se da el caso de que el propio maestro es capaz de conocer a cuatro vecinos. Por lo tanto, si comparamos los valores que se han obtenido anteriormente con los actuales, se observa que se han duplicado (al tener el doble de vecinos, es esperable obtener el doble de mensajes). Para ello, la figura 7.10 nos muestra los resultados obtenidos. Además, se adjunta la tabla 7.7 con los valores extraídos de la figura. La tasa de mensajes recibidos incrementa a 124,667 mensajes, mientras que la recepción aumenta hasta 227,556 mensajes.

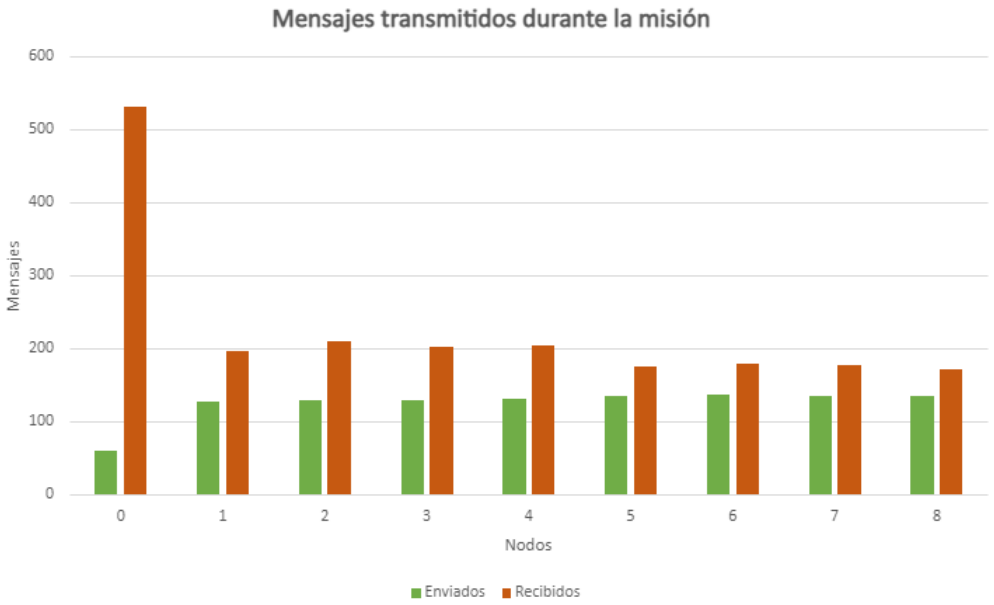


Figura 7.9: Envío/Recepción de mensajes en enjambre matricial de 9 nodos

Cuadro 7.7: Cantidad de mensajes transmitidos y recibidos en la misión.

Identificador de nodo									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Enviados	60	127	129	130	131	136	137	136	136
Media de Mensajes Enviados	124,667								
Recibidos	532	196	210	202	204	175	180	178	171
Media de Mensajes Recibidos	227,556								

El transcurso de la misión también mantiene cierta correlación con los experimentos anteriores. Según se observa, el nodo maestro es el primero en finalizar, y seguidamente van sus vecinos. Por último, finalizan los drones más alejados (en la periferia). Según se observa hay un desfase de dos segundos entre los drones más alejados, y un desfase de un segundo con los vecinos del maestro respecto a él. En la figura 7.10 se pueden observar dichos desfases temporales y, en la tabla 7.8 se puede ver al detalle los valores. Para concluir, se puede observar que el tiempo medio de vuelo es de 132,028 segundos, lo que equivale a 2 minutos y 23 segundos.

