UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Algoritmos de Enrutamiento Basados en Inteligencia de Enjambre: Caracterización, Simulación e Implementación

Trabajo de graduación presentado por Erick Abraham Roquel Pisquiy
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2022

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

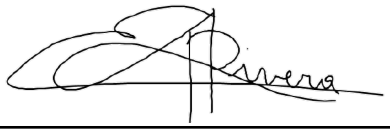
Algoritmos de Enrutamiento Basados en Inteligencia de Enjambre: Caracterización, Simulación e Implementación

Trabajo de graduación presentado por Erick Abraham Roquel Pisquiy
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

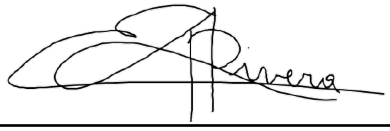
2022

Vo.Bo.:

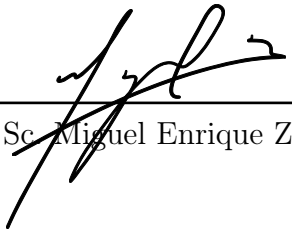
(f) 

Dr. Luis Alberto Rivera Estrada

Tribunal Examinador:

(f) 

Dr. Luis Alberto Rivera Estrada

(f) 

M. Sc. Miguel Enrique Zea Arenales

(f) 

M. Sc. Pedro Iván Castillo Rivera

Fecha de aprobación: Guatemala, 6 de enero de 2022.

Prefacio

La inspiración para este trabajo de graduación se basa en la curiosidad por enlazar dos áreas de interés, a saber, las redes computacionales y la inteligencia de enjambre. Esto involucra la combinación de conceptos de ambas disciplinas, siendo algunos las capas (*layers*) del lado de las redes y los algoritmos distribuidos por parte de la inteligencia de enjambre. Resulta de mucho interés cómo la combinación de estos conceptos permite resolver problemas para las nuevas clases de redes que el mundo está utilizando.

Agradezco a Dios por toda su bondad, que ha incluido mi trayecto por esta Universidad. También agradezco a la Fundación Juan Bautista Gutiérrez por ser un soporte invaluable durante mi carrera de muchas manera. También estoy agradecido por tener la oportunidad de desarrollar este trabajo en la Universidad del Valle, donde he conocido a docentes que me han ayudado a desarrollar habilidades y adquirir conocimiento que me será útil durante toda mi vida. En particular, quiero agradecer a los docentes y compañeros que me han ayudado a comprender que no se realizan esfuerzos por comprender nuevos temas solo por la utilidad que esto pueda tener, sino porque simplemente es satisfactorio conocer más.

Índice

Lista de figuras

Lista de cuadros

Resumen

Este trabajo de investigación se dedica al estudio de los algoritmos de enrutamiento basados en inteligencia de enjambre. En particular, se busca caracterizarles, evaluar su comportamiento por medio de simulaciones y a nivel de implementación física.

Estos algoritmos revelan tener características muy diferentes a los algoritmos de ruteo clásicos, como el enfoque en modelos del *trip time* en lugar de número de saltos. Otra diferencia importante es la información compartida, ya que las rutas no se comparten de manera directa sino por medio de estigmergia. Sin embargo, la manera de plantear el problema también es minimizar una función de costo.

Una de las plataformas más versátiles para la simulación de estos algoritmos es RMASE, la cual permite definir a conveniencia los diferentes componentes que integran el *stack* de protocolos que definen el comportamiento de la red. También es notable la facilidad que esta presta para obtener interacciones diversas con el usuario por medio de la interfaz gráfica.

El módulo transceptor MRF24J40MA junto con la plataforma de desarrollo que ofrece el Arduino NANO resultan ser un punto de partida excelente para implementar estos protocolos en una red física. Se validó que el comportamiento a nivel físico y en las simulaciones concuerdan con lo que se espera de las características de estos algoritmos.

Abstract

This research project is about the study of swarm-based routing protocols, specifically about their main features and challenges. Additionally, the goal is to evaluate their performance through simulations and by implementing those ideas in a physical network.

These routing algorithms reveal characteristics different from the classical routing methods. One example is the focus on a model of the trip time instead of hop count in order to define routes. Another difference is that information is not shared directly between routing devices but indirectly by updating data structures inside each node. Nevertheless, the goal is still to minimize a cost function.

RMASE is a flexible routing modeling application for these algorithms. It allows to define custom routing components in the protocol stack, thus defining the network's behaviour. It's also simple to develop interaction with the user through the graphical user interface features available.

MRF24J40MA is a certified radio transceiver module that along with Arduino NANO constitute a great platform for the implementation of swarm-based routing protocols. It was verified that the behaviour expected from the simulations and results gathered from the physical network where according to the principles set by the swarm-based algorithm.

CAPÍTULO 1

Introducción

El diseño de protocolos de enrutamiento es un área de interés en la actualidad, dada la alta demanda de conectividad que se tiene en varios niveles (comercial, de ocio, investigación, etc.). Las nuevas maneras en las que se interconectan dispositivos (por ejemplo IoT o redes inalámbricas de sensores) presentan retos únicos a ser evaluados por el diseñador del protocolo de comunicación.

Las características de ciertos conjuntos de animales han inspirado la rama de estudio de inteligencia de enjambre, la cual a su vez ha dado nacimiento a diversos algoritmos. Estos algoritmos permiten abordar problemas de optimización por medio de técnicas como la estigmergia, los algoritmos distribuidos, etc. Estas características resultan deseables para la resolución de problemas con muchos individuos con recursos limitados.

Antnet es uno de los algoritmos de enrutamiento que aprovecha las cualidades de inteligencia de enjambre para abordar el problema de enrutamiento. Este problema pretende hallar un camino o serie de nodos que un paquete debe atravesar para llegar a un destino en particular. En este trabajo de investigación se desea investigar las cualidades de estos algoritmos orientados a redes computacionales, simularlos para evaluar su rendimiento e implementarlos físicamente.

Es de interés poder determinar porqué es necesario utilizar estos algoritmos novedosos, es decir, qué prestaciones brindan en comparación con los algoritmos clásicos de enrutamiento. También es de interés determinar las nuevas cualidades y retos que introducen estos algoritmos al diseño y evaluación de nuevos protocolos de comunicación.

CAPÍTULO 2

Antecedentes

2.1. Algoritmos de enrutamiento basados en inteligencia de enjambre y redes inalámbricas de sensores

El surgimiento de nuevas tecnologías demanda actualizaciones en diversos niveles del proceso de comunicación entre dispositivos. El artículo en [1] expone el caso de las redes inalámbricas de sensores (*Wireless Sensor Network* o WSN), que incorporan dispositivos con características especiales, diseñados para desplegarse en cientos o miles de unidades. Las características de estos son: pequeños, de bajo costo, con capacidad de sensar alguna cantidad física, con capacidad limitada de procesamiento de la información, con limitaciones energéticas, y con capacidad limitada de comunicación inalámbrica. Las características de estas redes de sensores son diferentes de las redes convencionales, donde suele tenerse mayor capacidad de procesamiento, fuentes de alimentación, y mejores capacidades de comunicación inalámbrica (o bien comunicación por medio de cables) [2]. Esto ha generado la necesidad de idear nuevos protocolos que permitan procesar la información y enviarla de un nodo a otro de manera más efectiva.

El protocolo de enrutamiento cumple un rol fundamental en este proceso de comunicación, ejerciendo como el sistema nervioso de la red. La labor de este protocolo es identificar o descubrir una o más rutas conectando un par de nodos, bajo un conjunto definido de restricciones. Es decir, este protocolo permite que los paquetes de información lleguen desde un emisor hasta un receptor dentro de la red, atravesando diversos nodos en el proceso. Lo que se desea es hallar la mejor ruta que cumpla con estas especificaciones. Esto permite plantear el problema de enrutamiento como uno de optimización (el algoritmo de Dijkstra permite plantear esto como la minimización de una suma, ver ejemplo de aplicación en [3]).

Las comunidades de animales que cooperan en grandes cantidades para lograr un objetivo poseen características similares con estas redes de sensores, en particular que están conformados por entes individuales sencillos sin control centralizado. El comportamiento de estos animales ha inspirado el surgimiento de diversos algoritmos de optimización, que ha

su vez ha dado surgimiento a nuevos protocolos de enrutamiento, que se llaman basados en inteligencia de enjambre [1].

2.2. Comparación y rendimiento de diferentes protocolos de enrutamiento

El artículo en [4] expone diferentes algoritmos de enrutamiento de esta línea, así como las características de estos y su clasificación. También busca comparar los diferentes algoritmos bajo diferentes métricas: *throughput*, latencia, y consumo de energía. Los resultados generados se dieron para redes inalámbricas de sensores (WSN por sus siglas en inglés), y se concluyó que el mejor protocolo es función de la aplicación de uso. Sin embargo, uno de los mejores en términos generales resultó ser *Flooded piggyback ant routing*. Este protocolo está basado en *Ant Colony*, implementando ideas como combinar el uso de hormigas de *forward* y hormigas de datos. El surgimiento de protocolos posteriores a la publicación de este artículo y la falta de literatura que reúna sus características de implementación más importantes da lugar a nueva literatura que actualice el tema y cubra estos detalles para facilitar la evaluación e implementación de estos protocolos.

2.3. Descripción matemática de los protocolos de enrutamiento

El artículo en [5] expone las expresiones matemáticas detrás de los diferentes algoritmos como *Basic Ant Based Routing for WSN*, *Sensor driven and Cost-aware ant routing* y *Flooded Forward ant routing*. También se propone un ambiente de simulación basado en *MATLAB* llamado *Routing Modeling Application Simulation Environment* (RMASE). Bajo esta plataforma se obtienen las métricas de latencia, porcentaje de pérdidas, consumo de energía y eficiencia de energía. Estas métricas se obtuvieron bajo dos escenarios, el primero es estático, donde el nodo de *sink* (al cual van dirigidos los datos en una red WSN) no varía en el tiempo. El segundo escenario es dinámico, donde el tráfico va dirigido a diferentes nodos conforme transcurre el tiempo. Los autores propusieron un algoritmo que denominaron *Improved Energy Efficient Ant Based Routing* (IEEABR) que resultó tener buen rendimiento comparado con los otros protocolos analizados tanto para el escenario estático y dinámico. Esta plataforma de simulación, así como algunas otras propuestas que se mencionan más adelante, permiten dar un punto de partida para su modificación o extensión, de manera que puedan cubrirse más protocolos, más parámetros configurables por el usuario o más resultados generados.

2.4. Problemas a resolver y nuevas líneas de investigación

El diseño de nuevos protocolos debe considerar aspectos que la mayoría de protocolos existentes no ha considerado o lo ha hecho pobremente. El principal tema según [4] es la seguridad de la red en cuanto a soportar ataques análogos a DDOS (*Distributed Denial of*

Service, satura una conexión con tráfico ilegítimo), o bien el acceso a información sensible por parte de intrusos. La mayoría de protocolos no maneja conceptos como encriptación de los datos o control de flujo, por lo que son blanco fácil frente a estos ataques. Por otro lado, el artículo en [6] propone utilizar ciertos conceptos de centralización para mejorar el rendimiento de la red. Por ejemplo, en las redes WSN con un solo nodo *sink* se pueden tener ciertos nodos intermedios denominados *relay nodes* que filtren el tráfico para que solo se propaguen los paquetes que son relevantes que lleguen al *sink*. Esta literatura expone ideas interesantes que pueden ponerse a prueba en una implementación física para evaluar su desempeño y desafíos.

Un ejemplo de un nodo de una red inalámbrica de sensores se ilustra en Figura 1. Los autores de [7] exponen como el 'siguiente internet', el Internet de las Cosas, demanda nuevas abstracciones y herramientas para los desafíos que presenta. Esto es evidente al considerar este ejemplo, ya que el pequeño tamaño del dispositivo le permite aplicaciones que no tienen dispositivos como teléfonos móviles. Sin embargo, el diseño de los programas debe adaptarse a las limitaciones para generar aplicaciones que tengan un uso práctico.

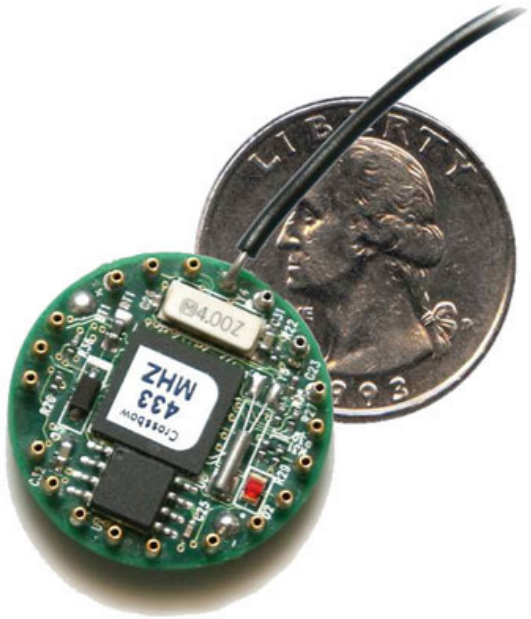


Figura 1: Ejemplo de nodo de una red inalámbrica de sensores [7].

CAPÍTULO 3

Justificación

La inteligencia de enjambre se entiende como una estrategia para la resolución de problemas que parte de la interacción de unidades de procesamiento sencillas. Se basa en principios como multiplicidad, estocasticidad y aleatoriedad, inspirado en el comportamiento de conjuntos de animales que colaboran en torno a un objetivo común [8]. Esta estrategia se ha aplicado en diferentes áreas, desde la robótica (un caso de uso se expone en [9]) hasta el enrutamiento de los circuitos súper complejos que surgen en nanoelectrónica (ver [10]). El área de redes computacionales también ha adoptado esta estrategia para situaciones que se adecúan, es decir, donde los nodos son sencillos, con recursos limitados y se tienen requerimientos como resiliencia ante cambios en la topología, escalabilidad y consumo eficiente de energía.

Las redes inalámbricas de sensores (WSN por sus siglas en inglés) son un ejemplo de redes que aprovechan los algoritmos de enrutamiento basados en inteligencia de enjambre. El artículo en [1] expone cinco áreas principales de aplicación: monitoreo del ambiente, hogar, salud, comercio y aplicaciones militares. El atractivo principal de estas redes es su rápido despliegue y bajo costo, razón por la cual también se han adaptado sensores químicos o biológicos para aplicaciones de automatización industrial o detección de enfermedades. Dadas estas aplicaciones y la importancia que los protocolos de enrutamiento basados en inteligencia de enjambre han adquirido, el propósito de este trabajo de investigación es proveer una base de entendimiento acerca de estos protocolos, cómo simular su comportamiento, y generar una guía de implementación de estos. Finalmente, se ilustrarán sus funcionalidades y retos de implementación con la implementación de un caso de uso.

A través de este proyecto se busca introducir esta nueva rama de aplicación de inteligencia de enjambre a las líneas de investigación de la Universidad del Valle, como una instancia interesante, nueva y útil de los sistemas de telecomunicaciones. La idea es que futuras investigaciones puedan partir de los conocimientos, herramientas y consejos de implementación expuestos en este documento para potencialmente generar proyectos de alto impacto con cientos o miles de nodos computacionales que colaboren en torno a un objetivo común. Esto se logrará a través del compendio y organización de investigaciones previas de estos algo-

ritmos, la extensión o modificación de plataformas de simulación generadas anteriormente, y la exposición de criterios para la selección de componentes concretos para implementar estas redes.

CAPÍTULO 4

Objetivos

Objetivo general

Establecer una base teórica y una plataforma de implementación de los algoritmos de enrutamiento basados en inteligencia de enjambre.

Objetivos específicos

- Caracterizar los protocolos de enrutamiento basados en inteligencia de enjambre.
- Desarrollar una plataforma de simulación para los protocolos de enrutamiento más importantes.
- Generar metodologías de implementación de estos protocolos.
- Implementar un caso de uso de estos protocolos en una red inalámbrica física.

CAPÍTULO 5

Alcance

El alcance de este proyecto se divide en tres áreas: caracterización, desarrollo de software, e implementación física de algoritmos seleccionados. Para el primer apartado se elige un algoritmo de enrutamiento basado en inteligencia de enjambre como caso de estudio, se analizan las ideas fundamentales de este así como los detalles a considerar para implementarlo.

Para el segundo apartado, se elige una plataforma de simulación como punto de partida. Esto se hace con base en características como robustez y facilidad de uso. Luego se define un modo de operación para poder escribir nuevos componentes de enrutamiento que posteriormente se pueden poner a prueba por medio de simulaciones.

Para el tercer apartado se eligen dispositivos para formar una red inalámbrica que implemente el algoritmo basado en inteligencia de enjambre elegido. Luego de la implementación se extraen resultados análogos a los generados vía simulaciones y se verifica que concuerden con los principios de estos algoritmos.

CAPÍTULO 6

Marco teórico

6.1. Conceptos generales de redes computacionales

Una red o *network* es un conducto que conecta dos o más computadoras u otros dispositivos. Estos dispositivos pueden ser computadoras de escritorio, dispositivos móviles (tabletas, teléfonos inteligentes), y muchos otros tipos. La diversidad de dispositivos y maneras en las que este conducto puede construirse generan una gran diversidad de tipos de redes. Hay dos tipos básicos según [11]: - LAN (*local area network*): consta de cualquier grupo de computadoras en una sola red delimitada geográficamente. - WAN (*wide area network*): es una red que conecta varias LANs sobre distancias geográficas amplias.

Un ejemplo de una red LAN es la de una casa, mientras que una WAN podría ser la que conecta la red de una casa con internet.

6.1.1. Una alegoría para la comunicación en una red

Una pregunta fundamental al hablar de redes de computadoras es ¿cómo se envía información de un dispositivo a otro dentro de una red? Esto, de acuerdo con [11], funciona muy parecido a la manera en que el servicio postal traslada cartas o paquetes de un emisor a un destinatario. La información original (lo que se desea hacer llegar) se encapsula dentro de un sobre o envoltorio de acuerdo con cierto estándar o protocolo de tal manera que este puede atravesar todas las entidades o nodos necesarios hasta llegar al destino. Este proceso puede ser complejo y para proveer un mejor entendimiento del mismo se han desarrollado modelos que identifican diferentes capas de funcionalidad que se usan en la comunicación. Existen varios modelos populares, como el OSI (*Open Systems Interconnection*) o TCP/IP, pero para este trabajo se utilizará el modelo de cinco capas (*Five-Layer Model*). Estos modelos definen de manera teórica la funcionalidad existente en cada capa, pero no especifican cómo debería realizarse esto; los protocolos, por su parte, definen formal y completamente cómo debe desarrollarse la funcionalidad requerida en la capa especificada.

6.1.2. El modelo de cinco capas

Este modelo identifica cinco capas de funcionalidad que existen en el proceso de comunicación dentro de una red [12]. Estas son:

- Capa 5: Aplicación (*Application*). Los protocolos en esta capa especifican detalle como la codificación de los datos, su compresión, encriptación, manejo de sesiones, etc.

- Capa 4: Transporte (*Transport*). Define qué protocolo de capa de aplicación debe ser usado para procesar los datos al ser recibidos en segmentos. También asigna números de puerto específicos que distinguen entre paquetes de información que serán manejados por las diferentes aplicaciones. Los protocolos disponibles en esta capa son TCP y UDP.

- Capa 3: Red (*Network*). En esta capa se añade un encabezado para brindar direcciones lógicas de destino y fuente a los paquetes. El protocolo más utilizado es IP.

- Capa 2: Enlace de datos (*Data Link*). Recibe los paquetes y añade un encabezado para formar tramas. En el encabezado se añade una dirección de capa 2 también conocida como dirección física. Esta capa realizar una detección de errores que descarta las tramas con errores por medio de un *checksum* que se añade al final de la trama. La porción de información que se añade al final se conoce como *trailer*. En esta capa también se convierten los datos a unos y ceros lógicos de manera que puedan ser entendidos por la capa inferior.

