FACULTAD DE INFORMÁTICA

UNIVERSIDAD POLITÉCNICA DE MADRID

TESIS DE MÁSTER

MÁSTER UNIVERSITARIO EN INTELIGENCIA ARTIFICIAL

ALGORITMOS INSPIRADOS EN SWARM INTELLIGENCE PARA EL ENRUTAMIENTO EN REDES DE TELECOMUNICACIONES

AUTOR: Arturo Imas Rodríguez

TUTOR: Nik Swoboda

JULIO, 2013

Resumen

En las últimas décadas hemos visto un rápido desarrollo de las redes de telecomunicación llegando a todos los rincones de la sociedad, bien a través de cable o bien de forma inalámbrica. Dichas redes, que cada vez son más grandes, dinámicas y complejas, integrando un mayor número de servicios y protocolos, requieren de un componente central que es el enrutamiento. El enrutamiento determina las estrategias a utilizar por los nodos de una red para encontrar las rutas óptimas entre un origen y un destino en el envío de información. Resulta difícil conseguir una estrategia que se adapte a este tipo de entornos altamente dinámicos, complejos y con un alto grado de heterogeneidad. Los algoritmos clásicos propuestos hasta la fecha suelen ser algoritmos centralizados que tratan de gestionar una arquitectura claramente distribuida, que en escenarios estacionarios pueden mantener un buen rendimiento, pero que no funcionan bien en escenarios donde se dan continuos cambios en la topología de red o en los patrones de tráfico. Es necesario proponer nuevos algoritmos que permitan el enrutamiento de forma distribuida, más adaptables a los cambios, robustos y escalables. Aquí vamos a tratar de hacer una revisión de los algoritmos propuestos inspirados en la naturaleza, particularmente en los comportamientos colectivos de sociedades de insectos. Veremos cómo de una forma descentralizada y auto-organizada, mediante agentes simples e interacciones locales, podemos alcanzar un comportamiento global "inteligente" que cumpla dichas cualidades. Por último proponemos Abira, un algoritmo ACO basado en AntNet-FA que trata de mejorar el rendimiento y la convergencia introduciendo mecanismos de exploración, de feedback negativo como la penalización y de comunicación de de las mejores rutas. Tras realizar una simulación y comparar los resultados con el algoritmo original, vemos que Abira muestra un mejor rendimiento.

Abstract

In recent decades we have seen the rapid development and deployment of telecommunications networks, they reach all corners of society, either through wired or wireless connection. Such networks, which are increasingly large, dynamic and complex, and which integrate a greater number of services and protocols, require of a core component which is routing. Routing determines the strategies to be used by network nodes to find appropriate paths between a source and a destination for sending information. It is difficult to get a strategy that fits this type of highly complex, dynamic environments, with a high level of heterogeneity. Classical algorithms proposed to date are usually centralized algorithms trying to manage a clearly distributed architecture, which before a stationary stage can maintain good performance, but do not work well in scenarios where there are continuous changes in the network topology or traffic patterns. Therefore, It is necessary to propose novel algorithms that allow distributed routing, more adaptable to changes, robust and scalable. Here we will try to review the algorithms inspired by nature, particularly the collective behaviors of insect societies. We will see how in a decentralized and self-organized manner, by simple agents and local interactions, we can achieve an "intelligent" global behavior that meets these qualities. Finally we propose Abira, an ACO algorithm based on AntNet-FA, which aims to improve convergence and performance by introducing some mechanisms of exploration, negative feedback (penalization) and the communication of the best routes. After performing a simulation and comparing the results with the original algorithm, I will conclude that Abira shows a better performance in the tested scenarios.

Índice

Índice de figuras

Índice de tablas

Parte I

Introducción

Las redes de telecomunicación han evolucionado mucho con el paso de los años. Desde las primeras redes de conmutación de circuitos, como el teléfono, en las que se utilizaban transmisiones extremo a extremo, hasta las redes de conmutación de paquetes como Internet. En las últimas décadas hemos visto un rápido desarrollo y despliegue de este tipo de redes llegando a todos los rincones de la sociedad, bien a través de cable o bien de forma inalámbrica. Además, cada vez son más grandes, complejas y dinámicas, integrando un mayor número de servicios, tecnologías y protocolos en los que la información se empaqueta y se envía a uno o varios destinos.

El enrutamiento determina la estrategia a utilizar por los nodos de una red para establecer una ruta entre un origen y un destino sobre la que enviar la información. Resulta difícil conseguir una estrategia que se adapte a este tipo de entornos altamente dinámicos, complejos y con un alto grado de heterogeneidad. Los algoritmos clásicos propuestos hasta la fecha suelen ser algoritmos centralizados que tratan de gestionar una arquitectura claramente distribuida, con un buen rendimiento en escenarios estacionarios, pero no así en escenarios donde los cambios en la topología de red o en los patrones de tráfico son continuos. Por este motivo creemos necesario proponer nuevos algoritmos de enrutamiento distribuidos, más adaptables a los cambios, robustos y escalables.

El comportamiento colectivo en las sociedades de insectos ha sido un éxito a la hora de resolver problemas en la naturaleza. De hecho, gracias a su capacidad de auto-organización han sido capaces de colonizar gran parte del planeta durante millones de años. Su éxito radica en su comportamiento colectivo perfectamente integrado y auto-organizado a la hora de llevar cabo una tarea. Como ejemplos más significativos, encontramos dos técnicas: el concepto de estigmergia en las colonias de hormigas (Grasse, 1959), una forma de comunicación indirecta entre los individuos mediante modificaciones realizadas en el entorno, o la técnica de "la danza de la abeja" (en inglés *waggle dance*), una forma de comunicación en los enjambres de abejas que permite transmitir información de distancia, dirección y calidad de una fuente de alimento.

Observar estos comportamientos en sociedades de insectos permite entenderlos y crear modelos que más tarde podrán ser adaptados y aplicados a la resolución de problemas similares en otro tipo de entornos. Muchos son los algoritmos que se inspiran en el comportamiento de las colonias de hormigas para encontrar el camino más corto entre el nido y la fuente de alimento mediante el mecanismo de segregación de feromonas. Pioneros fueron los trabajos de Goss, Beckers y Deneubourg [Goss et al., 1990; Beckers et al., 1992]. Este tipo de algoritmos pueden aplicarse a problemas complejos y problemas de optimización, ámbito en el que Dorigo propuso los algoritmos ACO [Dorigo, 1992; Dorigo et al., 2000].

La ventaja de estos planteamientos, frente a arquitecturas clásicas, es la posibilidad de construir sistemas adaptativos, flexibles, descentralizados y robustos. Pero también poseen

varias desventajas. Este tipo de sistemas son complejos, dinámicos y estocásticos, por lo que difícilmente puede predecirse su comportamiento.

Trataremos de hacer una revisión de los algoritmos de enrutamiento ya propuestos inspirados en la naturaleza, particularmente en los comportamientos colectivos de dos sociedades de insectos: colonias de hormigas y enjambres de abejas. Veremos cómo de una forma descentralizada y auto-organizada, utilizando agentes simples e interacciones locales entre ellos, podemos alcanzar un comportamiento global "inteligente" que permita obtener un mejor resultado con respecto a las alternativas convencionales. Actualmente existe una gran cantidad de trabajos dedicados a este ámbito, que podrían dividirse principalmente en tres categorías: redes cableadas sin conexión (con entrega de mejor esfuerzo o best-effort), redes cableadas orientadas a conexión con o sin calidad de servicio (QoS) y redes inalámbricas o redes MANET. Aquí nos centraremos en las primeras, revisando las principales propuestas: ABC [Schoonderwoerd et al., 1997], AntNet [Di Caro and Dorigo, 1998a,b] y BeeHive [Wedde et al., 2004].

Más adelante proponemos un nuevo algoritmo, *Abira*, basado en el algoritmo AntNet-FA[Di Caro and Dorigo, 1998a,b] al que se le incorporan un conjunto de técnicas adicionales para incrementar ciertas capacidades, como son la capacidad de exploración y la capacidad de adaptarse más rápido a los cambios de red. De ese modo se incrementa el tiempo de convergencia, la probabilidad de encontrar rutas óptimas y de evitar rutas sub-óptimas, lo que se traduce en un mayor rendimiento.

Para validar esta nueva propuesta, se diseña y pone en práctica un entorno de simulación compuesto por diferentes escenarios que consideramos suficientemente realistas. Una vez obtenidos los resultados y realizando una comparativa con el algoritmo original, se puede concluir que el rendimiento obtenido del nuevo algoritmo es mejor cuando la red sufre cambios a nivel de topología o de patrones de tráfico.

Este trabajo se organiza de la siguiente manera: en la primera parte se describe de forma general en qué consiste *swarm intelligence*, los conceptos básicos de red y por qué aplicar ACO al problema del enrutamiento; en la segunda parte se realiza un estudio del estado del arte en este ámbito, describiendo los principales algoritmos propuestos para más tarde comentar alguna de sus variantes; en la tercera parte se propone *Abira*, un algoritmo de enrutamiento basado en AntNet-FA, realizando una simulación y comparando los resultados; y por último, en la cuarta parte, se discuten las conclusiones sobre los resultados obtenidos de este trabajo y se describen algunas líneas de trabajo futuro.

1. Swarm intelligence

Existen muchas definiciones para la inteligencia de enjambre o swarm intelligence (SI). Básicamente, se trata de una rama de la inteligencia artificial inspirada en la naturaleza, concretamente en sistemas biológicos en los que pueden observarse comportamientos colectivos. Existen varios ejemplos como las colonias de hormigas, enjambres de abejas, comunidades de bacterias o bandadas de pájaros. Uno de los principales focos de investigación y en el que más nos centraremos serán las colonias de insectos.

La primera vez que se utilizó la expresión *swarm intelligence* fue en [Beni and Wang, 1989], un trabajo que se componía de agentes muy simples que formaban patrones y se auto-organizaban mediante interacciones con sus vecinos. El término se había usado en un ámbito muy específico y una década más tarde en [Bonabeau et al., 1999] se definió como: "the emergent collective intelligence of groups of simple agents". Una definición más amplia y bastante utilizada es la siguiente:

> "Swarm Intelligence (SI) is the property of a system whereby the collective behaviours of (unsophisticated) agents interacting locally with their environment cause coherent functional global patterns to emerge. SI provides a basis with which it is possible to explore collective (or distributed) problem solving without centralized control or the provision of a global model." (Stan Franklin, Coordination without Communication, talk at Memphis Univ., USA, 1996).

Algunos ejemplos de las tareas que podemos observar en la naturaleza son: la búsqueda de alimento en grupo, la construcción de nidos, el transporte cooperativo, la protección de la colonia o la clasificación colectiva. Para llevar a cabo estas tareas, los agentes siguen reglas muy simples y sin un control centralizado, dando lugar en muchos casos a la división de tareas o la especialización para coordinar sus acciones.

Una característica esencial en este tipo de comportamientos es la *auto-organización*, un término que surge del mundo de la física y de la química, y que posteriormente se extendió al mundo de la biología. En [Bonabeau et al., 1999] lo definen como:

> "Self-organization is a set of dynamical mechanisms whereby structures appear at the global level of a system from interactions among its lower-level components. The rules specifying the interactions among the system's constituent units are executed on the basis of purely local information, without reference to the global pattern, which is an emergent property of the system rather than a property imposed upon the system by an external ordering influence."

La auto-organización requiere de interacciones entre individuos. Este tipo de interacciones parten de una comunicación entre los individuos que puede darse de forma directa, cuando la comunicación se realiza de individuo a individuo, o de forma indirecta, cuando la comunicación surge de los cambios que cada uno de los individuos realiza en el entorno. Cuando el individuo modifica el entorno, el cambio afecta sobre el comportamiento del resto de la colonia y sobre suyo propio. Este mecanismo es denominado *estigmergia* y es utilizado principalmente por los insectos. El concepto surgió de la mano de Pierre-Paul Grassé (1959) al estudiar las colonias de hormigas, donde sus miembros depositan feromonas en el entorno como medio de comunicación.

Son varias las ventajas de la estigmergia. Dado que la mayor parte de la comunicación se realiza de forma indirecta, se reduce la comunicación entre los agentes y no se requiere de un ente central que la gestione. Facilita a sus miembros adaptarse mejor al entorno y les

ofrece mucha más información de la que podrían reunir por propios medios. Y por último, posibilita que la colonia pueda responder de forma colectiva a cualquier cambio, ya sea un cambio producido por un miembro de su propia colonia o por una acción externa.

El modo en que interactúan cada uno de los miembros de una colonia sienta las bases para que a partir de la auto-organización surja un patrón global o comportamiento colectivo. Podemos diferenciar dos modos de interacción:

- La retroalimentación positiva, que promueve los cambios en el sistema produciendo un efecto de amplificación. Por ejemplo, cuando una hormiga encuentra una fuente de alimento otras hormigas siguen el mismo camino, reforzando dicha trayectoria y provocando que más hormigas se unan a él.
- La retroalimentación negativa, que previene los cambios en el sistema. Permite estabilizar el sistema contrarrestando la retroalimentación positiva y evitando posibles fluctuaciones no deseadas. Siguiendo el ejemplo anterior, cuando una hormiga previamente ha encontrado una fuente de alimento se establece un camino. Al principio el camino será reforzado si un mayor número de hormigas lo eligen y depositan más feromona. Pero con el tiempo la cantidad de alimento disminuye y menos hormigas viajarán hasta él realizando el camino, por lo que el refuerzo cada vez será menor y la feromona se evaporará.

Una característica fundamental, además de la interacción entre los individuos, es el hecho de que las decisiones de los individuos mantengan una cierta aleatoriedad. De ese modo se amplia la capacidad de exploración y es posible descubrir nuevas alternativas que siguiendo la estrategia habitual no hubiera sido posible encontrar. Por ejemplo, una hormiga que sigue un camino previamente establecido hacia una fuente de alimento puede abandonarlo de forma aleatoria y caminar sin un rumbo fijo, pudiendo encontrar así nuevas fuentes de alimento. Aunque en muchas ocasiones el tiempo empleado en caminar sin un rumbo fijo no da resultados, es compensado por las ocasiones en las que se descubre una nueva fuente de alimento.

Respecto a las características de los agentes que podemos observar en este tipo de colonias, destaca el hecho de que sus capacidades cognitivas son muy limitadas. Es decir, sus capacidades no se basan en el aprendizaje, la memorización o el razonamiento, sino en el hecho de mantener un comportamiento individual muy fácil de describir condicionado por el entorno. Esto hace que su modelado y diseño sea más simple. Al mismo tiempo, son agentes que pueden solucionar problemas de una manera más flexible y robusta. Más flexible por el hecho de que pueden adaptarse fácilmente a los cambios del entorno, y más robustos porque la colonia puede mantener sus funciones y capacidades incluso aún después de que alguno de sus miembros no realicen sus tareas, ya sea por fallos o por su desaparición del sistema.

Derivada de esa simplicidad en el diseño, el hecho de utilizar el concepto de “agente” en swarm intelligence podría ser discutible. En [Kennedy et al., 2001] consideran que un agente debe tener cualidades como autonomía y especialización, que en este caso no se cumplen debido a que consideran que los miembros “swarm” son homogéneos y tienden

a seguir un programa de forma explícita. En este texto es posible que aparezca el término agente haciendo referencia a un miembro "swarm" por similitud, aunque no sea un agente propiamente dicho.

A partir de las metáforas extraídas de la naturaleza, algunos de los algoritmos o metaheurísticas desarrolladas han sido las siguientes:

ACO Ana Colony Optimization. Propuesto inicialmente en [Dorigo, 1992]. Se trata de un algoritmo probabilístico inspirado en las colonias de hormigas. Trata de solucionar problemas de búsqueda de trayectorias óptimas o cuasi-óptimas en grafos, como puede ser el problema del viajante. Se basa en la mecánica de las hormigas de depositar feromonas en las trayectorias utilizadas y que derivan en una fuente de alimento. Las mejores trayectorias atraerán un mayor número de hormigas que a su vez reforzarán depositando una nueva cantidad de feromona, dando así lugar a una retroalimentación positiva. En cambio, las trayectorias sub-óptimas sufrirán la evaporación de la feromona al obtener un menor número de hormigas que las transiten, evitando así converger en óptimos locales.

ABC Artificial Bee Colony. Propuesto por [Karaboga, 2005]. Se trata de un algoritmo de optimización inspirado en los enjambres de abejas. Se basa en la búsqueda de alimento de las abejas, donde la población se divide en abejas empleadas o trabajadoras, abejas espectadoras y abejas exploradoras. Distintas fuentes de alimento con un número equivalente de abejas se reparten en el espacio de búsqueda. Cada fuente contiene una cantidad distinta de néctar que equivale al fitness. Las abejas trabajadoras conocen las fuentes de alimento de su entorno y las seleccionan según su fitness. Las abejas espectadoras observan el intercambio de información de las trabajadoras, seleccionando las fuentes de mejor fitness. Las abejas exploradoras en cambio seleccionan fuentes de alimento de forma aleatoria, permitiendo ampliar el espacio de búsqueda.

PSO Particle Swarm Optimization. Propuesto por [Kennedy et al., 2001]. Se trata de un algoritmo de optimización inspirado en las bandadas de pájaros o bancos de peces. Dado un espacio de soluciones, se intenta mejorar una solución candidata con respecto a un criterio de calidad o fitness. La idea es moverse por el espacio de búsqueda en base a una idea de velocidad que engloba la inercia o movimiento de la partícula en el pasado, los resultados de la propia partícula y los resultados de sus vecinos. El objetivo es encontrar una mejor solución, que en ningún caso puede asegurar sea la óptima.

2. Modelo de red

El modelo de red utilizado en la actualidad y en el que aquí nos centraremos es el modelo de conmutación de paquetes, similar al que podemos encontrar en Internet.

En este tipo de redes la información se divide en paquetes de un determinado tamaño conocido. Cada paquete a su vez se divide en una cabecera con información de control, un área de datos (payload) que contiene la información y opcionalmente una cola (trailer) que incluye algún mecanismo de control de errores. La información de control dispone de la información necesaria para establecer la ruta entre el origen y el destino, que consiste básicamente en la dirección del emisor, la dirección del destinatario y otra información que puede ser útil como la prioridad del paquete. Conocido el tamaño máximo de cada paquete, si la cantidad de información a enviar es superior deberá ser dividida en varios paquetes. Más tarde el receptor recibirá cada una de las partes que deberán ser reensambladas hasta obtener la información original, de modo que si alguna de las partes se pierde la información no puede ser recuperada. En cuanto a la ruta a utilizar por cada uno de los paquetes de datos desde el emisor hasta el receptor, puede variar dependiendo de si se trata de una red con un protocolo orientado a conexión o no orientado a conexión. En el primer caso la ruta es fija y no varía una vez establecida la conexión, mientras que en el segundo caso cada paquete puede tomar diferentes caminos desde el origen hasta el destino.

2.1. Arquitectura de red

En las redes de conmutación de paquetes existe un arquitectura de interconexión organizada por diferentes capas de protocolos. El modelo teórico OSI[1] define esta arquitectura como una pila de protocolos de 7 capas: capa física, capa de enlace de datos, capa de red, capa de transporte, capa de sesión, capa de presentación y capa de aplicación. Adicionalmente y desde una visión más práctica se presenta la arquitectura TCP/IP, que de algún modo reorganiza el modelo OSI, especialmente las capas superiores, para dar lugar a una arquitectura de 5 capas: capa física, capa de acceso a la red, capa internet, capa de transporte y capa de aplicación. Ambas pueden compararse gráficamente en la Figura 1.

El trabajo de enrutamiento se realiza en la capa de red (OSI) o capa de red internet (TCP/IP). Las capas inferiores sirven para dar acceso al medio físico, mientras que las superiores se encargan de procesar y controlar la información enviada y recibida.

2.1.1. Protocolo de red (IP)

El protocolo IP equivaldría a la capa de red de la arquitectura OSI o a la capa internet de TCP/IP. Esta capa es la que se encarga de que cada paquete llegue desde su origen hasta su destino mediante el enrutamiento.

Lo datos son dispuestos en paquetes con una cabecera de control que sigue el formato de la Figura 2. Dicha cabecera contiene información utilizada para la gestión del envío y la recepción del paquete mediante diferentes indicadores: número de identificación del paquete, direcciones de origen y destino, tamaño de los datos, número de fragmento (si los datos han sido divididos en varios paquetes), tiempo de vida y control de errores.

[1] Open System Interconnection, es un modelo estándar creado en 1984, ISO/IEC 7498-1.

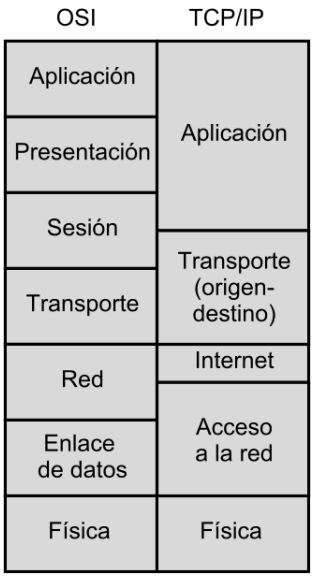


Figura 1: Equivalencia entre la arquitectura OSI y la arquitectura TCP/IP. Imagen extraída de [Stallings, 2006] (p41).

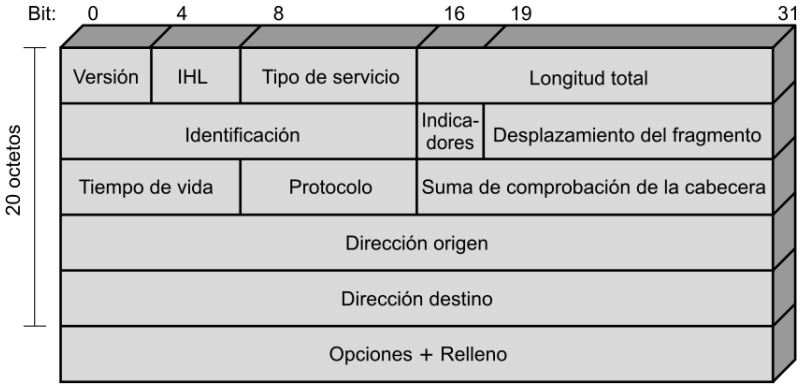


Figura 2: Formato de la cabecera IPv4. Imagen extraída de [Stallings, 2006] (p610).

Qué ofrece IP En este nivel se utiliza una estrategia sencilla como es *best-effort*, que trata simplemente de escoger la mejor ruta posible en la red. El mecanismo de enrutamiento podrá adaptarse mejor o peor al tráfico de la red, pero en una situación de congestión es posible que algunos paquetes sean descartados. Establece un tamaño máximo de paquete entre los nodos (MTU), que requiere dividir la información en paquetes del tamaño máximo para ser reensamblados en destino. También verifica de forma sencilla la integridad del paquete mediante sumas de comprobación con la información de las cabeceras.

Qué no ofrece IP En este nivel no se garantiza que el paquete sea entregado en su destino (es un servicio no fiable) y no pueden establecerse prioridades de entrega o una determinada calidad de servicio (QoS). No existen mecanismos de confirmación de la recepción del paquete, los paquetes podrían llegar desordenados o duplicados o incluso perderse en algún punto de la red. Para resolver este tipo de problemas será necesario utilizar una capa de transporte como puede ser TCP.

2.1.2. Protocolo de transporte (TCP/UDP)

La capa de transporte, capa número 4 en el modelo OSI, trata de solucionar el problema de IP que es ofrecer un servicio fiable. Para ello se establece un nuevo tipo de paquete o datagrama que incorpora una nueva cabecera, que además de establecer un enlace entre los datos y la capa de aplicación mediante "puertos", permite establecer mecanismos de control de errores, control de flujo y secuenciación de paquetes.

En general, existen dos tipos de protocolo en la capa de transporte, uno orientado a conexión y otro no orientado a conexión:

TCP Requiere establecer una conexión previa entre el emisor y el receptor para ofrecer un servicio fiable. Implementa mecanismos de confirmación de entrega y recepción, un sistema de control de flujo que garantiza el orden de los paquetes y capacidad de verificar los datos recibidos.

UDP No requiere establecer una conexión previa entre el emisor y el receptor, simplemente sirve de interfaz entre las capa de red y de aplicación. No garantiza la entrega, por lo que respecto a IP solamente ofrece la capacidad de verificar los datos

2.2. Enrutamiento

Una red equivale a un grafo ponderado $G = (V, E)$, donde V representa el conjunto de los distintos nodos de la red y E representa el conjunto de aristas o enlaces que conectan los nodos. Cada par de aristas (v_i, v_j) tiene un valor o peso asociado $p(v_i, v_j)$. El hecho de que G sea un grafo dirigido o un grafo no dirigido dependerá de si los enlaces de la red son unidireccionales (simétricos o asimétricos) o bidireccionales y simétricos. Dado un camino $P = (v_{i_1}, v_{i_2}, v_{i_3}, ..., v_{i_k})$ en G, el coste del camino se define como $C = p(v_{i_1}, v_{i_2}) + p(v_{i_2}, v_{i_3}) + \ldots + p(v_{i_{k-1}}, v_{i_k})$.