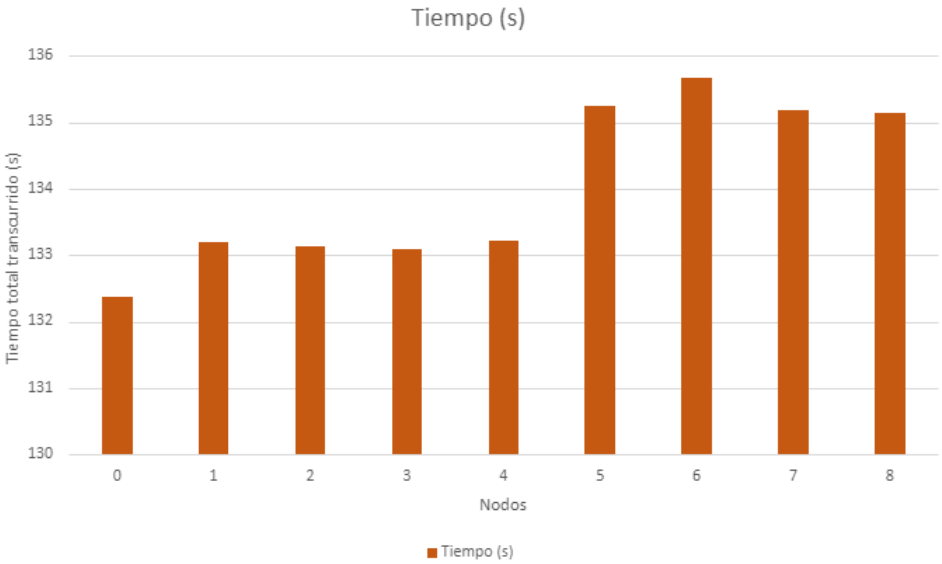


Figura 7.10: Tiempo transcurrido para finalizar la misión en enjambre matricial de 9 drones.

Cuadro 7.8: Tiempo de total en completar la misión.

Identificador de nodo									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
T (s)	132,368	133,203	133,145	133,084	133,211	135,241	135,673	135,181	135,147
Tiempo medio (s)				134,028					
T (m)	2,206	2,220	2,2190	2,2180	2,220	2,254	2,261	2,253	2,252
Tiempo medio (m)				2,233					

En este escenario la tabla de vecinos varía respecto a la formación linear. Para este escenario se observa como el nodo maestro conoce a cuatro esclavos distintos. Estos vecinos directos, a su vez, conocen a otros tres nodos del enjambre. Para finalizar, los más alejados de la periferia solo son capaces de localizar a dos vecinos. La tabla 7.9 muestra dicha correlación.

Cuadro 7.9: Tabla de vecinos para la formación matricial.

Identificador de nodo									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Vecinos	1,2,3,4	0,5,7	0,5,6	0,6,8	0,7,8	1,2	2,3	1,3	3,4

Para finalizar, el grafo de la figura 7.12 plasma perfectamente los resultados de la tabla anterior, llegando así a mostrar un grafo dirigido correctamente construido con los valores esperados.

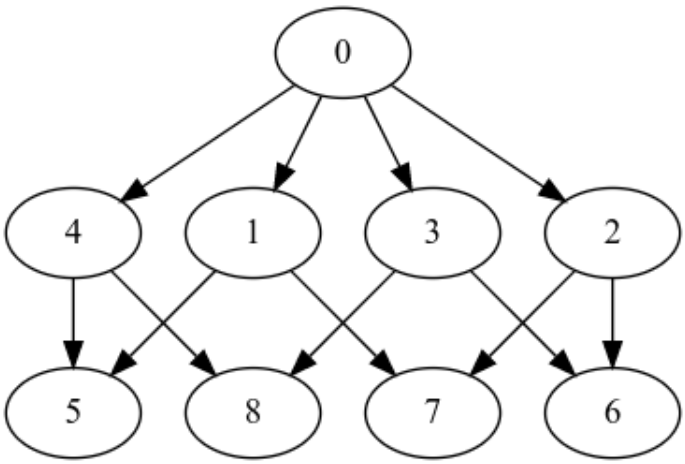


Figura 7.11: Grafo resultante de un enjambre compuesto por 9 drones.

7.3. Análisis y comparación entre ambas versiones de MUSCOP

Como última observación se ha planteado comparar ambas versiones de MUSCOP, aquella donde todos los nodos están en rango y no tienen ningún tipo de filtrado con la actual versión que soporta el problema de los nodos ocultos y utiliza mecanismos de filtrado y piggybacking. Primero se medirán las prestaciones previamente analizadas sobre la versión original, es decir, se medirán las tasas de mensajes y el tiempo transcurrido en el vuelo. Posteriormente, una vez se muestren los datos en las secciones pertinentes, se procederá a comparar y a concluir con las pruebas previamente expuestas.