- Capa 1: Física (*Physical*). Convierte los bits en señales eléctricas y los envía por el medio físico, lo cual puede hacerse por cable coaxial, fibra óptica, de manera inalámbrica, etc. En esta capa se definen parámetros como: voltajes, baudajes, conectores físicos, etc.

Este trabajo de investigación se centra en los protocolos de capa de red. Como se ejemplificó en la sección anterior, al momento que un conjunto de datos se quiere enviar de un emisor a un destino, se inicia en la capa superior o de aplicación. La información de esta capa se encapsula en la capa inferior, de transporte, al añadirle el encabezado correspondiente. Cada capa inferior encapsula la información de la capa superior hasta llegar al nivel de señales eléctricas. De modo que el destinatario pueda acceder a la infomación que se quería enviar originalmente, se realiza un proceso inverso. Este proceso se ilustra en Figura 2. En una red convencional, hay dispositivos que se dedican a la operación en diferentes capas, por ejemplo, un *router* o enrutador se enfoca en la capa de red y los *switches* en la capa den enlace de datos.

Figura 2: Proceso de comunicación entre dos puntos en una red [11].

6.1.3. Protocolos principales para comunicación en redes

El modelo de cinco capas define diferentes niveles de funcionalidad en el proceso de comunicación entre computadoras. En cada capa se han desarrollado protocolos populares que implementan esta funcionalidad. De acuerdo con [12] se tiene una familia o *suite* de protocolos estándar, que para la capa de aplicación contiene protocolos como DNS (sistema de nombres de dominio), HTTP (transferencia de hiper-texto), etc. Otros protocolos para el resto de capas son:

- IP (*Internet Protocol*): se utiliza para enviar datos de una computadora a otra por medio de identificación lógica. Cada dispositivo tiene al menos una dirección IP que los distingue de manera única del resto de máquinas. Este protocolo es de capa de red.
- UDP (*User Datagram Protocol*): es un protocolo muy sencillo para el envío de datos a nivel de capa 4. Trabaja bajo un estándar de mejor esfuerzo, que no provee notificación de entrega, detección de errores o procedimientos de recuperación de segmentos. Sin embargo, puede ser más rápido en comparación con TCP.
- TCP (*Transmission control protocol*): es un protocolo robusto para el envío de datos en capa 4. Provee acuse de recibido, verificación de errores, y procedimientos de recuperación de segmentos. También asegura una entrega de cada segmento tal que puede subir en orden a capa de aplicación.
- *Ethernet*: es un conjunto de estándares que define reglas acerca del formato de las tramas, así como la manera en que las computadoras se comunican con otras sobre el medio físico. También actúa en la capa física definiendo conectores, tipos de cable, distancias permitidas para los cables, etc.

6.1.4. Clases de direccionamiento en redes

Las capas 3 y 2 añaden direcciones a los paquetes (y tramas) que cumplen propósitos, distintos y necesarios. De acuerdo con [13], el direccionamiento lógico identifica de manera única a los dos dispositivos que se están comunicando, de manera que estas direcciones no

cambian mientras los datos viajan por la red. El emisor y receptor de esta comunicación también se conocen como *endpoints*. El direccionamiento físico identifica las paradas, nodos, o equipos que conforman la ruta desde el emisor hasta el receptor; estas direcciones cambian a lo largo de la ruta para identificar cada nodo.

6.2. Protocolos de enrutamiento

6.2.1. El problema de enrutar

El enrutamiento, ruteo o *routing* es una tarea que se da a nivel de capa 3 (y capa 4 y 5 en enfoques más generales). De acuerdo con [14] este consiste en hallar la ruta o conjunto de saltos (*hops*) que deben tomar los paquetes desde la fuente hasta el destino. También puede formularse como hallar el conjunto de nodos que debe tomar un paquete para llegar desde un dispositivo A hasta uno B. Esta es una tarea global, mientras que el *forwarding* es una tarea local que consiste en mover paquetes desde la entrada de un dispositivo capa 3 hasta la salida apropiada que lo conecta con el siguiente dispositivo de esa capa en la ruta designada. El objetivo final del enrutamiento es llenar la tabla de rutas, las cuales especifican la salida que debe tomar un paquete en el dispositivo con base en el destino (pueden tomarse otros parámetros en cuenta en visiones más generales). El objetivo del *forwarding* es mover paquetes de la entrada a la salida apropiada contando con la tabla de rutas como punto de partida.

6.2.2. Clasificación

En términos generales, se clasifican los protocolos de ruteo como clásicos y basados en inteligencia de enjambre. Los protocolos de ruteo clásicos son los protocolos convencionales diseñados para redes cableadas o inalámbricas como: *Routing Information Protocol* (RIP) , *Interior Gateway Protocol* (IGRP), *Open Shortest Path First* (OSPF), etc. Los protocolos de ruteo basados en inteligencia de enjambre están basados en el comportamiento colectivo de especies o insectos sociales, los cuales encuentran soluciones a problemas distribuidos sin ningún control o coordinación centralizado. Un ejemplo ilustrativo es el mecanismo de *Ant Colony*, que de hecho es sobre el cual se basan la mayoría de protocolos de ruteo de este tipo. Existen otras clasificaciones [15]:

1. Por el momento cuando se computan las rutas
    - Proactivos: estos protocolos computan las rutas previo a necesitarlas. Estas rutas son almacenadas en tablas en cada nodo. También se tienen rutas a cada nodo vecino. Estos protocolos cuentan con ciertas limitaciones, como alto tiempo de asentamiento o convergencia (*settling time*) y escalabilidad limitada.
    - Reactivos: estos encuentran las rutas cuando las requieren. La cantidad de *overhead* disminuye, esto es la cantidad de paquetes necesarios por mantenimiento y descubrimiento de rutas. El retardo suele ser mayor con estos protocolos ya que se deben computar las rutas cada vez que se requieran.

- Híbridos: en este, los nodos toman un enfoque proactivo en determinado número de saltos, mientras que más allá de ellos computa las rutas de forma reactiva.

2. Por la manera en que la red se organiza

   - De topología plana: trata todos los nodos de igual manera, se implementa en redes donde todos los nodos tiene la misma funcionalidad.
   - Basados en jerarquía: estos se aplican en redes heterogéneas donde algunos nodos tienen mayores capacidades que otros. También pueden elegirse nodos "cabeza" en algunas redes que agrupan los nodos en cúmulos (*clusters*). Los nodos cabeza son los que se comunican con el nodo de salida de la red (también conocido como *sink*).
   - Basados en la ubicación: en este tipo de enrutamiento, los nodos tienen la capacidad de saber su ubicación actual utilizando algún protocolo de localización. Esta información ayuda a mejorar el procedimiento de ruteo y permite a la red brindar servicios adicionales.

3. Por la forma de operación

   - *Multi-path*: estos generan varias rutas para cada destino, de manera que se pueda proveer balanceo de cargas, bajo retardo y mejor rendimiento de la red como resultado. También se obtiene redundancia en caso de que algún nodo desaparezca o falle. Este tipo de protocolos son preferidos en redes densas (con muchos nodos), pero requieren más *overhead* para determinar qué rutas continúan disponibles para enviar paquetes.
   - Basados en peticiones (*Query-based*): este tipo de ruteo se denomina iniciado por el receptor. Los nodos solo envían datos en respuesta a peticiones generadas por el nodo de destino.
   - Basados en negociación: en este, los nodos vecinos comparten la información de los recursos disponible, de manera que las decisiones de las rutas a tomar se toman luego de un proceso de negociación sobre esta base.
   - Basados en calidad de servicio (*QoS-based*): estos protocolos tratan de descubrir rutas desde la fuente hacia el destino que satisfagan métricas relacionadas a cierto servicios, estas pueden ser *throughput*, retardo, etc.
   - Ruteo coherente: en este los nodos llevan a cabo un mínimo de procesamiento en los datos antes de transmitirlos hacia los otros nodos. Se tiene un nodo que combina los datos provenientes de diferentes nodos para pasarlo al nodo destino.

4. Por el criterio utilizado

   - Basados en la dirección (*address centric*): bajo este paradigma, el único criterio para la selección de la ruta y la acción a llevar a cabo con el paquete es la dirección de destino. Esto quiere decir que se buscará en la tabla de rutas la dirección de destino para saber a dónde redirigir el paquete, sin importar su contenido.
   - Basados en datos (*data centric*): este paradigma toma en cuenta no solo la dirección de destino, sino también el contenido del paquete. De esta manera, por ejemplo, si el contenido del paquete no cumple un criterio por el cual sea relevante propagarlo a través de la red, entonces se descarta (*drop*).

6.3. Inteligencia de enjambre

La inteligencia de enjambre es un disciplina basada en los modelos del comportamiento de animales sociales, como hormigas, abejas, avispas, termitas, etc. Un enjambre es una configuración de individuos o agentes (decenas, cientos o miles) cuya operación ha convergido a un objetivo común. Esta inteligencia es el comportamiento complejo, auto-organizado, flexible y robusto de este grupo, que se basa en reglas sencillas [8]. Algunas características generales de esta disciplina son:

- Muchos individuos y una sola 'mente'. distribuida.
- Los agentes interactúan unos con otros y con su ambiente.
- Los agentes siguen reglas sencillas para realizar acciones.
- Las decisiones se toman de manera descentralizada.
- Adaptativo.
- La aleatoriedad permite la exploración de alternativas, permitiendo alcanzar un objetivo de optimización.
- Las decisiones que toman los agentes se basan en un balance entre un modelo simple de percepción-reacción, y un modelo aleatorio.
- Aplica conceptos inspirados en la naturaleza.

El emplear este tipo de inteligencia provee ventajas que incluyen aspectos como robustez, donde el enjambre puede cumplir el objetivo frente a perturbaciones internas (algunos agentes fallan en completar su tarea) y externas. Otra ventaja es que estos algoritmos distribuidos pueden escalarse fácilmente a muchos agentes, sin necesidad de un control central, y donde las soluciones a los problemas planteados son emergentes en lugar de definidos de antemano. También se considera que los cambios en la red pueden propagarse rápidamente y las operaciones de los agentes son inherentemente ejecutadas en paralelo [16]. El uso de esta tipo de inteligencia también presenta algunos retos, por ejemplo, el comportamiento puede tornarse difícil de predecir a partir de las reglas sencillas que rigen a cada individuo. Otro fenómeno relacionado es que cambios pequeños en estas reglas sencillas puede resultar en un comportamiento totalmente diferente a nivel de grupo.

Los principios en los que se basa la inteligencia de enjambre son:

- Proximidad: los agentes deben ser capaces de interactuar con su ambiente, respondiendo a los estímulos que provienen de este e influyendo a su vez en el mismo. Esto se conoce como estigmergia, donde los agentes agentes interactúan entre sí de manera indirecta por medio de modificaciones en el ambiente.
- Calidad: los agentes deben ser capaces de evaluar la calidad de una solución con base en diferentes factores

- Estabilidad: la población no debe cambiar su modo de operación con base en cambios ambientales (las reglas fundamentales que los rigen).
- Auto-organización: este principio tiene diferentes bases:
  - Retroalimentación positiva: se dice que un proceso tiene este tipo de retroalimentación cuando se amplifican los efectos de una pequeña perturbación. La regla que rige este comportamiento puede enunciarse como "A produce más de B, el cual a su vez produce más de A" [17]. En el contexto de inteligencia de enjambre, esto puede ayudar a que se explore una cantidad razonable de soluciones.
  - Retroalimentación negativa: esto se manifiesta en un proceso cuando los resultados de un cambio actúan precisamente para reducirlo. El uso de esto en el contexto es permitir que el enjambre converja a una solución. Esto puede enunciarse como "A produce B, que a su vez decrementa la producción de A".
  - Balance: se tiene una tensión continua entre retroalimentación positiva y negativa para asegurar que se explora una suficiente cantidad de soluciones y se converja a una. Este fenómeno se manifiesta en los mercados, fenómenos a nivel de células humanas, etc [17].

6.4. Algoritmos basados en inteligencia de enjambre

Existen diversos algoritmos basados en los principios de esta disciplina. Estos regularmente se plantean como algoritmos de optimización. Un problema de optimización puede plantearse de la siguiente manera:

$$\begin{aligned} &\text{mín}\,(f(\mathbf{x})) \\ &\text{Sujeto a } \mathbf{x} \in S \subset \mathbb{R}^N \end{aligned} \tag{1}$$

Donde

- $f(\mathbf{x})$ es la función objetivo o de costo.
- $\mathbf{x}$ es un vector N-dimensional cuyas componentes son las variables de decisión.
- S es la región viable del problema o espacio de trabajo.

6.4.1. *Ant Colony Optimization* (ACO)

Las hormigas exhiben un comportamiento social complejo que puede corroborarse al observar los "caminos" que forman al realizar tareas como transportar comida de un lugar a otro. El aspecto más sorpresivo de este comportamiento es que se ha encontrado que estos caminos son lo que se denomina "camino más corto" (*shortest path*) lo cual se ha determinado que las hormigas alcanzan por medio del uso de diferentes mecanismos. Uno de

los principales es el uso de feromonas para comunicarse entre ellas. ACO es un algoritmo de optimización inspirado en este comportamiento, y ha resultado ser unos de los más exitosos de su tipo. Los biólogos han mostrado que el comportamiento complejo de estas colonias puede ser explicado, en gran manera, usando solo comunicación por estigmergia y la idea para los algoritmos basados en estas colonias es aplicar una forma de estigmergia artificial para coordinar sociedades de agentes artificiales [18].

Los autores de [19] diseñaron un experimento para observar el uso de la estigmergia por parte de hormigas argentinas. En este experimento se tenía una conexión desde el nido hacia la fuente de aliento con una bifurcación donde ambas rutas tenían el mismo largo. Al inicio ninguna ruta tenía feromona, por lo que las hormigas elegían de manera aleatoria un camino. Sin embargo, una ligera cantidad mayor de hormigas transitando un camino hacía que eventualmente la colonia convergiera a un camino. Esto es un ejemplo de aplicación de la retroalimentación positiva. En un segundo experimento uno de los caminos era más largo y el mecanismo de comunicación de las hormigas las permitía converger al camino más corto. Esto puede describirse como sigue:

1. Al inicio, ningún camino tiene más feromona, por lo que la elección del camino es al azar (con igual probabilidad para ambos caminos).
2. Un camino es más corto, por lo que las hormigas que lo eligen son las primeras en llegar al otro lado y empezar su retorno.
3. El depósito de feromona en el regreso genera que esta se acumule más rápidamente en el camino más corto. Este es un punto que resulta crucial para que la colina elija el camino más corto. Los biólogos han demostrado que las hormigas que solo depositan feromona en su camino de ida (*forward*) o de regreso (*backward*).
4. La retroalimentación positiva hará que, eventualmente, la colonia converja al camino más corto.

A través de este experimento se notó que los mecanismos que dirigieron la convergencia hacia el camino más corto son la estigmergia, retroalimentación positiva y el largo diferencial de caminos (*differential path lenght*). Un punto importante es que, a pesar que hay un camino con el doble de largo del otro, no todas las hormigas usan el más corto, sino que un porcentaje pequeño utiliza el largo. Este uso de la aleatoriedad puede interpretarse como un tipo de exploración.

El objetivo es definir algoritmos que pueden ser usados para resolver los problemas de mínimo costo, en redes representadas por grafos complejos. Sea G un grafo conectado estático $G = (N, A)$, donde N es el conjunto de $n = |N|$ nodos y A es el conjunto de arcos no-dirigidos que los unen. Los dos puntos entre los cuales se quiere establecer el camino de mínimo costo se llaman fuente (*source*) y destino (*destination*). De acuerdo con los experimentos llevados a cabo con hormigas reales y en simulaciones, no se puede hallar el camino de mínimo costo con un modelo sencillo donde la feromona de *forward* solo incrementa en cada iteración. Esto podría generar bucles o impedir que se resuelva el problema de optimización. Una mejora que se le hace a las hormigas artificiales es añadir el concepto de memoria para que estas puedan almacenar los caminos parciales que han atravesado [18]. Esto permite introducir comportamientos útiles para la resolución de los problemas planteados, como:

- Construcción probabilística de la solución basado en los rastros (*trails*) de feromona, pero sin los problemas generados por la feromona de *forward*.
- Camino de *backward* determinista con eliminaciones de bucles.
- Evaluación de la calidad de las soluciones generadas para determinar la cantidad de feromona a depositar.

Otra adición importante a las consideraciones es la evaporación de la feromona. Esta característica permite aumentar la exploración de las soluciones, lo cuál es útil para hallar el mínimo global y resolver el problema de actualizar el camino más corto al ocurrir cambios en la configuración del grafo.

6.4.2. *Particle Swarm Optimization* (PSO)

El algoritmo de optimización de enjambre por partículas (PSO por sus siglas en inglés) es un algoritmo de búsqueda aleatoria basado en población desarrollado por Eberhart y Kennedy en 1995. Está inspirado en el comportamiento social de parvadas y cardúmenes. Ha sido utilizado en varios campos dada su fácil implementación y relativamente pocos parámetros. Este algoritmo se clasifica dentro de los métodos de búsqueda directa, donde en cada iteración un conjunto de puntos de prueba se genera y se comparan los valores de la función objetivo en estos puntos con los de la iteración previa. Esta información se usa para determinar el siguiente conjunto de puntos de prueba. También se considera como un método de búsqueda aleatoria ya que incorpora estrategias estocásticas en el proceso de búsqueda. Esto quiere decir que se utilizan características aleatorias o probabilísticas embebidas en el algoritmo [20].

Este método se considera de tipo de búsqueda localizada aleatoria ya que considera el proceso de búsqueda previo para generar los siguientes puntos de prueba. Una manera de implementar estos métodos es en la forma de un método metaheuristico, el cual es un proceso iterativo que combina inteligentemente diferentes conceptos para explorar y explotar el espacio de trabajo. En este contexto, exploración se entiende como la búsqueda global y explotación como búsqueda local. Hay muchos otros métodos metaheurísticos, uno de ellos se aborda en la siguiente sección.

Una variante básica de este algoritmo comienza con una población (llamada *swarm* o enjambre) de soluciones candidatas (llamadas partículas). Estas partículas es movilizan a través del espacio de soluciones de acuerdo con expresiones simples para la velocidad y posición de cada una. Estas expresiones toman en cuenta la mejor posición que ha alcanzado la partícula y la mejor que ha alcanzado el enjambre. El proceso se repite esperando poder hallar el máximo global (no está garantizado).

6.5. Comunicación inalámbrica

La comunicación inalámbrica es una manera común de implementar las redes que utilizan protocolos de enrutamiento orientados a inteligencia de enjambre. El tipo de comunicación

inalámbrica que se utiliza para estas redes se denomina de corta distancia, lo cual incluye generalmente varias de estas características [21]:

- Potencia de radio frecuencia en el orden de varios microwatts hasta 100 milliwatts.
- Rango de comunicación de centímetros hasta cientos de metros.
- Principalmente aplicaciones en interiores.
- Antenas omnidireccionales embebidas.
- Terminales relativamente pequeñas.
- Precio relativamente bajo.
- Operación sin necesidad de adquirir licencias (en términos del espectro a utilizar).
- Operación en la banda UHF (300 Mhz a 3GHz).
- Receptores y transmisores energizados vía baterías.

En el Cuadro 1 se describen las características principales de la comunicación inalámbrica de corto alcanza conforme su uso actual. La tendencia actual es el desarrollo de protocolos y *hardware* que permitan comunicación a altas tasas de datos para distancias de varios metros. Este es el caso de *Bluetooth* y *Zigbee*.

Aplicación	Frecuencias	Características
Sistemas de seguridad y alarmas	300-500, 800, 900 MHz	Facilidad de instalación
Accesorios de computadora	UHF	Alta tasa de datos, muy corta distancia
RFID	100kHz - 2.4GHz	Muy corta distancias, *transponder* pasivo
WLAN	2.4, 5-6 GHz	Ancho espectro, alta tasa de datos, modulación
WPAN	2.4 GHz	tasa de datos media, bajo costo
Micrófonos/audífonos inalámbricos	VHF, UHF	Modulación analógica, precio moderado
Domótica	UHF	Transmisores miniatura
IoT	UHF, 2.4GHz	Monitoreo sin intervención humana

Cuadro 1: Información extraída de [21].