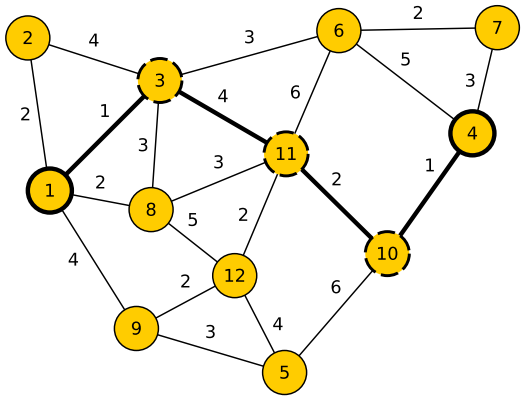


Figura 3: Red compuesta por 12 nodos con enlaces simétricos. Se establece un camino de coste mínimo entre el nodo 1 y el nodo 4.

El enrutamiento trata de establecer un camino entre un nodo origen y un nodo destino de forma que minimice el coste P, o dicho de otro modo, que maximice el rendimiento de la red. En la Figura 3 puede verse el ejemplo de una red con enlaces bidireccionales simétricos.

Para poder realizar el enrutamiento de paquetes, es necesario que cada uno de los nodos mantenga un modelo de la red, ya sea parcial o total, de forma que le permita en cierta medida conocer su topología y estado. Cada nodo mantiene una tabla de rutas con una estructura de datos común que le permite definir la política de reenvío. Dado un nodo origen s, sus entradas en la tabla de rutas reflejarán el nodo vecino al que reenviar el paquete si su objetivo es alcanzar el destino d. La forma en que se definen las tablas de rutas de cada uno de los nodos sigue un criterio o métrica determinada que determinará cómo serán las rutas que se formen a partir de ellas. Algunos criterios comúnmente utilizados suelen ser aquellos que tratan de establecer rutas minimizando el número de saltos o minimizando el tiempo de envío de los paquetes.

El algoritmo de enrutamiento debe adaptarse a la topología de la red, pero también a los patrones de tráfico que se den en ese momento, con el fin de evitar aquellas partes de la red que tengan problemas y/o sufran congestión. Esto requiere de la información y gestión local del nodo por un lado y de la información global del estado de la red por otra. En la parte local interviene la gestión de los recursos del propio nodo, como pueden ser los estados de los enlaces o las políticas de las colas del enlace. En la parte global en cambio interviene la información sobre el estado de la red intercambiada por los distintos nodos que la componen.

Tablas de rutas estáticas vs tablas de rutas dinámicas La tablas de rutas estática se establece de forma inicial bajo un criterio que comúnmente es el del camino más corto, sin tener en cuenta el estado de la red. Pueden establecerse rutas alternativas en caso de que alguno de los nodos deje de estar disponible. En cambio, las tablas de rutas dinámicas sí se actualizan y adaptan dependiendo del estado de la red, como pueden ser situaciones

de congestión y/o de errores, además de cambios en la topología.

Muestreo proactivo vs muestreo reactivo El muestro proactivo es un proceso en el que los nodos de la red están constantemente descubriendo rutas hacia diferentes destinos, ya sea para mejorar rutas ya disponibles en su tabla de enrutamiento o bien para generar nuevas rutas que puedan ser requeridas en un futuro. En cambio, el muestreo reactivo es un proceso de descubrimiento de rutas bajo demanda, generalmente para destinos que no hay información de enrutamiento.

Encaminamiento en nodos intermedios vs encaminamiento en origen A diferencia del encaminamiento tradicional en nodos intermedios, donde en cada salto el nodo decide el siguiente vecino al que reenviar el paquete en función de su tabla de enrutamiento, en la técnica de encaminamiento en el origen es el propio nodo origen el que especifica la ruta completa a seguir mediante la inclusión de una lista secuencial de nodos en el paquete de datos.

El hecho de que cada paquete pueda tomar diferentes rutas entre un emisor y un receptor tiene sus ventajas, como permitir un balanceo de carga en la red, adaptándose así a los diferentes patrones de tráfico de cada momento y repartiendo el trabajo entre los posibles caminos. También tiene sus inconvenientes, como la posibilidad de que los paquetes lleguen desordenados.

Establecer un encaminamiento en origen tiene otras ventajas, como la posibilidad de establecer criterios de prioridad o seguridad a la hora de enviar los paquetes.

3. Aplicación de ACO a las redes de telecomunicación

En 1992 Dorigo propuso Ant System en su tesis doctoral [Dorigo, 1992], un algoritmo inspirado en colonias de hormigas para dar solución al problema del viajante. A raíz de este algoritmo surgen diferentes variantes como AS, $\mathcal{MM}$AS o AS$_{rank}$, que tratan de aportar algunas mejoras respecto del algoritmo original[2]. Poco después se define la metaheurística ACO [Dorigo and Di Caro, 1999; Dorigo et al., 2000; Dorigo and Stützle, 2004] como un framework que engloba a esta familia de algoritmos.

Inspiración en las colonias de hormigas Observando en la naturaleza el comportamiento de la colonia de hormigas, se analiza paso a paso cómo resuelve el problema de establecer la ruta más corta entre el nido y la fuente de comida. En la Figura 4 se muestra gráficamente el proceso. Se establece como punto de origen n, en este caso el nido de la colonia de hormigas, y como punto de destino f , en este caso la fuente de alimento. Existen múltiples rutas que unen el origen n y el destino f, cada una de ellas con una distancia diferente. Las hormigas utilizan las feromonas, una sustancia química

[2]Puede verse una comparativa en [Mullen et al., 2009]

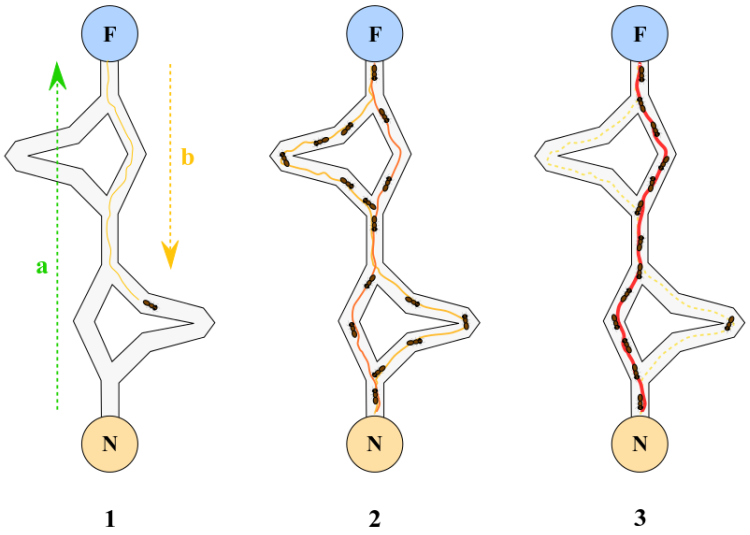


Figura 4: Ejemplo de proceso de búsqueda de la ruta más corta. Las hormigas parten del nido y al encontrar la fuente de alimento regresan depositando feromonas durante su trayecto. Las hormigas siguen aquellos caminos con mayor cantidad de feromonas que con el tiempo corresponden a las rutas más cortas, ya que el proceso de evaporación en las rutas más largas las hace menos atractivas. *Autor de la imagen: Johann Dréo*.

volátil con el paso del tiempo, como medio de comunicación con el resto de la colonia, ya sea segregando una nueva cantidad de feromonas durante su desplazamiento o bien siguiendo las feromonas que otras hormigas habían segregado en un momento anterior. Cuando una hormiga sale del nido n en busca de la fuente de alimento f, primero rastreará si existe un camino previo con restos de feromona. Si detecta feromonas, seleccionará con mayor probabilidad aquel camino en el que mayor cantidad de feromonas haya sido depositada. En cambio, si en su entorno no existe ningún sendero de feromonas que seguir, se establece un camino de forma totalmente aleatoria hasta encontrar la fuente de comida f. Una vez alcanzado f, la hormiga regresa al nido n seleccionando de nuevo con mayor probabilidad aquel camino con mayor cantidad de feromonas. Si la hormiga fue la primera en llegar al destino f, su camino de vuelta al nido n será el mismo camino de ida en sentido inverso, ya que es el único camino de feromonas que existe, siempre y cuando no sea lo suficientemente corto como para haberse evaporado, en cuyo caso la hormiga volverá a tomar un camino aleatorio. Otras hormigas saldrán desde el nido n hacia la fuente de comida f y todas seguirán utilizarán la misma mecánica de seguir y depositar las trayectorias con feromona o de moverse aleatoriamente en su defecto. Durante un periodo de tiempo, varios caminos serán utilizados para unir nido n y fuente de comida f, pero con el paso del tiempo aquellos caminos más cortos serán los más elegidos. Dos son las razones de que la colonia converja de ese modo a la solución óptima: al ser rutas más cortas la feromona depositada sufre menos evaporación que las rutas más largas, por lo que el nivel de feromona es mayor. Al mismo tiempo, cuantas más hormigas elijan la misma trayectoria, mayor nivel de feromonas acumulará la misma trayectoria, evitando depositar feromonas en el resto de rutas que con el paso del tiempo sufrirán aún una mayor evaporación.

Estructura de ACO Siguiendo la terminología utilizada en [Dorigo et al., 2000], ACO se utiliza para obtener un camino de coste mínimo factible sobre un grafo $G = (C, L, W)$. El hecho de que el camino sea factible dependerá de las restricciones impuestas Ω. $C = \{c_1, c_2, \ldots, c_n\}$ es el conjunto finito de componentes y $L = \left\{l_{c_i c_j} | c_i, c_j \in C\right\}$ es el conjunto de posibles conexiones entre los elementos de C. W es el conjunto de pesos asociados a los componentes de C, a los enlaces L o a ambos, y $\Omega(C, L)$ es un conjunto finito de restricciones sobre los elementos de C y L que puede variar en el tiempo.

Una vez obtenida la representación del grafo, una población de agentes similares a las hormigas recorre los diferentes enlaces explorando el espacio de búsqueda y recogiendo información. La información recogida se almacena en el sendero de feromonas τ_{ij} asociado a la conexión l_{ij}, codificando así la memoria a largo plazo de todo el proceso de búsqueda. El movimiento entre cada uno de los enlaces se realiza en base a una política estocástica con respecto a los valores de feromonas locales τ que representan la bondad de seleccionar un nodo. Combinando ambas funciones de recolección y selección, las hormigas construyen de forma incremental una solución con cada uno de los desplazamientos.

Además de la generación de hormigas, ACO incluye dos procedimientos: un proceso de evaporación de feromonas y "acciones de demonio". El proceso de evaporación decrementa la intensidad de los senderos de feromonas τ en el tiempo, lo que evita que el algoritmo converja rápidamente a posibles soluciones sub-óptimas y favorece la exploración de nuevas áreas del espacio de búsqueda. Las "acciones de demonio" son un proceso opcional que permiten realizar acciones de forma centralizada mientras se ejecutan paralelamente las acciones de la población de hormigas.

Aplicación de ACO ACO se aplica a problemas NP-complejos de optimización combinatoria que se caracterizan por su dinamismo, descentralización y complejidad computacional. Se trata de un problema dinámico, ya que las propiedades del gráfico que representa el problema pueden variar a lo largo del tiempo durante su resolución. Descentralizado, porque el sistema está computacionalmente distribuido, como es el caso actual de los nodos en las redes de telecomunicación. Y computacionalmente complejo, porque la dimensión de todas las soluciones del espacio de búsqueda es exponencial respecto a la representación del problema.

Relación entre ACO y las redes de telecomunicaciones Parece evidente la aplicación de la meta-heurística ACO a las redes de telecomunicación, donde un coste o en este caso un valor de feromona es asociado a cada par de nodos. Y se hace más evidente todavía cuando pensamos en protocolos no orientados a conexión, donde cada paquete de datos puede tomar una ruta diferente en base a la tablas de rutas del nodo en que se encuentre en ese momento.

Siguiendo la terminología anterior tenemos un grafo dirigido $G = (C, L, W)$, en el que $C = \{c_1, c_2, \ldots, c_n\}$ representa el conjunto de nodos de la red, $L = \left\{l_{c_i c_j} | c_i, c_j \in C\right\}$ el conjunto de enlaces o arcos dirigidos entre cada uno de los nodos y W el conjunto costes

para cada uno de los enlaces, que dependerán de las características físicas del enlace (ancho de banda y retraso del enlace) o del nivel de tráfico que soporten.

En este caso las hormigas son generadas y enviadas desde un nodo origen s hasta un nodo destino d, tratando de encontrar un camino factible entre ambos seleccionando nodos adyacentes. Los senderos de feromonas se acumulan en cada uno de los nodos en forma de tablas de enrutamiento, estructuradas en tantas filas como vecinos tenga el nodo actual k y en tantas columnas como destinos menos el nodo actual tenga la red, de modo que τ_{nd}^{k} representa la bondad de seleccionar el vecino n para alcanzar el destino d. Las tablas de feromonas pueden ser utilizadas tanto para mantener el proceso de búsqueda en el espacio de soluciones por parte de las hormigas, como para enrutar la información en la red.

La ventaja de utilizar ACO es que permite obtener una tabla de rutas que maximice algún criterio de evaluación de forma dinámica, adaptándose a los cambios de tráfico de la red al mismo tiempo que se produce el enrutamiento de la información. Además, como consecuencia de la selección estocástica del siguiente nodo en base a la bondad de las entradas de la tabla de feromonas, permite redistribuir la carga entre los diferentes nodos de la red.

Diferencias entre ACO y los algoritmos convencionales Los algoritmos clásicos construyen estimaciones de costes a partir de la observación pasiva de los flujos de tráfico de datos a nivel local, propagando posteriormente estas estimaciones a otros nodos de la red. Las hormigas, en cambio, exploran la red haciendo uso de las tablas de enrutamiento, permitiendo obtener desde cada uno de los nodos de la red una visión global del estado de la red, tanto de las rutas a nivel topológico y físico, como de los flujos del tráfico de información.

Parte II

Estado del arte

4. Algoritmos de enrutamiento para redes cableadas

Muchos han sido los trabajos dedicados a crear nuevos algoritmos de enrutamiento de red inspirados en sociedades de insectos. Principalmente dos han sido las fuentes de inspiración: las colonias de hormigas y las colonias de abejas, especialmente la primera. El origen está en la capacidad de las colonias de hormigas para descubrir las rutas más cortas entre el nido y las fuentes de comida utilizando el mecanismo de estigmergia, capacidad observada y modelada para dar lugar a la meta-heurística de optimización propuesta por Dorigo en su tesis de doctorado [Dorigo, 1992] conocida como ACO. Los algoritmos inspirados en la comunicación que se produce en los enjambres de abejas para la búsqueda de alimento son más recientes. Una meta-heurística de optimización inspirada en este comportamiento es la propuesta por Karaboga [Karaboga, 2005].

El primer algoritmo de enrutamiento de redes inspirado en sociedades de insectos, concretamente en colonias de hormigas, fue propuesto por Schoonderwoerd [Schoonderwoerd et al., 1997]. El algoritmo denominado ABC (Ant-Based Control) está dedicado a redes de conmutación de circuitos, donde los canales son medios de conexión de llamadas telefónicas y su objetivo es conseguir establecer y distribuir las llamadas a lo largo de la red. Siguiendo la misma filosofía poco después apareció el primer algoritmo de enrutamiento para redes de conmutación de paquetes de la mano de Di Caro y Dorigo [Di Caro and Dorigo, 1998a] denominado AntNet. Su objetivo es maximizar el rendimiento de la red distribuyendo el tráfico entre las diferentes rutas posibles, optimizando así los tiempos de entrega en base a las características de la ruta elegida: retraso, velocidad y nivel de congestión. AntNet fue puesto a prueba en diferentes simulaciones y comparado con algunos de los algoritmos convencionales más utilizados, resultando demostrar un rendimiento superior.

En lo referente a los algoritmos de enrutamiento inspirados en enjambres de abejas las propuestas son mucho más recientes. El primer algoritmo de este tipo fue presentado por Wedde et al. fue BeeHive [Wedde et al., 2004]. BeeHive se basa en cómo se comunican las abejas dentro del enjambre para indicar la distancia y la calidad de las fuentes de comida. Farooq, uno de los autores del algoritmo, realiza varias aportaciones durante años posteriores dentro de este campo [Farooq and Caro, 2008; Farooq, 2009]. En la Figura 5 puede observarse una taxonomía de los principales protocolos de enrutamiento para redes cableadas.

A raíz de estos dos algoritmos de referencia ABC y AntNet, han derivado multitud de nuevos algoritmos con diferentes variantes que tratan de solventar algunas deficiencias o que tratan de resolver nuevos escenarios que no se habían tenido en cuenta. Un claro ejemplo es AntNet-FA [Di Caro and Dorigo, 1998b], algoritmo propuesto por los mismos

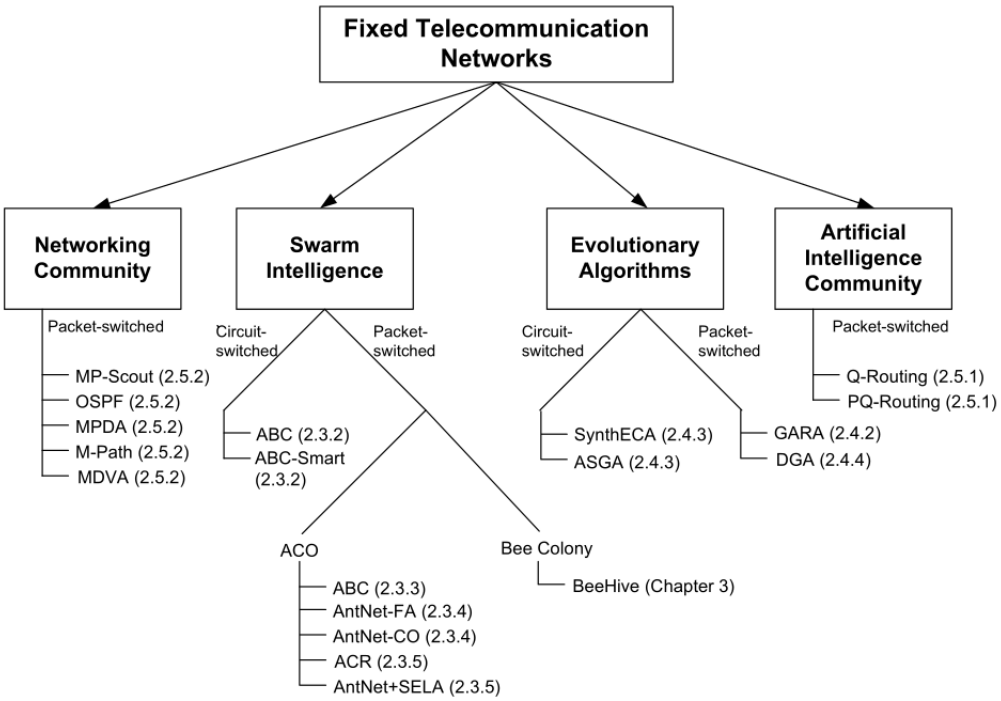


Figura 5: Taxonomía de protocolos de enrutamiento para redes cableadas. Imagen extraída de [Farooq, 2009], página 26.

autores de AntNet que incrementa el rendimiento modificando el modo en que se recorren y evalúan las rutas.

Además de algoritmos dedicados al enrutamiento para redes cableadas sin conexión como son los anteriores, han surgido nuevos algoritmos que tratan de adaptarse a las necesidades reales en el ámbito de las redes. Ninguna de las anteriores propuestas cumple con uno de los requisitos de red más demandados hoy en día, como es un protocolo orientado a conexión. De modo que surgen nuevas propuestas siguiendo la misma filosofía y que sí mantienen la conexión, como por ejemplo MACO [Sim and Sun, 2002]. Cumpliendo con un requisito más de los protocolos orientados a conexión como es garantizar una mínima calidad de servicio (QoS), aparecen algoritmos como AntNet-SELA [Di Caro and Vasilakos, 2000] o AntNet-QoS [Carrillo et al., 2004a]. Y si algo evoluciona rápidamente son las redes inalámbricas, con dispositivos móviles que requieren ser más eficientes debido en gran parte a la necesidad del uso de baterías. Es en éste último ámbito donde existe un mayor número de investigaciones, con múltiples propuestas como AntHocNet [Di Caro et al., 2005], ARA [Gunes et al., 2002], HOPNET [Wang and Lee, 2009] o BeeAdHoc [Wedde et al., 2005].

Existen también algunos estudios sobre el rendimiento de una implementación real de un algoritmo no orientado a conexión como AntNet y diferentes implementaciones TCP [Gadomska and Pacut, 2007a]. Se demuestra un buen rendimiento a pesar de perder el potencial de análisis previos como consecuencia de no poder redistribuir el tráfico entre diferentes rutas y de la necesidad de reenviar los paquetes de datos que con cumplen con el control de flujo que establece la capa de transporte.

El uso de algoritmos basados en swarm intelligence dedicados a la gestión de rutas y la distribución de tareas desata un interés creciente, ya no solamente en el ámbito de las telecomunicaciones, sino también en otros ámbitos como pueden ser la gestión de sistemas de computación en grid [Ludwig and Moallem, 2011] o la gestión de los sistemas de tráfico [Ghazy et al., 2012].

Dada la amplitud del número de trabajos propuestos en el ámbito hasta la fecha, no es posible realizar una revisión exhaustiva de cada una de las categorías, ya que daría lugar a un estudio demasiado largo, al mismo tiempo que dejaríamos de lado el objetivo inicial que es ver cómo se puede aplicar swarm intelligence al enrutamiento de redes, especialmente las meta-heurísticas ACO. Por lo tanto, nos centraremos solamente en algoritmos aplicados a redes cableadas no orientadas a conexión con entrega de mejor esfuerzo.

A continuación describimos de forma detallada los principales algoritmos de referencia en orden cronológico de aparición: ABC, AntNet y Beehive. Más adelante enumeramos alguna de las variantes más interesantes que han ido surgiendo en los últimos años. Por último, describimos las características que debería tener un algoritmo de enrutamiento equilibrado y algunas de lás técnicas que se han utilizado para alcanzar cada una de ellas.

4.1. ABC

El algoritmo presentado por [Schoonderwoerd et al., 1997], denominado ABC (Ant-Based Control), es la primera aplicación de un algoritmo ACO en tareas de enrutamiento, concretamente al enrutamiento de llamadas en redes telefónicas.

El hecho de considerar una red telefónica en lugar de una red de conmutación de paquetes da lugar a un escenario de características diferentes respecto al resto de algoritmos que describiremos a continuación. Aún así, el impacto que tuvo en su momento provocó un aluvión de nuevas propuestas dedicadas al enrutamiento de paquetes de datos.

ABC fue probado en un modelo de la red de British Telecom y su rendimiento a la hora de establecer llamadas fue comparado con un algoritmo basado en agentes móviles desarrollado Appleby y Steward en 1994 [Appleby and Steward, 1994]. Los resultados determinaban que ABC tenía un rendimiento significativamente superior, enrutando el tráfico y realizando al mismo tiempo un balanceo de carga de una forma descentralizada y robusta.

4.1.1. Modelo de red

Los nodos de la red de telecomunicaciones contemplada en ABC tienen las siguientes características:

- Cada nodo dispone de una capacidad máxima de conexiones simultáneas que puede manejar.

- Los enlaces entre cada uno de los nodos son bidireccionales, simétricos y de capacidad infinita. Dado un enlace, una conexión puede establecerse en ambos sentidos y el número de conexiones que soportan solamente está limitado por la capacidad de conexiones disponible en cada nodo.

- Cada uno de los nodos k en una red de N nodos mantiene una tabla de feromonas $\mathcal{T}^k$, estructurada en tantas filas como destinos $N-1$ posibles existan y en tantas filas como vecinos $|\mathcal{N}_k|$ tenga el nodo k en cuestión. Cada entrada τ_{nd} de la tabla representa la probabilidad de seleccionar el nodo vecino $n \in \mathcal{N}_k$ para alcanzar el destino d, de forma que $\sum \tau_{nd} = 1$.

4.1.2. Descripción del algoritmo

En cada paso de tiempo y de forma proactiva, se envían agentes hormiga desde cualquier nodo de la red s hacia diferentes destinos d seleccionados aleatoriamente. Desde el nodo actual en el que se encuentra el agente se selecciona un nodo intermedio n para alcanzar d en base a su probabilidad $\tau_{nd} \in \mathcal{T}^k$, dando lugar a una ruta $A_{s \to d}$ compuesta por los nodos $s \to n_t \to n_{t+1} \to \ldots \to n_{t_n} \to d$. En cada salto, el agente actualiza la entrada de la tabla de feromonas en el sentido inverso de su dirección[3], es decir, se actualiza la probabilidad τ_{is} de utilizar el vecino i para alcanzar el origen s, de modo que se incrementa el valor de la feromona respecto al nodo previo. Una vez alcanzado el destino d el agente hormiga es destruido.