7.3.1. Formación Linear

Para empezar, la formación linear ha sido configurada con todos los nodos dentro del rango de comunicaciones del maestro con una separación de 40 metros entre ellos. Además, el enjambre está compuesto por una cantidad de siete nodos. El primer análisis viene dado por las tasas de envío y recepción de mensajes. Tal y como se puede observar en la figura 7.12 el comportamiento mostrado es similar a cuando se emplea multisalto salvo por una cosa: la cantidad de mensajes. Según se puede observar en la gráfica, la cantidad de mensajes recibidos duplica los valores previamente obtenidos. Esto demuestra que los mecanismos de filtrado aplicados en su versión mejorada están aliviando el canal de comunicaciones. Respecto a la tasa media de envío y

recepción, se pueden observar que se recibe una media de 390,857 mensajes, mientras que se envían 113,857. La tabla 7.10 detalla los datos de la figura.

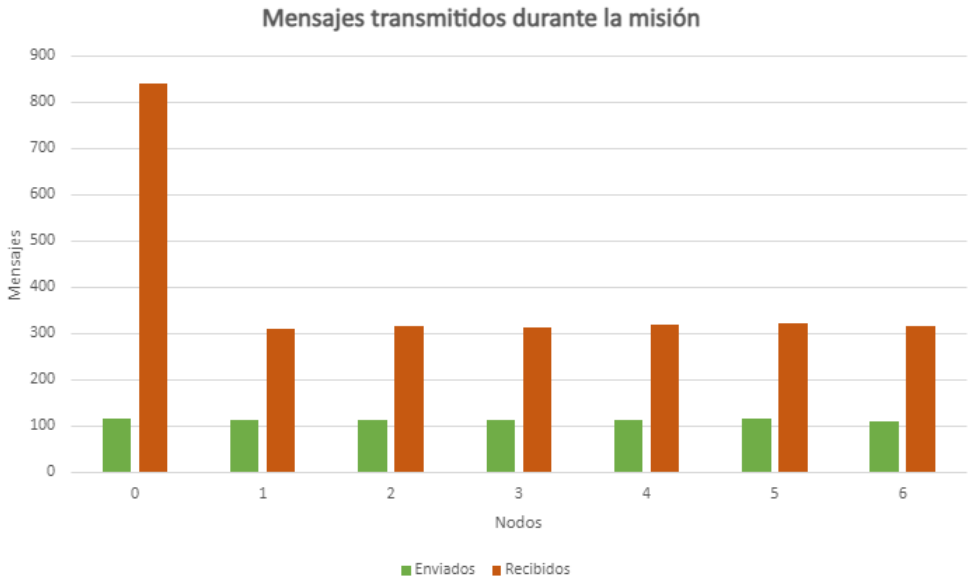


Figura 7.12: Envío/Recepción de mensajes en MUSCOP

Cuadro 7.10: Cantidad de mensajes transmitidos y recibidos en la misión.

Identificador de nodo							
	0	**1**	**2**	**3**	**4**	**5**	**6**
Enviados	116	112	113	114	114	117	111
Media de Mensajes Enviados	113,857						
Recibidos	842	309	315	314	319	321	316
Media de Mensajes Recibidos	390,857						

A continuación, el análisis del transcurso total de la misión puede observarse como una mejora también. Si se analiza la figura 7.13 se puede observar como algunos esclavos finalizan la misión antes que el propio maestro. Este comportamiento se debe a la saturación del canal. Como el maestro es bombardeado con 6 mensajes simultáneos cada segundo (un mensaje por dron), tarda más en procesar los datos y, por lo tanto, mientras se procesa el mensaje de un nodo ya hay otros a la espera. Como dicho nodo ha recibido la respuesta, pasa a finalizar la misión mientras que los otros aún siguen a la espera de que se procese su mensaje. Respecto a los tiempos, como se puede observar, al estar todos en rango, la máxima variabilidad que hay es de un o

dos segundos. El tiempo medio para finalizar la misión es de 178,7 segundos, aunque convertido a minutos es representado como 2,97 minutos. La tabla 7.11 representa los valores extraídos de la figura