6.5.1. Modelos para la propagación de ondas por radio

Un modelo de propagación por radio determina la potencia de una señal transmitida en un punto particular del espacio para todos los transmisores del sistema [22]. De acuerdo

con la sección anterior, se vio que regularmente las antenas utilizadas son omnidireccionales, por la que la potencia transmitida solo depende de la distancia desde el emisor. Un modelo sencillo para esta propagación está dado por:

$$P_{\text{rec, ideal}} = \frac{P_{\text{transmit}}}{1 + d^{\gamma}} \tag{2}$$

Donde d es la distancia desde el transmisor y $2 \leq \gamma \leq 4$ es un parámetro. Por otro lado, para incluir los factores aleatorios propios del entorno y el tiempo se modela lo siguiente:

$$P_{\text{rec}} = P_{\text{rec, ideal}}(d_{i,j})(1 + \alpha(i,j))(1 + \beta(t)) \tag{3}$$

Donde α y β son variables aleatorias con distribuciones normales $N(0, \sigma_a), N(0, \sigma_b)$, respectivamente. Bajo este modelo, α es estático y depende solo de las ubicación en el punto j relativo a la fuente i; por otro lado, β es dinámico, cambiando en el tiempo. Una manera sencilla de modelar la recepción correcta de ciertos datos transmitidos inalámbricamente es el uso de un umbral. Toda señal con potencia arriba del umbral se recibe correctamente. Se dice que una colisión ocurre si dos señales de transmisores distintos cumplen con esta condición. Otros modelos de propagación por radio son los de Rayleigh (tanto para propagación como recepción), que consideran aspectos como la relación señal a interferencia y ruido (SINR).

CAPÍTULO 7

AntNet

AntNet es un algoritmo de enrutamiento desarrollado por Gianni A. Di Caro basado en la plataforma de ACO, explotando los mecanismos detrás de la manera en que las colonias de hormigas hallan el camino más corto entre dos puntos. Se considera como el primer algoritmo de este tipo basado en ACO, cuyos orígenes datan de 1997. Exhibe propiedades como el de funcionar en una manera totalmente distribuida, con la capacidad de adaptarse a los cambios en las condiciones de la red y siendo robusto frente a fallas generadas por los agentes involucrados [23]. A continuación se exponen las ideas principales y el detalle de implementación a nivel de algoritmo.

7.1. Características principales

AntNet es un protocolo de enrutamiento pensado para redes inalámbricas y cuenta con las siguientes características (según la clasificación general expuesta anteriormente) [18]:

- Reactivo: en intervalos de tiempo fijos Δt se envían hormigas de *forward* hacia destinos elegidos aleatoriamente (según una distribución que se discute adelante) dentro de los nodos de interés a nivel de aplicación.
- De topología plana: no se hace distinción entre los nodos, es decir, estos pueden actuar tanto como usuario (*host*) y dispositivo de enrutamiento en cualquier instante de tiempo.
- Basado en calidad de servicio (*QoS-based*): las rutas se eligen con base en una estructura de feromona $\mathcal{T}_i$ que determina el cuán eseable es la ruta del siguiente salto j, desde el nodo i para un paquete dirijido al destino d. La feromona, a su vez, cambia en el tiempo con base en la calidad de la ruta que conecta la fuente con el destino.

Esta calidad se determina por medio de un modelo de la red que almacena cada nodo i, que se identifica con $\mathcal{M}_i$. Este modelo almacena el promedio $\mu_{i,d}$, la varianza $\sigma^2_{i,d}$ y el mejor valor $\mathcal{W}_{i,d}$ del timepo de ida (*trip time*) desde el nodo i hasta el destino d.

- *Address Centric*: las tablas de rutas se llenan solo con base en la dirección de destino, de manera que para elegir el siguiente salto no se realiza algún análisis sobre la información para determinar la próxima acción a realizar con el paquete.

Otras clasificaciones para este protocolo son:

- *Shortest-path*: este protocolo da resolución al problema de enrutar desde la perspectiva de un solo par fuente-destino. Otros protocolos tienen un enfoque de ruteo óptimo (*optimal routing*) donde las rutas se eligen con el objetivo de minimizar una función de costo que toma en cuenta los flujos existentes en toda la red (entendiendo flujo el conjunto de paquetes con la misma 2-tupla fuente-destino).
- *Distance-vector*: este paradigma tiene que ver con la información que cada nodo utiliza para construir la tabla de rutas. Esta consiste de 3-tuplas en la forma (destino, distancia, siguiente salto), la cual se define para todos los destinos en la red y para todos los nodos en la vecindad del nodo. Esto quiere decir que para computar las rutas solo se toma en cuenta el *fitness* de enviar el paquete al siguiente salto, tomando en cuenta el nodo de destino.

7.2. Ideas fundamentales

Este protocolo es una extensión directa de la plataforma ACO y tiene como ideas principales las siguientes [24]:

- En intervalos regulares, y conjuntamente con el tráfico (esto quiere decir a nivel de implementación: en las mismas colas (*queues*)) se envían hormigas de *forward* a destinos elegidos según la siguiente distribución:

$$p_{sd} = \frac{f_{sd}}{\sum_{i=1}^{n} f_{si}} \tag{4}$$

  Donde P_{sd} es la probabilidad de enviar una hormiga desde el nodo s hacia el destino d, f_{si} es una medida del flujo desde el nodo s hacia el nodo i (en bits o paquetes) y f_{sd} es el caso particular para el nodo d. Esto quiere decir que, mientras más paquetes de tráfico se envíen a cierto destino, es más probable que se envíen hormigas para poder evaluar nuevamente la mejor ruta hacia ese destino.
- Las hormigas artificiales actúan simultáneamente y de manera independiente, comunicándose de manera indirecta (vía estigmergia) por medio de las feromonas que leen y escriben en la estructura $\mathcal{T}_i$ de cada nodo.
- Cada hormiga artificial busca un camino de mínimo costo que una la fuente con el destino.

- Cada hormiga se mueve, paso a paso, hacia el destino eligiendo el siguiente salto por medio de una política que utiliza:

  1. Información local de la estructura $\mathcal{T}_i$.
  2. Información local heurística del problema: en este caso la información utilizada es la cantidad de bits en cada una de las colas.

- Mientras se mueven, las hormigas recolectan información del tiempo que les toma desplazarse de nodo a nodo, así como el estatus de la congestión y los identificadores del camino seguido

- Al llegar al destino, las hormigas regresan al nodo fuente por el mismo camino de la hormiga de *forward* pero en la dirección opuesta.

- Durante el camino de regreso, los modelos locales del estatus de la red y la estructura de feromona de actualizan como función del camino seguido y su 'calidad'.

- Al llegar de regreso a la fuente, las hormigas son borradas del sistema.

7.3. Estructuras de datos

7.3.1. Matriz de feromona

La matriz de feromona se representa con $\mathcal{T}_i$ donde i denota el identificador del i-ésimo nodo. Los elementos τ_{ijd} de la matriz representan qué tan deseable es para un paquete dirigido hacia el destino d tomar como siguiente salto el vecino j. Se definen las entradas τ_{ijd} como normalizadas según la siguiente ecuación:

$$\sum_{j \in \mathcal{N}_i} \tau_{ijd} = 1,\ d \in [1, n] \text{ y } \forall i \tag{5}$$

Esto quiere decir que la sumatoria de toda la feromona para un mismo destino d debe ser igual a 1, lo cual aplica para todos los n destinos, y para cualquiera de los i-ésimos nodos. El conjunto $\mathcal{N}$ representa los vecinos del nodo i. En términos de implementación estos son los vecinos que son alcanzables por el medio físico (ya sea cableado o inalámbrico) directamente (en la misma LAN). Esta estructura se representa en Figura 3, donde se ve que según la normalización definida y la disposición de columnas y filas dad, la sumatoria de las entradas de una misma columna debe ser 1.

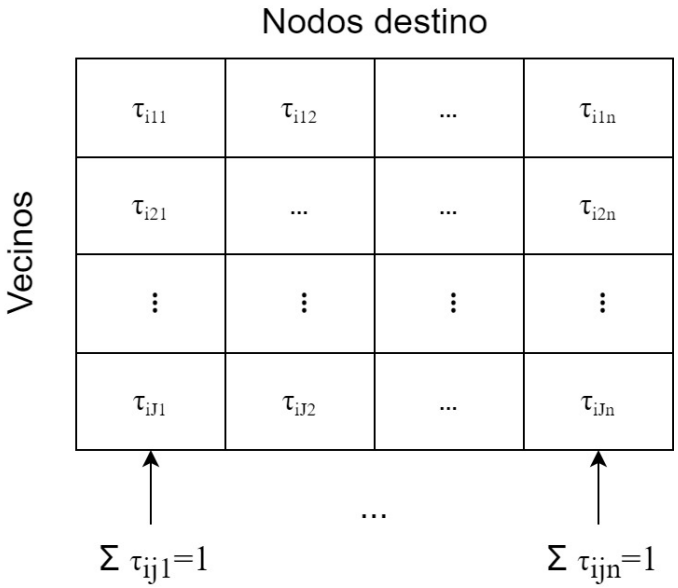


Figura 3: Matriz $\mathcal{T}_i$ de feromonas con J = $|\mathcal{N}_i|$ (modificado de [18]).

7.3.2. Modelo de la red

Esta estructura se encarga de modelar la condición de la red desde la perspectiva de la pareja (fuente, destino). Es útil recordar que AntNet es un algoritmo de tipo *shortest-path*, lo que quiere decir que se busca optimizar individualmente cada ruta, en lugar de tomar en cuenta algún indicador global de rendimiento. En este caso la estructura $\mathcal{M}_i$ donde i es el identificador de cada nodo se encarga de almacenar esta información, por medio de tres variables:

- $\mu_{i,d}$ es el promedio del tiempo de ida desde el nodo i hacia el destino d, y $\sigma^2_{i,d}$ la varianza del mismo.
- $\mathcal{W}_{i,d}$ contiene el mejor valor del tiempo de ida en la ventana consistente de w muestras. Se define $w = 5 * c/\sigma$ para que en el caso de $c = 1$ concuerden los conjuntos de observaciones para los modelos, como se verá más adelante.

Las ecuaciones para la actualización de estos valores son:

$$\begin{aligned} \mu_{i,d} &\leftarrow \mu_{i,d} + \varsigma(o_{i\to d} - \mu_{i,d}) \\ \sigma^2_{i,d} &\leftarrow \sigma^2_{i,d} + \varsigma((o_{i\to d})^2 - \sigma^2_{i,d}) \end{aligned} \tag{6}$$

Donde $o_{i\to d}$ es el último valor medido del tiempo de ida. Al considerar la ecuación de diferencias para la actualización de la media se ve que es un filtro pasa-bajas paramétrico del tiempo de ida. Haciendo el cambio de variables $x[n] = o_{i\to d},\ y[n] = \mu_{i,d}$ y definiendo $X(z) = \mathcal{Z}(x[n]),\ Y(z) = \mathcal{Z}(y[n])$ se tiene:

$$H(z) \equiv \frac{Y(z)}{X(z)} = \frac{\varsigma z}{z + \varsigma - 1} \tag{7}$$

Este filtro es estable para $\varsigma \in (0, 2)$. La estimación común para el número de observaciones que se toman en cuenta para el valor actual de $\mu_{i,d}$ es $5/\varsigma$ [18]. La respuesta al escalón del sistema para diversos valores de ς se presenta en Figura 4, donde se ve que el valor de la media responde de manera más rápida a los cambios conforme el parámetro ς se elige más grande y viceversa. Por otro lado, en Figura 5 se ve que para $\varsigma < 1$ el comportamiento es el esperado, de pasa baja, mientras que para valores más grandes de sigma se tienen mayores ganancias a frecuencias más altas. Esto último no parece una característica deseable para la media. Se ve que al actualizarse la media en tiempo discreto podrían haber cambios tan rápidos en la red que se genere el fenómeno de *aliasing* y se le dé el mismo tratamiento que para algún cambio más suave, por lo que siempre se tiene la cota del tiempo de muestreo. Un punto interesante aquí es que el tiempo de muestro es el mismo tiempo de ida ya que al mismo tiempo que se recibe una muestra se genera la medición.

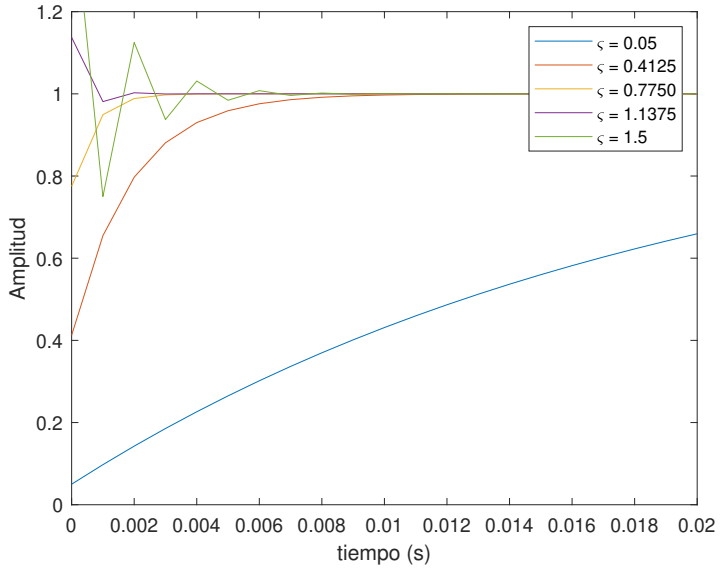


Figura 4: Respuesta al escalón del sistema para actualización de la media con $T_s = 1$ms.

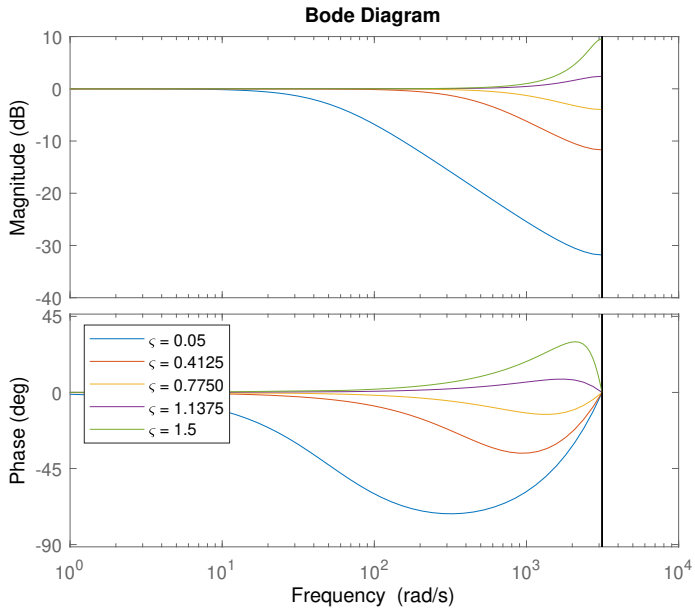


Figura 5: Diagramas de magnitud y fase del sistema para actualización de la media con $T_s = 1$ms.

7.4. Construcción de las soluciones

7.4.1. Procesos ejecutados en el camino hacia el destino

Considerando las ideas fundamentales del algoritmo, surge la necesidad de decidir hacia qué destino enviar las hormigas de *forward*. Para esto se toma en cuenta la medida del flujo de datos (a nivel de aplicación) hacia un destino en particular, lo cual puede medirse en bits o número de paquetes y se representa como f_{sd}. El índice s representa la fuente, y d el destino. La probabilidad de enviar una hormiga de *forward* desde una fuente s hacia un destino d es:

$$p_{sd} = \frac{f_{sd}}{\sum_{i=1}^{n} f_{si}} \tag{8}$$

Esto quiere decir que la probabilidad de que se inicie un proceso de descubrimiento y evaluación de rutas hacia un destino depende de la carga de trabajo existente hacia ese destino. La hormiga de *forward* se denota como $F_{s \to d}$

Estas hormigas deben almacenar dos variables por cada nodo en el camino organizadas en una pila que almacena una 2-tupla en cada registro que contiene el identificador del nodo visitado, y el tiempo que le tomó llegar ahí desde que salió de la fuente s. Cabe destacar la ligera distinción en los términos de ruta y camino, donde el primero se refiere regularmente al conjunto red destino-próximo salto, y el segundo al conjunto de saltos hasta el destino. Esto se representa con $S_{s \to d\,,i}$, donde i representa el i-ésimo nodo en la ruta hacia d. Luego surge la pregunta de cómo se forman las rutas hacia el destino. Esto se resuelve considerando

el proceso de aprendizaje que se ha llevado a cabo por medio de la actualización de las estructuras $\mathcal{T}_i$ y $\mathcal{M}_i$; también se considera un valor heurístico relacionado con la demanda actual del recurso físico hacia el j-ésimo vecino. Este último valor se calcula como sigue:

$$\eta_{ij} = 1 - \frac{q_{ij}}{\sum_{l=1}^{|\mathcal{N}_i|} q_{il}} \tag{9}$$

Donde q_{ij} es el tamaño de la cola (en bits o paquetes) desde el nodo i hacia el vecino j. Esto permite que para la elección de cada salto (y, por extensión, para la formación de las soluciones) se tome en cuenta no solamente el aprendizaje llevado a cabo sino la situación instantánea de la red al hacer el envío de la hormiga. La probabilidad de que una hormiga de *forward* en un nodo i con destino d elija al vecino j es:

$$P_{ijd} = \frac{\tau_{ijd} + \alpha\eta_{ij}}{1 + \alpha(|\mathcal{N}_i| - 1)} \tag{10}$$

Se ve que el valor de α pondera la importancia del valor heurístico en la elección del próximo salto. Algunos otros detalles existen para las hormigas de forward.

- Al detectar un ciclo (la hormiga regresa a un nodo visitado previamente) se remueve del stack de memoria todo lo relacionado con los nodos del ciclo.
- Si el ciclo es más grande que la mitad de la vida (TTL) de la hormiga, esta se desecha. Este parámetro se conoce como max_life.

7.4.2. Procesos ejecutados en el camino de vuelta

Cuando la hormiga de *forward* llega al destino se genera una hormiga de *backward* que se denota $B_{d\to s}$, a la cual se transfiere toda la memoria del agente que la creó, y esta última se desecha. Las hormigas de *backward* siguen el mismo hacia la fuente pero en colas de mayor prioridad porque el objetivo es que la información que se ha recolectado se propague rápidamente. En este punto, es posible realizar la actualización de las estructuras de memoria $\mathcal{T}_i$ y $\mathcal{M}_i$ relacionadas con el destino d. Esto se ejemplifica en Figura 6, donde se ve que en el camino de regreso, esta actualización es posible para las hormigas $i = 1, 2, 3$ con respecto del destino $d = 4$. El caso para $i = 1$ era la intención principal, sin embargo los sub-caminos (*subpaths*) formados en el proceso permiten la actualización de los nodos intermedios conforme:

- Se actualizan $\mathcal{M}_i(\mu_{i4}, \sigma^2_{i4}, \mathcal{W}_{i4})$ y las entradas de la matriz $\mathcal{T}_i$: τ_{ij4} para $i = 1, 2, 3$. En Figura 6 este es el caso para el camino principal y los sub-caminos en rojo, todos con destino $d = 4$.
- Hay otros sub-caminos que se generan donde podrían considerarse como destinos los nodos intermedios $d' = s + 1, ..., d - 1 = 2, 3$, estos se ilustran en verde. Es importante recordar que AntNet es *address centric*, lo que quiere decir que las rutas (y el proceso para hallarlas) se basa en la dirección de destino. Como la dirección de destino real

no era alguno de los nodos d' hay que proceder con cuidado para realizar las actualizaciones de las estructuras $\mathcal{M}_i(\mu_{id'}, \sigma^2_{id'}, \mathcal{W}_{id'})$, $\tau_{ijd'}$. El criterio utilizado es que si la métrica obtenida es 'buena', entonces se ha recorrido a costo cero un camino que es útil para actualizar estas estructuras. Por el contrario, si le métrica es pobre, entonces no puede confiarse en la medición para esta actualización ya que es potencialmente sub-óptima. La calidad del tiempo de ida medido se determina por medio de:

$$o_{i \to d'} < \mu_{id'} + I(\mu_{id'}, \sigma_{id'}) \tag{11}$$

Donde I denota un estimado de un intervalo de confianza para $\mu_{id'}$.