Actualización de la tabla de feromonas Cuando un agente hormiga llega a cualquier nodo intermedio k de la ruta $A_{s \to d}$, la entrada de la tabla de feromonas τ_{fs}, correspondiente a seleccionar el enlace del nodo previo f desde el que acaba de realizarse el último salto para alcanzar s se actualiza del siguiente modo:

$$\tau_{fs} = \frac{\tau_{fs} + \triangle p}{1 + \triangle p}$$

donde $\triangle p$ es el incremento de feromona a aplicar al enlace utilizado. Al mismo tiempo, el incremento se ve normalizado en el resto de valores de feromonas de los nodos vecinos $\tau_{ns} \in \mathcal{T}^k$, $n \neq f$:

$$\tau_{ns} = \frac{\tau_{ns}}{1 + \triangle p}$$

El algoritmo utiliza un método de recompensa-inacción, de modo que los valores de feromona solamente pueden ser decrementados por normalización.

[3]La posibilidad de realizar la actualización en sentido inverso deriva de considerar que la red es simétrica.

Incremento de probabilidad $\triangle p$ El algoritmo se fija dos objetivos a la hora de generar rutas: encontrar rutas cortas y al mismo tiempo evitar aquellas zonas de la red más congestionadas. Por este motivo el valor $\triangle p$ representa el tiempo de vida de la hormiga, que se corresponde con el número de saltos que ha realizado. Cuanto más corta sea la ruta encontrada, mayor será el incremento de feromona $\triangle p$. Al mismo tiempo, al tiempo de vida se le añaden retrasos de forma proporcional al grado de congestión que presente cada uno de los nodos de la ruta. Aumentar el tiempo de vida de los caminos más congestionados se traduce en un incremento mayor, favoreciendo así escoger rutas alternativas. La forma utilizada para calcular estos valores es la siguiente:

$$\triangle p = \frac{0{,}08}{age} + 0{,}005 \qquad delay = \left\lfloor 80 \times e^{-0{,}075 \times s} \right\rfloor$$

donde s es la capacidad máxima de conexiones del nodo.

Enrutamiento de llamadas Recibida una nueva llamada en un nodo s con destino el nodo d, se genera una ruta seleccionando de forma determinista los nodos intermedios n que tengan una mayor probabilidad τ_{nd} para alcanzar d. La llamada será aceptada o rechazada dependiendo de si es posible o no encontrar capacidad disponible en cada uno de los nodos que componen la ruta.

Introducción de ruido Para evitar el estancamiento en algunos de los valores de la tabla de feromonas $\mathcal{T}^k$, se introduce un factor de ruido que permita descubrir rutas alternativas y de ese modo poder hacer frente a aquellas situaciones en las que se produzca un cambio en la red. Una hormiga generará una ruta aleatoria con probabilidad f y utilizará los valores de las tablas de feromonas con probabilidad $1 - f$.

4.2. AntNet

AntNet, propuesto por Di Caro y Dorigo [Di Caro and Dorigo, 1998a], es un algoritmo de enrutamiento basado en la meta-heurística ACO. Está inspirado en las técnicas desarrolladas por Marco Dorigo durante los años noventa [Dorigo, 1992] y en el algoritmo de enrutamiento de líneas telefónicas ABC [Schoonderwoerd et al., 1997]. Siguiendo la misma idea de las colonias de hormigas, se utilizan agentes que simulan hormigas para lograr la distribución y el enrutamiento de tráfico de una red de conmutación de paquetes cableada mediante una estrategia best-effort.

Los agentes "hormigas" se comunican entre sí de un modo indirecto y asíncrono, analizando y actualizando las rutas que conectan los nodos origen y destino mediante la segregación de feromonas. A diferencia de ABC, se utilizan dos tipos de hormigas: "forward ants" y "backward ants", dividiendo el análisis del estado de la red y la actualización de las tablas de rutas en dos fases diferentes.

Los agentes "forward ant" realizan el camino desde el origen hasta el destino, manteniendo una estructura de datos que consiste en un listado de los nodos visitados y los tiempos de viaje entre cada uno de ellos. En cada salto entre nodos intermedios utilizan las mismas estructuras de red que los paquetes de datos, analizando así el tráfico de red al obtener parámetros como el tiempo de envío o el estado de congestión de las colas de los nodos. Al llegar al nodo destino el agente "forward ant" se convierte en "backward ant", heredando toda su información y realizando el viaje en sentido inverso. En el camino desde el destino al origen se actualizan los nodos intermedios en base los parámetros registrados previamente en el camino de ida. El hecho de actualizar durante la fase "backward" puede evitar errores, como por ejemplo reforzar rutas cuando un agente está en un bucle sin ser consciente de ello.

Dado que es un algoritmo destinado a redes cableadas donde se asume una cierta estabilidad topológica, el algoritmo no prevee cambios en la estructura de la red. Es un algoritmo puramente proactivo, que necesita una fase inicial en la que la red se inunda de agentes para encontrar las mejores rutas y establecer las tablas de enrutamiento de cada uno de los nodos. Si una ruta desaparece o deja de funcionar debido a fallos de enlace, el algoritmo reaccionará incrementando el número de paquetes de datos en espera en las colas de los nodos.

El objetivo de AntNet es adaptarse a los cambios y fluctuaciones en los patrones de tráfico, realizando un balanceo de carga mediante la distribución de paquetes en diferentes rutas gracias a un enrutamiento estocástico. Aunque se muestra robusto, es un algoritmo en el que intervienen muchos parámetros que podrían afinarse para mejorar su rendimiento.

A continuación se describen en detalle las dos fases en las que se divide el comportamiento de los agentes: la fase hacia adelante o "forward phase" y la fase hacia atrás o "backward phase", y de cómo finalmente se realiza el enrutamiento de paquetes a partir de los modelos de red obtenidos en los nodos que la componen.

4.2.1. Fase forward

De forma proactiva, se envían agentes "forward ant" desde cada nodo de la red s en intervalos de tiempo regulares Δt hacia diferentes destinos de la red d seleccionados de forma aleatoria, dando lugar un camino $F_{s \to d}$.

El nodo de destino s es seleccionado de forma aleatoria, bien uniformemente (con baja probabilidad) o bien mediante un modelo probabilístico sesgado que favorece a los destinos más solicitados por los paquetes de datos:

$$p_d = \frac{f_{sd}}{\sum_{d=1}^{N} f_{sd}} \tag{1}$$

donde f_{sd} es el número de paquetes de datos que han sido enviados desde s a d.

La tasa de generación de hormigas $1/\triangle t$ es un parámetro fijo del algoritmo establecido en base a los experimentos realizados. El objetivo es encontrar un equilibrio entre el número

de hormigas que consigan reducir la varianza en las estimaciones y al mismo tiempo evitar una sobrecarga de tráfico de enrutamiento que podría tener un impacto negativo en el rendimiento.

La "forward ant" mantienen una estructura de memoria $\mathcal{H}$ en la que almacenan una lista de los nodos visitados $V_{v_0 \to v_m} = [v_0, v_1, ..., v_m]$, el tiempo de viaje transcurrido en cada uno de ellos $T_{v_0 \to v_m} = \left[t_{v_0 \to v_1}, t_{v_1 \to v_2}, ..., t_{v_{m-1} \to v_m}\right]$ y el momento en que fue creada.

Para realizar la ruta $F_{s \to d}$, la hormiga debe seleccionar en cada nodo intermedio k del trayecto el siguiente nodo vecino $n \in \mathcal{N}_k$ al que moverse. La selección se hace teniendo en cuenta la memoria privada de la hormiga $\mathcal{H}(k)$, los valores de la matriz de feromonas (o tabla de enrutamiento) $\tau_{nd} \in \mathcal{T}_k$ y el estado de las colas de cada uno de los enlaces $l_n \in \mathcal{L}^k$. l_n es un valor normalizado $[0,1]$ proporcional a longitud de la cola y los datos que están esperando ser enviados en el enlace del nodo n:

$$l_n = 1 - \frac{q_n}{\sum_{n=1}^{|\mathcal{N}_k|} q_n} \tag{2}$$

El caso más común es cuando no todos los nodos vecinos posibles han sido visitados $\exists n \mid n \notin V_{s \to k}$. Teniendo en cuenta τ_{nd} que representa el aprendizaje continuo de los agentes de forma colectiva y l_n que representa el tiempo de espera estimado en la cola, el vecino n es seleccionado en base a una probabilidad de salto P_{nd} definida de la siguiente manera:

$$\begin{cases} p_{nd} = \frac{\tau_{nd} + \alpha l_n}{1 + \alpha(|\mathcal{N}_k| - 1)} & \forall n \in \mathcal{N}_k \wedge n \notin V_{s \to k} \\ p_{nd} = 0 & eoc \end{cases} \tag{3}$$

donde$\alpha \in [0,1]$ define la importancia de cada uno de los factores a la hora de establecer la probabilidad. Con un valor cercano a 0 la probabilidad depende completamente de la tabla de feromonas y con un valor cercano a 1 del estado de las colas de enlace.

En caso de que todos los nodos vecinos hayan sido visitados $n \in V_{s \to k}\, \forall n \in \mathcal{N}_k$, se escoge uno con probabilidad uniforme, a excepción del nodo k desde el que el que llegó la hormiga con para evitar así bucles:

$$\begin{cases} p_{nd} = \frac{1}{|\mathcal{N}_k| - 1} & \forall n \in \mathcal{N}_k \wedge (n \neq v_{i-1} \vee |\mathcal{N}_k| = 1) \\ p_{nd} = 0 & eoc \end{cases} \tag{4}$$

Si al forzar a la hormiga a volver a un nodo previamente visitado se detecta un ciclo en su memoria privada $V_{k \to k}$, todos los nodos que componen el ciclo, así como los tiempos que se habían registrado $T_{v_k \to v_k}$ son eliminados.

En la transmisión de una hormiga de un nodo a otro, se comparten las mismas colas que los paquetes de datos, de forma que experimentan los los mismos problemas de congestión que los paquetes de datos permitiendo analizar mejor el estado de carga de la red. El tiempo en espera incrementará el tiempo total del trayecto.

La hormiga puede ser eliminada en dos posibles escenarios: si al detectar un ciclo el tiempo que lleva la hormiga dentro del mismo supone más de la mitad del tiempo total de la ruta $F_{s\to k}$ o si la hormiga no alcanza el destino d antes llegar al tiempo máximo de vida (*TTL* o time to live).

Cuando un agente "forward ant" $F_{s\to d}$ alcanza el destino d, se convierte en un agente "backward ant" $B_{d\to s}$ heredando toda su información con el objetivo de realizar el camino inverso. El trayecto $F_{s\to d}$ ya completo puede ser evaluado para obtener una medida de bondad.

4.2.2. Fase backward

El agente "backward ant" realiza el camino inverso del agente "forward ant", recorriendo los nodos intermedios k de la lista $V_{s\to d}$ desde d hasta s. A diferencia los agentes "forward ant", los agentes "backward ant" no comparten las mismas colas de enlace que los paquetes de datos sino que utilizan las colas de prioridad alta, ya que su objetivo es propagar rápidamente a los nodos las actualizaciones necesarias.

En base a los tiempos obtenidos $T_{s\to d}$ en el trayecto $F_{s\to d}$ se consigue una medida de bondad que permite actualizar las estructuras de información de cada uno de los nodos:

$\mathcal{M}^k$ Es un modelo estadístico que representa el estado local de la red del nodo actual k respecto al nodo de destino d.

$\mathcal{M}^k$ es un vector de $N-1$ estructuras $\left(\mu_d, \sigma_d^2, W_d\right)$, donde μ_d y σ_d^2 son la media y la varianza del tiempo $T_{k\to d}$ experimentado y W_d es el mejor tiempo obtenido en una ventana de w observaciones. Las estimaciones de la media y la varianza se realizan del siguiente modo:

$$\mu_d = \mu_d + \eta\left(o_{k\to d} - \mu_d\right) \tag{5}$$

$$\sigma_d^2 = \sigma_d^2 + \eta\left(\left(o_{k\to d} - \mu_d\right)^2 - \sigma_d^2\right) \tag{6}$$

donde $o_{k\to d}$ es el tiempo de viaje desde el nodo k hasta el nodo d obtenido en el camino de ida $T_{k\to d}$ y η es un factor que establece el peso de las anteriores muestras que afectan a la media.

El número máximo de observaciones está directamente ligado a η, calculado como:

$$w = 5(c/\eta)\, c \in (0,1] \tag{7}$$

El valor W_d se actualiza constantemente dentro de la ventana de tiempos w. Una vez alcanzado el límite de la ventana t_{w+1}, el valor de W_d se establece como el valor de la media en este instante$\mu_d\left(t_{w+1}\right)$ y la ventana se restablece volviendo a la posición inicial.

$\mathcal{T}^k$ Tabla de feromonas que representa el aprendizaje colectivo, organizada de forma similar a una tabla de rutas.

En $\mathcal{T}^k$ existe una entrada $\tau_{nd} \in [0,1]$ para cada nodo vecino $n \in \mathcal{N}_k \mid \sum_{n \in \mathcal{N}_k} \tau_{nd} = 1$, que representa la bondad de elegir el nodo vecino n para alcanzar d. A la hora de actualizar la tabla, se distinguen dos casos dependiendo de si la entrada del nodo a actualizar es el nodo desde el que llegó el agente tras el último salto τ_{fd} o si la entrada corresponde al resto de vecinos que no forman parte de la ruta τ_{nd} :

$$\begin{cases} \tau_{fd} = \tau_{fd} + r\left(1 - \tau_{fd}\right) & si\, f \in V_{k \to f \to d} \\ \tau_{nd} = \tau_{nd} - r\tau_{nd} & si\, \forall n \in \mathcal{N}_k, n \neq f \end{cases} \tag{8}$$

donde r es un valor de refuerzo adaptativo dependiente del tiempo experimentado $T_{k \to d}$:

$$r = c_1 \left(\frac{W}{T}\right) + c_2 \left(\frac{I_{sup} - W}{(I_{sup} - W) + (T - W)}\right) \tag{9}$$

- W es el límite inferior e I_{sup} es el límite superior de un intervalo de confianza estimado:

$$I_{sup} = \mu + z(\sigma/\sqrt{w}) \tag{10}$$

  Siendo $z = 1/\sqrt{1-\gamma}$, γ un coeficiente de confianza con un valor previamente establecido en un rango $[0{,}6, 0{,}8]$.

- c_1 y c_2 son los coeficientes que regulan la relevancia del primer y segundo término respectivamente. El primer factor mide cómo de bueno es el tiempo actual con respecto al mejor tiempo en una ventana de w muestras y el segundo es un factor de corrección que aporta estabilidad al tener en cuenta el intervalo de tiempos.

Cuando la entrada a actualizar es el vecino del que procede $B_{d \to s}$ en su camino inverso se produce un incremento en la probabilidad de τ_{fd}, siendo mayor el incremento ante menores valores de probabilidad con el objetivo de potenciar la capacidad de exploración. En cambio, cuando la entrada a actualizar es la del resto de nodos se produce un decremento de probabilidades por normalización.

En la Figura 6 puede verse un esquema completo de las estructuras de información de un nodo AntNet.

Dada la naturaleza del algoritmo, actualizar todos los nodos intermedios que componen la ruta seguida por el agente "forward ant" en $F_{s \to d}$ es correcto bajo condiciones estacionarias, pero puede no serlo en situaciones no estacionarias. Siendo $V_{k \to d}$ una subruta

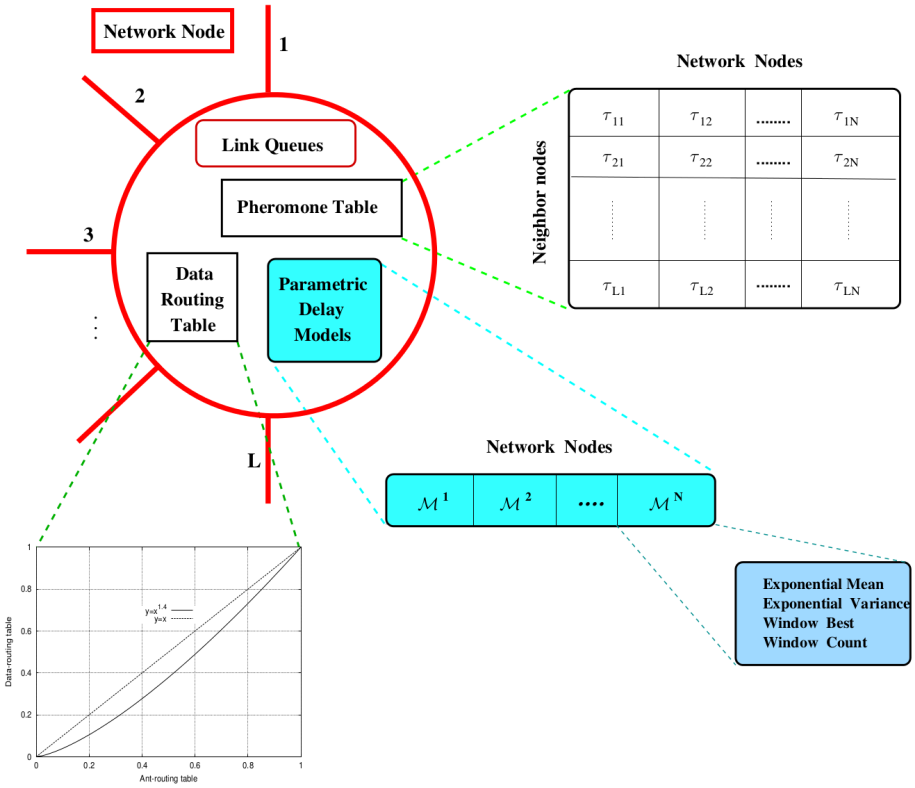


Figura 6: Estructura de datos de un nodo AntNet con L vecinos y una red de N nodos. Imagen extraída de [Di Caro, 2004], página 205.

de $F_{s\to d}$, los resultados obtenidos por un nuevo agente $F'_{k\to d}$ podría diferir mucho del los resultados de evaluar la subruta ante fluctuaciones de tráfico en la red. Por este motivo se establece un filtro que limite las actualizaciones en los nodos intermedios de modo que el tiempo $T_{k\to d}$ esté dentro de un intervalo de confianza estimado, en este caso por debajo de I_{sup}.

Cuando un agente "backward ant" llega al nodo origen se actualizan los modelos ($\mathcal{M}^k$ y $\mathcal{T}^k$) y el agente es eliminado.

4.2.3. Paquetes de datos

Los paquetes de datos comienzan a enviarse una vez completada la fase inicial, previamente acotada, en la que la red se inunda de agentes para ser analizada y establecer las rutas adecuadas.

El enrutamiento de los datos se realiza de forma estocástica a partir de una tabla de enrutamiento $\mathcal{R}^k$ que mantiene una estructura similar a la tabla de feromonas $\mathcal{T}^k$. Cada una de las entradas de $\varphi_{nd} \in \mathcal{R}^k$ del nodo k es obtenida a partir de una transformación exponencial de su correspondiente entrada $\tau_{nd} \in \mathcal{T}^k$ y su posterior normalización a 1, con el objetivo de propiciar la selección de los nodos más prometedores al mismo tiempo que no se descarte la posibilidad de distribuir el tráfico en múltiples rutas:

$$\varphi^k_{nd} = (\tau_{nd})^{\varepsilon}\,,\ \varphi^k_{nd} = \frac{\varphi^k_{nd}}{\sum\limits_{i\in\mathcal{N}_k} \varphi^k_{id}} \tag{11}$$

donde ε es el exponente de la función de transformación con un valor fijo $\varepsilon = 1{,}4$ como resultado de la experimentación.

4.3. Beehive

Los algoritmos de enrutamiento inspirados en la organización y el comportamiento de las abejas son muchos más recientes que los inspirados en colonias de hormigas. Beehive es el primer algoritmo de este tipo propuesto por Wedde et al. [Wedde et al., 2004].

En este caso, la inspiración proviene de la metodología de abejas de la miel a la hora de encontrar fuentes de comida. Cuando descubren una fuente de alimento la evalúan y regresan a la colmena para realizar una "danza" informando de la ubicación y distancia a la que se encuentra. Dependiendo de la calidad de la fuente, otras abejas se unirán al baile y a la recolección del alimento al que se hace referencia para ser explotado. Beehive abstrae este baile en una tabla de enrutamiento, donde los agentes abeja son lanzados desde un nodo origen para ser difundido entre sus diferentes vecinos. Una vez alcanzado un nodo cualquiera, intercambian la información necesaria para modelar el estado de la red y evaluar las posibles rutas.

Se proponen dos tipos de agentes: los agentes abeja de corta distancia, que obtienen información de los nodos cercanos al nodo origen limitados por un número máximo de saltos, y los agentes abeja de larga distancia, que a diferencia de los primeros viajan por toda la red.

Diferencias con AntNet A diferencia de AntNet, que utiliza el principio de estigmergia para la comunicación entre los agentes, aquí se utiliza un sistema blackboard o arquitectura de pizarra, donde la pizarra es actualizada por todos los agentes con sus resultados para alcanzar una solución de forma colectiva. Además, los nodos de la red se organizan por regiones de forma jerárquica, de modo que las estructuras de tablas de ruta a almacenar en la memoria de cada uno de los nodos es menor que si tuviesen que mantener una tabla con todos los vecinos y cada uno de los destinos de la red. Tampoco es necesario gestionar un modelo estadístico para mantener un control sobre los patrones de tráfico de la red. Respecto a los agentes, solamente se requiere el camino de ida, eliminando parte del tráfico de control generado, y tampoco necesitan mantener un listado de los nodos visitados con sus respectivos tiempos, por lo que se elimina la necesidad de utilizar una sincronización global entre los nodos de la red. En los experimentos realizados, Beehive muestra un rendimiento similar o superior comparado al obtenido con AntNet.

Una característica a tener en cuenta de este algoritmo es que fue diseñado bajo un framework propuesto en [Farooq and Caro, 2008], que tiene en cuenta las capacidades y limitaciones reales de la red. Esto permite ser implementado en redes físicas. Beehive que fue implementado en routers Linux para ser comparado con un algoritmo real como OSPF (capítulo 5 de [Farooq and Caro, 2008]). El rendimiento obtenido por Beehive respecto a OSPF fue significativamente superior.

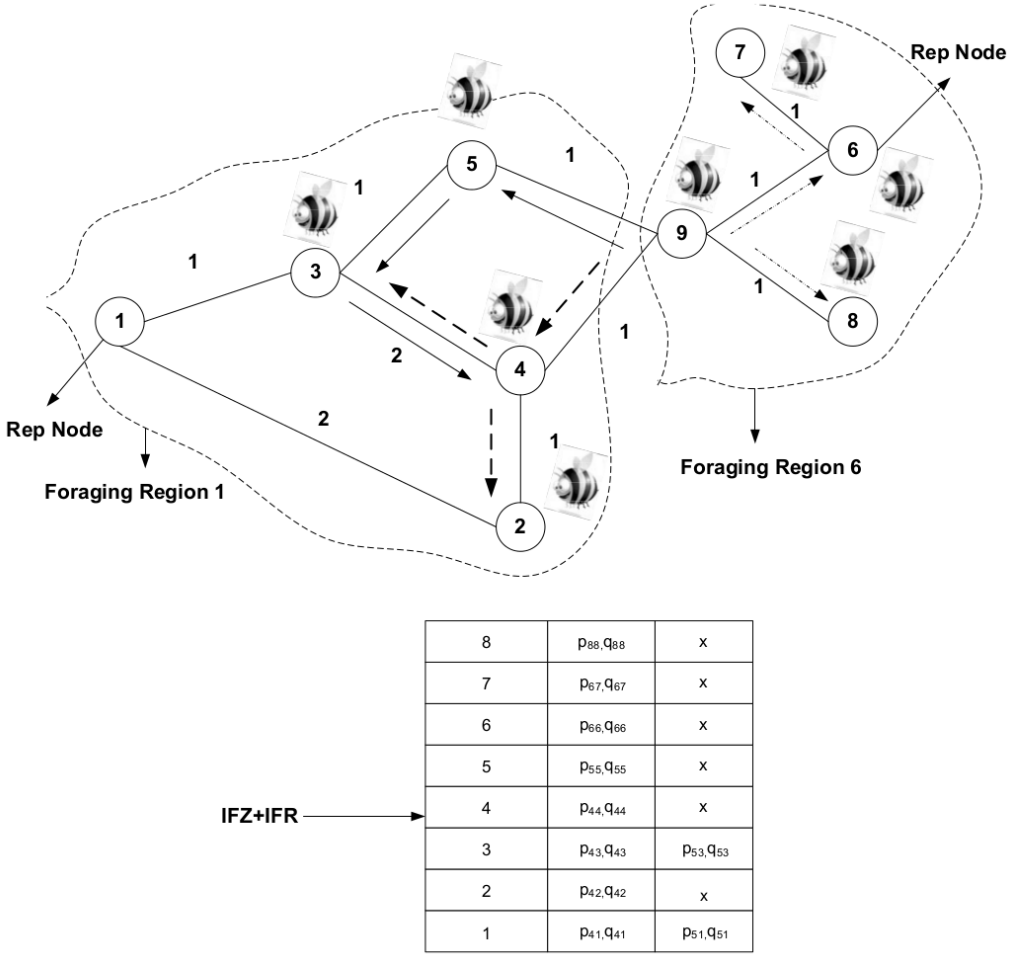
4.3.1. Descripción del algoritmo

La red se organiza en diferentes regiones: las zonas de forraje, que son las zonas que alcanzan cada uno de los nodos utilizando agentes abeja de corta distancia, y las regiones de forraje en las que se elige un nodo representativo. Como nodo respresentativo se elige al nodo que tenga una dirección IP más baja dentro de la región de forraje.