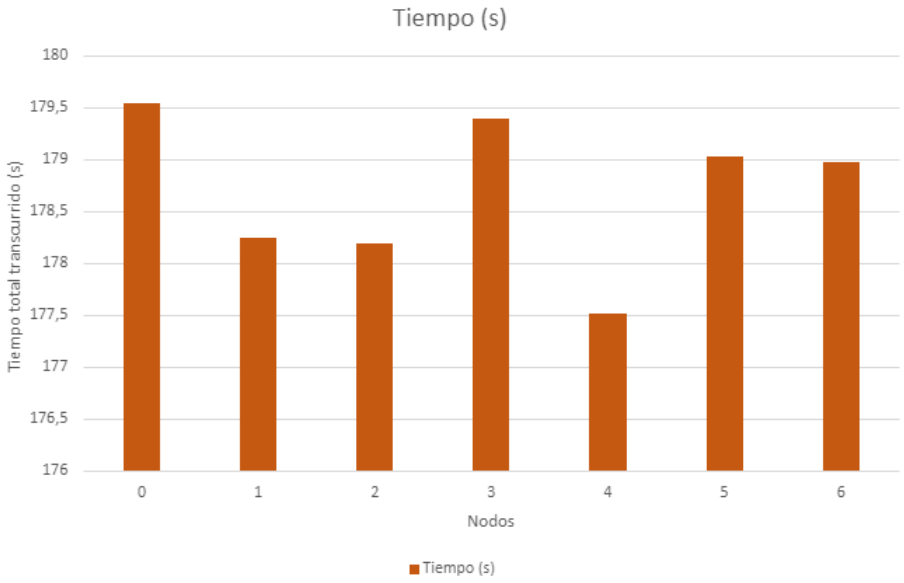


Figura 7.13: Envío/Recepción de mensajes en MUSCOP.

Cuadro 7.11: Tiempo de total en completar la misión

Transcurso total de la misión							
	0	**1**	**2**	**3**	**4**	**5**	**6**
Tiempo (s)	179,55	178,252	178,194	179,391	177,514	179,027	178,974
Tiempo medio (s)			178,700				
Tiempo (m)	2,9925	2,970866667	2,9699	2,98985	2,958566667	2,983783333	2,9829
Tiempo medio (m)			2,978				

Para finalizar, al analizar ambos escenarios, se pueden observar varios detalles. En un primer lugar, al emplear multi-salto, hay un mayor envío de mensajes respecto al original; esto se debe a que los nodos situados en los extremos (hojas) empiezan a transmitir los mensajes, y estos van pasando por los nodos intermedios hasta llegar al maestro. Al tener agentes intermedios es normal que se incremente un poco la cantidad de mensajes enviados, aunque si se desglosan los mensajes, se puede ver que dichos mensajes enviados con multisalto hacen uso del piggybacking, y por lo tanto contendrán la información más compactada. Este comportamiento se ve reflejado en la tasa de recepción donde multisalto tiene un menor índice de recepción debido a que

los datos de cada extremo vienen compactos en un único mensaje, mientras que MUSCOP original envía los datos constantemente, siendo redundante. Este último comportamiento lleva a una mayor saturación del canal ya que, si se observan los datos obtenidos, se puede ver como la versión original llega a procesar, en la recepción, casi más de 200 mensajes en comparación con su versión mejorada. Por otra parte, los tiempos de ejecución en la versión optimizada son mucho menores que en los de su versión original. Hay una diferencia de 32 segundos entre ambas. lográndose unas mejores prestaciones en la nueva versión. Como conclusión se puede afirmar que la versión actualizada es mucho mejor que la original. A pesar de tener una peor tasa de envíos (bastante ínfima), en todos los otros aspectos mejora considerablemente. Estos datos se pueden ver contrastados en la tabla 7.12.

Cuadro 7.12: Comparativa entre ambas versiones de MUSCOP.

	Original	**Multisalto**
Media de Mensajes Enviados	113,857	138,857
Media de Mensajes Recibidos	390,857	207,428
Tiempo medio (s)	178,7	146,771
Tiempo medio (m)	2,978	2,495

7.3.2. Formación Matricial

Para la formación matricial se han empleado los mismos parámetros que para la formación linear, es decir, se ha establecido un rango fijo de 320 metros con una separación de 40 metros entre los nodos. La diferencia notable es el número total de drones que componen el enjambre, que en estos experimentos han sido de nueve. Para esta formación se hace el mismo análisis empezando por la tasa de envíos, y finalizando con las comparaciones.

Si comparamos ambas versiones, de primeras se ve otra vez un comportamiento similar donde los esclavos tienen un índice de recepción similar, salvo el maestro que excede en creces sus valores. La tasa de envíos se mantiene casi constante en todos los nodos, pero la preocupación principal viene dada en la recepción del maestro, que llega casi a los 1000 mensajes recibidos. Aquí se puede observar claramente como, al emplear multisalto, el número de mensajes se ve reducido prácticamente a la mitad, dando un respiro al canal de comunicaciones. La figura 7.14 muestra de forma visual la representación de los datos, mientras que la tabla 7.13 contiene dichos valores para su mejor compresión.