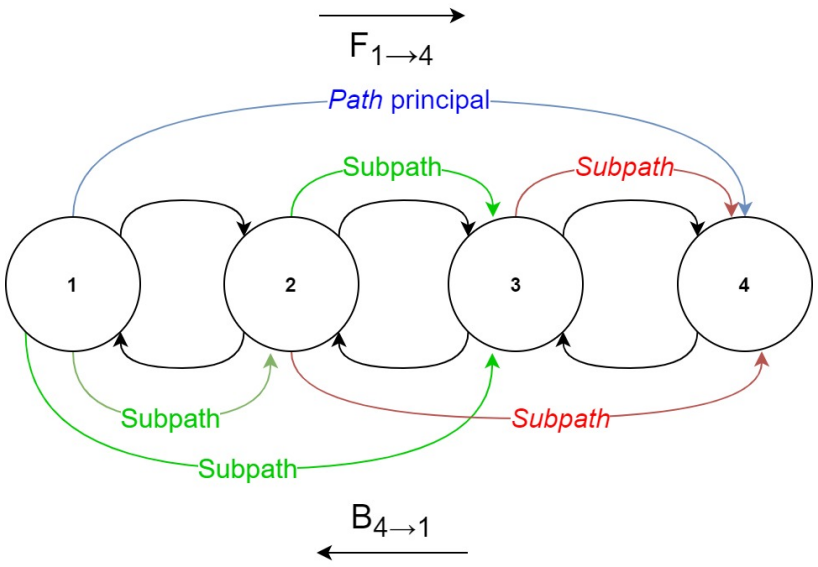


Figura 6: Ilustración del flujo de las hormigas de *forward* y *backward* con los caminos (*pathsy* sub-caminos (*subpaths*)) formados en el proceso (modificado de [18]).

7.4.3. Actualización de las estructuras de datos

La estructura $\mathcal{M}_i$ almacena un modelo paramétrico del tiempo que se demora un paquete en viajar del nodo i hacia el nodo d'. Este modelo es necesario ya que juega un papel fundamental en el proceso de actualización de la feromona, y la posterior instalación de los resultados en las tablas de rutas. Esto es porque para una nueva medición $o_{i \to d'}$ es necesario determinar si es 'buena' o 'mala'. Su actualización se describe en la ecuación (6). Qué tanto se va a recompensar una solución con base en su calidad es un problema de aprendizaje reforzado [18]. El objetivo es: para una nueva medición del tiempo de ida actualizar la feromona $\tau_{ifd'}$ donde f es el vecino que se tomó para llegar al destino d', y actualizar las feromonas $\tau_{ijd'}$ con j un elemento de la vecindad $\mathcal{N}_i$ y $j \neq f$ tal que se mantenga la ecuación 5. El refuerzo $r \equiv r(o, \mathcal{M}_i)$ con $0 < r \leq 1$ es función del modelo de la red hacia el destino d' y el tiemo de ida actual o. Con base en esto se tiene:

$$\tau_{ifd'} \leftarrow \tau_{ifd'} + r \cdot (1 - \tau_{ifd'}) \tag{12}$$

$$\tau_{ijd'} \leftarrow \tau_{ijd'} - r \cdot \tau_{ijd'},\ j \neq f \tag{13}$$

La regla dada por la ecuación (12) pretende proveer siempre un aumento de feromona para la ruta recorrida, y favorecer na rápida explotación de rutas nuevas con buena calidad. Esto permite aprovechar el efecto de camino diferencial ya que, no solamente entra en juego el refuerzo, sino el efecto de camino diferencia: la tasa de llegada desde un vecino f hasta el nodo de fuente s. De manera, correspondiente, la regla dada por la ecuación (5) pretende evaporar las otra feromonas por normalización para el mismo destino.

Definiendo el refuerzo r

El problema de asignación de crédito de aprendizaje reforzado puede resolverse de una manera sencilla asignando $r = \text{cte}$, de manera que se aproveche únicamente el efecto de camino diferencial discutido anteriormente. Esto sería muy parecido al comportamiento de las hormigas naturales. Otra manera es utilizar la información del modelo de la red que se tiene, en conjunto con el tiempo de ida medido para realizar la asignación de crédito. Los autores de [18] proponen utilizar lo siguiente:

- Definir límites I_{sup}, I_{inf} de un intervalo de confianza para μ. Se asigna $I_{inf} = \mathcal{W}$ e $I_{sup} = \mu + z(\sigma/\sqrt{w})$ donde es útil recordar que w es el número de observaciones sobre las cuales se computó $\mathcal{W}$. El parámetro z ayuda a definir un intervalo de confianza por medio de [18]:

  $$z = \frac{1}{\sqrt{1-v}} \tag{14}$$

  Donde $0 < v < 1$ provee el nivel de confianza seleccionado (véase que confirme $v \to 1,\ z \to \infty$ y la cota superior del intervalo se dispara.

- Con base en lo anterior, se propone formar r' con dos términos, uno que favorezca un buen tiempo de ida o en el contexto del intervalo de confianza, y otro que lo haga relativo a $\mathcal{W}$:

  $$r' = c_1(\frac{\mathcal{W}}{o}) + c_2(\frac{I_{sup} - I_{inf}}{(I_{sup} - I_{inf}) + (o - I_{inf})}) \tag{15}$$

- Finalmente se utiliza una función de *squash* para normalizar r' y favorecer buenos resultados, y saturar a cero las recompensas para malos resultados.

  $$\begin{aligned} r &= s(r')/s(1) \\ s(x) &= (1 + exp(\frac{a}{x \cdot |\mathcal{N}_i|}))^{-1} \end{aligned} \tag{16}$$

  Con esto se mantiene $|r| < 1$ con a real positivo.

7.5. Resumen de parámetros de AntNet

Símbolo	Significado	Cotas/valores comunes
Δt	Intervalo para envío de hormigas de *forward*	10ms
c	Fracción de observaciones para el cálculo de $\mathcal{W}$	1
ς	Peso relativo de la respuesta a cambios en la media del tiempo de ida	0.05
α	Peso del valor heurísitico para la selección del destino de hormigas de *forward*	$0 < \alpha < 1$
v	Intervalo de confianza para μ	$0 < v < 1$
c_1, c_2	Pesos para actualización de r	0.5, 0.5

Cuadro 2: Parámetros de AntNet. Los valores comunes se dan referidos a [18].

7.6. Variantes de AntNet

El corazón del algoritmo basado en las colonias de hormigas para redes computacionales se describió en las secciones anteriores. Las variantes para el algoritmo tienen que ver con las maneras en las que se seleccionan los parámetros (vía huerísticas e hiper-parámetros). También se tiene la definición de otras formas para capturar el modelo de la red de una forma 'fiel' o representativa del estado real de la congestión hacia los destinos. Otras variaciones importantes son para la forma en la que se recompensan las rutas con base en el modelo de la red y el tiempo de ida medido (el problema de aprendizaje reforzado).

En el siguiente capítulo se desarrolla una plataforma de simulación para el algoritmo AntNet que permita realizar la exploración en los aspectos descritos. También se pretende que sea posible plantear diferentes casos de uso del algoritmo, así como condiciones diversas (topologías, anchos de banda, potencias inalámbricas) para realizar pruebas.

CAPÍTULO 8

Desarrollo de la plataforma de simulación

La simulación de AntNet es un punto importante para caracterizar por completo este algoritmo y tener una perspectiva más amplia para predecir su comportamiento. El comportamiento de algoritmos distribuidos puede ser contra-intuitivo en ocasiones, por lo que es importante generar simulaciones para ver los resultados bajo escenarios concretos. Un ejemplo de esto es la aparición del problema de conteo al infinito en los algoritmos de vector-distancia como RIP (*Routing Information Protocol*) donde la actualización del costo de un enlace se propaga lentamente si es sub-óptimo. Esto genera un transiente no deseado con fluctuaciones en las rutas [25]. En este capítulo se describe el proceso de desarrollo de la plataforma de simulación, tomando en cuenta detalles como el proceso de selección del punto de partida y las características principales de su entorno de desarrollo.

8.1. Entradas y salidas

Para la plataforma de simulación se definen como entradas lo siguiente:

- Topología de la Red: cantidad de nodos y su disposición relativa.
- Parámetros de AntNet: estos se definen en el Cuadro 2. Se contempla además permitirle al usuario añadir o remover parámetros conforme se necesite.
- Eventos: capacidad para definir qué nodo ejecutará una labor de ruteo, enviando proactivamente paquetes para cumplir con objetivos e implementando reglas de asignación como que que se presenta en la ecuación (8).
- Parámetros comunes de la redes: direccionamiento de los nodos, anchos de banda, TTL (esto ya se implementa como max_life en AntNet), etc.

Para las salidas del sistema hay varios parámetros interesantes que se desearían obtener. Por un lado están las métricas clásicas de rendimiento para redes (latencia, pérdida de paquetes, *jitter*) y por otro lado las variables propias del algoritmo AntNet: gráficas de la feromona hacia un destino conforme avanza el tiempo, los modelos del tiempo de ida hacia los diferentes destinos, etc. Para este trabajo de investigación se estará evaluando en particular la feromona, como un indicador de la preferencia o asignación de calidad hacia una solución en particular.

8.2. Selección de la plataforma base

Existen varias maneras de partir, sin embargo el mismo autor de AntNet, Gianni A. Di Caro, ofrece un panorama general de las plataformas donde se han hecho implementaciones del algoritmo:

- OmNet++: es una plataforma de desarrollo basada en C++ para eventos discretos. Ha resultado útil para el desarrollo de simuladores de redes por medio de diversos proyectos. Originalmente diseñado para ambientes Linux, se desarrolló una versión para Windows. Muddassar Farooq llevó a cabo una implementación de AntNet bastante cercana algoritmo original [26].
- NS-2: es un simulador de eventos discretos diseñado para la investigación de redes computacionales, basado en C++ y provee soporte para la simulación de componentes de ruteo sobre redes cableadas e inalámbricas. Similar a OmNet++, se desarrolló originalmente para Linux y se adaptó posteriormente a Windows. Lavina Jain y Richardson Lima han creado versiones de AntNet para esta plataforma [27].
- RMASE: (*routing modeling application simulator environment*) los autores de [28] utilizaron esta herramienta para llevar a cabo la comparación de varios algoritmos de enrutamiento en Matlab. Se hizo tanto para algoritmos clásicos como para los basados en inteligencia de enjambre.

Luego de la realización de varias pruebas, se vio que la plataforma más amigable con un ambiente Windows es RMASE, que por otro lado está basada en Matlab. La ventaja de usar esta herramienta es la familiaridad que se tiene con el modo de operación de Matlab.

8.2.1. Generalidades de RMASE

Esta plataforma de simulación se desarrolló con el objetivo de comparar diferentes algoritmos para redes inalámbricas de sensores [29]. A su vez, RMASE se basa sobre Prowler (*probabilistic wireless network simulator*) provee una manera fácil de prototipar aplicaciones en ambientes inalámbricos, con sistemas distribuidos, desde la capa de aplicación hasta la capa física. El rol principal de Prowler es:

- Simular modelos de propagación de las ondas de radio: la implementación de la Universidad de Vanderblit permite simula transmisiones, recepciones y colisiones de tramas

por medio de los modelos que se discutieron en la sección del marco teórico (dependiente de la distancia, probabilístico, considerando fallas aleatorias, etc.).

- Implementar la capa de comandos y eventos a nivel físico: se implementa el modelo de TinyOS, el cual distingue entre eventos y comandos para el flujo de información. El modo de operación se ilustra en Figura 7 donde *Send_packet* es un comando de envío procedente de una capa superior y *Packet_Sent* es un evento que se sube a la capa superior luego de que el comando se ha completado. Como se ve, al inico se espera cierto tiempo y se evalúa si el canal de transmisión está en uso, si no lo está se empieza la transmisión, caso contrario se espera un tiempo *backoff_time* antes de volver a sensar el medio e intentar transmitir. Finalmente, la transmisión se realiza en un intervalo de tiempo *transmission_time*.

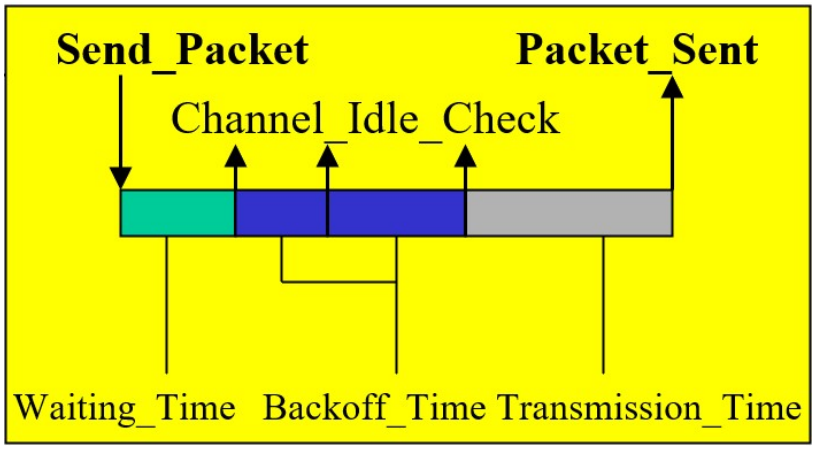


Figura 7: Implentación de la capa MAC de TinyOS de Prowler [30].

Para este trabajo de investigación se estará utilizando RMASE por su orientación a redes inalámbricas, simulando colisiones, tomando en cuenta la aleatoriedad del medio, etc. Con base en esto se pueden crear mejores modelos para redes con este tipo de limitantes o dinámica.

8.2.2. Arquitectura de RMASE

RMASE consta de tres componentes principales: un modelo de la topología de la red, una modelo de la aplicación y un modelo de rendimiento. Por otro lado, la arquitectura de RMASE está organizada (convenientemente) en capas. Esto se ilustra en Figura 8. Las capas superiores se acercan gradualmente al nivel de aplicación, mientras que las inferiores al nivel físico de la comunicación. Se considera que los eventos 'suben' o se propagan de capas inferiores a superiores (si se dan las condiciones especificadas por cada capa), mientras que los comandos 'bajan' o son transmitidos de capas superiores a inferiores conforme se necesite/desee.

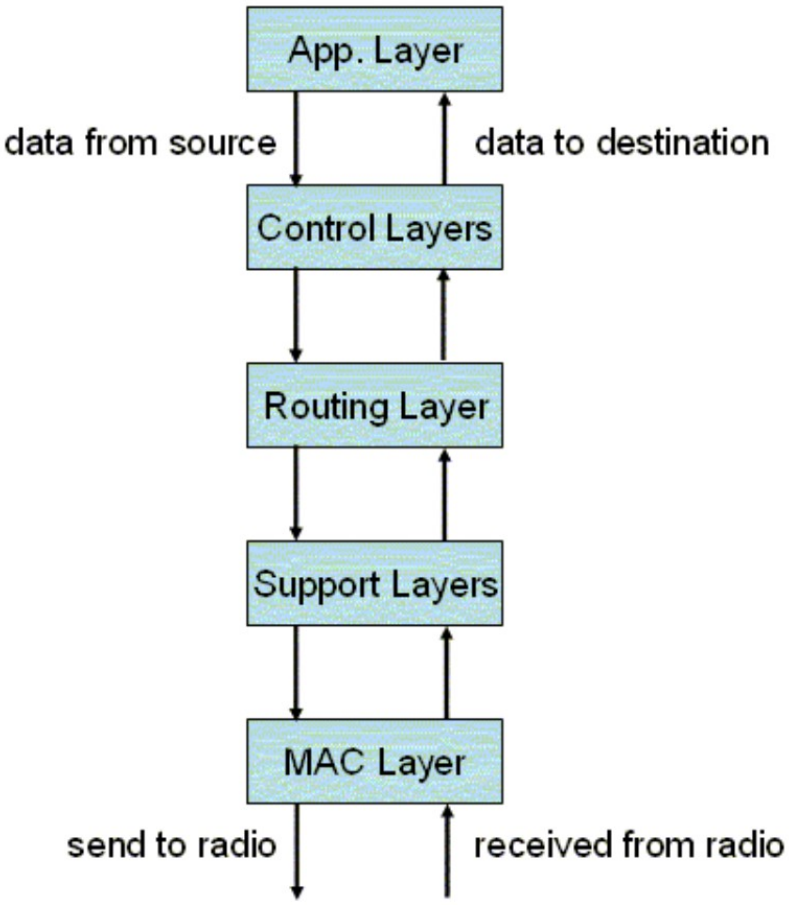


Figura 8: Arquitectura modular y por capas de RMASE [29].

Definiendo la topología de la red

Existen varias formas de ingresar la topología de la red, conforme se comparte en la documentación oficial de la aplicación, sin embargo, una de las más útiles es definirlo de manera explícita por medio de los IDs de cada nodo, y las ubicaciones relativas. Puede definirse un archivo `topology.m` (con nombre arbitrario) que debe retornar las ubicaciones de los nodos (en una matriz de $N \times 2$, donde N es la cantidad de nodos) y sus IDs, respectivamente. Para poder utilizarlo este debe estar en el directorio de trabajo de Matlab y ser marcada la opción de la elección de parámetros del programa.

Para el correcto funcionamiento de la aplicación es necesario añadir todas las carpetas de `Rmase-1.1share` al directoria de trabajo. Adicionalmente a lo que se tiene originalmente, se añadió la carpeta de `topologies` para almacenar los archivos de topologías.

Iniciando RMASE

Una vez añadidos los archivos de interés al directorio de trabajo puede utilizarse RMASE bajo la línea de comandos, o bien por medio de una interfaz gráfica, para iniciar la interfaz

gráfica basta con escribir `prowler` la línea de comandos. La ventana se ilustra en Figura 9.

Figura 9: Vista inicial de la interfaz gráfica de Prowler.

Es útil recordar que RMASE se implementa como una aplicación de Prowler. En el menú *drop down* para elegir el nombre de la aplicación que Prowler va a ejecutar se selecciona RMASE. Luego, se puede elegir si se desea utilizar un archivo para especificar la topología, como se ilustra en Figura 10.

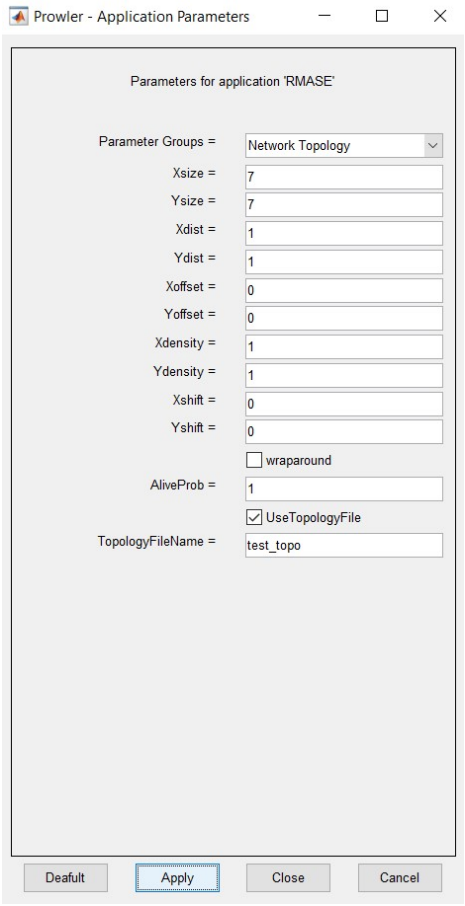


Figura 10: Elección del archivo para definir la topología.