A partir de esta organización, la información que mantiene cada nodo se estructura de la siguiente manera:

IFZ Intra Foraging Zone. Es una tabla de rutas R_i compuesta de $N(i)$ filas y $D(i)$ columnas, donde $D(i)$ son los destinos dentro de la región de forraje y $N(i)$ el vecino a escoger. Cada entrada de la tabla $P_{jk} = \left(q_{jk,} p_{jk}\right)$ es un indicador del retraso que experimentará un paquete en la cola y en la propagación respectivamente al elegir el nodo vecino j para alcanzar el destino k.

IFR Inter Foraging Region. Es una tabla de rutas similar a la de IFZ, compuesta de $N(i)$ filas y $D(i)$ columnas, donde $D(i)$ son los distintos nodos representativos de cada región de forraje y $N(i)$ el vecino a escoger. Cada entrada de la tabla



8	p_{88},q_{88}	x
7	p_{67},q_{67}	x
6	p_{66},q_{66}	x
5	p_{55},q_{55}	x
4	p_{44},q_{44}	x
3	p_{43},q_{43}	p_{53},q_{53}
2	p_{42},q_{42}	x
1	p_{41},q_{41}	p_{51},q_{51}

Figura 7: Ejemplo de división de una red de 9 nodos. Los agentes abeja de corta distancia tienen un límite de 2 saltos y sobre cada uno de los enlaces se indica el coste asociado. Puede observarse el lanzamiento de réplicas desde el nodo 9. En la parte inferior se muestra la que sería la tabla de rutas del nodo 9. Imagen extraída de [Farooq and Caro, 2008], página 42.

$P_{jk_r} = \left(q_{jk_r}, p_{jk_r}\right)$ es un indicador del retraso que experimentará un paquete en la cola y en la propagación al elegir el nodo vecino j para alcanzar el nodo representativo k de la región de forraje r.

FRM Foraging Region Membership. Es una tabla donde se asigna a cada uno de los destinos la región de forraje a la que pertenecen.

En la Figura 7 puede observarse un ejemplo de una estructura una red y una tabla de enrutamiento.

Periodo de formación de regiones de forraje Hay un periodo de inicio para la formación de de las regiones de forraje IFR. Cada uno de los nodos crea su propia región en la que es el nodo representativo y lanza los primeros agentes abeja de corta distancia a cada uno de sus nodos vecino $n \in \mathcal{N}_k$ para propagar su dirección. Si un nodo recibe un agente

de corta distancia cuya dirección del nodo representativo es menor que la suya, deja su región inicial para unirse a esa nueva región de forraje. Si el nodo representativo de algún nodo se une a otra región, volverá a comenzar de nuevo el proceso. Una vez finalizada la división de la red en regiones de forraje no solapadas, cada nodo informa al resto de los nodos de la red a qué región pertenece. Si se produce algún cambio en la topología, es necesario volver a repetir el proceso para reorganizar las regiones de forraje.

Una vez supera la fase de formación, cada nodo no representativo difunde en intervalos de tiempo regulares Δt una réplica de un agente abeja de corta distancia $Bsd_{s\to n}$ a cada uno de sus nodos vecino $n \in N(i)$, estableciendo un número máximo de saltos (o tiempo de vida) h_{sd}. Los nodos representativos lanzan de forma similar agentes abeja de larga distancia $Bld_{s\to n}$, con la diferencia de que el límite del número de saltos h_{ld} es mucho mayor.

Los agentes abeja utilizan las colas de prioridad para conseguir una rápida difusión de la información de enrutamiento. Durante el camino, cada agente mantiene en memoria el número de saltos, el retraso acumulado en las colas q_{ps} y el retraso acumulado en la propagación p_{ps} entre el último nodo visitado p y el nodo origen s.

Cuando una de las réplicas del agente abeja lanzado desde el nodo s llega a un nodo determinado, se actualiza la información de rutas del nodo actual i y la información de retrasos del propio agente. La información a actualizar dentro del nodo i dependerá de si el agente se encuentra dentro de su zona de forraje o fuera de ella. En el primer caso actualizará la entrada $\left(q_{jk,}p_{jk}\right)$ de la tabla de enrutamiento IFZ y en el segundo la entrada $\left(q_{jk_r,}p_{jk_r}\right)$ de la tabla IFR.

Las acciones a realizar sobre el agente dependerán de si han llegado más réplicas lanzadas desde el mismo nodo s. Si han llegado más réplicas, se seleccionará aquella más prometedora para ser actualizada y difundida de nuevo entre sus vecinos, a excepción del vecino p desde el que llegó. Se actualizarán los retrasos en las colas q_{ks} y en la propagación p_{ks} del propio agente incorporando los retrasos del resto de réplicas en proporción a la calidad de sus rutas. El proceso de difusión continúa hasta que el número de saltos del agente llegue al límite establecido (el tiempo de vida del agente expire) o hasta que una réplica de este agente ya haya sido recibida, en cuyo caso la réplica será eliminada si no era la más prometedora.

Los agentes utilizan el siguiente modelo para estimar el tiempo de viaje de un paquete $T_{i\to s}$ desde el nodo actual i hasta el nodo origen s utilizando en enlace de llegada:

$$T_{i\to s} \approx \frac{q_{l_{in}}}{b_{in}} + tx_{in} + pd_{in} + t_{ns}$$

donde ql_{in} es el tamaño en bits de la cola del vecino n para el nodo i, b_{in}, tx_{in} y pd_{in} son el ancho de banda, el retraso de transmisión y el retraso de propagación entre el nodo i al nodo n, y t_{ns} es el tiempo de viaje entre n y s. El ancho de banda y los retrasos de propagación de los enlaces se calculan de forma aproximada mediante la transmisión de paquetes hello.

4.3.2. Enrutamiento de paquetes de datos

Dado que los dos tipos de agente (de corta y larga distancia) viajan por la red, cada nodo mantiene la información de enrutamiento actualizada para llegar a otros nodos, ya sea dentro o fuera de su zona de forraje. En el caso de que el destino esté fuera de su zona, se utilizará el nodo representativo de la región de forraje a la que pertenezca.

Cuando un paquete de datos d_{sd} llega al nodo i, primero se comprueba si el destino d es el propio nodo. De no ser así, hay que elegir un nuevo vecino para reenviar el paquete del siguiente modo:

- Primero es necesario comprobar si el nodo destino pertenece a la zona de forraje del nodo actual, en cuyo caso se consultará la tabla IFZ para obtener los retrasos $\left(q_{jk,}p_{jk}\right)$ hacia d.
- Si el nodo no pertenece a la zona de forraje, entonces será necesario consultar primero la tabla FRM para obtener el nodo representante w de la zona. Una vez conocido el nodo representante w, es necesario consultar la tabla IFR para obtener los retrasos $\left(q_{jk_r,}p_{jk_r}\right)$ hacia w, que será el nodo hacia el que se enrutará el paquete.

Una vez obtenido el par de retrasos de cada uno de los nodos vecinos, el siguiente salto para un paquete de datos es seleccionado de forma aleatoria en base a la bondad definida para cada uno de ellos. El hecho de ser una selección estocástica da lugar a que no siempre se escojan las mejores rutas, sino que a que haya un reparto de tráfico que ayudará a maximizar el rendimiento de la red.

El modo en que se calcula la bondad de un nodo j respecto a sus vecinos $n \in \mathcal{N}_k$ para alcanzar el destino d se define de la siguiente manera:

$$g_{jd} = \frac{\frac{1}{p_{jd}}\left(e^{-\frac{q_{jd}}{p_{jd}}}\right) + \frac{1}{q_{jd}}\left(1 - e^{-\frac{q_{jd}}{p_{jd}}}\right)}{\sum_{n=1}^{\mathcal{N}_k}\left(\frac{1}{p_{nd}}\left(e^{-\frac{q_{nd}}{p_{nd}}}\right) + \frac{1}{q_{nd}}\left(1 - e^{-\frac{q_{nd}}{p_{nd}}}\right)\right)}$$

El retraso total de un enlace está condicionado por dos elementos, las colas del enlace y el propio retraso del enlace:

q_{jd} Cuando un nodo experimenta una carga de tráfico alta, es la cola la que condiciona el retraso total del enlace $q_{jd} \gg p_{jd}$, de modo que $g_{jd} \approx \frac{\frac{1}{q_{jd}}}{\sum_{n=1}^{\mathcal{N}_k}\frac{1}{q_{kd}}}$.

p_{jd} Cuando un nodo experimenta una carga de tráfico baja, es el propio retraso del enlace el condiciona el retraso total del enlace $p_{jd} \gg q_{jd}$, de modo que $g_{jd} \approx \frac{\frac{1}{p_{jd}}}{\sum_{n=1}^{\mathcal{N}_k}\frac{1}{p_{kd}}}$.

4.4. Otros algoritmos de enrutamiento inspirados en hormigas

A continuación describimos de forma cronológica otros algoritmos de enrutamiento para redes cableadas que han ido surgiendo a raíz de los tres algoritmos básicos ABC, AntNet y en menor medida BeeHive. La mayoría son modificaciones o han sido directamente inspirados en AntNet. El número de trabajos propuestos de enrutamiento inspirados en ACO es increíblemente grande, por lo que solamente se describen algunos de ellos.

4.4.1. Uniform Ant Algorithm, ABC para redes de conmutación de paquetes

A partir del algoritmo ABC [Schoonderwoerd et al., 1997], ideado para redes de conmutación de circuitos de coste simétrico, Subramanian et al. desarrollan un algoritmo adaptado a las redes de conmutación de paquetes con enrutamiento multi-trayecto [Subramanian et al., 1997]. La diferencia respecto a ABC radica en el uso de dos tipos de hormiga: las hormigas regulares que mantienen la misma filosofía que ABC y las hormigas uniformes.

Las hormigas regulares, que realizan la actualización en sentido inverso y se mueven por la red en base a la bondad $\tau_{nd} \in \mathcal{T}_k$ de cada uno de los enlaces con sus vecinos n, solamente se diferencian respecto a ABC en el cálculo del incremento de la probabilidad$\triangle p$:

$$\triangle p = \frac{k}{f(c)}$$

donde $k > 0$ es una constante denominada "ratio de aprendizaje" generalmente menor que 0,1 y $f(c)$ es una función no decreciente de c, donde c es el coste del enlace evaluado.

Las hormigas uniformes son generadas del mismo modo que las hormigas regulares, pero con una estructura más sencilla al no incluir un destino concreto dentro de la red y un tiempo de vida máximo. La selección del siguiente enlace de salto se realiza de forma uniforme $p = \frac{1}{|\mathcal{N}_k|}$, donde k es el nodo actual y $\mathcal{N}_k$ sus vecinos. El objetivo de este tipo de hormigas, que se mueven de forma aleatoria sin rumbo fijo, es ampliar la exploración en la red y evitar una rápida convergencia hacia soluciones sub-óptimas.

En los experimentos realizados se compara el rendimiento del algoritmo de hormigas uniformes con el rendimiento de algoritmos convencionales (de estado de enlace y vector distancia) en entornos donde se producen cambios en la red. Aunque el algoritmo propuesto muestra un mejor rendimiento, el hecho de basarse en una red donde todos sus enlaces son de coste simétrico es poco realista en redes de conmutación de paquetes.

4.4.2. AntNet-FA

AntNet-FA, también llamado AntNet-CO, se trata de una variante de AntNet propuesta por Di Caro y Dorigo [Di Caro and Dorigo, 1998b]. Surge como la necesidad de corregir algunos problemas en el procedimiento de evaluación de las rutas con respecto a los agentes "forward ant".

En la versión original de AntNet los agentes "forward ant" hacen uso de las mismas colas de baja prioridad que utilizan los paquetes de datos, de forma que pueden experimentar los mismos problemas de congestión traduciéndose en un incremento del tiempo de viaje. Este mecanismo tiene sus inconvenientes, ya que el retraso de la actualización en el camino inverso del agente "backward ant" puede ocasionar que el estado de la ruta sea diferente al que se había evaluado inicialmente. Además, si existe congestión en la ruta se le otorga un refuerzo muy pequeño, pudiendo dar lugar a una cierta polarización hacia rutas menos congestionadas.

Para evitar esto, se propone que los agentes "forward ant" utilicen las colas de prioridad al igual que los agentes "backward ant", de modo que la actualización de las tablas de rutas de los nodos ya no se realiza con tiempos obtenidos experimentalmente en el camino de ida, sino mediante el cálculo de estimaciones. De ese modo, la hormiga actualizará las tablas de rutas sin los retrasos producidos por la espera en las colas de datos, además de no requerir una memoria con los tiempos de viaje.

Las estimaciones de tiempos de viaje se realizan en base la cantidad de información o paquetes esperando en la cola, el ancho de banda y el retraso de propagación del enlace. El tiempo $T_{i \to k}$ de enviar un paquete de tamaño s_p desde el nodo actual k al nodo vecino n desde el cual llega el agente "backward ant" se estima del siguiente modo:

$$T_{k \to n} = d_l + \frac{q_l + s_a}{b_l} \tag{12}$$

donde d_l, q_l y b_l son respectivamente el retraso, el tamaño en bits de los paquetes que hay esperando en la cola y el ancho de banda del enlace. s_a es el tamaño del paquete de la hormiga.

Según los experimentos realizados, AntNet-FA tiene un rendimiento superior a AntNet, principalmente en redes de mayor tamaño.

4.4.3. AntNet-FA mejorado

Poco después de que AntNet fuera mejorado por sus propios autores presentando AntNet-FA que utiliza las colas de prioridad y tiempos estimados, Barán y Sosa presentan una versión mejorada de AntNet-FA [Baran and Sosa, 2000].

Las modificaciones realizadas son las siguientes:

- Inicialización de las tablas de rutas. Durante la inicialización de cada uno de los nodos, AntNet asigna de manera uniforme entre sus vecinos la probabilidad de alcanzar un destino d determinado: $p_{id} = \frac{1}{n_k}$, $\forall n_k \in \mathcal{N}_i$. La propuesta consiste en asignar una mayor probabilidad a aquellos enlaces en los que el destino coincida con el nodo vecino, ya que sería el camino más corto (un salto). De ese modo, $p_{id} = \frac{1}{n_k} + \frac{3}{2} \times \frac{(n-1)}{n_k^2}$ si el nodo destino d es el nodo vecino n.

- Mecanismo de recuperación ante fallos en la red. Integran un mecanismo de recuperación ante fallos de nodo de la red que AntNet no contempla, de modo que si el nodo j cae, la probabilidad p_{jd} de elegir el enlace l_{kj} para alcanzar el destino d es 0 y se distribuye uniformemente entre el resto de enlaces vecinos: $p_{ij} = p_{ij} + \frac{p_{jd}}{|\mathcal{N}_i|-1} \forall i \in \mathcal{N}_i, i \neq j$. Además, si el nodo caído vuelve a recuperarse, en lugar de reinicializar las probabilidades como si del instante t_o se tratara, utilizan las probabilidades del momento en el que el nodo dejó de funcionar aplicando un factor de "memoria" λ: $p_{id}(t_{up}) = (1-\lambda) \times p(t_0) + \lambda \times p(t_{down}), 0 \leq \lambda < 1$.

- Introducción de ruido. Para evitar el estancamiento a la hora de seleccionar un nodo vecino con valores de probabilidad cercanos a 0 ó 1 y para mejorar la capacidad de descubrir nuevas rutas, se establece un factor de ruido f. De ese modo, la selección del siguiente nodo se realiza de forma uniforme con una probabilidad f y en base a las tablas de rutas con probabilidad $1-f$.

- Combinación de selección determinista y estocástica del siguiente nodo. A diferencia de AntNet, que selecciona el siguiente salto de forma estocástica en base a la bondad de cada enlace, aquí se selecciona de forma estocástica con probabilidad p y de forma determinista con probabilidad $1-p$.

- Control de del número de agentes generado. Para evitar la congestión, se establece un límite en la población de hormigas de cuatro veces el número de nodos de la red.

4.4.4. AntNet Loop-Free

En este trabajo presentado por Doi y Yamamura [Doi and Yamamura, 2004], analizan cómo afecta la topología de red al rendimiento de algoritmos como AntNet. Internet mantiene una jerarquía donde el grado de los nodos que la componen es muy diferente. Para llevar a cabo el estudio plantean experimentos utilizando redes libres de escala y tres algoritmos diferentes, AntNet-FA y dos variaciones de éste que consisten en lo siguiente:

- AntNet-FAA-Ln. Consiste en una variante que evita bucles para todos aquellos nodos de grado igual o mayor que n. Por ejemplo, si el umbral n es 3, no se permitirá un bucle en la ruta en aquellos nodos que tengan 3 ó más conexiones con nodos vecinos.

- AntNet-FAA. Incluye una modificación que aprovecha aquellos agentes "forward ant" en los se ha detectado un bucle si el tiempo dentro del bucle no es mayor que el tiempo fuera de él. En este caso, se elimina el ciclo detectado, el nodo actual pasa a ser el destino y el agente se convierte en "backward ant" camino al origen inicial.

De los resultados del estudio se extrae el hecho de que evitar bucles puede ser beneficioso o puede ser un problema dependiendo del grado de los nodos de la red. Por ejemplo, en una red dispersa o con un grado muy bajo beneficia evitar los bucles, pero en redes con hubs o nodos de un grado elevado se comporta mejor el algoritmo cuando sí permite bucles.

4.4.5. Ant Swarm-based Routing (ASR)

Se trata de un nuevo algoritmo propuesto por Lü et al. [Lü et al., 2004] denominado Ant Swarm-based Routing (ASR). Basado en la versión clásica de AntNet y destinado a redes de conmutación de paquetes, trata de incrementar la velocidad de convergencia y de mejorar la estabilidad introduciendo una variación del aprendizaje por refuerzo y una técnica denominada "momentum".

A diferencia del modelo estadístico $\mathcal{M}^k$ utilizado en AntNet, ASR mantiene un array $\mathcal{D}^k$ que contiene las dos últimas estimaciones de retrasos $D^k_{jd}(t)$ y $D^k_{jd}(t-1)$ experimentados por el agente desde el nodo k a cualquier nodo destino d utilizando el nodo vecino j. Para actualizar esta estructura se basa en el momentum, que además de utilizar el retraso actual experimentado d^k_{jd}, incorpora también parte del retraso previo:

$$D^k_{jd}(t) = D^k_{jd}(t) + \eta\left((1-\alpha)\triangle D^k_{jd}(t) + \alpha\triangle D^k_{jd}(t-1)\right)$$

donde:

$$\triangle D^k_{jd}(t) = d^k_{jd} - D^k_{jd}(t)$$
$$\triangle D^k_{jd}(t-1) = D^k_{jd}(t) - D^k_{jd}(t-1)$$

η es el factor de aprendizaje y α es el factor de momentum.

A partir de esta información, cada entrada τ^k_{jd} de la ruta de tablas $\mathcal{T}^k$ del nodo k, que indica la probabilidad de elegir el enlace $k \to j$ para alcanzar el destino d se actualiza de la siguiente forma:

$$\tau^k_{jd} = \frac{\left(D^k_{jd}(t)\right)^{-\beta}}{\sum_{n\in\mathcal{N}_k}\left(D^k_{nd}(t)\right)^{-\beta}}$$

donde $\beta > 1$ es un parámetro de no linealidad para favorecer las rutas más cortas o con menor retraso.

Además de estas dos diferencias respecto a AntNet, incorpora algunas de las modificaciones presentadas en [Baran and Sosa, 2000], como son la mejora de la inicialización de las tablas de rutas para acelerar la convergencia o la incorporación de ruido a la hora de seleccionar el siguiente nodo con el objetivo de incrementar la exploración.

Respecto al rendimiento, los resultados aportados indican que el rendimiento de ASR es superior a AntNet. Aun así, el modo en que se actualiza el modelo de red $\mathcal{D}^k$ podría no ser correcto, ya que se basa en el principio de Bellman al actualizar todas las subrutas intermedias $k \to d$ al evaluar la ruta completa $s \to d$. Ese principio podría no ser válido en un entorno no estacionario como es una red de telecomunicación donde cambian los patrones de tráfico, tal y como se explica en [Di Caro, 2004].

4.4.6. AntNet con recompensa-penalización

Lalbakhsh et al. [Lalbakhsh et al., 2010] pretenden aumentar las capacidades de exploración y de adaptabilidad de AntNet introduciendo algunas modificaciones en el proceso de actualización de las tablas de rutas. A diferencia del algoritmo clásico, en el que las tablas de rutas solamente son actualizadas cuando el tiempo de viaje está dentro de un intervalo de confianza, en este caso las tablas se actualizan siempre. Consideran la actualización como un aprendizaje por refuerzo en el que aplicar un refuerzo cuando el el tiempo de viaje es satisfactorio y una penalización cuando no lo es. Dada la naturaleza dinámica del entorno, proponen una estrategia que se adapte al estado de la red, ya sea en la correspondiente evaluación del tiempo de viaje o en los valores de recompensa y penalización.

Para detectar los casos en los que los tiempos de viaje no son satisfactorios debido a fallos o a cambios en los patrones de tráfico, se propone una estrategia que analiza un conjunto de tiempos de viaje que no superan el intervalo de confianza establecido en AntNet. Dado un factor de no optimalidad $\Omega(T) = 1 - \frac{I_{sup}}{T_{np}}$, en el que T_{np} es el tiempo de viaje no válido respecto a I_{sup}, se obtiene una evaluación de los últimos tiempos de viaje $\sum \Omega_i(T_{np})$. Si la evaluación del conjunto analizado supera un umbral determinado, se considera que hay un problema con el correspondiente enlace y se aplica un factor de penalización, además de reiniciar los datos del modelo estadístico $\mathcal{M}^k$.

En cuanto al factor de penalización $F(T_{np})$, se calcula de forma proporcional a la distancia entre el tiempo de viaje respecto al límite superior I_{sup} del intervalo de confianza:

$$F(T_{np}) = \frac{1}{1 + \frac{\left(T_{np} - a \times I_{sup}\right)^2}{\sqrt{\left(b \times I_{sup}\right)}}}$$

Dicho factor actualiza la entrada del enlace en la tabla de rutas decrementando su valor, a la vez que incrementa las de el resto para mantener la normalización:

$$P_i(n+1) = P_i(n) - (F(T_{np}) \times P_i(n))$$
$$P_j(n+1) = P_j(n) + \frac{\left(F\left(T_{np}\right) \times P_j(n)\right)}{\sum j}$$

4.5. Objetivos de calidad

A continuación, describimos las características que de algún modo consideramos definen la calidad de los algoritmos de enrutamiento, así como algunas técnicas que han sido utilizadas para lograr cada una de ellas. Es posible que una sola técnica proporcione más de una característica.

Rendimiento. Dado que hablamos de algoritmos ACO y se trata de resolver un problema de optimización combinatoria, el objetivo será maximizar o minimizar algún

criterio de evaluación de los experimentos realizados por cada uno de los agentes de la red, como los tiempos de envío o el número de saltos. Cuanto mejor se resuelva el problema, mayor rendimiento espera obtenerse de la red. Dos medidas de rendimiento habituales son el throughput[4] y el tiempo de entrega de los paquetes (end-to-end delay).

Varias técnicas se han mostrado efectivas a la hora de mejorar el rendimiento. Una de ellas es la selección estocástica del siguiente salto en base a la probabilidad de cada entrada de la tabla de rutas. A diferencia de la selección determinista, permite establecer un enrutamiento multi-trayecto dando lugar a la distribución del tráfico en la red. Otra de las técnicas es tratar de establecer un control de la sobrecarga de tráfico generado para la gestión de la red, en este caso la población de agentes hormiga. Generar muy pocos agentes hormiga no permite explorar la red y establecer las rutas correctamente, pero una cantidad elevada puede provocar una sobrecarga que da lugar a un efecto negativo, por lo que es necesario establecer un equilibrio.

Convergencia. Diremos que un algoritmo converge, si a partir de una secuencia de iteraciones, cada vez se aproxima más a la solución óptima. Un algoritmo tendrá una mayor capacidad de convergencia que otro, si necesita un número menor de interaciones para aproximarse a dicha solución óptima. En este caso, siempre que un camino sea factible entre dos nodos, podrá ser descubierto mediante el envío de agentes hormiga. La cuestión está en aplicar técnicas que favorezcan que el tiempo necesario para converger hacia la solución óptima sea menor.