Figura 7.14: Envío/Recepción de mensajes en MUSCOP

Cuadro 7.13: Cantidad de mensajes transmitidos y recibidos en la misión.

Identificador de nodo									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Enviados	117	118	119	116	116	119	117	118	117
Media de Mensajes Enviados	117,444								
Recibidos	959	323	323	325	321	318	323	319	320
Media de Mensajes Recibidos	392,333								

Para analizar el transcurso de la misión es importante fijarse en los desbalances de la figura 7.15. Se puede observar como el dron maestro, en este caso, es el primero en finalizar, pero a partir de este punto cada nodo finaliza en un periodo de tiempo más variable. Esto se debe principalmente a la cantidad de mensajes que ocupan el canal de comunicaciones y, al ser un escenario donde hay más drones, es completamente normal que haya más saturación de por si. Para estos desfases se puede afirmar que hay una varianza de uno a cinco segundos. Si se observa el tiempo medio se puede apreciar como de media se tarda 176,735 segundos (2,945 minutos) en finalizar la misión. Se pueden contrastar dichos datos en la tabla 7.14.

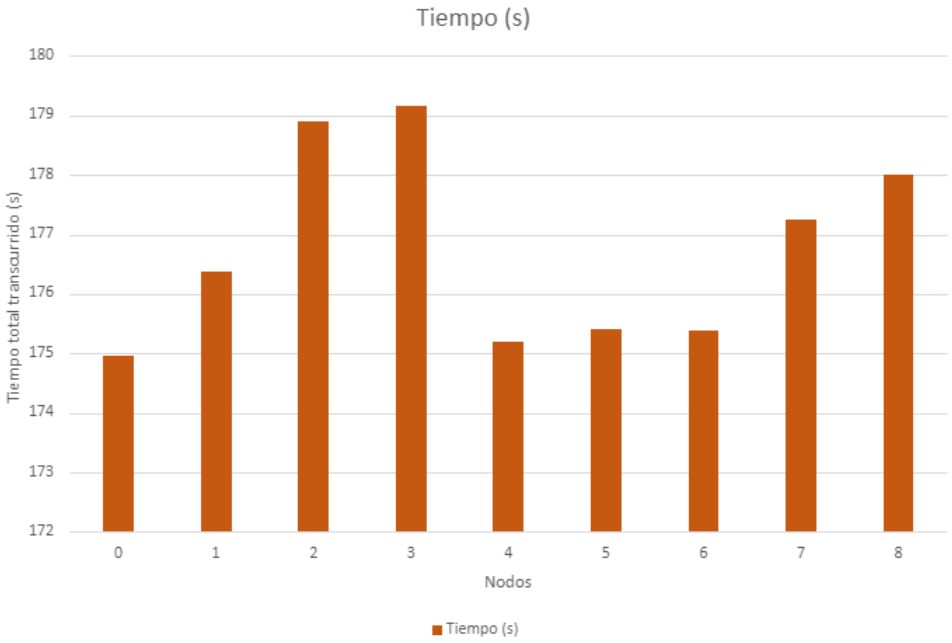


Figura 7.15: Envío/Recepción de mensajes en MUSCOP.

Cuadro 7.14: Tiempo de total en completar la misión.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Tiempo (s)	174,965	176,384	178,897	179,165	175,183	175,406	175,378	177,241	178,002
Tiempo medio (s)				176,735					
Tiempo (m)	2,916083333	2,939733333	2,981616667	2,986083333	2,919716667	2,923433333	2,922966667	2,954016667	2,9667
Tiempo medio (m)				2,945					

Para concluir, ambas versiones de MUSCOP aportan ciertas prestaciones, pero cabe destacar que la optimización ha aportado ciertas mejoras bastante considerables en todos los ámbitos. La tasa de envíos es prácticamente la misma, ya que solo hay una pequeña varianza de menos de 10 mensajes. Por lo tanto, se puede considerar que, para este sector el multisalto ha conseguido adecuarse correctamente. Por otra parte, la tasa de recepción es mucho más elevada en la versión original. En este punto se puede apreciar la buena función de multisalto respecto al filtrado de mensajes, ya que ha conseguido reducir considerablemente las recepciones aliviando así el propio canal. Por último, los tiempos medios de vuelo muestran que la mejora aplicada sobre MUSCOP consigue reducir el transcurso de la misión unos 45 segundos, aproximadamente, siendo esto un gran avance. La comparación puede ser apreciada en la tabla 7.15.