Al presionar aplicar, y luego correr la aplicación sin especificar algo adicional se obtiene algo parecido a lo que se ilustra en Figura 11.

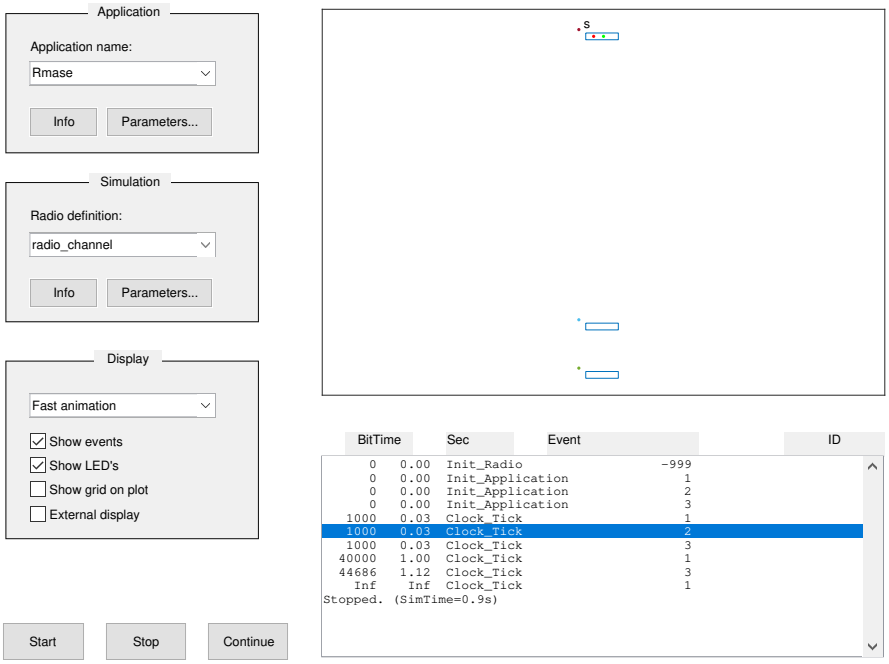


Figura 11: Despliegue de la topología inicial en la aplicación.

Escribiendo un nuevo componente de ruteo

Es posible definir un nuevo componente que actúe como una capa, permitiendo definir el comportamiento en esta. Este archivo debe incluir:

- Definiciones para los eventos de recepción de paquete, paquete enviado, etc. (en general todos lo eventos de interés).
- Definiciones para los comandos que se desean originar en esa capa.

Para fines de este trabajo se crea el archivo `mainAntNet_layer.m` con una estructura parecida a lo que se presenta a continuación:

```
function status = mainAntNet_layer(N, S)

% ...

switch event
case 'Init_Application'
    % Evento de inicio del programa

case 'Send_Packet'
    % Evento para enviar paquete

case 'Packet_Received'
```

```
        % Evento para recibir paquete

case 'Clock_Tick'
        % Evento de temporizador
end

% ...

% definir comandos para el componente

end
```

Los comandos a generar pueden ser tanto a nivel de visualización (dibujar una línea para que se despliegue en el visualizador, encender/apagar una LED) como a nivel funcional en el componente de ruteo. El componente de ruteo es el enfoque particular de este trabajo. Este define cómo los diferentes nodos de la red se coordinarán para generar rutas hacia destinos particulares.

8.3. Utilizando RMASE para simular AntNet

8.3.1. Generalidades

Funcionamiento de TinyOS y descubrimiento de vecinos

La capa de acceso al medio se implementa según el algoritmo definido en Figura 7. En un primer experimento se desea que los nodos envíen paquetes a sus vecinos (alcanzables por el medio físico inmediato, a solo un salto). Esto de manera que, en la recepción de los paquetes, los nodos puedan descubrir vecinos. Para esto se definen las siguientes capas:

- `mainAntNet_layer`: es la capa de enrutamiento general, responsable de disparar los comandos de envío de paquetes.
- `neigborhood_layer`: es una capa intermedia (de soporte, ver Figura 8) que se encarga de actualizar la estructura `NEIGHBORS`, donde `NEIGHBORS{ID}` denota el conjunto de vecinos para el nodo con identificador `ID`. Su operación de manera simplificada se muestra a continuación:

```
function status = neigborhood_layer(N, S)
% ...
switch event
case 'Send_Packet'
    %se identifica al salto anterior
    data.from = ID;
case 'Packet_Received'
    % agregar el nuevo vecino
    NEIGHBORS{ID} = [NEIGHBORS{ID}, rdata.from];
```

```
    remover_duplicados(NEIGHBORS{ID});
% ...
```

La pila general de capas para este experimento se ilustra en Figura 12. En este caso la capa de aplicación se ve tachada ya que no se utilizó directamente, sino por requerimiento de RMASE para generar estadísticas que, por el momento, no eran relevantes.

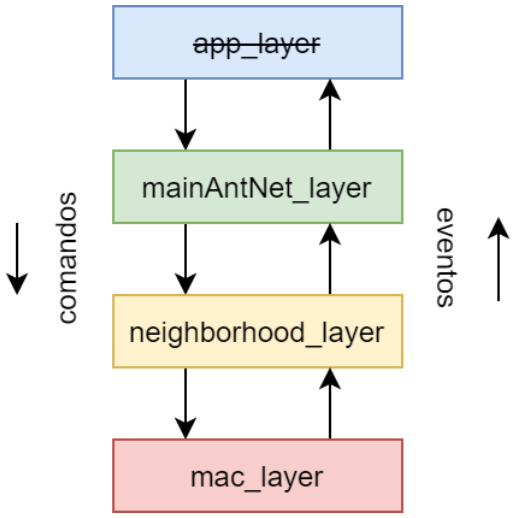


Figura 12: Pila para descubrimiento de vecinos sobre TinyOS

La topología de prueba se muestra en Figura 13, donde el nodo 1 está abajo, el 2 en el medio y el 3 arriba. Para este experimento se definió que en el instante de tiempo (BitTime) $n = 1000$ (se tiene la relación $n = 100 \rightarrow t = 3ms$) todas las partículas generen un evento de reloj, que a su vez ejecute un envío de paquetes tipo *broadcast*. Se ve que cuando $n = 1000$ los tres nodos empiezan el proceso de solicitud de canal.

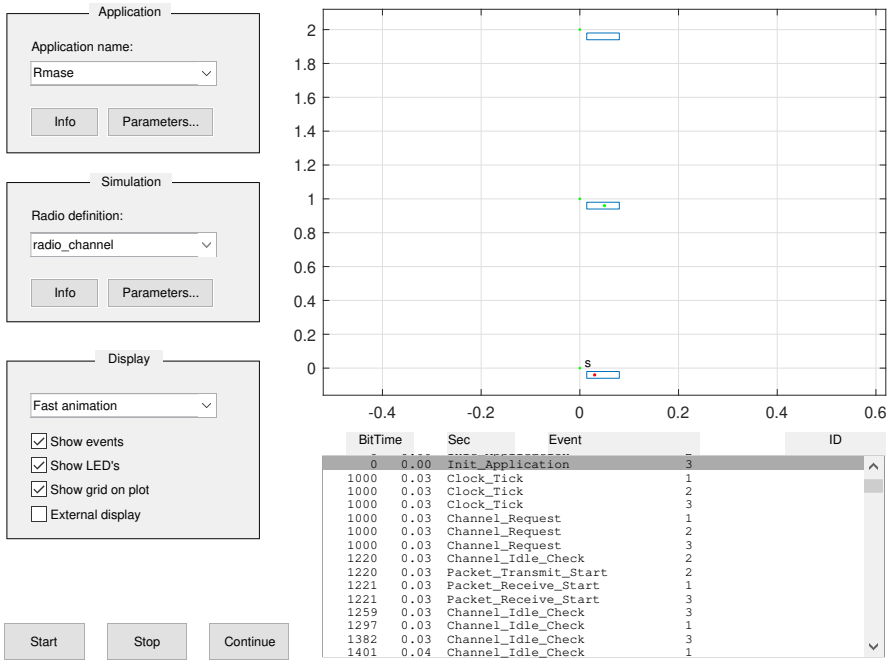


Figura 13: Disposición para prueba de TinyOS

Luego, se generan los siguientes eventos:

- 1. Los nodos esperan un tiempo aleatorio antes de sensar el canal.
- 2. Para este caso, el nodo 2 es el primero en sensar el canal, y al hallarlo libre empieza la transmisión.
- 3. Los nodos 1 y 3 reconocen el inicio de la transmisión e incian la recepción.
- 4. Mientras reciben el paquete, los nodos 1 y 3 sensan el canal, sin encontrarlo libre, por lo que no empiezan su transmisión.
- 5. El nodo 2 finaliza su transmisión
- 6. Los nodos 1 y 3 finalizan la recepción.

Posterior a esto, los nodos 1 y 3 siguen sensando el canal hasta hallarlo libre y logran enviar sus paquetes, cada uno en su turno. El resultado final es que la estructura `NEIGHBORS` tiene tres listas con dos vecinos en cada lista (se tiene registro de seis vecinos en total).

Colisiones de tramas

Bajo el mismo escenario de prueba de la sección anterior, no siempre sucede que todas las tramas son transmitidas de manera exitosa hacia todos los nodos vecinos. En otra prueba sucedió que el tiempo aleatorio para el sensado del canal fue muy cercano para dos nodos,

por lo que uno empezó a transmitir y el otro no pudo acusar la recepción adecuadamente. Esto generó un evento de `collided packet received` en el nodo que empezó la recepción tardía, lo cual no le permitió descubrir al vecino. Esto se ilustra en Figura 14.

BitTime	Sec	Event	ID
1000	0.03	Clock_Tick	1
1000	0.03	Clock_Tick	2
1000	0.03	Clock_Tick	3
1000	0.03	Channel_Request	1
1000	0.03	Channel_Request	2
1000	0.03	Channel_Request	3
1210	0.03	Channel_Idle_Check	1
1210	0.03	Packet_Transmit_Start	1
1211	0.03	Channel_Idle_Check	3
1211	0.03	Packet_Receive_Start	2
1211	0.03	Packet_Receive_Start	3
1265	0.03	Channel_Idle_Check	2
1337	0.03	Channel_Idle_Check	3
1393	0.03	Channel_Idle_Check	2

BitTime	Sec	Event	ID
2170	0.05	Packet_Transmit_End	1
2170	0.05	Packet_Sent	1
2171	0.05	Packet_Receive_End	2
2171	0.05	Packet_Receive_End	3
2171	0.05	Packet_Received	2
2171	0.05	Collided_Packet_Received	3
2258	0.06	Channel_Idle_Check	3
2258	0.06	Packet_Transmit_Start	3
2259	0.06	Packet_Receive_Start	1
2259	0.06	Packet_Receive_Start	2
2283	0.06	Channel_Idle_Check	2
2409	0.06	Channel_Idle_Check	2
2517	0.06	Channel_Idle_Check	2
2637	0.07	Channel_Idle_Check	2

Figura 14: Generación de evento de colisión por similitud en tiempo de medición

Mejorando el descubrimiento de vecinos

Los resultados de las pruebas preliminares demostraron si se desea tener mayor seguridad de descubrir los vecinos del nodo, se deben tomar consideraciones adicionales. Además, el descubrimiento de vecinos forma un paso escencial para el correcto funcionamiento del algoritmo de ruteo, ya que aumenta la cantidad de soluciones disponibles. Los autores de [31] describen el problema del descubrimiento de vecinos, brindando dos tipos de soluciones:

- Descubrimiento directo: bajo este paradigma, un nodo se reconoce como vecino si y solo si se recibe un paquete con su dirección como fuente (a nivel de capa dos).
- Descubrimiento indirecto: en este enfoque los nodos vecinos pueden compartir información para informar a otro vecino de manera explícita la existencia de un vecino.

Estas soluciones, a su vez, se pueden dar para dos tipos de redes:

- Síncronas: (*slotted*) en estas existe una sincronía que permite alinear eventos o programarlos de manera que están coordinados. A cada bloque de tiempo que permite alinear eventos se le conoce como tiempo de trama.
- Asíncronas: en estas los eventos que genera cada nodo no siguen un patrón y orden con respecto de la ocurrencia de los eventos generados por otros nodos.

Con base en el trabajo de estos autores se implementó el algoritmo para hallar vecinos que se muestra en Figura 15. Este flujo es de descubrimiento directo y para una red síncrona. El valor de P_t es la probabilidad que optimiza el número de vecinos que la red puede conocer

por cada tiempo de trama y sigue la regla en (17) cuando el nodo de vecinos alcanzables (o en la vecindad) k satisface $k >> 1$.

$$P_t \approx \frac{2\pi}{k\theta} \tag{17}$$

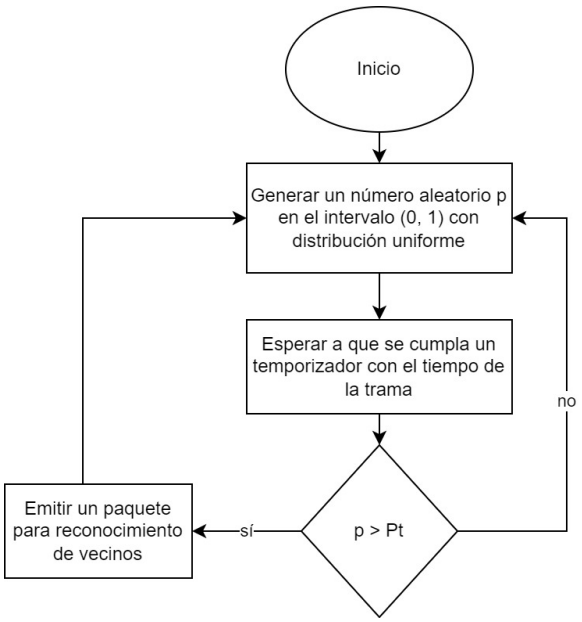


Figura 15: Flujo para descubrimiento de vecinos [31].

El descubrimiento de vecinos puede ejecutarse de manera periódica, es decir, aplicando el algoritmo propuesto a cada cierto tiempo, o una sola vez al inicio del programa, si se asume que todos los nodos inician sus funcionalidades al mismo tiempo. Para el caso de estas simulaciones se toma esta última manera de proceder, ya que todos los nodos efectivamente empiezan a operar al mismo tiempo. Para determinar la funcionalidad del algoritmo se planteó el escenario de pruebas que se muestra en Figura 16. Es una cuadrícula de 7×7 que con nodos separados 1 unidad tanto horizontal como verticalmente.

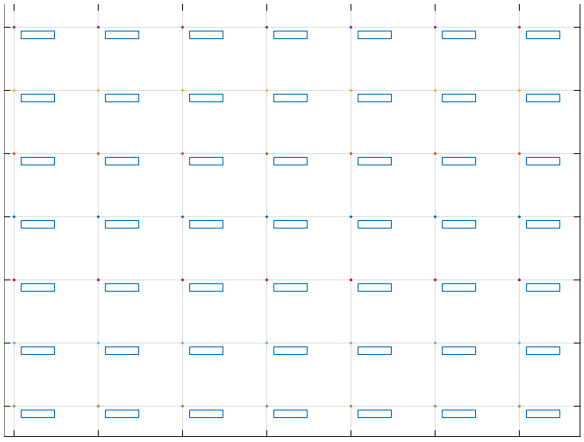


Figura 16: Topología utilizada para las pruebas

Se tomó el tiempo de trama como una sobrestimación del tiempo que le toma a un nodo enviar un paquete hacia un vecino. Esto se compone de:

- El tiempo de espera aleatorio de TinyOS.
- El tiempo de *backoff* aleatorio en caso el canal esté ocupado a la hora del envío de paquete.
- El tiempo que le toma al paquete atravesar el medio.

Los parámetros anteriores pueden consultarse o modificarse por medio de la interfaz gráfica al dirigirse a *radio channel/parameters* en la pantalla principal. Se muestra algo similar a Figura 17.

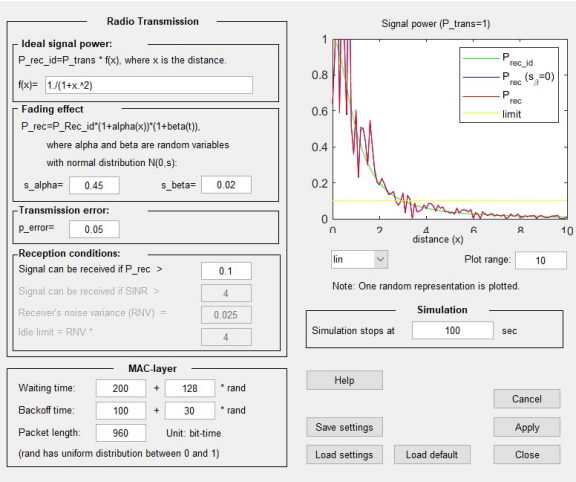


Figura 17: Pantalla de selección de parámetros de transmisión

Se tomó como estimación arbitraria que si se tiene N como el número de nodos en la red, y se sobreestima que todos los posibles destinos pueden ser vecinos (es decir, $N-1$ vecinos), entonces el número de tiempos de trama a esperar para conocer una cantidad aceptable de vecinos es $(N-1)/2$. Esto surgió luego de prueba y error y para que no se prolongara demasiado el tiempo de simulación. Con estos valores se obtuvo en promedio 11 vecinos luego de 50 corridas. Cabe destacar que el radio de cobertura teórico que tienen los nodos está dado por (18).

$$r = \sqrt{\frac{1}{T} - 1} \tag{18}$$

Donde T es el umbral de potencia para considerar que el paquete es posible recibirlo (esta expresión se obtiene al considerar que la potencia decrece con el cuadrado de la distancia desde el punto de transmisión). Para el caso de nuestro escenario de pruebas se tiene $r = 3$ y por tanto 24 vecinos teóricos alcanzables. Por ende se puede concluir que para las condiciones de prueba el método sincronizado permite hallar cerca del 50 % vecinos posibles.

8.3.2. Específico de AntNet

Remover bucles

Como se vio en la sección de caracterízación de AntNet, es necesario remover los bucles antes del envío de las hormigas de *backwards* de regreso a la fuente para que las noticias de la nueva muestra se propaguen rápidamente. Para esto se utilizó el procedimiento propuesto en Figura 18 con una ligera modificación, donde en lugar de hacer la búsqueda del primer índice sin con valor repetido en el arreglo, se busca el último índice posible para que se elimine el mayor bucle posible.

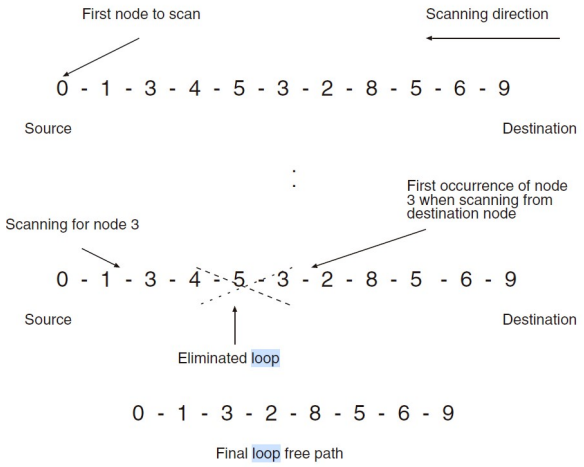


Figura 18: Algoritmo para remoción de bucles [18].

Definiendo tipos de paquetes

Para la implementación en MATLAB se definieron 3 tipos de paquetes a ser transmitidos/recibidos dentro de la red:

- Paquetes de descubrimiento de vecinos: estos solo actualizan la estructura de vecinos conocidos, no generan impacto a nivel de ruteo.
- Paquetes de hormigas de *forward*
- Paquetes de hormigas de *backward*

Para esto se definió una estructura del paquete donde el campo con nombre *MSG_ID* identifica numéricamente el tipo de paquete.