Una de las técnicas que tratan de mejorar la convergencia es el modo en que se inicializan las tablas de rutas. Así como en AntNet se realiza de manera uniforme, en [Baran and Sosa, 2000] proponen una inicialización en la que se otorga mayor probabilidad a las entradas de aquellos nodos vecinos que al mismo tiempo son nodos destino. Otras técnicas son la evaporación de feromonas o mecanismos de feedback negativo. En el primero se produce una evaporación de feromonas periódicamente respecto a un factor de decaimiento, de modo que si la entrada con un valor de feromona más alto lleva un tiempo sin ser incrementado, pueden favorecerse otras rutas alternativas, como en el caso de [Tekiner and Ghassemlooy, 2005]. En el segundo se introduce un feedback negativo que decrementa el valor de feromonas de aquellos enlaces con problemas, como en [Lalbakhsh et al., 2010].

Exploración. Si muchas hormigas seleccionan un camino sub-óptimo al inicio del proceso de aprendizaje, otras hormigas seleccionarán la misma ruta y la reforzarán. De hecho, incluso después de haber encontrado el camino óptimo, es posible que ante un cambio en las condiciones de la red deje de ser la solución óptima y una alternativa mejor no pueda ser descubierta. Es necesario por tanto aplicar alguna técnica que favorezca la exploración, evitando que las hormigas puedan quedar estancadas en caminos sub-óptimos.

[4]Podemos definir throughput como una medida relativa del volumen de información que fluye a través del medio por unidad de tiempo.

En este caso, son muchas las técnicas para incrementar la exploración y evitar que los valores de feromona se estanquen. Una de las técnicas más utilizadas y efectivas en este aspecto ha sido la introducción de ruido en la política de selección de salto, de modo que con probabilidad p_{noise} realiza una selección de manera uniforme entre cualquiera de los vecinos no visitados y con probabilidad $1 - p_{noise}$ realiza una selección en base a la bondad que represente la entrada de la tabla de feromonas de cada vecino. Pueden verse ejemplos de uso en [Schoonderwoerd et al., 1997; Baran and Sosa, 2000]. En el caso de [Subramanian et al., 1997] van más allá e introducen un nuevo tipo de hormiga, la hormiga uniforme, que se dedica exclusivamente a la exploración de la red. En AntNet por su parte, utilizan una técnica que consiste en limitar los valores mínimos τ_{min} y máximos τ_{max} de feromonas con el objetivo de evitar el estancamiento. La técnica de evaporación de feromonas comentada en el apartado de convergencia también sería un método efectivo para evitar el estancamiento.

Parte III

Algoritmo propuesto

5. Abira

Se propone Abira (**A**nts **B**ehaviour **I**nspired **R**outing ***A***lgorithm), un algoritmo basado en AntNet-FA, dedicado al enrutamiento de tráfico para redes cableadas con entrega de mejor esfuerzo. La idea es ofrecer un conjunto de mejoras utilizando las bases generales de ACO y otras aportaciones de los principales trabajos en este ámbito: el trabajo de Schoonderwoerd et al. [Schoonderwoerd et al., 1997], el primer algoritmo inspirado en el comportamiento de las colonias de hormigas para el enrutamiento en redes telefónicas, y el trabajo de Di Caro y Dorigo [Di Caro and Dorigo, 1998a,b], con su propuesta orientada a redes de conmutación de paquetes. En menor medida, la propuesta de Wedde et al. [Wedde et al., 2004], con un algoritmo basado en el comportamiento de los enjambres de abejas, también se tiene en cuenta.

AntNet sentó una base para los algoritmos de enrutamiento de redes de conmutación de paquetes, permitiendo que otras investigaciones en el ámbito de los algoritmos inspirados en ACO hicieran nuevas propuestas tanto para redes cableadas, como para redes inalámbricas. Algunos ejemplos de algoritmos basados en AntNet dirigidos a redes cableadas ya han sido comentados en la segunda parte de este trabajo [Baran and Sosa, 2000; Lalbakhsh et al., 2010; Doi and Yamamura, 2004; Liang et al., 2006; Sim and Sun, 2002]

En este caso, tratamos de recoger varias de esas ideas y de proponerlas como nuevas mejoras dentro del algoritmo AntNet-FA. Cada una de ellas tiene el objetivo incrementar sus capacidades o de cubrir algunas de sus carencias. AntNet es un algoritmo que consigue un buen balanceo de carga redistribuyendo el tráfico de datos entre los diferentes nodos de la red, pero todavía tiene cierto margen de mejora. A pesar de ser muy difícil incrementar el rendimiento en un estado estacionario, sí es posible incrementar el tiempo de convergencia o la capacidad de reacción ante estados transitorios, ya sean cambios producidos en la propia topología de la red o en los patrones de tráfico.

En esta sección, primero describimos de forma general el algoritmo y las modificaciones que proponemos, para más adelante describir de un modo más detallado cada una de ellas.

5.1. Descripción del algoritmo

En líneas generales, dado que el algoritmo se basa en AntNet-FA [Di Caro and Dorigo, 1998b], la estrategia a seguir es la misma que la descrita en la parte II, puntos 4.2 y 4.4.2. Hemos incorporado algunas modificaciones que describimos a continuación, con el objetivo de mejorar los aspectos que comentábamos en el punto 4.5: rendimiento, convergencia y exploración.

Algoritmo base, AntNet-FA Se envían agentes "forward ant" $F_{s\to d}$ de forma proactiva y en intervalos de tiempo regulares $\triangle t$ desde cada nodo de la red hacia diferentes destinos d. La forma de seleccionar el nodo destino d se realiza o bien de forma uniforme con probabilidad p o bien mediante un modelo probabilístico sesgado hacia los destinos más solicitados por los paquetes de datos con probabilidad $1-p$. La tasa de generación de hormigas $1/\triangle t$ es fija, intentando asignar un valor que mantenga un equilibrio entre el número de agentes que ofrecen un mejor rendimiento y que al mismo tiempo no provocan una sobrecarga de tráfico en la red.

Los agentes "forward ant", a diferencia de los paquetes de datos, utilizan las colas de prioridad. Una vez alcanzado el destino d, el agente se convierte en "backward ant" $B_{s\to d}$, heredando toda la información y realizando el camino inverso. En cada uno de los nodos de la red se realiza una estimación de tiempo de viaje entre éste y el destino, y entre éste y todos los diferentes sub-trayectos posibles. La estimación se realiza en base a dos factores: las propiedades físicas del enlace (el ancho de banda de los enlaces y el retraso de propagación) y la carga de las colas del nodo. Si el tiempo estimado es menor que el límite superior de un intervalo de confianza I_{sup} derivado del modelo estadístico $\mathcal{M}^k$, se actualizan tanto las tablas de feromonas $\mathcal{T}^k$ como el propio modelo estadístico $\mathcal{M}^k$.

Modificaciones o mejoras propuestas La intención de Abira es mejorar la respuesta del algoritmo AntNet-FA en escenarios transitorios. Para ello, se proponen las siguientes tres modificaciones:

- Introducción de un mecanismo que mejora la capacidad de exploración. El objetivo de esta propuesta es evitar en un mayor grado el estancamiento de los valores de feromonas, además de mantener un mayor número de rutas alternativas ante posibles situaciones de cambio en la red.
- Introducción de un mecanismo de difusión de caminos óptimos. En este caso, la idea es que los nodos de la red compartan con sus vecinos las rutas óptimas encontradas en un espacio de tiempo determinado, de forma que el tiempo que necesita el algoritmo para converger a una solución óptima se reduzca de forma considerable. Mantener una ventana de tiempo permite que el algoritmo se adapte a la situación de la red en cada momento, de manera que las rutas óptimas a comunicar también varíen.
- Introducción de un mecanismo de recompensa-penalización (*negative feedback*). A diferencia de AntNet, que utiliza un mecanismo de recompensa-inacción, donde solamente se aplican refuerzos proporcionales al tiempo de viaje experimentado, en esta propuesta se utiliza un mecanismo de recompensa-penalización, que además de aplicar refuerzos utilizando los tiempos óptimos, aplica penalizaciones si el tiempo experimentado está lejos de la solución óptima. Utilizar una mayor parte de los resultados, incrementando o decrementando el valor de las feromonas, ayuda a que el tiempo que el algoritmo necesita para converger sea menor.

En los siguientes apartados, se explica de un modo más detallado cada una de estas modificaciones. Se describe primero la idea, luego cómo funciona en el algoritmo AntNet y más tarde se detalla cómo se aplican estos cambios en Abira.

5.2. Mejora de la capacidad de exploración: introducción de ruido

Podría darse el caso de que algunas entradas de la tabla de feromonas τ_{nd} tuvieran valores cercanos o iguales a 0 y 1, fenómeno conocido en ACO como *estancamiento*. Este tipo de situaciones provocan que solamente una de las rutas sea utilizada, independientemente de que sea o no la ruta óptima, no permitendo que otras alternativas sean tenidas en cuenta.

Cómo evitar el estancamiento en AntNet A la hora de entregar un paquete desde cualquier nodo origen s a un nodo destino d, existe un criterio de selección del nodo vecino al que reenviar el paquete que varía dependiendo de si se trata de un paquete de datos o de una hormiga. En cualquiera de los dos casos, los valores de la tabla de feromonas $\mathcal{T}^k$ están involucrados, donde cada entrada $\tau_{nd} \in \mathcal{T}^k$ representa la bondad de seleccionar el enlace del nodo n para alcanzar el destino d. Mientras que en el caso de las hormigas, la selección depende tanto de los valores de feromonas como del estado de las colas de cada uno de los nodos vecinos (ver ecuación 3), en el caso de los paquetes de datos la selección depende totalmente de los valores de feromonas, que sufren una transformación exponencial siguiendo la ecuación 11.

Sin embargo, en AntNet el estancamiento no podría llegar a darse si consideramos que la actualización de las tablas de feromonas es realizada por las hormigas y que existen dos situaciones en las que podrían alterar estos valores:

- Como puede observarse en la ecuación 3, la selección del siguiente salto se realiza en base al valor previo de la tabla de feromonas τ_{nd} y el estado de las colas del enlace l_n. Con un valor $\alpha > 0$, llegados a un punto en el que la saturación provoca que la ruta más prometedora del vecino f con valor $\tau_{fd} \simeq 1$ deje de ser la ruta más prometedora, empiezan a evaluarse rutas alternativas.
- Cuando un agente "forward ant" $F_{s \to d}$ se encuentra en un nodo k y todos los nodos vecinos ya han sido visitados $\forall n \in \mathcal{N}_k \mid n \in V_{s \to d}$, cualquier nodo puede ser seleccionado de forma uniforme con probabilidad p_{nd} según la ecuación 4.

En cualquiera de los dos casos se aplica un refuerzo $r > 0$, dando lugar a un incremento, en mayor o menor medida, de los valores $\tau_{nd} \simeq 0$, $n \in \mathcal{N}_k$, $n \neq f$ y un decremento de τ_{fd}. Adicionalmente y de manera similar a MMAS, se limitan los valores de forma artificial estableciendo un valor τ_{max} y un valor τ_{min}.

Problemas de bloqueo y atajo Tal y como indican en [Schoonderwoerd et al., 1997], aunque este tipo de algoritmos se basen en la auto-organización en lugar del aprendizaje

por refuerzo, las hormigas sufren un fenómeno similar al sobreentrenamiento que da lugar a los dos problemas descritos en [Sutton, 1991]: el problema de bloqueo y el problema de atajo. El problema de bloqueo se produce cuando una ruta fuertemente reforzada en la tabla de feromonas $\mathcal{T}^k$ deja de estar disponible, de modo que transcurre un tiempo relativamente largo hasta que rutas alternativas son consideradas. El problema del atajo, en cambio, se produce cuando una nueva ruta más corta[5] aparece y tarda en ser incorporada debido a que el resto de entradas τ_{nd} son demasiado elevadas para que los agentes la visiten.

Modificación propuesta A pesar de las medidas de limitar los valores τ_{nd}, utilizar la saturación de las colas en las evaluaciones o seleccionar un nodo de forma uniforme en caso de ciclo, aún es posible mejorar la capacidad exploración y con ello la posibilidad de descubrir nuevas rutas. Estas medidas no tienen en cuenta, por ejemplo, aquellas situaciones en las que no habiendo llegado a una situación real de estancamiento, existe una entrada en la tabla de feromonas cercana a 1 ó τ_{max} que pertenece a un camino sub-óptimo y un valor cercano a 0 ó τ_{min} que pertenece a un camino óptimo. Para evitar esta situación, se introduce un factor de ruido p_{noise} en la selección del siguiente salto cuando el paquete a enrutar es una hormiga:

- Con probabilidad p_{noise}, la selección del siguiente nodo se realiza de manera uniforme (ver ecuación 4), evitando converger rápido hacia soluciones sub-óptimas y ayudando a encontrar nuevos caminos en caso del que se produzcan cambios en la red.
- Con probabilidad $1 - p_{noise}$, la selección se realizará de la forma habitual siguiendo las ecuaciones 3 y 4.

5.3. Mecanismo de difusión de caminos óptimos: agentes comunicadores

Los algoritmos ACO tienen en cuenta el descubrimiento y la gestión de las soluciones o rutas encontradas, utilizando tanto la acumulación, como la evaporación de feromonas. Es una técnica eficiente y sencilla a la hora de encontrar soluciones, pero puede ser un mecanismo demasiado lento, más todavía si se trata de un sistema en tiempo real como es la tarea de enrutamiento de paquetes.

AntNet y el modelo estadístico $\mathcal{M}^k$ En AntNet existen dos estructuras diferentes para modelar el estado de la red: un modelo estadístico $\mathcal{M}^k$, que determina el estado local de la red, y una tabla de feromonas $\mathcal{T}^k$, que recoge el estado global de la red.

[5] Según la paradoja de Braess, añadir una ruta más corta y/o menor coste puede dar lugar a un incremento en la congestión en la red.

La estrategia para mantener ambos modelos $\mathcal{M}^k$ y $\mathcal{T}^k$, se basa en utilizar los tiempos estimados por la hormiga "backward ant" $B_{d\to s}$ en el camino inverso de la ruta del agente "forward ant" $F_{s\to d}$. Teniendo en cuenta que las actualizaciones podrán producirse tanto para la ruta completa $s \to d$, como para todas las posibles sub-rutas desde cada uno de los nodos intermedios $k \in V_{s\to d}$, $k \neq d$, denominamos el tiempo estimado a evaluar como $T_{k\to d}$. El tiempo $T_{k\to d}$ solamente se tendrá en cuenta para actualizar los modelos, si el nodo k es el nodo origen s o si $T_{k\to d}$ está dentro de un intervalo de confianza $I\left(\mu_d, \sigma_d^2\right)$ obtenido a partir del modelo $\mathcal{M}^k$, donde el límite superior I_{sup} del intervalo se calcula utilizando la ecuación 10.

Si el tiempo $T_{k\to d}$ no supera el límite superior I_{sup} o corresponde al nodo origen s desde el que se inició el experimento, se procede a actualizar el modelo estadístico $\mathcal{M}^k$ y la tabla de feromonas $\mathcal{T}^k$. Los valores a actualizar en el modelo estadístico $\mathcal{M}^k$ son: la media y la varianza, utilizando la ecuación 5, el valor del tiempo mínimo W_d observado en la ventana de tiempos w y la propia ventana de tiempos w.

A partir de este modelo estadístico $\mathcal{M}^k$, para AntNet la ruta óptima es aquella que puede entregar los paquetes en un tiempo W_d.

Modificación propuesta Para agilizar el tiempo que necesita el algoritmo para converger a soluciones óptimas, se ha recurrido a un mecanismo que no contempla ACO y que es parecido a la técnica de compartir la información entre los nodos vecinos de PSO [Kennedy et al., 2001], la difusión de paquetes que se utiliza en BeeHive [Wedde et al., 2004] o a la mecánica de algoritmos de enrutamiento convencionales: compartir la información de las mejores rutas. La idea es generar un tercer tipo de hormiga (adicional a las hormigas "forward ant" y "backward ant"), con el objetivo de que comunique las mejores soluciones o mejores trayectos observados a sus vecinos. Compartir esta información reduce el tiempo de convergencia, potenciando las posibilidades de encontrar una mejor solución, además de actuar con mayor celeridad ante cambios en los patrones de tráfico en la red e incrementando el rendimiento. A este tipo de agentes los denominamos "communicator ants", a los que habitualmente haremos referencia como "agentes comunicadores".

Utilizar un nuevo tipo de hormiga tiene sus ventajas e inconvenientes. Por un lado, cuantos más agentes se envíen, mayor será la información sobre la situación de la red, pero por otro lado, cuantos más agentes se envíen mayor será también la congestión generada. La cuestión es encontrar un punto de equilibrio entre el número de hormigas generadas y la mejora de rendimiento obtenida. El modo de acotar el número de agentes comunicadores generados, pasa por establecer un mecanismo de control que se adapte de forma dinámica a cada situación de la red.

Ventana de tiempos óptimos $\mathcal{W}_d^k$ El hecho de obtener un tiempo $T_{k\to d}$ menor que el mejor tiempo observado W_d durante una ventana de tiempos de tamaño w, no es un motivo suficiente para que la ruta sea comunicada al resto de vecinos. Cada vez que se alcanza el final de la ventana de tiempos t_{w+1}, el valor de W_d es reinicializado al valor de la media

en ese instante $\mu_d(t_{w+1})$, por lo que el valor mínimo cambia frecuentemente y serían demasiadas las notificaciones a enviar al resto de nodos vecinos, congestionando de ese modo la red y obteniendo un resultado negativo.

Por esa razón, se establece un modelo de ventana de tiempos óptimos $\mathcal{W}_d^k$ de tamaño w_{best}, a la que se incorporan solamente aquellos tiempos experimentados por las hormigas menores a W_d. A partir de este modelo, es posible obtener una referencia concreta de los últimos mejores tiempos experimentados en un momento determinado, facilitando obtener un límite inferior que permite determinar cuándo debería comunicarse el nuevo mejor tiempo para alcanzar el destino d:

$$I_{best} = min\left(\mathcal{W}_d^k\right) \tag{13}$$

Generación y envío de agentes comunicadores Una vez generada la ventana de tiempos óptimos $\mathcal{W}_d^k$, cualquier tiempo $T_{k\to d}$ menor que el límite inferior I_{best}, dará lugar a la generación de agentes comunicadores $C_{k\to n}$ en el nodo k. De forma reactiva, se enviará un agente "communicator ant" $C_{k\to n}$ a cada nodo vecino n, a excepción del nodo que pertenece a la ruta evaluada $n \in \mathcal{N}_k$, $n \notin V_{k\to d}$.

Los agentes $C_{k\to n}$ incluyen, entre otra información básica, el nodo de destino d y el tiempo estimado $T_{k\to d}$ para la ruta $V_{k\to d}$. Además, al igual que los agentes "forward ant" y "backward ant" en AntNet-FA, los agentes comunicadores utilizan las colas de prioridad para llegar a cada uno de los nodos vecinos.

Procesamiento de los agentes comunicadores Cuando cualquiera de los nodos vecinos $n \in \mathcal{N}_k$ recibe un agente $C_{k\to n}$, lo primero que hace es obtener los datos del nodo destino d y del tiempo $T_{k\to d}$ comunicados por el nodo k. El tiempo $T_{k\to d}$ representa el tiempo necesario para alcanzar el destino d desde el nodo k que generó el agente. Para poder evaluar el tiempo desde el nodo actual, hay que sumar al tiempo comunicado $T_{k\to d}$ el tiempo estimado adicional que sería necesario para el trayecto del último enlace $n \to k$, que une el nodo actual n con el nodo k desde el que se recibió el agente. Para estimar el tiempo $T_{n\to k}$ entre ambos enlaces, se utiliza la ecuación 12 del siguiente modo:

$$T_{n\to k\to d} = \left(d_{l_k} + \frac{q_{l_k} + s_a}{b_{l_k}}\right) + T_{k\to d}$$

donde d_{l_k}, q_{l_k} y b_{l_k} son respectivamente el retraso, el tamaño en bits de los paquetes que hay esperando en la cola y el ancho de banda del enlace que une el nodo n y el nodo k. s_a es el tamaño del paquete de la hormiga.

Una vez calculado $T_{n\to k\to d}$, si el tiempo estimado es menor que el mejor tiempo observado W_d para alcanzar el destino d, dentro de la ventana w de tiempos del nodo n, se procede a actualizar el modelo estadístico $\mathcal{M}^k$ y la tabla de feromonas $\mathcal{T}^k$ con el nuevo valor.

Pseudocódigo

Algorithm: Communicator Ants

```
// Procedimiento a ejecutar en el nodo donde se están evaluando los tiempos de ruta.
// Si detecta un tiempo óptimo, se generan y envían los agentes comunicadores
procedure update_statistical_traffic_model(dest_node, time_to_dest)
   update_mean(dest_node, time_to_dest)
   update_variance(dest_node, time_to_dest)
   if(time_to_dest < best_time[dest_node])
      update_best_time(dest_node, time_to_dest)
      Ibest ← min(ibest_times[dest_node])
      if(time_to_dest < Ibest)
         launch_communicator_ants(dest_node, time_to_dest)
      end if
      update_ibest_times_model(dest_node, time_to_dest)
   end if
end procedure

// Procedimiento a ejecutar en el nodo receptor que recibe el agente comunicador
// indicando el nodo destino y el tiempo de viaje estimado
procedure process_communicator_ant(comm_ant)
   neighbor_node ← comm_ant.source
   dest_node ← comm_ant.comm_dest
   time_to_dest ← comm_ant.comm_time_to_dest
   estimated_time_to_neighbor← propagation_delay + queue_delay
   new_trip_time ←time_to_dest + estimated_time_to_neighbor
   if(new_trip_time < best_time[dest_node])
      update_all_models(dest_node, neighbor_node, new_trip_time)
   end if
end procedure
```

5.4. Mecanismo de recompensa-penalización (*negative feedback*)

La mayoría de los algoritmos ACO utilizan una metodología de *recompensa-inacción* similar al aprendizaje por refuerzo, que consiste en reforzar de manera constante aquellas alternativas que proporcionan un mejor resultado, convergiendo poco a poco hacia una solución óptima o al menos cercana a ella.

El caso de AntNet es similar, ya que durante la fase "backward" solamente se realizan incrementos de feromona, independientemente de que el tiempo estimado sea óptimo o no.

El valor del incremento o refuerzo de feromona aplicado al enlace seleccionado es proporcional al resultado del experimento, mientras que los valores correspondientes al resto de nodos vecinos es decrementado por normalización. A continuación se describimos en detalle este proceso.

AntNet y la tabla de feromonas $\mathcal{T}^k$ Cada entrada $\tau_{nd} \in [0,1]\,, \sum_{n \in \mathcal{N}_k} = 1$ de la tabla de feromonas $\mathcal{T}^k$ del nodo k, representa la bondad de seleccionar el nodo vecino $n \in \mathcal{N}_k$ para alcanzar el destino d. Como ya hemos comentado en el punto 5.3, las actualizaciones de los valores de feromonas podrán producirse tanto para la ruta completa $s \to d$, como para todas las posibles sub-rutas desde cada uno de los nodos intermedios $k \in V_{s \to d}$, $k \neq d$. Siendo k el nodo actual, el valor de cada entrada τ_{nd} se actualiza de acuerdo a los tiempos de viaje $T_{k \to d}$, siempre que $T_{k \to d}$ no supere el límite superior I_{sup} del intervalo de confianza $I\left(\mu_d, \sigma_d^2\right)$. Si la entrada a actualizar es la del nodo vecino f, correspondiente al enlace que ha sido elegido previamente para llegar hasta d, la probabilidad se ve incrementada mediante la ecuación 8:

$$\tau_{fd} = \tau_{fd} + r\left(1 - \tau_{fd}\right)$$

Respecto a las entradas del resto de nodos vecinos τ_{nd}, $n \in \mathcal{N}_k$, $n \neq f$, su probabilidad se ve decrementada por normalización:

$$\tau_{nd} = \tau_{nd} - r \times \tau_{nd}$$

donde r es el refuerzo a aplicar, calculado mediante la ecuación 9 a partir del tiempo $T_{k \to d}$ y de los datos recogidos en el modelo estadístico $\mathcal{M}^k$.

Di Caro, uno de los autores de AntNet, describe el incremento de feromonas como la suma de refuerzos explícitos e implícitos [Di Caro, 2004]. Los refuerzos explícitos derivan de aplicar incrementos con un valor de refuerzo r que depende de la calidad de la ruta y los refuerzos implícitos no dependen tanto del valor de refuerzo, sino de la frecuencia de llegada de hormigas.

Modificación propuesta En AntNet se ignoran los tiempos por encima del límite suerior I_{sup} del intervalo de confianza, tiempos que podrían indicar ciertas situaciones de cambio en los patrones de tráfico o de problemas en la red. Nuestra propuesta consiste en utilizar una metodología de *recompensa-penalización*, aplicando un refuerzo positivo cuando el resultado es óptimo o aplicando una penalización cuando el resultado está a una distancia considerable de la solución óptima. De este modo, se acelera la adaptación de las tablas de feromonas con respecto al mecanismo *recompensa-inacción*, reduciendo el tiempo que necesita el algoritmo para converger a la solución óptima.