Cuadro 7.15: Comparativa entre ambas versiones de MUSCOP.

	Original	**Multisalto**
Media de Mensajes Enviados	117,4	124,667
Media de Mensajes Recibidos	392,33	227,556
Tiempo medio (s)	176,735	132,028
Tiempo medio (m)	2,945	2,23

Una vez analizados todos y cada uno de los experimentos, se puede concluir que el desarrollo de la idea ha sido correctamente implementado. La adición de esta nueva capa al simulador ha conseguido reducir la tasa de recepción y los tiempos de vuelo. Además, a pesar de que la tasa de envío haya sido mayor, simplemente ha aumentado de forma ínfima debido al proceso de encaminamiento. Por lo tanto, la implementación de los filtrados, el encaminamiento, y el piggybacking, se pueden considerar completamente funcionales. Seguidamente, el descubrimiento de vecinos y la construcción de grafos en base a estos ha sido un rotundo éxito.

Capítulo 8

Conclusión y Trabajos Futuros

Para finalizar, en este último capitulo se exponen las conclusiones del proyecto desarrollado. Además, se adjuntan una serie de trabajas futuros que pueden ser sumamente interesantes para futuros alumnos que busquen realizar una tesis en este área.

8.1. Conclusión

Ardusim es un simulador extremadamente revolucionario que nos permite desarrollar nuestros propios protocolos, misiones y complementos (como el detector de colisiones). Debido a esto, surge la necesidad de añadir e implementar nuevas funcionalidades que puedan ser beneficiosas para el simulador. Una de estas necesidades tiene que ver con el problema de los nodos ocultos vigente en el simulador, es decir, las comunicaciones entre todo el enjambre tiene que surgir dentro de un rango determinado ya que, más allá de este, no hay comunicación posible. Con este problema vigente nace la idea desarrollada en esta memoria, una capa que permita las comunicaciones multisalto y que, así, se aumente la escalabilidad y funcionalidad.

En la memoria se ha visto como opera la nueva capa. En un primer lugar, para emplear el multi-salto, se hace un descubrimiento de vecinos para así tener unas nociones básicas de la topología de red; tras esto, mediante el uso de grafos, tanto dirigidos como no dirigidos, se busca completar la construcción del grafo para así tener una visión completa y acíclica de la misma.

Cuando se forma la red, la nueva capa permite al enlace de comunicaciones tener distintos mecanismos de filtrado que permiten agilizar la transmisión de mensajes. Además, no solo hay un mejor filtrado, sino que, por otra parte, los mensajes pueden acumularse siempre y cuando compartan un objetivo

común. Con estas dos implementaciones el enlace recibe una menor tasa de mensajes, provocando así un alivio en el propio canal o, mejor dicho, provoca una menor congestión del canal, permitiendo que el tráfico pueda fluir mejor.

La experimentación ha corroborado que el desarrollo de la idea ha sido implementado correctamente. Observando los resultados, mediante el uso de MUSCOP, se ha visto como la nueva capa es capaz de crear una red coherente a los resultados esperados y, además, ha demostrado reducir la cantidad de mensajes que se envían y reciben en cada nodo. Sin embargo, no solo ha agilizado el tráfico, sino que ha conseguido reducir el transcurso de la misión, reduciendo los tiempos totales de cada nodo.

A continuación, se ha demostrado que la nueva capa es operativa sobre MUSCOP, pero también se ha realizado un análisis riguroso sobre su versión original. Los experimentos realizados han permitido concluir que la mejora aplicada ha superado con creces a la versión original de MUSCOP, es decir, la nueva capa multisalto ha permitido unas mayores prestaciones en todos los ámbitos al compararse con la capa de comunicaciones actual, además de las ventajas obvias que ofrece el multi-salto, ya que aumenta la distancia máxima que se puede cubrir con el enjambre.

Para concluir, se puede afirmar rotundamente que el desarrollo e implementación de la idea ha sido un rotundo éxito.