Funciones importantes

Dentro del código se diseñaron dos funciones principales que pueden ser modificadas por el usuario:

- *update_T_matrix* esta función permite actualizar el modelo de la red así como la cantidad de feromona para cada entrada de la matriz. El usuario puede modificar estas líneas de código que tiene a su disposición todos los parámetros necesarios de manera que pueda probar distintos modelos de red, diferentes a los que se proponen en 6 (referirse al código documentado para los diferentes argumentos disponibles).

- *get_reinforcement* esta función permite calcular el refuerzo r con base en la media del RTT, su varianza, su mejor valor durante las últimas W muestras, su nuevo valor y los hiperparámetros de AntNet. El usuario puede elegir retornar cualquier valor de r tal que se mantenga en el intervalo de $(0, 1)$.

8.3.3. Resultados

Para esta sección se utilizan los hiperparámetros estándar de AntNet que se presentan en la sección de caracterización. El escenario de pruebas es el que se presenta en Figura 19. Se define de manera fija un destino y una fuente, como se ilustra en la figura.

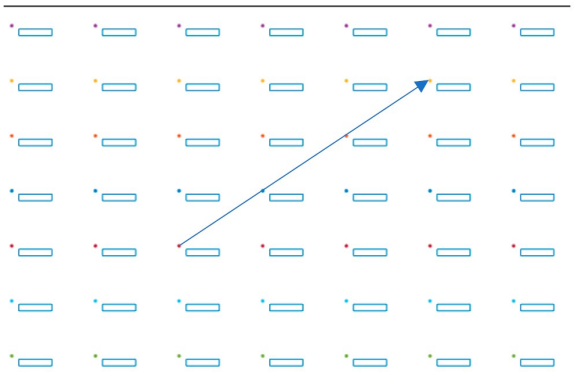


Figura 19: Objetivo de las pruebas: establecer una conexión (ruta) entre dos nodos

Los resultados consisten de lo siguiente: una gráfica que presenta la evolución de la feromona asignada a cada vecino para el destino previamente seleccionado. Esto simboliza el grado de aceptación que el algoritmo provee a ese vecino como solución al problema de ruteo para el destino fijo. También se obtuvieron representaciones de las diferentes soluciones que genera el algoritmo antes de converger. Se define la convergencia del algoritmo como sigue: la condición de paro se da cuando la entrada con el valor máximo de la matriz de feromonas es diferente de la unidad por 0.001. Esto también puede interpretarse como un *porcentaje de error* de 0.1 %.

En Figura 20 se ve el comportamiento típico de las primeras iteraciones del algoritmo (para ese caso se define como nodo fuente el 15 y como destino el 19). Se denomina una iteración como un proceso donde se logra hallar un camino de ida y vuelta al destino. Se ve que al inicio, y como se esperaba, básicamente se tiene un comportamiento aleatorio para la búsqueda del siguiente salto. En el transiente para hallarlo se ve cómo se prueban multitud de saltos distintos antes de que se produzca una sucesión de saltos válida.

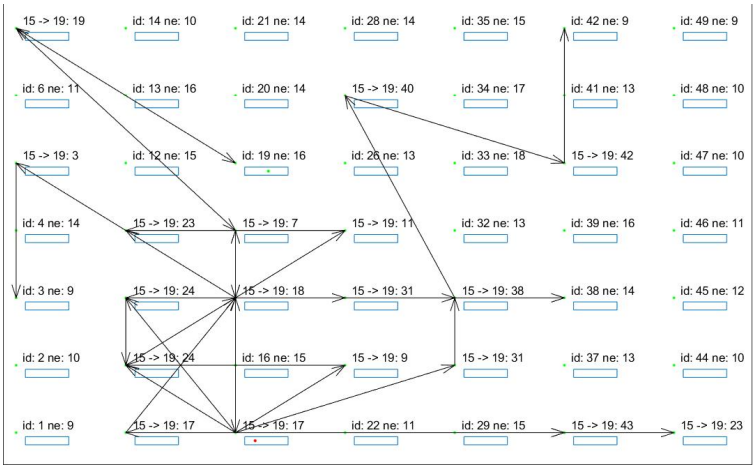


Figura 20: Primera iteración del algoritmo (se halla la primera solución completa, una ruta de ida y vuelta al nodo de interés)

La segunda iteración del algoritmo se presenta en Figura 21. Se ve cómo los mecanismos de AntNet entran en acción ya que se tiene una sucesión de saltos sin extravíos hacia el destino.

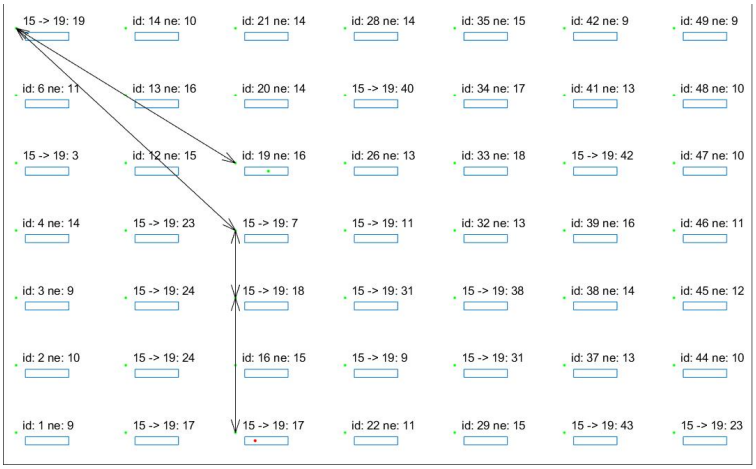


Figura 21: Segunda iteración del algoritmo

Una tercera iteración del algoritmo se presenta en Figura 22. Se ve cómo la característica de aleatoriedad del algoritmo aún permite extravíos en los saltos, sin embargo la cantidad es mucho menor.

Figura 22: Tercera iteración del algoritmo

Al algoritmo le toman 10 iteraciones para converger, en este caso la ruta elegida es la de la segunda iteración. En términos del tiempo y la feromona los resultados se presentan en Figura 23. Esta visualización permite obtener un resumen de lo que sucedió a través del algoritmo:

- El tiempo de convergencia fue de 13.29 s

- El vecino elegido como siguiente salto para el destino elegido tiene ID 17

- La feromona para la ruta elegida crece y es *dominante* durante la ejecución del algoritmo, es decir, el algoritmo converge casi de manera directa a esa solución, sin deambular por otras.

Este tipo de visualización, junto con la de cada salto del algoritmo pueden ayudar a generar un mejor entendimiento de AntNet para proponer mejoras o adiciones que se consideren convenientes.

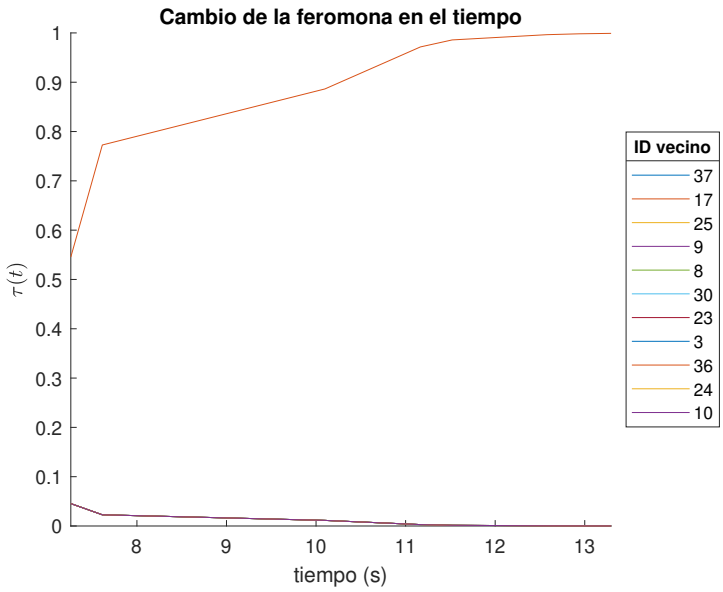


Figura 23: Gráfica de feromona para el experimento inicial

La primeras tres iteraciones para otra topología se ilustran en las Figuras 24 a 26. De manera similar que para la topología anterior, al inicio se tienen varios saltos de tipo extravío, luego se va convergiendo a una solución. La ruta convergente en este caso es la de Figura 25. En total al algoritmo le toman 27 iteraciones para converger. La gráfica de feromona se presenta en Figura 27. Para este experimento se ve cómo el algoritmo osciló entre dos soluciones, a pesar de que convergió a la solución esperada. Por este comportamiento también le tomó 11.58 segundos converger, lo cual es bastante tiempo al considerar que para el experimento anterior el tiempo de convergencia fue similar aún teniendo una topología con más nodos (más decisiones qué tomar).

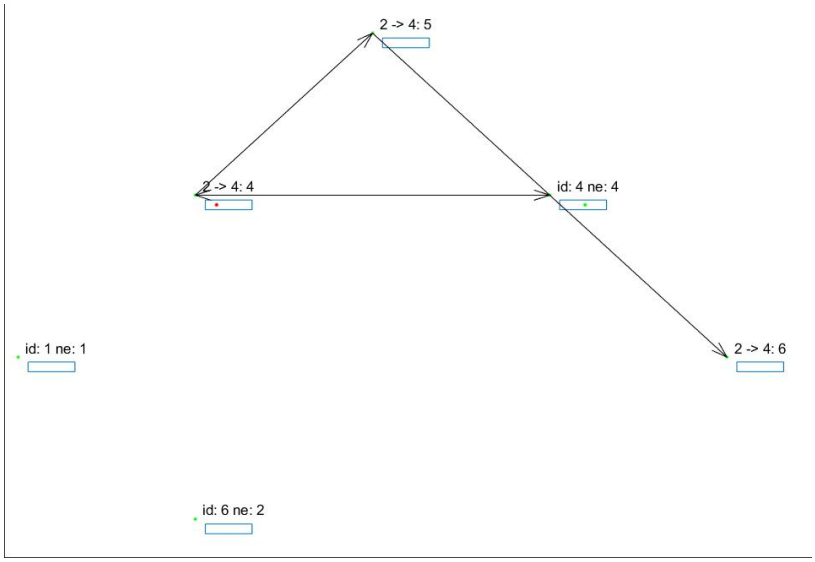


Figura 24: Primera iteración, segundo experimento

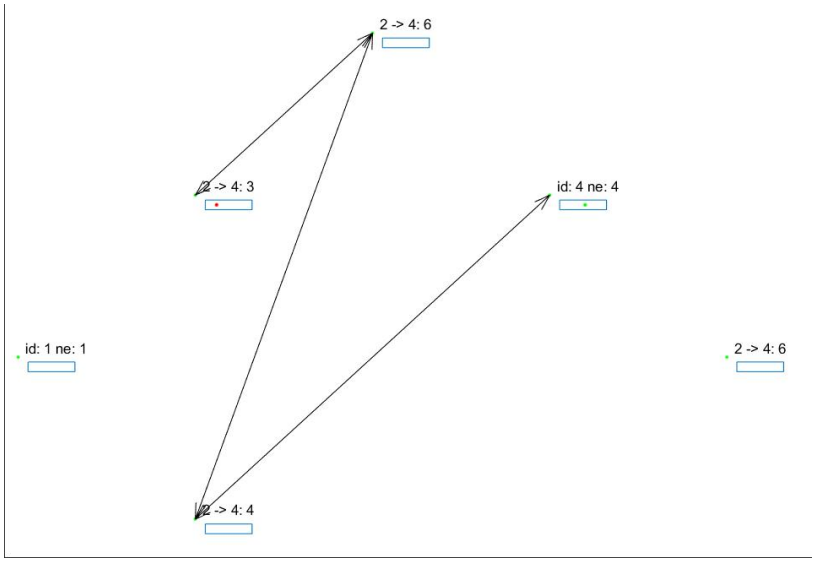


Figura 25: Segunda iteración, segundo experimento

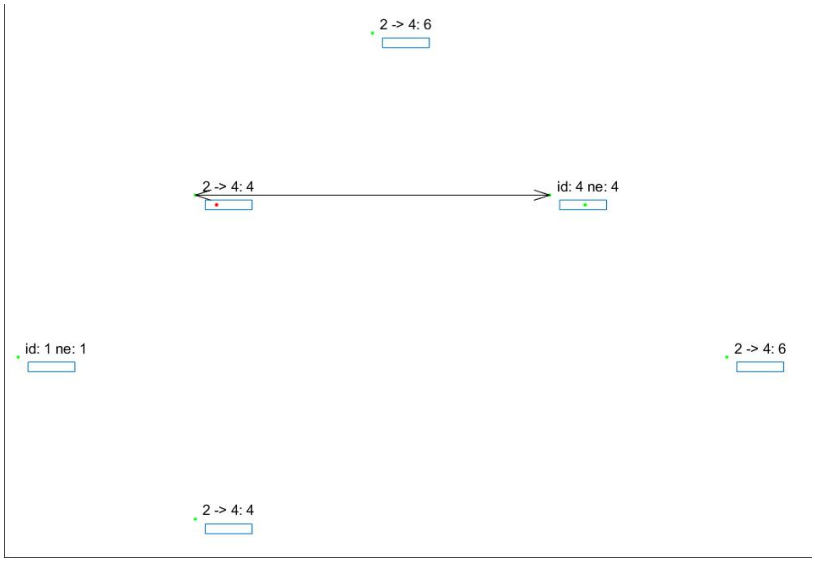


Figura 26: Tercera iteración, segundo experimento

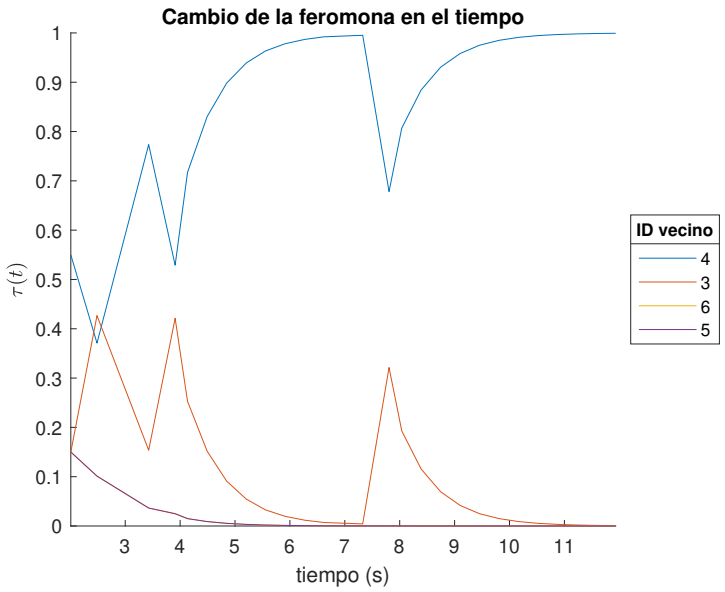


Figura 27: Gráfica de feromona, segundo experimento

Como resultado ilustrativo final se tiene el estado final presentado en Figura 28 para el tercer experimento. En este experimento se ve cómo el algoritmo converge a una ruta claramente sub-óptima. Para esa fuente y destino en particular, y en varias corridas del experimento, se tuvo este tipo de comportamiento. Esto ilustra también cómo puede ser necesario para ciertas topologías hacer un refinamiento de los hiper-parámetros a ser utilizados o bien de la manera en la que se otorga el refuerzo.

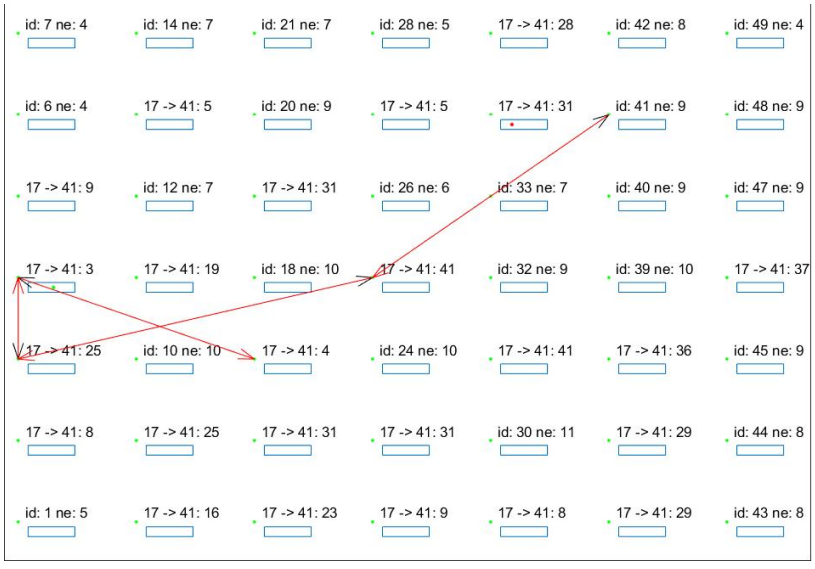


Figura 28: Ruta convergente, tercer experimento

CAPÍTULO 9

Implementación de la red física

La implementación de un algoritmo en dispositivos físicos genera nuevas consideraciones y retos a tomar en cuenta. En este capitulo se exponen las decisiones y criterios tomados para la elección de los componentes físicos de la red y los objetivos planteados, tomando en cuenta la característica principal de estas: muchos nodos con capacidad limitada.

9.1. Componentes físicos de la red

Hay muchos estándares y soluciones para la conectividad de los nodos de manera inalámbrica. Entre las opciones principales están:

- *Thread* y *Zigbee*: pueden conectar sensores en la banda de 2.4 GHz, con un ancho de banda de 250 kbps. Ya poseen una pila de protocolos definido y el factor determinante para utilizarlos o no es si es posible redefinir del protocolo de enrutamiento implementado por defecto.
- *Z-wave*: opera en la banda de 915 MHz con menor ancho de banda de 250 kbps (menor frecuencia de modulación, mayor distancia de cobertura). Posee la misma pregunta fundamental del primer inciso.
- *Arduino IoT Nano 33*: esta solución es conveniente ya que la Universidad ya cuenta con algunos.
- Transceptor general: esta forma de implementar la conectividad nos permite hacerlo a la medida, desde cero. Esto puede aumenta el tiempo de implementación pero permite tener mejores mediciones del rendimiento del algoritmo puro. Una opción popular es el módulo transceptor nRF24L01, que implementa la comunicación inalámbrica en la banda de 2.4GHz, con comunicación SPI hacia algún módulo de procesamiento.

El principal criterio para este trabajo de investigación es tener la capacidad de variar los parámetros y funciones que el algoritmo usa para enrutar, sin restricciones, de manera que se tenga la posibilidad de explorar diversas formas de enrutar. Por este motivo se optó por un transceptor general, en particular el MRF24J40MA. Este es un módulo transceptor certificado, que opera en la banda de 2.4 GHz y cuenta con soporte para ZigBee™ y otros protocolos, sin embargo se utilizaron solo sus funcionalidades de capa 1 y 2.

9.2. Consideraciones para la implementación de AntNet

9.2.1. Componentes utilizados

Hardware:

- Arduino Nano (funcional con el *old bootloader* en el IDE de Arduino.
- MRF24J40MA
- Una placa que sirve como *shield* e interfaza entre el transceptor y el Arduino Nano. Esta fue desarrollada por Hee Chan Kim, estudiante de ingeniería mecatrónica de esta universidad.

Software:

- IDE de Arduino, con el paquete de librerías base para Arduino Nano
- Libería *mrf24j40ma.h* desarrollada por Hee Chan Kim. Se utilizaron las funcionalidades de asignación de dirección, selección de canal, envío de datos y recepción de datos. Se añadió una función para enviar datos con base en un puntero a un arreglo de tipo *uint8_t*
- Liberería *Adafruit_NeoPixel* al igual que en los ejemplos de uso de la librería de Hee.