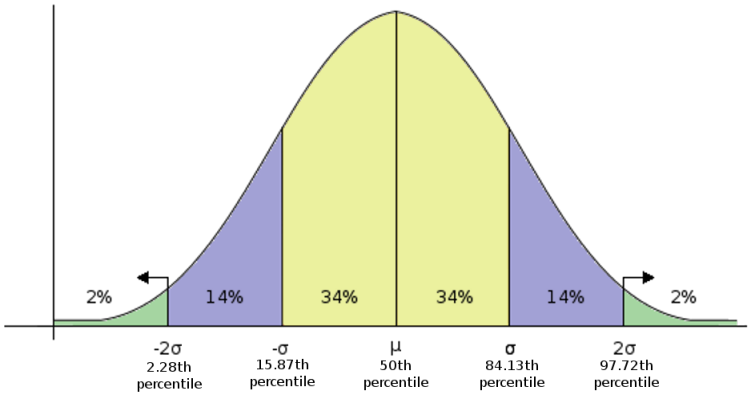


Figura 8: Representación de la distribución de los datos respecto a la media y la desviación típica.

Ventana de tiempos no óptimos $\mathcal{U}_d^k$ El hecho de obtener tiempos fuera del intervalo de confianza $I\left(\mu_d, \sigma_d^2\right)$ del modelo $\mathcal{M}^k$, no indica que haya ocurrido un cambio en la red, ya que pueden existir varias rutas posibles desde un mismo nodo k a un nodo d, y quizás algunos de los tiempos, dependiendo de la ruta elegida, pueden variar lo suficiente como para superar el límite I_{sup}.

Para poder determinar cuándo es necesario penalizar un tiempo $T_{k\to d}$, se establece una ventana de tiempos no óptimos $\mathcal{U}_d^k$ de tamaño w_{worst} que incorpora los tiempos que estén por encima del límite superior I_{sup}. A partir de esta ventana, es posible obtener un intervalo de confianza con un nuevo límite superior I_{worst}:

$$I_{worst} = \mu\left(\mathcal{U}\right) - \left(\beta \times \sigma\left(\mathcal{U}\right)\right) \tag{14}$$

donde β es el factor que establece cómo limitar el valor de I_{worst} respecto de la media de tiempos observados $\mu\left(\mathcal{U}\right)$, al utilizar la desviación típica $\sigma\left(\mathcal{U}\right)$. En la Figura 8 puede observarse cómo β determina la cantidad de muestras a tener en cuenta a partir de una distribución normal de los datos.

Sin embargo, $\mathcal{U}_d^k$ no es una ventana de tiempos como tal, sino una ventana de distancias entre el límite superior I_{sup} y el tiempo observado $T_{k\to d}$, de modo que cada entrada $u_d^k \in \mathcal{U}_d^k$ se calcula del siguiente modo:

$$u_d^k = 1 - \frac{I_{sup}}{T_{k\to d}}$$

La razón de utilizar la distancia, en lugar del tiempo $T_{k\to d}$, es que el límite superior I_{sup} varía con el tiempo.

Detectar tiempos penalizables Una vez obtenido un modelo de tiempos no óptimos $\mathcal{U}_d^k$, es posible obtener un nuevo intervalo de tiempos con límite inferior I_{sup} y límite superior I_{worst}, que permite discriminar tiempos que habitualmente pueden superar el límite $I_{sup}\left(\mu_d, \sigma_d\right)$, de aquellos que son altos debido a un posible cambio en la red. Mientras

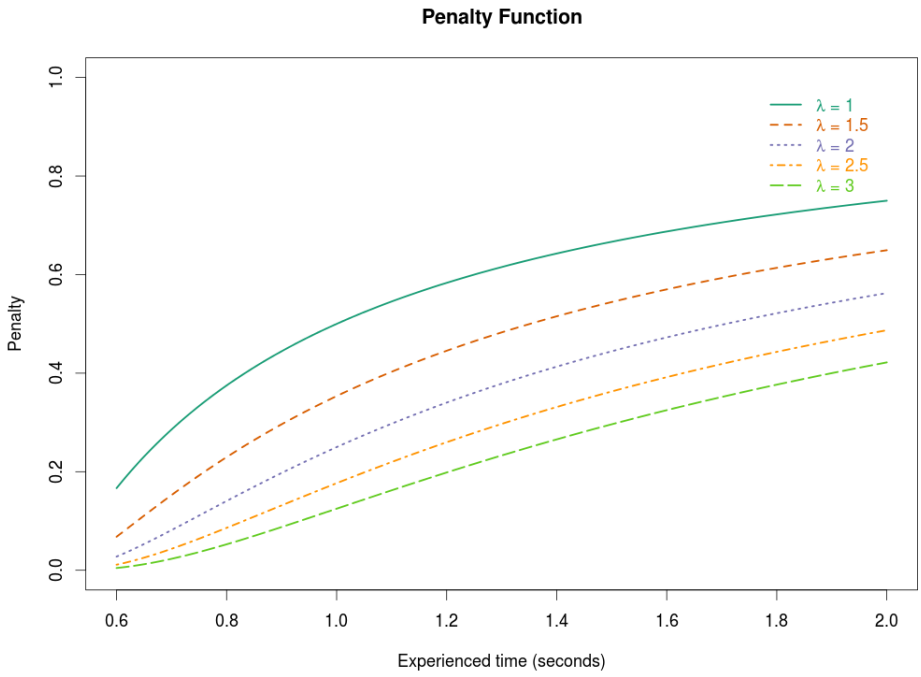


Figura 9: Representación de la transformación exponencial para el cálculo de la penalización.

que los primeros normalmente estarán por encima del límite I_{sup} y dentro del límite del intervalo de tiempos I_{worst}, los segundos estarán por encima del límite I_{worst}.

Cualquier tiempo $T_{k \to d}$ mayor que el nuevo límite superior I_{worst}, dará lugar a una penalización en la correspondiente entrada de feromonas.

Cálculo del factor de penalización Una vez detectado un tiempo T_{high} superior al intervalo de confianza I_{worst}, el paso siguiente será calcular el factor de penalización ρ a aplicar. Dicho factor dependerá de la distancia entre el tiempo T_{high} y el límite superior del intervalo de confianza I_{sup} actual:

$$\rho = \left(1 - \frac{I_{sup}}{T_{high}}\right)^{\lambda}$$

donde λ es un factor que incrementa o decrementa el nivel de penalización respecto a la diferencia entre T_{high} e I_{sup}. En la Figura 9 puede observarse cómo afecta el valor de λ.

Aplicación de la penalización El factor de penalización ρ decrementa el valor de la tabla de feromonas para la entrada τ_{fd}, siendo f el enlace del nodo vecino utilizado para alcanzar el destino d en el que se ha obtenido un tiempo elevado $T_{high} > I_{worst}$:

$$\tau_{fd} = \tau_{fd} - \rho\,\tau_{nd} \tag{15}$$

La cantidad de feromona que acaba de ser decrementada en el enlace penalizado, es distribuida entre el resto de los enlaces de los nodos vecinos $n \in \mathcal{N}_k$, $n \neq f$ del siguiente modo:

$$\tau_{nd} = \tau_{nd} + \frac{\rho\,\tau_{nd}}{|\mathcal{N}_k| - 1} \tag{16}$$

Pseudocódigo

Algorithm: Penalty

```
// Procedimiento a ejecutar en el nodo en el que se están evaluando los tiempos
// de ruta. Si detecta un tiempo fuera del límite, aplica la penalización
procedure update_routing_table(dest_node, neighbor)
   Isup←calculate_isup()
   if(time_distance > isup_times_model)
      time_distance ← 1 - (Isup / time_to_dest_node)
      Iworst ← mean(isup_distance[dest_node]) - beta * stddev(isup_distance[dest_node])
      if(time_distance > Iworst)
         apply_penalty(dest_node, neighbor, time_distance)
      end if
      update_isup_distance_model(dest_node, time_distance)
   end if
end procedure

// Procedimiento a ejecutar en el nodo receptor que recibe el agente comunicador
// indicando el nodo destino y el tiempo de viaje estimado
procedure apply_penalty(dest_node, neighbor, time_distance)
   penalty ← calculate_penalty_factor(time_distance)
   pheromone ← get_pheromone(dest_node, neighbor)
   pheromone_diff← pheromone * penalty
   update_pheromone(dest_node, neighbor, pheromone - pheromone_diff)
   for(each neighbor n except the link of the penalized neigbour)
      pheromone ← get_pheromone(dest_node, n)
      new_pheromone_value ← pheromone + pheromone_diff / (num_neigbours - 1)
      update_pheromone(dest_node, n, new_pheromone_value)
   end if
end procedure
```

6. Simulación y resultados

En esta sección describimos de forma general cómo han sido implementados los algoritmos, la configuración y el entorno utilizado para realizar las simulaciones.

Concretamente, describimos las herramientas de simulación utilizadas para implementar los algoritmos, AntNet-FA y el algoritmo propuesto Abira, el modelo de red considerado en ambas implementaciones y la configuración utilizada para las simulaciones, que se compone de: la topología de red, los parámetros de red, los parámetros de los algoritmos y los diferentes escenarios en los que hemos realizado los experimentos.

6.1. Entorno de simulación

Un simulador de red permite modelar cualquier tipo de red de telecomunicación especificando el comportamiento de los nodos y los enlaces de comunicación que la componen. La mayoría de las herramientas de simulación de red utilizan entornos de simulación basados en eventos discretos (DES).

Un entorno de simulación de eventos discretos es un sistema donde los eventos (o cambios de estado) ocurren en instantes de tiempo concretos y procesarlos no suponen un incremento adicional de tiempo en la simulación. A diferencia de los entornos dinámicos continuos, en un entorno discreto no ocurre nada de especial relevancia entre dos eventos consecutivos, por lo que no hay cambios de estado en el sistema entre cada uno de los eventos.

Tras revisar las diferentes alternativas sobre entornos de simulación en tiempo discreto dedicados a la simulación de red y algunos análisis [Weingartner et al., 2009; Pan and Jain, 2008], elegimos el entorno OMNeT++ [Varga, 2011]. Las razones son varias: su enfoque abierto, su estructura de programación orientada a objetos, su equilibrado rendimiento y capacidad de escalabilidad, la cantidad de modelos y librerías existentes, y otras características como ofrecer una rica interfaz gráfica de usuario y un lenguaje de modelado abstracto, sencillo y funcional.

Una simulación diseñada bajo OMNeT++ se compone de dos tipos de módulos: módulos simples y módulos compuestos. Los módulos simples, programados en C++, son el núcleo de la simulación, ya que describen de forma individual el comportamiento del modelo. Los módulos compuestos engloban uno o varios módulos simples y son escritos en NED, un lenguaje descriptivo propio de OMNeT++ que permite especificar los parámetros del modelo de un modo sencillo y sin necesidad de volver a compilar.

6.2. Modelo de simulación adoptado

Los algoritmos de enrutamiento trabajan en la capa de red IP (ver punto 2.1), que utiliza una estrategia de entrega de mejor esfuerzo tratando simplemente de escoger la mejor

ruta posible en la red. Sin una capa de transporte que controle el tráfico de red, el protocolo IP es un servicio no fiable y no orientado a conexión, en el cual los paquetes al ser distribuidos por diferentes rutas pueden llegar desordenados[6] o simplemente no llegar.

En este caso, el objetivo es centrarnos en la funcionalidad de enrutamiento, comprobar su capacidad y su rendimiento, de modo que implementar una capa de red que establezca un control sobre los paquetes, requiera de confirmaciones de recepción o el establecimiento de una conexión, como hace TCP, es algo que descartamos. Cada componente de control adicional puede tener un un impacto considerable en el rendimiento del algoritmo de enrutamiento, siendo bastante difícil evaluar y estudiar cómo afecta cada uno de estos componentes. Por ese motivo y al igual que en las simulaciones realizadas en el trabajo original de AntNet, consideramos una capa de transporte similar a UDP, donde los paquetes son encapsulados en datagramas y enviados a través de la red sin que se haya establecido previamente una conexión, sin confirmación de recepción ni control de flujo, pudiendo sufrir cualquiera de los problemas que el protocolo IP de pérdida de paquetes o la llegada desordenada. En cualquier caso, ninguno de los protocolos (IP y UDP) se implementan de forma completa, sino que se ajustan lo máximo posible a la realidad centrándonos en aquello que realmente nos interesa, que es procesar los paquetes y enrutarlos.

De forma general, consideramos que la red a simular tiene las siguientes características:

- Se considera un modelo de red simplificado, en el que la red se compone solamente de nodos genéricos que actúan tanto de host como router, realizando las funciones de generar tráfico y de enrutarlo.

- Asumimos una estabilidad topológica en la red y suponemos que todos los nodos de la red conocen los identificadores de cada uno de los nodos que la componen. Por ejemplo, no se tienen en cuenta que un nuevo nodo pueda ser conectado a la red. Aunque el descubrimiento de nodos puede considerarse una tarea más de cualquier router, es un procedimiento diferente al enrutamiento y en este caso carece de valor. Además, dado que el modelo que contemplamos es sobre redes cableadas, no es una tarea que podamos considerar fundamental para el desarrollo de la simulación.

- Los enlaces que unen cada uno de los nodos se caracterizan por tener un ancho de banda determinado y un retraso de propagación asociado. Por lo tanto, el tiempo que tarde un paquete en llegar desde un nodo a su vecino dependerá del tamaño del propio paquete y del retraso de propagación del enlace.

- Generación de paquetes de datos en base a sesiones de tráfico. Entendemos las sesiones como una aplicación de usuario, que de forma estocástica, genera tráfico de red desde cualquier nodo s actuando como host, hacia un nodo destino d. La selección del nodo de destino d se realiza de forma aleatoria y con probabilidad uniforme entre todos los nodos de la red. El tiempo medio de llegada entre dos sesiones y el tiempo medio de llegada entre los paquetes de cada una de las sesiones se establecen según una distribución exponencial negativa de media $MSIA$ y $MPIA$.

[6] La variación en el tiempo en la llegada de los paquetes se conoce como *jitter*.

Tanto los paquetes de datos, como el número de paquetes que componen la sesión tienen un tamaño fijo. Cada una de las sesiones mantiene un control del número de paquetes que todavía quedan por ser enviados.

- No existe un control sobre los paquetes. Cada paquete enviado se considera como correctamente recibido. No hay ningún mecanismo de confirmación de la recepción, control del orden de secuencia o control de errores en los datos. En consecuencia, tampoco se contempla un mecanismo de reenvío en caso de error o pérdida.

- Los nodos de la red siguen una política "store and forward". Existe un buffer por enlace que mantiene dos tipos de colas de paquetes: la cola regular y la cola de prioridad. Dependiendo del tipo de paquete a almacenar en el buffer será incorporado a la cola regular o a la cola de prioridad. Los paquetes de cada una de las colas se sirven bajo una política FIFO (first-in-first-out) y los paquetes de la cola de prioridad siempre tienen prioridad sobre los de la cola regular. Cuando un paquete es recibido por cualquier nodo intermedio k se comprueba su destino d y se selecciona el enlace al que reenviar el paquete para llegar a él en base a las entradas de la tabla de feromonas $\tau_{nd} \in \mathcal{T}^k$. Si el enlace seleccionado está libre, el paquete se reenvía directamente al nodo vecino, pero si está ocupado, el mensaje se almacena en la cola correspondiente del buffer del enlace. El tiempo de servicio de un paquete encolado dependerá del número de paquetes en la cola con mayor prioridad u orden, y de las características físicas del enlace (ancho de banda y retraso de propagación). Si la cola en la que se intenta incorporar el nuevo paquete está llena el paquete se descarta y se pierde.

Medidas de rendimiento A la hora de utilizar una métrica para evaluar y comparar la bondad de los caminos $F_{s \to d}$ realizados por las hormigas, hemos mantenido el mismo indicador que el algoritmo AntNet original: el tiempo tiempo de viaje estimado $T_{s \to d}$. $T_{s \to d}$ representa el tiempo acumulado que depende del estado del tráfico en ese momento, de las propias características de la red (el ancho de banda y el retraso de propagación de cada uno los enlaces) y del número de saltos entre ambos nodos.

Podíamos haber utilizado una alternativa diferente, como el número de saltos, pero consideramos que no es una buena medida de rendimiento en una red cableada. Minimizar el número de saltos no tiene sentido en una red cableada si en consecuencia obtenemos rendimiento menor al aumentar el tiempo de entrega (end-to-end delay) o al reducir la cantidad de información entregada por unidad de tiempo. Sí sería una medida a tener en cuenta en redes en las que es relevante minimizar el número de saltos, aún a pesar de incurrir en una pérdida de rendimiento, como es el caso de redes MANET. En este tipo de redes los equipos suelen utilizar baterías, de modo que minimizar el número de saltos significa minimizar los recursos utilizados, dando lugar a un menor consumo.

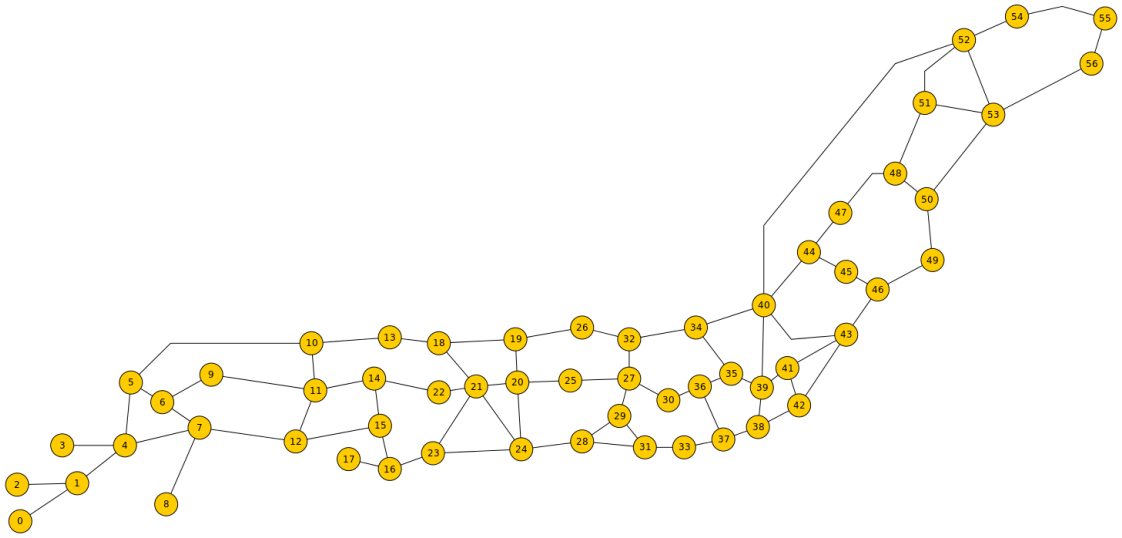


Figura 10: Red troncal de NTT, Japón. Se trata de una red real compuesta de 57 nodos y 162 enlaces bidireccionales con un ancho de banda de 6 Mbit/sec y un retraso de propagación de 2 a 5 milisegundos.

6.3. Topología de red, parámetros y escenarios de simulación

Tras diseñar el modelo de simulación, es necesario configurar el entorno, que se compone de la topología de red a utilizar, los parámetros de los algoritmos y los escenarios que se pretenden llevar a cabo en la simulación.

Podríamos dividir los parámetros de la simulación en tres categorías diferentes: los parámetros de la red, que establecen las capacidades de los nodos de la red y el formato de los paquetes a utilizar; los parámetros del algoritmo AntNet, que también son utilizados en Abira; y los parámetros del propio algoritmo Abira.

A continuación describimos cómo se han definido cada uno de estos apartados.

Topología de red Hemos elegido una red real de un tamaño medio y con nodos muy distribuidos. Se trata de la red troncal de fibra óptica de la empresa japonesa NTT (Nippon Telegraph and Telephone Corporation). En la Figura 10 puede observarse la topología de la red. Es una red de 57 nodos y 162 enlaces de conexión bidireccionales con un ancho de banda de 6 Mbits/sec y un retraso de propagación que va de 2 a 5 milisegundos.

Parámetros de la red El tamaño de las sesiones establecidas entre cada par de nodos se ha fijado en 2 Mbytes, el de los paquetes de datos en 512 bytes y el de los paquetes de hormigas (independientemente del tipo) en 48 bytes. El tiempo medio de llegada entre dos sesiones (*MSIA*) y el tiempo medios de llegada entre los paquetes de cada una de las sesiones (*MPIA*) dependen del experimento a realizar. Las colas del buffer de cada uno de los nodos tienen un tamaño máximo de 1000 paquetes, lo que equivale a 512 Kbytes por cola o 1 Mbyte por buffer, ya que mantiene la cola regular y la cola de prioridad.

window coefficient c	window factor η	queue weight α	reinforcement factor c_1	reinforcement factor c_2	confidence level z	rescaling factor ε
0.3	0.15	0.45	0.8	0.2	1.72	1.2

Tabla 1: Lista de parámetros base utilizados en la simulación de AntNet y Abira.

exploration factor p_{noise}	comm window size w_{best}	penalty window size w_{worst}	pentalty range β	penalty factor β
0.001	100	50	1.5	3

Tabla 2: Lista de parámetros utilizados en el algoritmo Abira.

Parámetros básicos del algoritmo AntNet Dado que Abira es un algoritmo basado en AntNet, ambos comparten los mismos parámetros básicos. El número de parámetros es bastante elevado, por lo que es difícil afinar el algoritmo hasta obtener un rendimiento aceptable. En la mayoría de los casos hemos optado por utilizar aquellos valores que se sugerían en la propuesta original [Di Caro and Dorigo, 1998a,b; Di Caro, 2004]. En la tabla 1 se detallan algunos de los valores que hemos utilizado.

Parámetros del algoritmo Abira De forma adicional a los parámetros base de AntNet, el algoritmo Abira requiere utilizar cinco nuevos parámetros para las tres modificaciones. El factor de ruido p_{noise} establece la capacidad de exploración. El tamaño de ventana w_{best} permite ajustar el envío de agentes comunicadores. Y respecto a los parámetros para ajustar cómo aplicar las penalizaciones, tenemos el tamaño de ventana w_{worst}, el rango del intervalo β y el factor de penalización β. En la tabla 2 se muestran los valores utilizados durante las simulaciones.

Escenarios de simulación Podemos encontrar muchas situaciones de tráfico de red o topologías diferentes, y por ese motivo, hemos considerado realizar aquellos experimentos que mayor relevancia pueden tener a la hora de comparar el rendimiento de ambos algoritmos. Por ejemplo, una situación estacionaria donde la carga de tráfico es baja no aportará ningún tipo de información, ya que los algoritmos se comportan de forma similar. Lo que nos interesa es estudiar la evolución del rendimiento en situaciones transitorias, como pueden ser:

- Un cambio en los patrones de tráfico. Por ejemplo, a partir de una situación estacionaria con una carga de tráfico baja, el tráfico se ve incrementado de forma considerable en un momento determinado y de forma temporal. En este caso es interesante evaluar cómo las rutas óptimas hasta ese momento se congestionan y el tráfico empieza a ser redistribuido por rutas alternativas.
- Un cambio en las rutas disponibles. Podría ocurrir que en un determinado momento una de las rutas dejara de estar disponible, ya sea debido a un fallo en la red o

una desconexión del nodo. En este caso es interesante ver cómo el resto de rutas se adaptan para redistribuir la carga de tráfico que ya no puede viajar por la ruta desaparecida.

6.4. Resultados

Durante los experimentos hemos observado que los parámetros utilizados, tanto los parámetros base de AntNet, como los de las nuevas propuestas de Abira, son altamente sensibles. Algunas variaciones determinan obtener un rendimiento cercano al esperado o un rendimiento negativo. El hecho de que estos parámetros se definan de forma experimental, limita poder encontrar los valores óptimos que aporten el mejor rendimiento posible.

Problemas con la modificación propuesta de penalización La modificación propuesta que penaliza aquellos tiempos por encima de un límite superior, calculado a partir del propio límite superior I_{sup} de AntNet, no ha podido ser aplicada con resultados positivos, bien debido a un problema de planteamiento, o bien por un problema de ajuste de los parámetros. De modo que durante el apartado de resultados, solamente las dos primeras modificaciones propuestas, la de exploración y la de comunicación de las mejores rutas, han sido implementadas en Abira. Hemos incluido un último punto al final de esta sección analizando los problemas de inestabilidad que nos hemos encontrado.

Todos los datos mostrados en los resultados son los valores medios obtenidos de 10 ejecuciones independientes diferentes. Cada ejecución equivale a 1000 segundos de tiempo real, de los cuales los 500 primeros se han utilizado para inicialización de las tablas de feromonas en ausencia de tráfico de datos y el resto es tiempo utilizado para evaluar el comportamiento del algoritmo. Observando los resultados, consideramos que 500 segundos es tiempo suficiente para poder evaluar los estados transitorios inducidos en cada escenario y devolver la red de nuevo al estado estacionario. Utilizar 500 segundos en el periodo de inicialización resulta excesivo, pero se ha mantenido así por ser el tiempo que utliza el algoritmo original AntNet.