8.2. Trabajos Futuros

El ámbito científico de los drones es un campo extremadamente extenso que, con el paso del tiempo, ha ido creciendo a pasos agigantados. Por lo tanto, como trabajos futuros, puede haber una gran variedad de posibilidades a implementar sobre el simulador. En la siguiente lista se muestran unas cuantas ideas bastante interesantes para posibles trabajos:

- Optimización del simulador. En estos momentos el simulador está siendo adaptado poco a poco para mejorar sus prestaciones. Por lo tanto, una vez haya sido mejorado, sería una buena idea dedicarse a optimizar ciertas partes para así agilizar las simulaciones y solucionar errores menores.

- Desarrollo de un protocolo capaz de implementar inteligencia artificial para el reconocimiento de catástrofes. Actualmente el simulador dispone de un protocolo capaz de procesar imágenes con una cámara. La idea sería juntar este protocolo con inteligencia artificial para poder desarrollar una nueva variante.

- Adición de nuevos tipos de aeronaves sobre el simulador. Hoy en día el simulador solo dispone de un tipo de nave, siendo este el UAV, y sería interesante considerar nuevos dispositivos para así poder expandirse hacia nuevos campos.
- Aplicación de nuevos patrones de comunicación. De momento la comunicación entre los nodos es realizada mediante el uso de un patrón maestro-esclavo, pero sería interesante implementar un patrón de consenso sobre el enjambre.

Bibliografía

[1] Francisco Fabra y col. «ArduSim: Accurate and real-time multicopter simulation». En: *Simulation Modelling Practice and Theory* 87 (2018), págs. 170-190. ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2018.06.009. URL: https://www.sciencedirect.com/science/article/pii/S1569190X18300893.

[2] Liguo Weng y col. «Immune network-based swarm intelligence and its application to unmanned aerial vehicle (UAV) swarm coordination». En: *Neurocomputing* 125 (2014). Advances in Neural Network Research and Applications Advances in Bio-Inspired Computing: Techniques and Applications, págs. 134-141. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2012.06.053. URL: https://www.sciencedirect.com/science/article/pii/S0925231213001653.

[3] Francisco Fabra y col. «MUSCOP: Mission-Based UAV Swarm Coordination Protocol». En: *IEEE Access* 8 (2020), págs. 72498-72511. DOI: 10.1109/ACCESS.2020.2987983.

[4] Carlos Sampedro y col. «A flexible and dynamic mission planning architecture for UAV swarm coordination». En: *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2016, págs. 355-363. DOI: 10.1109/ICUAS.2016.7502669.

[5] Liang Hong y col. «Toward Swarm Coordination: Topology-Aware Inter-UAV Routing Optimization». En: *IEEE Transactions on Vehicular Technology* 69.9 (2020), págs. 10177-10187. DOI: 10.1109/TVT.2020.3003356.

[6] Jiansong Miao y col. «Secrecy Energy Efficiency Maximization for UAV Swarm Assisted Multi-Hop Relay System: Joint Trajectory Design and Power Control». En: *IEEE Access* 9 (2021), págs. 37784-37799. DOI: 10.1109/ACCESS.2021.3062895.

[7] Davide Caputo y col. «Genetical swarm optimization of multihop routes in wireless sensor networks». En: *Applied Computational Intelligence and Soft Computing* 2010 (2010).

[8] Alexis Pospischil. «Multihop routing of telemetry data in drone swarms». En: International Foundation for Telemetering. 2017.

[9] Francisco Fabra y col. «A Distributed Approach for Collision Avoidance between Multirotor UAVs Following Planned Missions». En: *Sensors* 19.10 (2019). ISSN: 1424-8220. DOI: 10.3390/s19102404. URL: https://www.mdpi.com/1424-8220/19/10/2404.

[10] Jamie Wubben y col. «Accurate Landing of Unmanned Aerial Vehicles Using Ground Pattern Recognition». En: *Electronics* 8.12 (2019). ISSN: 2079-9292. DOI: 10.3390/electronics8121532. URL: https://www.mdpi.com/2079-9292/8/12/1532.

[11] Francisco Fabra y col. «Automatic system supporting multicopter swarms with manual guidance». En: *Computers & Electrical Engineering* 74 (2019), págs. 413-428. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2019.01.026. URL: https://www.sciencedirect.com/science/article/pii/S0045790618323577.

[12] Dimitrios Michail y col. «JGraphT—A Java Library for Graph Data Structures and Algorithms». En: *ACM Trans. Math. Softw.* 46.2 (mayo de 2020).

[13] Kairanbay Magzhan y Hajar Mat Jani. «A review and evaluations of shortest path algorithms». En: *International journal of scientific & technology research* 2.6 (2013), págs. 99-104.