9.2.2. Tipos de datos utilizados

En la documentación de la librería *mrf24j40ma.h* se detallan estructuras a las que se puede escribir o de las que se puede leer el momento del envío o recepción. Además de esas estructuras se utiliadon otras principales. Con fines de apartar la memoria en el MCU y para garantizar la funcioinalidad general del algoritmo es importante definir:

- El número de nodos en la red: cada nodo debe tener este dato o una noción del mismo para saber cuánta memoria apartar para las matrices de feromona y los modelos de la red.
- El número de saltos máximos permitidos: esto pone un límite en el tamaño máximo del paquete que atraviesa la red. Una particularidad de AntNet es que el *payload* de

los paquetes aumenta en cada salto, por lo que es necesario máximo de saltos para apartar memoria para su buffer.

- El número de muestras sobre las cuales se va a calcular el mínimo histórico. Esto también se conoce como tamaño de la ventana y define el tamaño de una matriz de arreglos circulares.

- Los IDs para los tipos de mensajes en la red. Como se explicó en el capítulo de simulaciones, para el caso de AntNet son tres, incluyendo los mensajes de descubrimiento de vecinos.

9.2.3. El direccionamiento

Los módulos MRF24J40MA cuentan con un direcciones capa dos de dieciséis bits. Todas las estructuras que almacenan direcciones son de tipo de datos *uint16_t*. El tamaño de las direcciones también es un parámetro crítico en cuanto al tamaño de variables y paquetes.

9.2.4. Manejo de tiempo

Un aspecto crucial de AntNet es el cálculo del *trip time* o bien el *round trip time*. Ya que calcular la primera cantidad requiere sincronía de relojes entre los miembros, se puede opta por la segunda opción. Arduino implementa las funciones *millis()* y *micros()* que permiten saber cuántos milisegundos y microsegundos han pasado desde el inico del programa, respectivamente. Dependiendo de la precisión que se quiera, se puede elegir una función u otra. Para estos experimentos se usaron microsegundos.

9.2.5. La feromona

En la implementación de Matlab la feromona se utilizó como una cantidad de punto flotante, lo cual también existe en Arduino (no necesariamente con la misma resolución). Ya que el chip de los Arduinos Nano, el ATmega328, no tiene una FPU, las operaciones con punto flotante deben abstraerse en software, lo cual puede hacer al algoritmo lento. Para estas pruebas se utilizó punto flotante, sin embargo puede considerarse utilizar enteros como feromona y realizar pruebas. Esto requería nuevas formulaciones para garantizar que concuerde con la implementación esperada de AntNet.

9.2.6. El modelo de la red

Para esta implementación se utiliza una simplificación del modelo de la red que solo utiliza el RTT mínimo sobre la ventana de medición hacia cada destino. Para esto se utilizó una matriz de valores de cuatro bytes que almacenan los tiempos en microsegundos.

9.2.7. El refuerzo

Se ajusta el refuerzo a uno de los términos de la expresión original que ideó el autor de AntNet. Este considera solo el mínimo histórico del RTT en comparación del RTT actual, se da en (19).

$$r = \text{mín}\,(0.5, \frac{t_m}{t_a}) \tag{19}$$

Donde t_m es el RTT mínimo sobre la ventana de medición y t_a es el RTT calculado en la iteración actual. Véase que mientras más pequeño es el RTT actual respecto del RTT mínimo hitórico, entonces se le asigna más refuerzo. Esto busca favorecer las rutas con menor RTT.

9.2.8. La estructura del paquete

Se ideó un paquete capa 3 que contienen los campos mostrados en Figura 29.

Índice	0	1	2	4	6	8
Tamaño	1 byte	1 byte	2 bytes	2 bytes	2 byte	4 bytes
Nombre	data_type	payload_size	source_address	destination_address	hop_address_1	time_address_1

Figura 29: Estructura del paquete para AntNet

- *data_ type*: identifica qué tipo de paquete es
- *payload_ size*: es el tamaño de todo el paquete capa tres, (*payload* del paquete capa dos).
- *source_ address*: dirección fuente a nivel de capa tres.
- *destination_ address*: dirección destino a nivel de capa tres.
- *hop_ address_ 1*: es la dirección del primer salto hacia el destino.
- *time_ address_ 1*: es el tiempo con el que el dispositivo con dirección 1 marca el paquete. Es útil para calcular el RTT en el camino de regreso.

Pueden haber tantos *hop_ address* y *time_ address* como saltos esté permitido en la red.

9.2.9. Funcionalidades

Existen tres eventos principales en AntNet, los cuales deben tener funciones que ejecuten las respuestas esperadas:

- Se recibe una paquete de reconocimiento de vecinos: si el vecino es nuevo, activar una bandera donde se le reconoce como vecino y aumentar el contador de vecinos.

- Se vence el contador para enviar una hormiga de *forward*:
  - Elegir el destino de la hormiga
  - Elegir un vecino para ir a ese destino, con base en la probabilidad de la matriz de feromonas
  - Llenar el paquete con los encabezados de dirección de destino, fuente y tipo de dato
- Se recibe una hormiga de *forward*
  - Si es el destino, remover bucles de la pila de direcciones y enviar una hormiga de *backward* hacia el nodo que originó el paquete, con base en la última dirección de la pila.
  - Si no es el destino re-direccionar el paquete con base en la matriz de feromonas y añadir al paquete la dirección y tiempo del nodo al final del paquete, aumentar el tamaño indicado en el índice 1.
- Se recibe una hormiga de *backward*
  - Calcular el RTT correspondiente
  - Identificar el vecino por donde se hizo *forward* a ese paquete y el destino del paquete.
  - Actualizar el modelo de la red
  - Actualizar la matriz de feromonas
  - Si no es el destino del paquete, redireccionar con base en la operación *pop()* sobre la pila de direcciones al final del paquete

9.3. Resultados

Para los experimentos se utilizó un tiempo de 300 ms como intervalo de envío de hormigas de *forward*. El primer experimento consistió en un escenario sencillo: tres nodos en el mismo radio de alcance, con una fuente y un destino. Lo que se quiere validar es si el algoritmo permite elegir la ruta óptima, que claramente es enviar directamente el paquete al destino y no generar dos saltos para hacerlo. El escenario de pruebas fue como se muestra en Figura 30 y la recolección de la información se hizo por medio del monitor serial de Arduino como se ve en Figura 31.

Figura 30: Escenario para las primeras pruebas, tres nodos en el mismo radio de alcance.

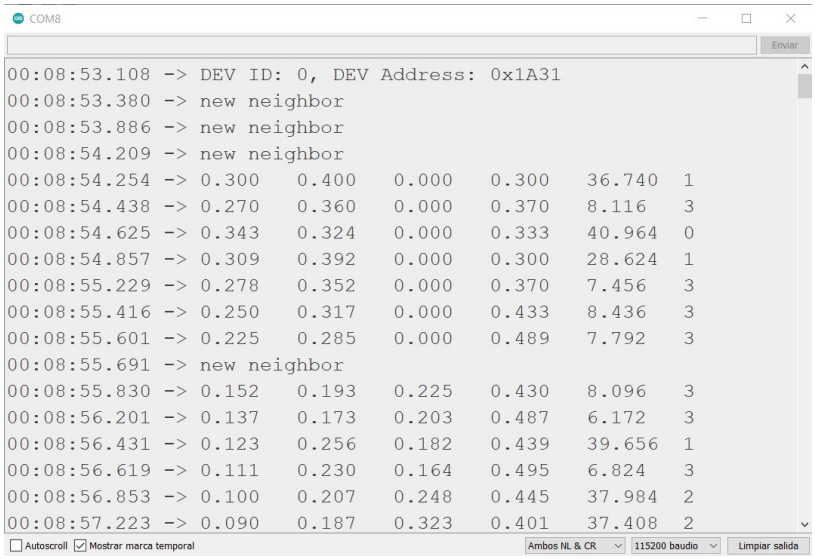


Figura 31: Recolección de los datos por medio del monitor serial.

Se hicieron diez corridas, obteniendo la gráfica de feromonas y el RTT para cada instante durante la prueba. Utilizando una ventana de medición de tres muestras se obtuvo que siete veces convergió el algoritmo a hacer dos saltos (sub-óptimo) y tres veces al resultado óptimo. Resultados ilustrativos se ven en las Figuras 32 a 35.

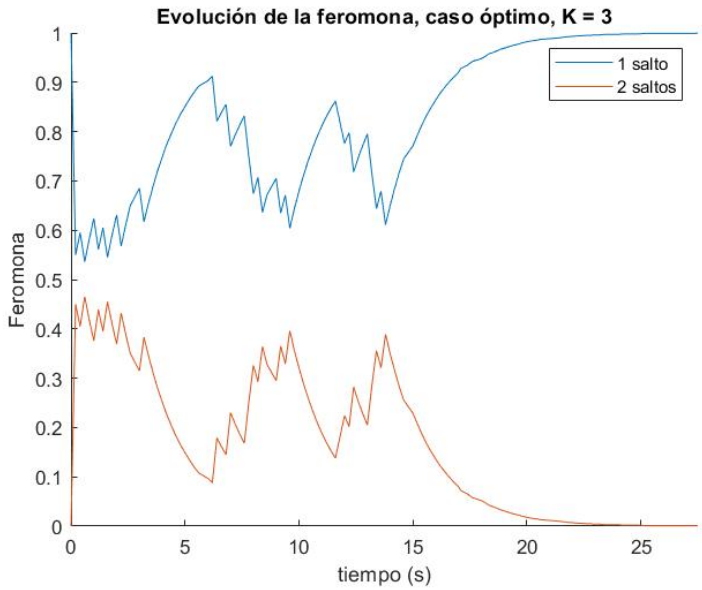


Figura 32: Evolución de la feromona para una ventana de medición de tres muestras, caso óptimo

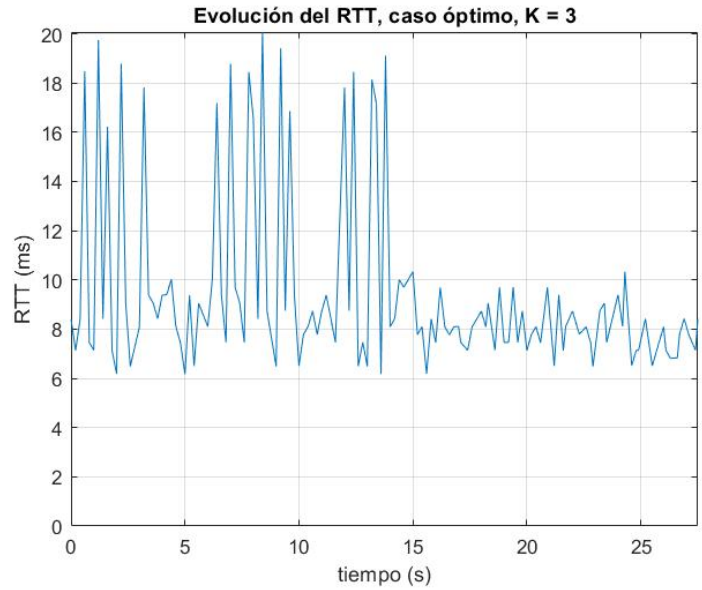


Figura 33: Evolución del RTT para una ventana de medición de tres muestras, caso óptimo

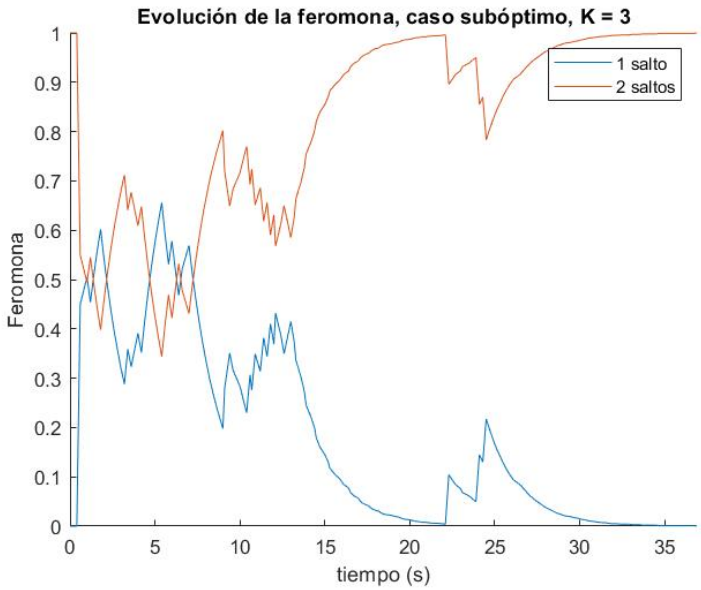


Figura 34: Evolución de la feromona para una ventana de medición de tres muestras, caso sub-óptimo

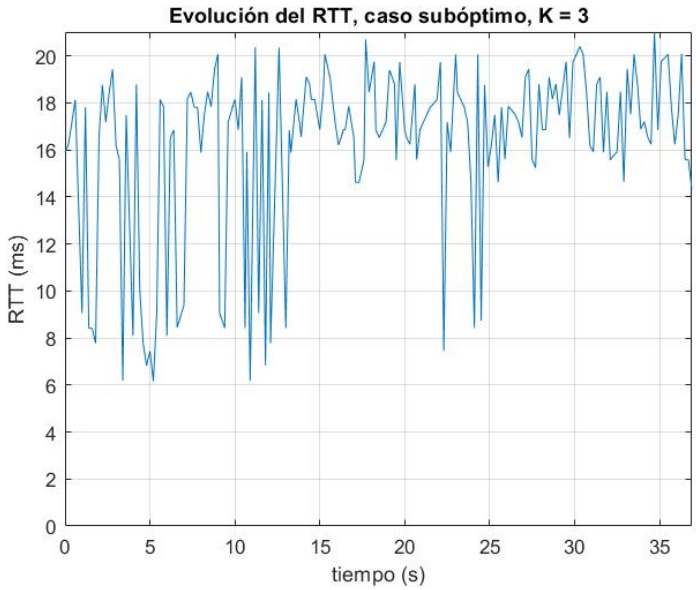


Figura 35: Evolución del RTT para una ventana de medición de tres muestras, caso sub-óptimo

Al evaluar la causa de este comportamiento se vio que había una diferencia significativa en la variación del RTT cuando se hacían dos saltos comparado con un salto. Al notar en las Figuras 33 y 35 se ve que la amplitud de la gráfica del RTT en el caso óptimo es menor (alrededor de 2.5 ms) mientras que la del caso sub-óptimo es cerca del doble. Esto quiere decir que si se da una sucesión de paquetes que llenan el arreglo de RTTs históricos, se favorecerá la feromona de la ruta que tenga más variación. Este comportamiento suecede ya que la regla de asignación para la feromona considera solo el valor mínimo histórico del RTT en este caso.

Se validó el comportamiento al aumentar la ventana de medición a diez muestras y hacer otras diez corridas. Los resultados ilustrativos están en las Figuras 36 a 39. Para este caso se tuvo que siete veces se convergió a la solución óptima, validando la hipótesis hecha con base en las observaciones del experimento anterior. Esto a su vez valida el correcto funcionamiento del protocolo en los dispositivos ya que responde al comportamiento esperado al generar el cambio de ventana de medición.

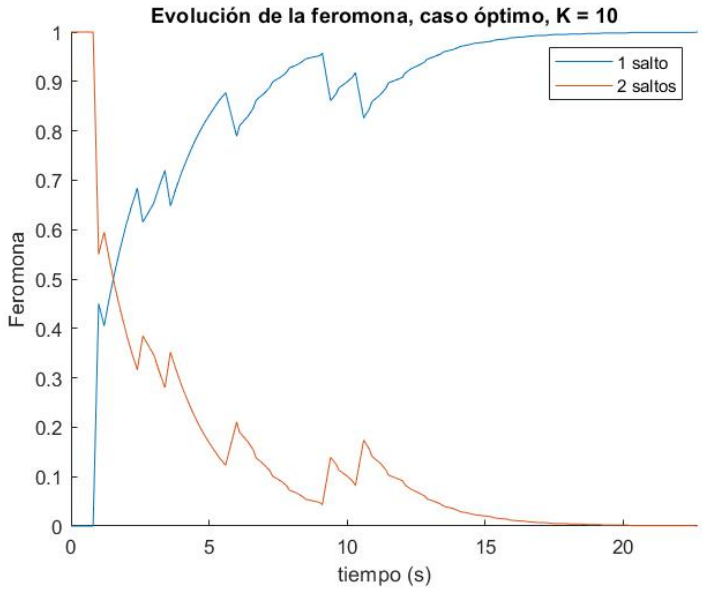


Figura 36: Evolución de la feromona para una ventana de medición de diez muestras, caso óptimo

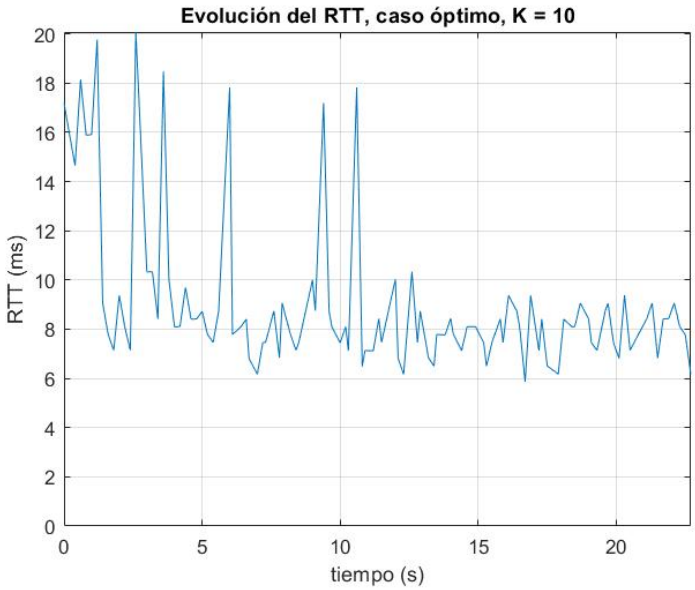


Figura 37: Evolución del RTT para una ventana de medición de diez muestras, caso óptimo

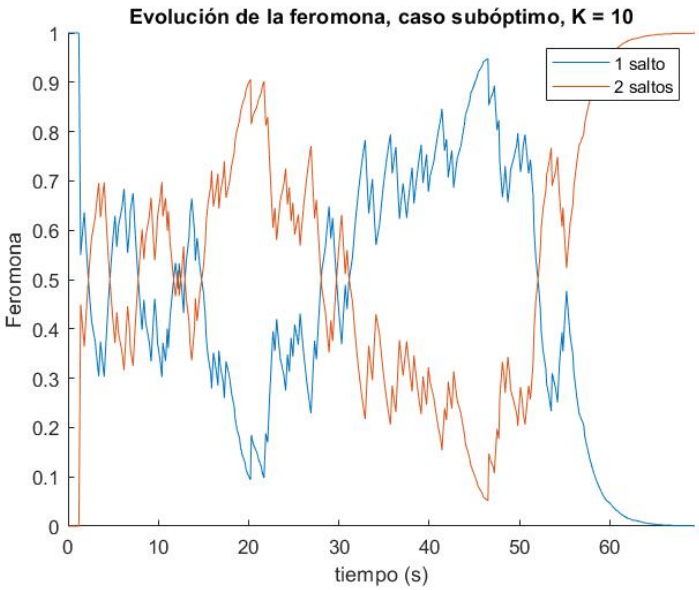


Figura 38: Evolución de la feromona para una ventana de medición de diez muestras, caso sub-óptimo

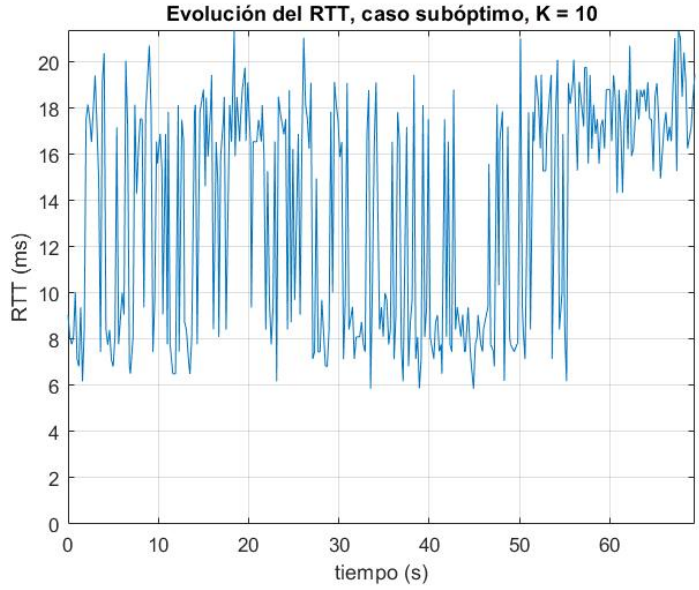


Figura 39: Evolución del RTT para una ventana de medición de diez muestras, caso sub-óptimo

Como experimento final se utilizaron cinco miembros en la red con una fuente y un destino. Se dispusieron al azar, sin tomar en cuenta el radio de alcance y se validó el comportamiento con la ventana de diez mediciones. Nuevamente, en siete de los diez casos se obtuvo el resultado esperado, aún debiendo elegir cada uno de los cienco miembros de manera adecuada para obtener el valor mínimo del RTT. Esto valida nuevamente el funcionamiento de AntNet en esta implementación. Los resultados ilustrativos están en las Figuras 40 a 43.