Las medidas de rendimiento que hemos utilizado para comparar los resultados en cada uno de los experimentos han sido las habituales en este ámbito: la cantidad de datos entregados por unidad de tiempo (throughput) y el tiempo de entrega de los paquetes (end-to-end delay). En los resultados, presentamos cada una de estas medidas siguiendo también el mismo criterio que el trabajo original de AntNet [Di Caro and Dorigo, 1998a]:

- Las medidas de throughput se muestran como la media de datos entregados por unidad de tiempo. En este caso no se tiene en cuenta la varianza, ya que no existe una variabilidad en los datos como para ser relevante.
- Las medidas de los tiempos de entrega se presentan mediante la media y mediante el uso del percentil 90. La razón es que los tiempos de entrega pueden sufrir grandes variaciones, dependiendo de la distancia entre los nodos (por ejemplo, nodos adyacentes o nodos que están en cada extremo de la red) o de la capacidad física de

los enlaces por los que deben viajar los paquetes de datos. Por lo tanto, la varianza tampoco puede resultar significativa y en su lugar se mantiene una medida que representa al 90 % de los datos.

Las comparativas entre ambos algoritmos muestran diferencias mucho más pequeñas a nivel de throughput que a nivel de tiempos de entrega. Esto se debe a que los paquetes de datos se acumulan en las colas del buffer de los nodos de la red, aumentando retrasos en la entrega sin afectar demasiado al número de paquetes entregados. A diferencia de la pérdida de rendimiento en los retrasos de entrega, la pérdida de rendimiento en el throughput se produce cuando las colas del buffer están llenas y los paquetes se descartan, algo que ocurre en situaciones de alta congestión o fallos en los enlaces. Ambas medidas, throughput y retrasos de entrega, están de algún modo interrelacionadas, ya que el incremento en los retrasos de entrega durante un amplio espacio de tiempo da lugar a que las colas se llenen y se descarten paquetes. Por el contrario, el hecho de que los paquetes no sean entregados debido a su acumulación, da lugar a un incremento del tiempo de entrega.

Tras realizar diferentes experimentos, modificando los patrones de tráfico de la red y de la propia topología, los resultados muestran que en cualquiera de las condiciones, Abira muestra un rendimiento similar o superior que el de AntNet-FA.

A continuación, comentamos los resultados obtenidos describiendo en detalle las diferencias entre AntNet-FA y Abira.

6.4.1. Escenario 1: Estado estacionario

El propósito de este escenario es comprobar el comportamiento de ambos algoritmos ante situaciones de tráfico estacionarias.

En general, dado un algoritmo ACO, puede garantizarse con probabilidad muy cercana a 1 que converge hacia una solución óptima siempre que el algoritmo haya sido ejecutado un número lo suficientemente elevado de veces [Stützle et al., 2002; Dorigo and Stützle, 2004].

En este caso, entendemos la solución óptima como la ruta que entrega los paquetes de datos en un menor tiempo, al combinar las rutas más cortas con aquellas que menor carga de tráfico tienen. Podemos decir entonces que cualquier algoritmo de enrutamiento basado en ACO siempre converge hacia la solución óptima en condiciones estacionarias de tráfico, con la única diferencia del tiempo que necesite para hacerlo.

En un escenario estacionario los resultados esperados son muy similares, ya que ambos algoritmos tienen la misma base y las modificaciones propuestas tienen un efecto sobre la capacidad de adaptación a los cambios en la red. Las diferencias de rendimiento pueden derivar del tiempo inicial para converger o de situaciones de congestión.

Para esta simulación se han elegido tres tipos de carga de tráfico diferentes: baja, media y alta. En cada una de ellas se ha establecido un tiempo medio de llegada entre dos sesiones $MSIA = 4{,}7s$, $2{,}7s$, $1{,}7s$, y un tiempo medio de llegada entre paquetes $MPIA = 0{,}005s$. De este modo, incrementamos el número de sesiones que se van a establecer de forma

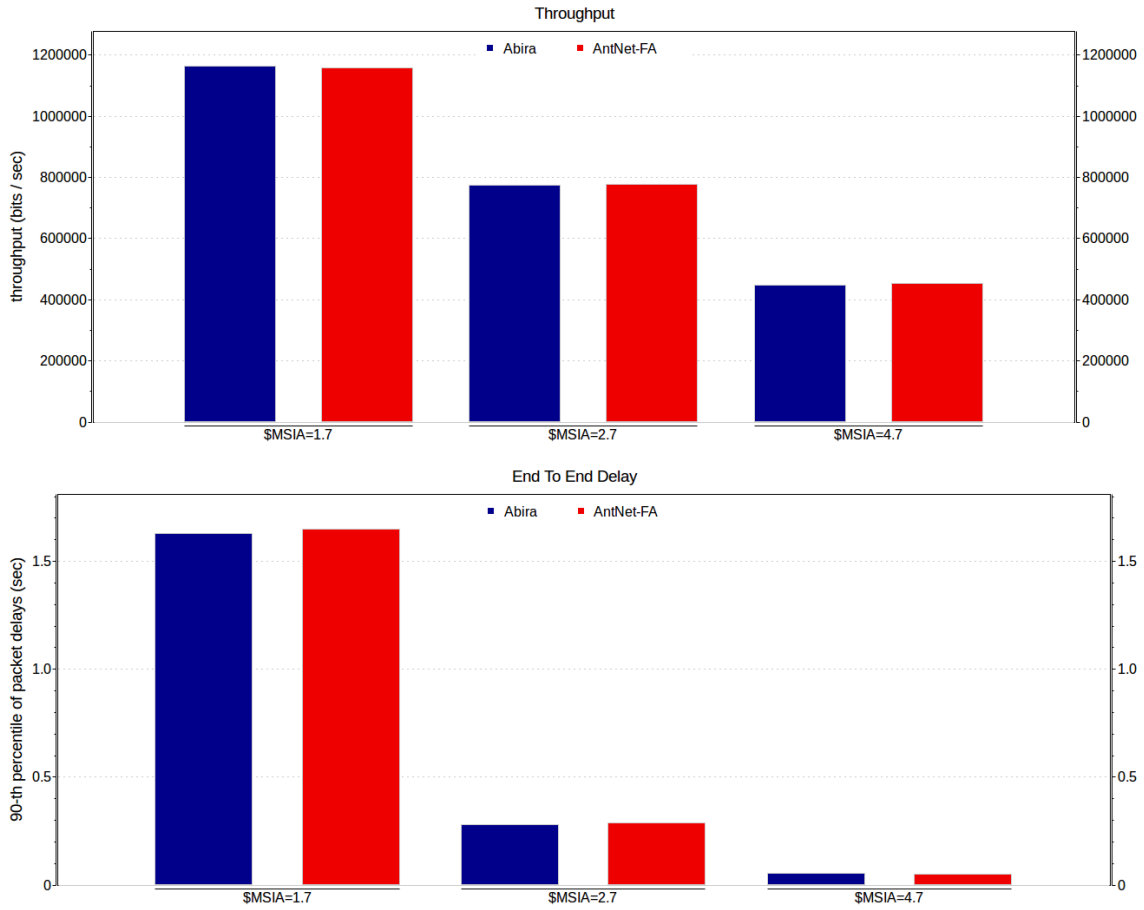


Figura 11: Comparativa de Abira y AntNet-FA en un escenario estacionario. Se muestran el throughput y tiempos de entrega obtenidos en diferentes situaciones de carga de tráfico al reducir el tiempo medio de llegada entre dos sesiones: $MSIA = 4{,}7s$, $MSIA = 2{,}7s$ y $MSIA = 1{,}7s$.

simultánea entre cada par de nodos de la red, manteniendo en cada una de estas sesiones un tiempo medio de generación de paquetes común.

La Figura 11 muestra los resultados obtenidos en las diferentes simulaciones. Puede observarse que los valores de throughput son muy similares en ambos casos, un poco más altos en Abira cuando la carga de tráfico está en una situación de saturación $MSIA = 1{,}7s$. Ocurre algo parecido en el caso de los tiempos de entrega, donde Abira obtiene un mejor resultado a medida que que se incrementa la cantidad de tráfico en la red.

En situaciones donde la carga de tráfico es baja podemos observar que ambos algoritmos muestran un rendimiento similar, y a medida que el tráfico se incrementa, aparecen pequeñas diferencias de rendimiento. La razón es que, en situaciones de congestión, las rutas deben sufrir más adaptaciones y en este caso Abira lo hace más rápido.

6.4.2. Escenario 2: Cambios en la topología de red

El propósito de este escenario es estudiar el efecto de un cambio en las rutas disponibles de la red.

AntNet no está pensado para gestionar un cambio temporal o permanente en la topología de red, como puede ser el fallo de un nodo, un fallo de enlace o que nuevos nodos sean incorporados a la red. Para gestionar de forma adecuada este tipo de situaciones sería necesario dotar de una capacidad adicional al algoritmo de enrutamiento. Un ejemplo de este tipo de mecanismos es el utilizado en AntHocNet [Di Caro et al., 2005], en el que existe un proceso de monitorización que permite descubrir fallos en la red. Ante la detección de error de un enlace, actualiza las tablas de enrutamiento y envía mensajes de notificación al resto de nodos de la red.

A pesar de no implementar mecanismos de control de fallos, consideramos interesante comprobar cómo cada uno de los algoritmos consigue recomponer las tablas de feromonas cuando una de las rutas deja de estar disponible, ya que determina la velocidad para converger a una nueva solución óptima. El algoritmo debería ser capaz de evaluar rutas alternativas entre las que redistribuir la carga de tráfico, además de hacerlo en un tiempo relativamente corto para evitar la congestión en los nodos involucrados y la consiguiente pérdida de paquetes.

Para asegurarnos de que la causa de la congestión o pérdida de paquetes es el cambio en la topología de red, establecemos una carga de tráfico baja, con un tiempo medio de llegada entre dos sesiones $MSIA = 4{,}7s$ y un tiempo medio de llegada entre paquetes $MPIA = 0{,}3s$. En este escenario, el nodo 21 dejará de procesar los paquetes desde el segundo 200 hasta el segundo 300. De forma adicional, el nodo 40 dejará de funcionar desde el segundo 250 hasta el segundo 350. Durante un corto espacio de tiempo, ambos nodos no estarán funcionando (ver en la Figura 10 la topología de red), lo que ocasionará la pérdida de paquetes y el embotellamiento. Aunque exista un inevitable incremento inicial de la congestión de tráfico al producirse cada uno de los fallos, éste debería poco a poco ser redistribuido entre rutas alternativas.

La Figura 12 muestra los resultados obtenidos en el experimento. Puede observarse que en ambos casos el nivel de throughput es muy parecido y que no muestra ningún tipo de degradación del rendimiento debido al envío de agentes comunicadores. En cambio, sí que podemos observar una evidente mejora del rendimiento en los tiempos de entrega de Abira respecto a los tiempos de entrega de AntNet-FA.

En el segundo 200, cuando el router 21 cae, se produce un pequeño incremento de tiempo en ambos algoritmos que parece que consigue solucionarse redistribuyendo el tráfico entre los nodos vecinos 18, 19, 20, 22, 23 y 24. Sin embargo, cuando en el segundo 250 cae el nodo 40, uno de los principales nodos de acceso a la parte alta de la red, y con el nodo 43 como única alternativa, el incremento en el tiempo de entrega es considerable. Es en este punto en el que Abira se comporta mejor que AntNet-FA, permitiendo encontrar rápidamente la ruta alternativa, tanto en la caída del segundo nodo, como ante la recuperación del primero.

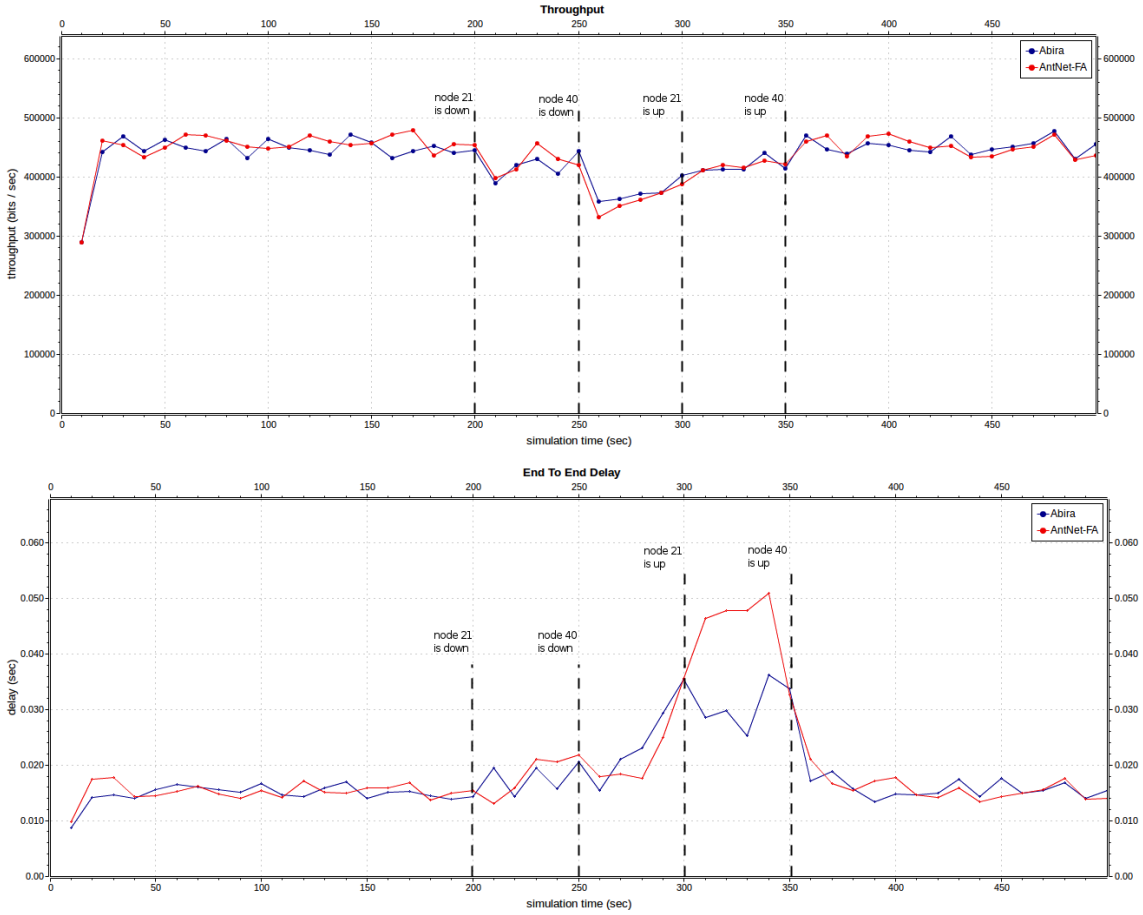


Figura 12: Comparativa de Abira y AntNet-FA en un escenario de fallo: (a) throughput, (b) tiempos de entrega. Parámetros: $MSIA = 4{,}7s$, $MPIA = 0{,}3s$. Situación: el nodo 21 falla durante el intervalo $200-300s$ y el nodo 40 falla durante el intervalo $250-350s$.

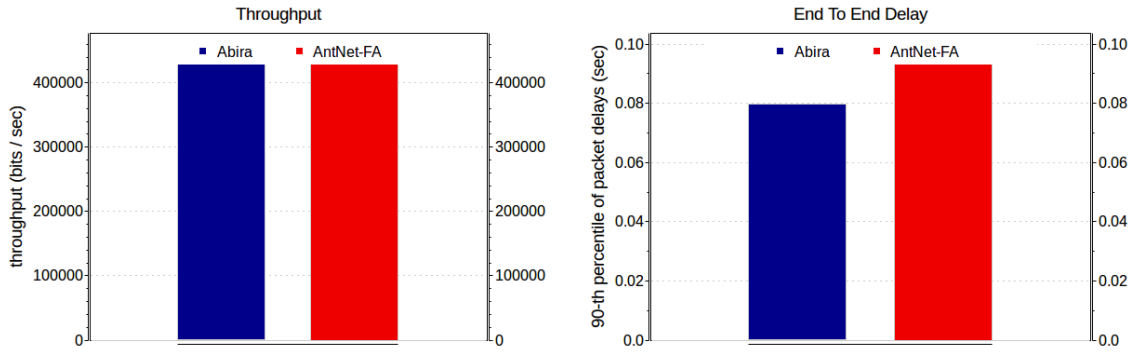


Figura 13: Comparativa de Abira y AntNet-FA en un escenario de fallo: (a) throughput, (b) tiempos de entrega - percentil 90. Parámetros: $MSIA = 4{,}7s$, $MPIA = 0{,}3s$. Situación: el nodo 21 falla durante el intervalo $200 - 300s$ y el nodo 40 falla durante el intervalo $250 - 350s$.

En la Figura 13 se muestran los resultados promedio de throughput y el percentil 90 de los tiempos de entrega. Como puede apreciarse, mientras el valor de throughput es similar, los tiempos de entrega en el caso de Abira son claramente mejores. Hay que tener en cuenta que el tiempo de entrega comparado en esta figura representa el 90 % de las muestras, frente al 30 % que corresponde al tiempo que los nodos han permanecido caídos y que es el momento donde realmente se aprecia una diferencia considerable como podemos ver en la Figura 12.

La Figura 14 muestra el número de agentes comunicadores enviados durante la simulación. Se puede observar cómo existe un periodo inicial donde se envían algunos agentes hasta el segundo 200, que corresponde con la caída del primer nodo. En ese momento, dado que los resultados de los experimentos no mejoran los tiempos registrados, y que de forma adicional el segundo nodo se detiene en el segundo 250, el envío de agentes se detiene. Vemos que a partir del segundo 300 se envía una pequeña cantidad de agentes como consecuencia de que el primer nodo vuelve a funcionar. Pero no es hasta el segundo 350, momento en el que el segundo nodo se recupera, cuando una mayor cantidad de agentes se envían como consecuencia de que muchos de los experimentos realizados por las hormigas "forward" y "backward" están encontrando mejores alternativas. De este modo, los agentes comunicadores ayudan a reducir el tiempo necesario para adaptarse a la nueva situación de la red, al mismo tiempo que la medida de throughput no se ve reducida con respecto a AntNet-FA, lo que indica que la necesidad de enviar nuevos agentes no degrada el rendimiento.

6.4.3. Escenario 3: Cambio en los patrones de tráfico (*hotspot*)

El propósito de este escenario es estudiar el efecto de una sobrecarga de tráfico temporal en la red.

La idea es mantener una carga de tráfico relativamente baja, hasta que en un momento determinado, uno de los nodos actúe como *hotspot*[7] recibiendo una gran cantidad de tráfico

[7]El concepto de hotspot suele utilizarse como un punto de acceso público que ofrece acceso a Internet a

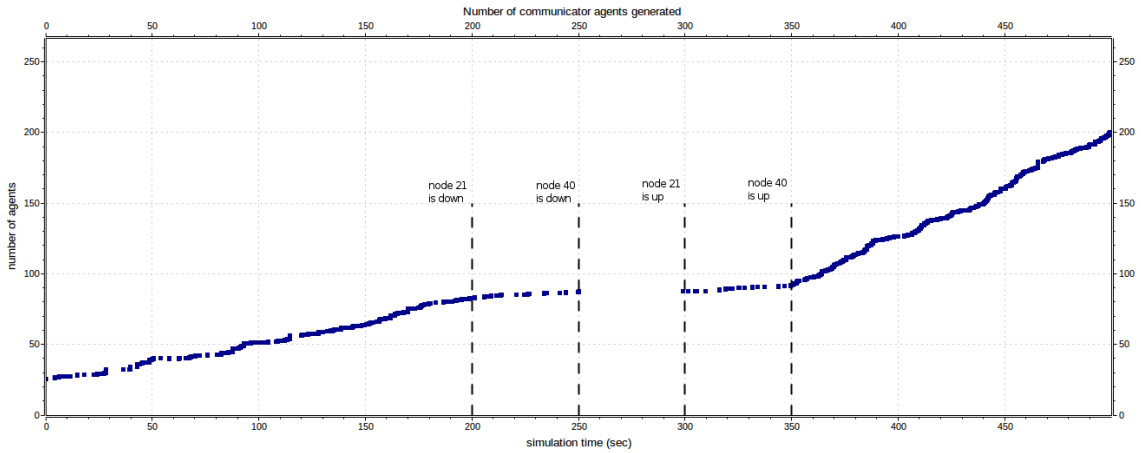


Figura 14: Agentes "communicator ant" enviados durante la simulación en un escenario de fallo. El nodo 21 falla durante el intervalo $200-300s$ y el nodo 40 falla durante el intervalo $250-350s$.

desde todos los nodos de la red. Al principio, las sesiones se establecen de forma aleatoria entre cada par de nodos como es habitual. Más tarde, de forma adicional al tráfico que ya existe, todos los nodos de la red establecen una sesión fija con el nodo que actúa como hotspot. Pasado un espacio de tiempo, las sesiones fijas terminan y la red vuelve a mantener una carga de tráfico baja.

Para la carga de tráfico baja se ha establecido un tiempo medio de llegada entre dos sesiones $MSIA = 2{,}7s$ y un tiempo medio de llegada entre paquetes $MPIA = 0{,}3s$. Como nodo hotspot se ha seleccionado el nodo 4 (ver en la Figura 10 la topología de red) y el tiempo medio de llegada entre paquetes para las sesiones fijas establecidas con este nodo es $HOSPOT\_MPIA = 0{,}03s$. El nodo hotspot permanece activo desde el segundo 250 hasta el segundo 350.

La Figura 15 muestra los resultados obtenidos en el experimento. De nuevo, puede observarse que en ambos casos el nivel de throughput es muy parecido, no mostrando ningún tipo de degradación debido al envío de agentes comunicadores. Respecto a los tiempos de retraso, a pesar de no obtener una mejora tan significativa como en el escenario anterior, sí que podemos observar una mejora en Abira respecto de AntNet-FA, debido a que todos los nodos de la red están enviando datos al mismo nodo y al mismo tiempo, generando un estado de saturación.

Durante la primera parte de la simulación, en la que se da una situación estacionaria con una carga de tráfico baja, se observa cómo los paquetes son entregados sin problemas, manteniendo un tiempo de entrega también muy bajo. En el segundo 250, que es el momento en que todos los nodos establecen una sesión con el nodo 4 enviando tráfico de forma más intensa, se produce un incremento de tiempo en ambos algoritmos debido a la cantidad de tráfico que necesita ser redistribuido y que el nivel de congestión es generalizado. En este caso, es difícil que el algoritmo encuentre rutas más prometedoras si todas se congestionan al mismo tiempo. Sin embargo, dada una mayor agilidad de Abira a la hora de corregir las tablas y mejorar la convergencia, vemos que consigue un mejor

través de una red inalámbrica. En este caso utilizamos el término simplemente como "punto caliente".