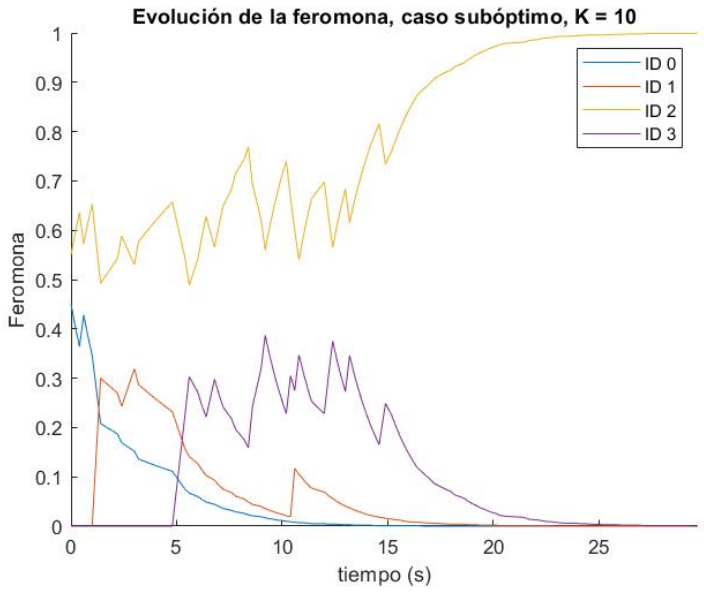


Figura 40: Evolución de la feromona con cuatro vecinos, caso sub-óptimo

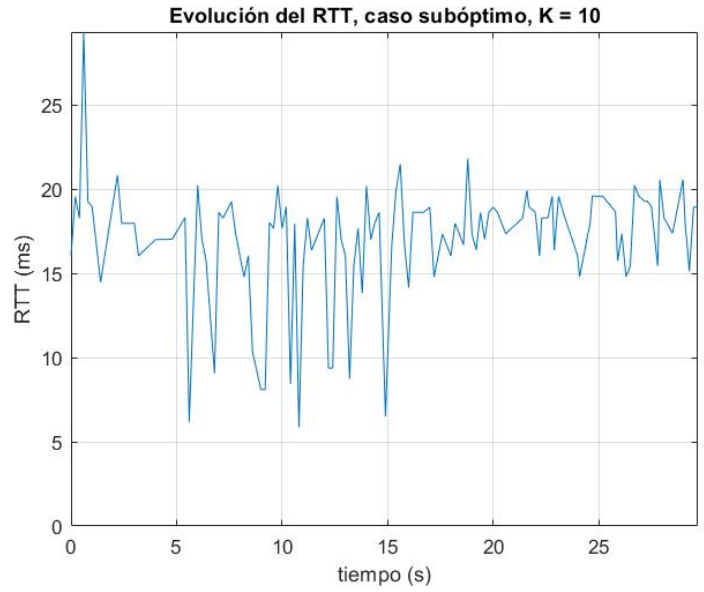


Figura 41: Evolución del RTT con cuatro vecinos, caso sub-óptimo

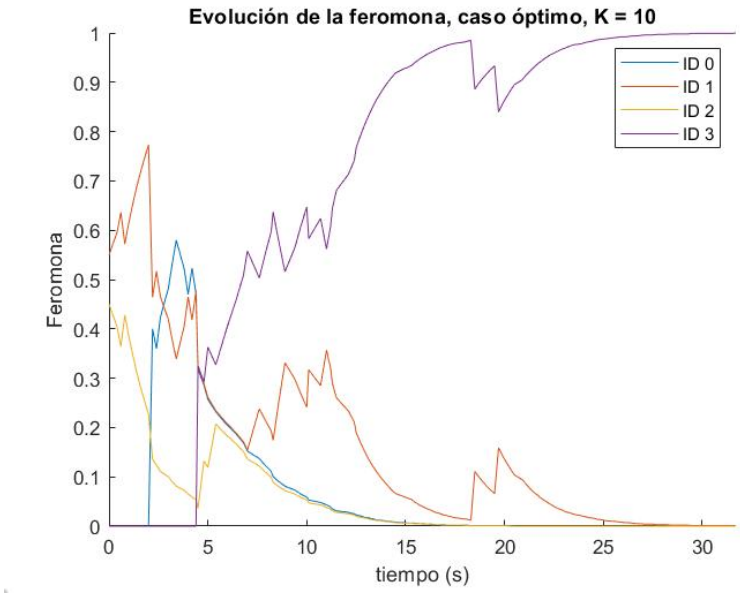


Figura 42: Evolución de la feromona con cuatro vecinos, caso óptimo

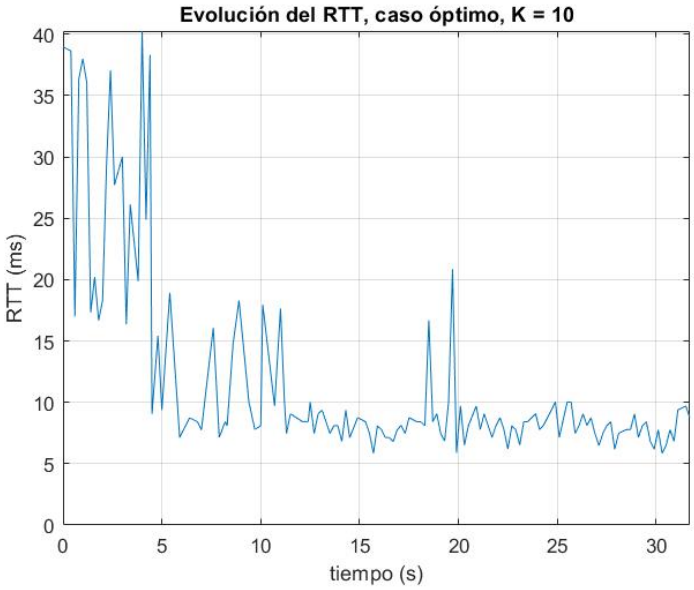


Figura 43: Evolución del RTT con cuatro vecinos, caso óptimo

CAPÍTULO 10

Conclusiones

- Se caracterizó AntNet como un algoritmo de enrutamiento basado en inteligencia de enjambre, explicando sus ideas principales, características como algoritmo de optimización y también como algoritmo útil para redes computacionales.
- Se desarrolló una plataforma de simulación que permite visualizar el comportamiento de AntNet salto por salto y en cada iteración (formación de una solución), y la feromona a través del tiempo. Esto para parámetros que puede definir el usuario conforme se necesite, así como los estándar de AntNet. Esta plataforma permite también obtener parámetros de rendimiento como el tiempo de convergencia de las soluciones, entre otros.
- Se idearon flujos, convenciones, tipos de datos y propuestas de componentes físicos que pueden servir como metodologías base o de referencia para la implementación de algoritmos de enrutamiento basados en las cualidades de la inteligencia de enjambre.
- El modelaje de la media y varianza del tiempo de ida de los paquetes en AntNet por medio de la ecuación de diferencias en la ecuación (6) permite un fácil cálculo a nivel de microcontroladores. Sin embargo, introduce la elección del parámetro ς (varsigma) para el cual no existen criterios de rendimiento asociados que permitan elegirlo analíticamente, solo estimados de la ventana de observaciones a considerar en la actualización de las variables.
- Los parámetros a elegir de AntNet (ver Cuadro 2) no permiten una visualización directa de cómo impactarán en las decisiones de ruteo del algortimo. Esto contrasta con protocolos como OSPF, donde atributos como *route preference* permiten definir de manera más explícita la manera de operación que se desea en la red. Con base en esto, en la implementación básica de AntNet se ve poco probable que sea utilizable en redes como las que provee un ISP (*internet service provider*) ya que dificulta la implementación de políticas.
- Prowler y RMASE resultaron ser un punto de partida que permite evaluar el desempeño de un protocolo de enrutamiento de manera modular. La interfaz gráfica también

permitió adquirir una mejor entendimiento de los eventos que suceden en la red y el acceso total al código base permite modificar casi cualquier aspecto que se requiera, lo cual brindó mucha flexibilidad.

- El módulo transceptor de radio MRF24J40MA, junto con las funcionalidades de programación de Arduino permitieron la implementación de AntNet en una red física inalámbrica, tal como se demuestra en el comportamiento en las curvas de RTT y feromona obtenidas.
- Un intervalo de envío de hormigas de forward de 300 ms fue funcional para apreciar la dinámica del algoritmo de AntNet, así como para capturar estos datos de manera efectiva
- La ventana de muestras para calcular el máximo histórico en AntNet puede ser un parámetro crítico si se utiliza la expresión en (19) para calcular el refuerzo. Una elección cuidadosa del mismo permitió que se hallara el mínimo del RTT en la mayoría de experimentos realizados.

CAPÍTULO 11

Recomendaciones

- Según lo que se describió en el Capítulo de Antnet, utilizar el modelo del retardo de los enlaces para la actualización de las estructuras de datos del algoritmo permite considerar diversas variables de interés (latencia, *jitter*, pérdidas) para las decisiones de ruteo. Sin embargo, hace falta un mejor sustento teórico para el modelaje de la media para lo cual se recomienda hallar o proponer otras ecuaciones y validar con experimentos qué modelo puede ser el más representativo.

- El problema de asignación de crédito a las rutas con base en el tiempo de ida medido es un tema que puede profundizarse. En particular, para el caso que se expuso se tiene la complejidad de implementar la función exponencial para el cálculo, lo cual puede presentar desafíos a nivel de microcontroladores. Se recomienda proponer nuevas reglas de asignación y evaluar, para una misma sucesión de retardos medidos, qué regla genera la selección de las mejores rutas y es factible implementar a nivel físico.

- A nivel de simulación, para futuros trabajos podrían elegirse nuevos módulos o plataformas que permitan variar el modelo de la capa física y de acceso, que en este trabajo se tomó como premisa la implementación de TinyOS y un modelo simple de radio. También sería posible llevar las ideas de inteligencia de enjambre hacia otras tecnologías como MPLS, que actúa en una capa intermedia entre la capa de acceso al medio y la capa de red.

- Implementar nuevos métodos para descubrir vecinos (como *gossip based*), y el paradigma asíncrono de estos. Esto es relevante ya que sirve como base para la funcionalidad apropiada de AntNet (y cualquier otro algoritmo de enrutamiento para redes inalámbricas).

- Investigar e implementar estimaciones analíticas de la cantidad de tiempos de trama a usar como periodo de descubrimiento de vecinos. Esto es importante para garantizar que en una red sincronizada todos los vecinos al alcance sean reconocidos. El artículo de [32] pueder servir como base.

- Idear en RMASE cómo generar pruebas en masa, para poder automatizar las comparaciones entre diferentes modelos de red o formas de refuerzo. Se puede consultar las pruebas en *batch* como comentan los autores de RMASE en la documentación oficial.
- Utilizar la plataforma desarrollada sobre RMASE para evaluar diferentes casos de uso, otros modelos de red y otras formas de calcular el refuerzo.

CAPÍTULO 12

Bibliografía

[1] M. Z. Adamu, "Energy-efficient routing algorithms based on swarm intelligence for wireless sensor networks," Tesis doct., University of Nottingham, 2013.

[2] K. Sohraby, D. Minoli y T. Znati, *Wireless sensor networks: technology, protocols, and applications*. John wiley & sons, 2007.

[3] M. Razzaq, G.-R. Kwon y S. Shin, "Energy efficient Dijkstra-based weighted sum minimization routing protocol for WSN," en *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, IEEE, 2018, págs. 246-251.

[4] B. Velusamy y C. Pushpan, "A Review on Swarm Intelligence Based Routing Approaches," *International Journal of Engineering and Technology Innovation*, vol. 9, no. 3, págs. 182-195, 2019.

[5] A. M. Zungeru, L.-M. Ang y K. P. Seng, "Performance evaluation of ant-based routing protocols for wireless sensor networks," *ArXiv Preprint*, 2012.

[6] C. Shin y M. Lee, "Swarm-Intelligence-Centric Routing Algorithm for Wireless Sensor Networks," *Sensors*, vol. 20, no. 18, pág. 5164, 2020.

[7] T. Mens, A. Serebrenik y A. Cleve, *Evolving Software Systems*. Springer, 2014, vol. 190.

[8] J. Kennedy, "Swarm intelligence," en *Handbook of nature-inspired and innovative computing*, Springer, 2006, págs. 187-219.

[9] G. Beni, "From swarm intelligence to swarm robotics," en *International Workshop on Swarm Robotics*, Springer, 2004, págs. 1-9.

[10] X. Chen, G. Liu, N. Xiong, Y. Su y G. Chen, "A survey of swarm intelligence techniques in VLSI routing problems," *IEEE Access*, vol. 8, págs. 26 266-26 292, 2020.

[11] J. Networks, *Getting started with networking*, Juniper online education, Extraído de: `https://learningportal.juniper.net/juniper/user_activity_info.aspx?id=769`.

[12] J. Kurose y K. Ross, *Computer Networking: A Top-down Approach*. Pearson, 2016, ISBN: 9781292153599. dirección: `https://books.google.com.gt/books?id=lQNZMQAACAAJ`.

[13] H. Berkowitz, *Designing Addressing Architectures for Routing and Switching*, ép. Macmillan network architecture and development series. Macmillan Technical Pub., 1999, ISBN: 9781578700592. dirección: https://books.google.com.gt/books?id=0yQfAQAAIAAJ.

[14] S. K. Kushwaha, A. Kumar y N. Kumar, "Routing Protocols and Challenges Faced in Ad hoc Wireless Networks," *Adv. Electron. Electric Eng*, vol. 4, n.º 2, págs. 207-212, 2014.

[15] N. Shabbir y S. R. Hassan, "Routing protocols for wireless sensor networks (WSNs)," *Wireless Sensor Networks-Insights and Innovations*, 2017.

[16] F. Chan y M. Tiwari, *Swarm Intelligence: focus on ant and particle swarm optimization*. BoD–Books on Demand, 2007.

[17] B. Zuckerman, D. Jefferson, D. R. Jefferson y col., *Human population and the environmental crisis*. Jones & Bartlett Learning, 1996.

[18] D. Merkle y M. Middendorf, *Ant Colony Optimization, Marco Dorigo, Thomas Stützle, MIT Press (2004), ISBN: 0-262-04219-3*, 2006.

[19] J.-L. Deneubourg, S. Aron, S. Goss y J. M. Pasteels, "The self-organizing exploratory pattern of the argentine ant," *Journal of insect behavior*, vol. 3, n.º 2, págs. 159-168, 1990.

[20] S. Jun, L. Choi-Hong y W. Xiao-Jun, *Particle Swarm Optimization: Classical and Quatum Perspectives*. Taylor & Francis Group, LLC, 2012.

[21] A. Bensky, *Short-range wireless communication*. Newnes, 2019.

[22] H. L. Bertoni, *Radio propagation for modern wireless systems*. Pearson Education, 1999.

[23] G. Di Caro y M. Dorigo, "Ant colony optimization and its application to adaptive routing in telecommunication networks," Tesis doct., PhD thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles . . ., 2004.

[24] ——, "AntNet: Distributed stigmergetic control for communications networks," *Journal of Artificial Intelligence Research*, vol. 9, págs. 317-365, 1998.

[25] C. Steigner, H. Dickel y T. Keupen, "RIP-MTI: A new way to cope with routing loops," en *Seventh International Conference on Networking (icn 2008)*, IEEE, 2008, págs. 626-632.

[26] A. Varga, "OMNeT++," en *Modeling and tools for network simulation*, Springer, 2010, págs. 35-59.

[27] Y. Xue, H. S. Lee, M. Yang, P. Kumarawadu, H. H. Ghenniwa y W. Shen, "Performance evaluation of ns-2 simulator for wireless sensor networks," en *2007 Canadian Conference on Electrical and Computer Engineering*, IEEE, 2007, págs. 1372-1375.

[28] A. M. Zungeru, L.-M. Ang y K. P. Seng, "Classical and swarm intelligence based routing protocols for wireless sensor networks: A survey and comparison," *Journal of Network and Computer Applications*, vol. 35, no. 5, págs. 1508-1536, 2012.

[29] Y. Zhang, G. Simon y G. Balogh, "High-level sensor network simulations for routing performance evaluations," en *Third International Conference on Networked Sensing Systems (INSS06)*, Citeseer, 2006.

[30] G. Simon, P. Volgyesi, M. Maróti y A. Ledeczi, "Simulation-based optimization of communication protocols for large-scale wireless sensor networks," en *IEEE aerospace conference*, vol. 3, 2003, págs. 31 339-31 346.

[31] S. Vasudevan, J. Kurose y D. Towsley, "On neighbor discovery in wireless networks with directional antennas," en *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, IEEE, vol. 4, 2005, págs. 2502-2512.

[32] S. Vasudevan, D. Towsley, D. Goeckel y R. Khalili, "Neighbor discovery in wireless networks and the coupon collector's problem," en *Proceedings of the 15th annual international conference on Mobile computing and networking*, 2009, págs. 181-192.

CAPÍTULO 13

Glosario

Enjambre: es una colección de individuos con un objetivo común, regularmente con funcionalidades limitadas a nivel individual pero con características globales más amplias. 18

Enrutador (*router*): es un miembro de la red que se encarga de realizar las tareas de enrutamiento y direccionamiento dentro de la red, coordinando con otros enrutadores las rutas para los diferentes destinos y encaminando los paquetes por estas. 14

Estigmergia: es un proceso de comunicación indirecto entre individuos por medio de alteraciones en el ambiente. 18

IP: protocolo de internet (*internet protocol*) define características de la capa de red como el direccionamiento y la naturaleza de la comunicación (orientada a paquetes). 14

LAN: red de área local (*local area network*) se define formalmente como el conjunto de dispositivos bajo un mismo dominio de *broadcast*. 13

Optimización: es la búsqueda de la generación del mejor resultado posible sujeto a diferentes restricciones. Se plantea formalmente como hallar el mínimo de una función de costo. 19

Siguiente salto: (*next hop*) es la dirección del elemento de red al que debe enviarse un paquete para determinado destino. Conocer el siguiente salto para los destinos de interés es el objetivo de los protocolos de erutamiento. 27

Tabla de rutas: es una tabla que contiene registros que especifican redes, siguientes saltos y un medio físico de salida para paquetes con base en la dirección de destino. 16

WAN: red de área amplia (*wide area network*) es una red relavitamente amplia, no sujeta a solo una ubicación geográfica, compuesta por varias LANs. 13