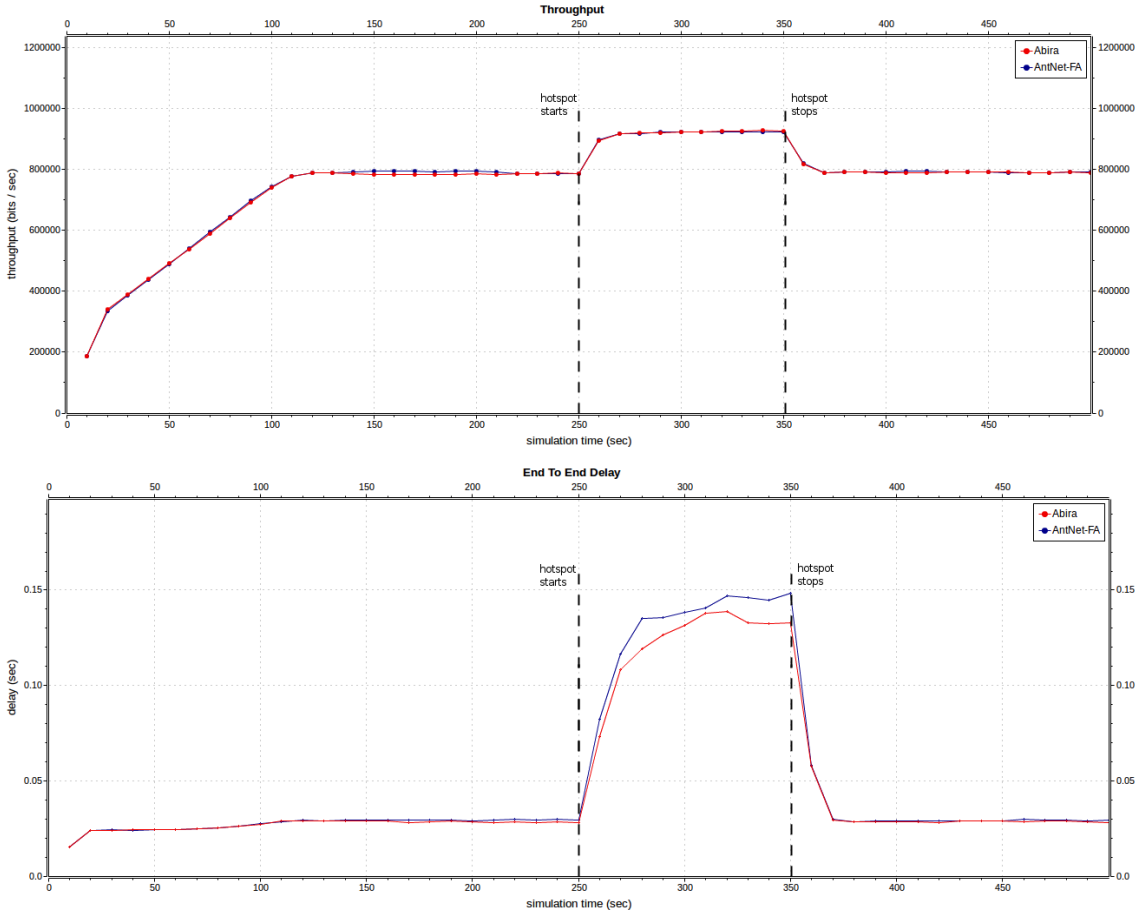


Figura 15: Comparativa de Abira y AntNet-FA en un escenario de cambio en los patrones de tráfico: throughput y tiempos de entrega. Parámetros: $MSIA = 4{,}7s$, $MPIA = 0{,}3s$, $HOSPOT\_MPIA = 0{,}03s$. Situación: el nodo 4 actúa como hotspot durante el intervalo $250 - 350s$, recibiendo nuevas sesiones de todos los nodos de la red.

rendimiento que el obtenido por AntNet-FA. Más tarde, cuando en el segundo 350 las sesiones fijas con el nodo 4 terminan, el tráfico vuelve a un estado estacionario con carga de tráfico baja, donde ambos algoritmos se comportan igual.

En la Figura 16 se muestran los resultados promedio de throughput y el percentil 90 de los tiempos de entrega. Al igual que en el caso anterior, el valor de throughput es similar y los tiempos de entrega siguen siendo mejores en el caso de Abira. De nuevo, hay que tener en cuenta que el tiempo de entrega comparado en esta figura representa el 90 % de las muestras, frente al 20 % que corresponde al tiempo que el hotspot ha estado activo congestionando la red y que es el momento donde Abira se comporta mejor.

La Figura 17 muestra el número de agentes comunicadores enviados durante la simulación. Se puede observar cómo durante el periodo inicial, donde la carga de tráfico es baja, el número de agentes que se envían es muy reducido. En el segundo 250, como consecuencia del cambio producido en el tráfico de red, podemos observar un ligero incremento del número de agentes generados. A pesar de que los tiempos han empeorado, la carga de tráfico es elevada y el nivel de congestión obliga a que las tablas de rutas

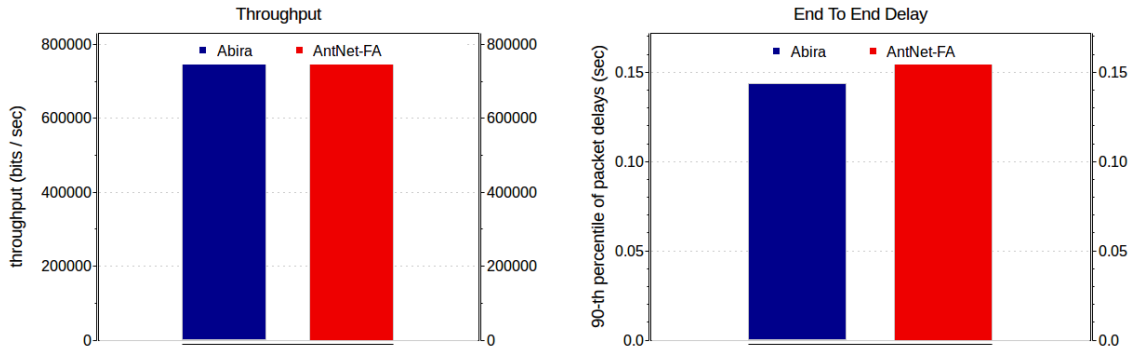


Figura 16: Comparativa de Abira y AntNet-FA en un escenario de fallo: (a) throughput, (b) tiempos de entrega - percentil 90. Parámetros: $MSIA = 4{,}7s$, $MPIA = 0{,}3s$. Situación: el nodo 21 falla durante el intervalo $200 - 300s$ y el nodo 40 falla durante el intervalo $250 - 350s$.

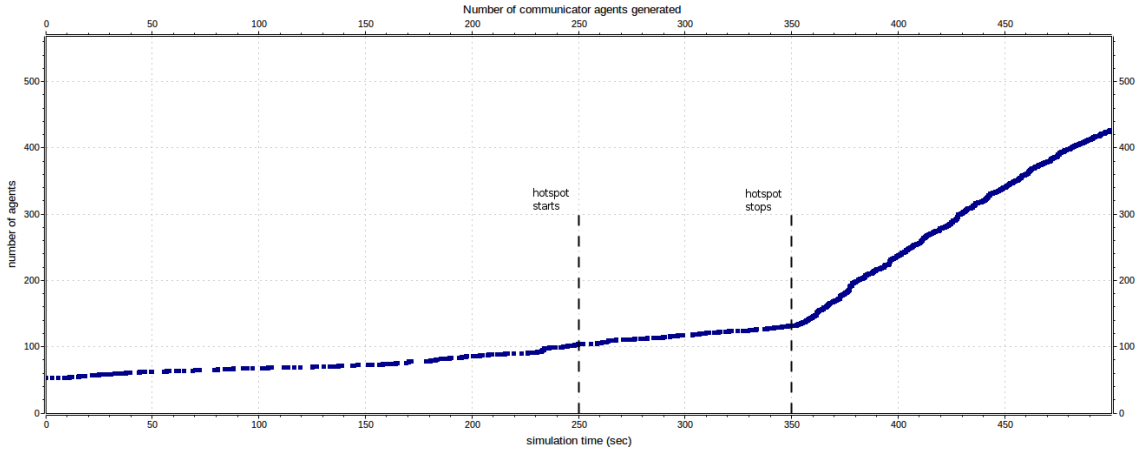


Figura 17: Agentes "communicator ant" enviados durante la simulación de cambio en los patrones de tráfico. El nodo 4 actúa como hotspot durante el intervalo $250 - 350s$, recibiendo nuevas sesiones de todos los nodos de la red.

se adapten constantemente a la nueva situación de la red encontrando nuevos mejores caminos. Más tarde, en el segundo 350, el nodo hotspot se desactiva y todas la sesiones establecidas con el resto de los nodos de la red terminan. Se produce una brusca caída de la congestión, por lo que el número de rutas más prometedoras encontradas es mucho mayor y el número de agentes comunicadores sufre un incremento significativo.

6.4.4. Problemas con la modificación propuesta de penalización

Tal y como describíamos en el punto 5.4, nuestra propuesta de penalización consiste en utilizar una metodología de recompensa-penalización, aplicando un refuerzo positivo en las entradas de feromonas de los enlaces en los que se haya obtenido un tiempo óptimo, o aplicando una penalización en las entradas de feromonas de los enlaces en los que se haya obtenido un resultado que esté fuera de un intervalo de tiempos cercanos al óptimo. Sin embargo, utilizar ambas acciones en un entorno dinámico como es una red de

telecomunicaciones, puede dar lugar a un sistema inestable. Es muy difícil establecer un criterio de actualización que permita construir una solución sin que se produzcan grandes variaciones en el modelo.

En los resultados de las simulaciones, hemos visto que los valores de feromonas sufrían grandes variaciones al aplicar las penalizaciones. A pesar de haber limitado las situaciones en las que aplicar una penalización mediante un nuevo intervalo de confianza, que discrimina los tiempos no óptimos que superan el límite I_{sup} de los tiempos que indican que existe un cambio en la red, la distancia que existe entre I_{sup} y los tiempos experimentados en estos casos es muy elevada. Dado que que el valor de la penalización a aplicar es proporcional a esta distancia, cuando la distancia es elevada, la cantidad feromona se ve decrementada intensamente, situándola en valores cercanos a τ_{min}. Al producirse grandes diferencias en los valores de feromonas, observamos que se producen picos en el rendimiento del algoritmo.

En la Figura 18 se muestra el resultado de aplicar la modificación de penalización en el escenario de fallo, donde dos nodos dejan de funcionar: el nodo 21 durante el intervalo de tiempo $200 - 300s$ y el nodo 40 durante el intervalo $250 - 350s$. Se trata de una comparativa de los tiempos de entrega entre AntNet-FA y Abira, además de un gráfico que muestra en qué momento y en qué cantidad se aplican las penalizaciones a lo largo de la simulación.

Puede observarse que durante todo el proceso de simulación se aplican algunas penalizaciones, obteniendo más o menos un resultado estable. Pero es durante el periodo de tiempo que corresponde a los nodos caídos, desde el segundo 200 hasta el segundo 350, cuando las penalizaciones se aplican en mayor número, lo que da lugar a una inestabilidad que produce picos en los tiempos de entrega, como puede apreciarse en el segundo 350, cuando ambos nodos se recuperan. Éste es el motivo por el que hemos desestimado el uso de la propuesta de penalización.

Si comparamos nuestra propuesta de penalización con la una propuesta presentada en [Lalbakhsh et al., 2010], la diferencia principal es el modo en que se aplican las penalizaciones. Mientras que en nuestro caso se aplican a modo de "corrección en línea", Lalbakhsh et al. solamente las aplican cuando se determina que existe un error en la red. Para ello, monitorizan el tráfico utilizando un mecanismo al que denominan *Occurrence Detection*. Una vez detectado el error, aplican la penalización y reinician el modelo estadístico $\mathcal{M}^k$ de AntNet. Al no coexistir penalizaciones y refuerzos durante el proceso continuo de actualización de los valores de feromonas, no sufren este tipo de inestabilidad.

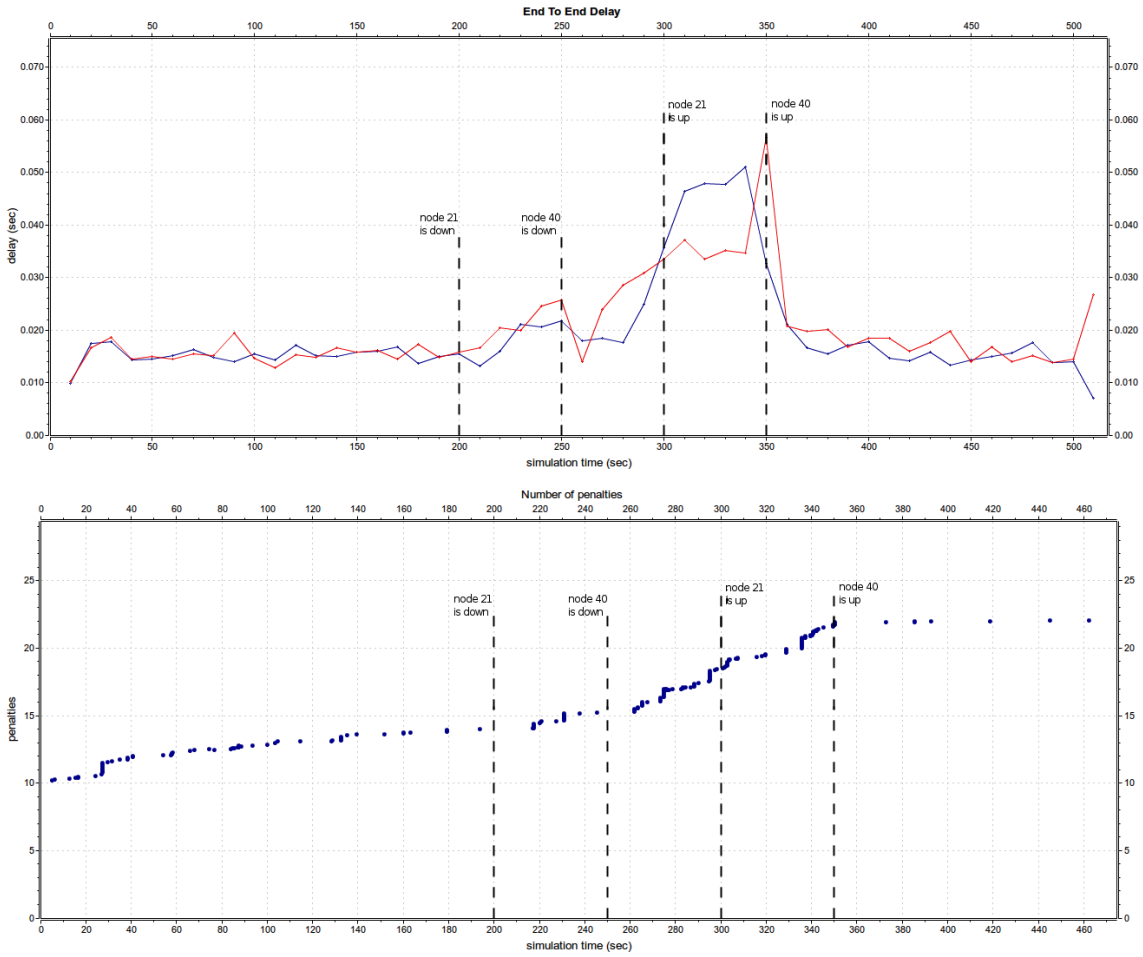


Figura 18: (a) Comparativa de tiempos de entrega entre Abira con penalización y AntNet-FA en un escenario de fallo. (b) Número de penalizaciones aplicadas a lo largo de la simulación de Abira. Parámetros: $MSIA = 4{,}7s$, $MPIA = 0{,}3s$. Situación: el nodo 21 falla durante el intervalo $200 - 300s$ y el nodo 40 falla durante el intervalo $250 - 350s$.

Parte IV

Conclusiones

7. Conclusiones y trabajo futuro

La disciplina de *swarm intelligence* (SI) poco a poco ha ido ganando terreno en muchos ámbitos, entre ellos las redes de telecomunicación, donde los algoritmos SI se posicionan como una posible alternativa a los algoritmos de enrutamiento convencionales. Gracias a la interacción de agentes muy simples y utilizando reglas sencillas, consiguen un comportamiento colectivo que resuelve el problema de enrutamiento, adaptándose mejor a los cambios, sin necesidad de un ente central, de forma escalable y con una alta capacidad de tolerancia a fallos. Estas son características que encajan perfectamente con la filosofía de este tipo de redes.

A lo largo de este trabajo, hemos realizado un análisis de los principales algoritmos inspirados en diferentes colonias de insectos, como las hormigas o las abejas. Las primeros se inspiran en cómo las colonias de hormigas descubren las rutas más cortas entre el nido y las fuentes de comida mediante el mecanismo de estigmergia, un comportamiento observado y modelado que da lugar a la meta-heurística ACO, la base de este tipo de algoritmos. Los segundos, en cambio, se inspiran en la comunicación de las abejas dentro del enjambre para indicar la distancia y la calidad de las fuentes de comida mediante el mecanismo de *waggle dance*. Dada la cantidad de propuestas que existen, nos hemos centrado en los algoritmos de enrutamiento para redes cableadas, no orientadas a conexión y con una estrategia de entrega de mejor esfuerzo. Hemos descrito el diseño general y los principios básicos de cada uno de ellos. A pesar de que el diseño general varía, principalmente en la heurística utilizada, la mayoría mantiene las mismas características, como son la adaptación de las tablas de rutas al estado de la red (en base a los tiempos de viaje y la congestión), o el enrutamiento multi-trayecto, que distribuye el tráfico entre las diferentes rutas de forma estocástica, maximizando así el ancho de banda utilizado. En todas las comparativas realizadas entre algoritmos SI y algoritmos convencionales, hemos visto que los primeros suelen tener un rendimiento igual o superior, algo que confirma que este nuevo enfoque satisface las necesidades de las redes de telecomunicación.

Hemos presentado Abira, un algoritmo de enrutamiento basado en AntNet, el primer y principal algoritmo de redes de conmutación de paquetes inspirado en ACO. Se trata de un algoritmo para redes cableadas con entrega de mejor esfuerzo, que pretende mejorar el rendimiento y la convergencia aplicando técnicas como la exploración o la penalización (negative feedback) y una técnica a la que hemos denominado “agentes comunicadores”, que consiste en compartir las rutas óptimas con los nodos vecinos. Para validar estas propuestas, se ha implementado un entorno de simulación con los algoritmos AntNet-FA y Abira, utilizado el framework OMNet++. Una vez obtenidos los resultados, hemos comprobado que la propuesta de penalización planteada en Abira no es estable, por lo que ha sido descartada. Tras realizar varias simulaciones incluyendo las dos primeras

modificaciones, la de exploración y la de agentes comunicadores, hemos comprobado que el tiempo que necesita el nuevo algoritmo para adaptarse a los cambios de la red es menor que el algoritmo original, lo que permite obtener un rendimiento superior. Para finalizar, hemos realizado un análisis de por qué la propuesta de penalización no funciona correctamente, llegando a la conclusión de que variaciones muy amplias en los valores de feromonas hacen el modelo inestable.

Posibles mejoras del algoritmo propuesto Existen varias mejoras que quedan pendientes para un trabajo futuro. El primero, resolver el problema encontrado al aplicar la modificación de penalización. Es posible que estableciendo ciertos límites sobre el factor de penalización o utilizando unos parámetros diferentes, pueda alcanzarse una solución estable. Al mismo tiempo, sería interesante utilizar una algoritmo PSO para optimizar y afinar los parámetros que se utilizan en el modelo, tanto los de Abira, como los de AntNet. Por último, también sería posible introducir un mecanismo que permitiera adaptar de forma dinámica el número de hormigas generadas, forward/backward o comunicadoras, a la situación de tráfico en la red.

Futuro de los algoritmos SI como alternativa a los algoritmos convencionales A pesar de las ventajas demostradas en los algoritmos SI, son pocas las implementaciones llevadas a cabo más allá de los entornos de simulación, ya que en la mayoría de los casos se tratan de pruebas de concepto. En el ámbito de las redes cableadas, solamente AntNet y BeeHive han sido implementados en entornos reales a pequeña escala [Verstraete et al., 2006; Farooq, 2009]. El hecho de que este tipo de algoritmos no transciendan más allá de lo teórico, es debido a que no es posible predecir su comportamiento en diferentes situaciones, además de ser altamente sensibles a la hora de afinar cada uno de sus parámetros. Este problema es derivado de la propia naturaleza estocástica del algoritmo, donde el resultado surge del comportamiento colectivo y varía en función del estado de la red. A la hora de establecer los valores de los parámetros no se sigue un criterio formal, sino que se establecen de manera experimental, por lo que se desconoce en todo momento si el rendimiento del algoritmo es el óptimo, pudiendo variar el resultado en función de la situación en la que haya sido realizado cada experimento.

A a los requerimientos de fiabilidad, se unen otras razones como la falta de una infraestructura adecuada para este tipo de sistemas o un proceso lento de modernización de las redes, en gran parte debido a la necesidad de mantener una compatibilidad con los sistemas convencionales. Quizás en otro tipo de redes más modernas, donde todavía existe un campo de investigación creciente y donde se necesitan enfoques novedosos de enrutamiento, como pueden ser las redes MANET (mobile ad hoc network), es donde este nuevo tipo de algoritmos tienen más posibilidades de aplicación.

Es evidente que los algoritmos inspirados en sociedades de insectos encajan perfectamente con los requisitos de las redes de telecomunicación. Si se consiguieran resolver los problemas de fiabilidad, estos algoritmos podrían sustituir o complementar perfectamente a los algoritmos convencionales.

Con vistas al futuro, y llegado el momento en el que este tipo de planteamientos sean utilizados en sistemas reales, será necesario definir un amplio marco seguridad. Las redes son un entorno sensible dado que su objetivo es el envío de información, lo que las convierte en un entorno hostil. Los algoritmos propuestos hasta la fecha no contemplan en ningún caso cómo protegerse ante situaciones de riesgo, como verificar si la información de los agentes procede de una fuente segura o si la información que portan no ha sido alterada, por lo que será necesario introducir mecanismos que eviten este tipo de vulnerabilidades.

Referencias

Appleby, S. and Steward, S. (1994). Mobile software agents for control in telecommunications networks. *BT Technology Journal*, 12(2):104–113.

Baran, B. and Sosa, R. (2000). A new approach for antnet routing. In *Computer Communications and Networks, 2000. Proceedings. Ninth International Conference on*, pages 303–308. IEEE.

Beckers, R., Deneubourg, J.-L., and Goss, S. (1992). Trails and u-turns in the selection of a path by the ant lasius niger. *Journal of theoretical biology*, 159(4):397–415.

Beni, G. and Wang, J. (1989). Swarm intelligence in cellular robotic systems. In *Proceedings of NATO Advanced Workshop on Robots and Biological Systems*, pages 26–30.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford.

Bonabeau, E., Hénaux, F., Guérin, S., Snyers, D., Kuntz, P., and Theraulaz, G. (1998). Routing in telecommunications networks with "smart.ªnt-like agents. *Intelligent Agents for Telecommunication Applications*, 1437:60–71.

Carrillo, L., Marzo, J. L., Fàbrega, L., Vilà, P., and Guadall, C. (2004a). Ant colony behaviour as routing mechanism to provide quality of service. In *Ant Colony Optimization and Swarm Intelligence*, pages 418–419. Springer.

Carrillo, L., Marzo, J. L., Vilá, P., and Mantilla, C. (2004b). Mants-hoc: A multi-agent ant-based system for routing in mobile ad hoc networks. In *Proceedings of Sete Congrés Catala d'Intelligencia Artificial (CCIA'2004)*, pages 285–292.

Çelik, F., Zengin, A., Tuncel, S., and Çobanoğlu, B. (2010). A survey on swarm intelligence based routing protocols in wireless sensor networks. *International Journal of the Physical Sciences*, 5:2118–2126.

Dhillon, S. and Mieghem, P. V. (2007). Performance analysis of the antnet algorithm. *Computer Networks*, 51(8):2104 – 2125.

Di Caro, G. (2004). *Ant colony optimization and its application to adaptive routing in telecommunication networks*. PhD thesis, Université Libre de Bruxelles.

Di Caro, G. and Dorigo, M. (1998a). Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9(1):317–365.

Di Caro, G. and Dorigo, M. (1998b). Two ant colony algorithms for best-effort routing in datagram networks. In *Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98)*, pages 541–546.

Di Caro, G., Ducatelle, F., and Gambardella, L. (2005). Anthocnet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications*, 16(5):443–455.

Di Caro, G. and Vasilakos, T. (2000). Ant-sela: Ant-agents and stochastic automata learn adaptive routing tables for qos routing in atm networks. In *ANTS'2000 - Ant Colonies to Artificial Ants: Second International Workshop on Ant Colony Optimization*.

Doi, S. and Yamamura, M. (2004). An experimental analysis of loop-free algorithms for scale-free networks. In Dorigo, M., Birattari, M., Blum, C., Gambardella, L., Mondada, F., and Stützle, T., editors, *Ant Colony Optimization and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 278–285. Springer Berlin Heidelberg.

Dorigo, M. (1992). *Optimization, learning and natural algorithms*. PhD thesis, Politecnico di Milano, Italy.

Dorigo, M., Bonabeau, E., and Theraulaz, G. (2000). Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871.

Dorigo, M. and Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2. IEEE.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. A Bradford book. University Press Group Limited.

Ducatelle, F., Caro, G., and Gambardella, L. (2006). An analysis of the different components of the anthocnet routing algorithm. In Dorigo, M., Gambardella, L., Birattari, M., Martinoli, A., Poli, R., and StÃŒtzle, T., editors, *Ant Colony Optimization and Swarm Intelligence*, volume 4150 of *Lecture Notes in Computer Science*, pages 37–48. Springer Berlin Heidelberg.

Farooq, M. (2009). *Bee-Inspired Protocol Engineering: From Nature to Networks*. Springer Berlin Heidelberg, 1 edition.

Farooq, M. and Caro, G. A. D. (2008). Routing protocols for next generation networks inspired by collective behaviors of insect societies: An overview. In *Swarm Intelligence*, pages 101–160. Springer.

Flikkema, P. and Leid, J. (2005). Bacterial communities: a microbiological model for swarm intelligence. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 416–419.

Gadomska, M. and Pacut, A. (2007a). Performance of ant routing algorithms when using tcp. *Applications of Evolutionary Computing*, 4448:1–10.

Gadomska, M. and Pacut, A. (2007b). Recent progress in ant algorithms for fixed telecommunication networks: A review. *Evolutionary Computation and Global Optimization*, pages 79–89.

Garcia-Serrano, A. and Ossowski, S. (1998). Inteligencia artificial distribuida y sistemas multiagente. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 2(6):12–23.

Ghazy, A. M., EL-Licy, F., and Hefny, H. A. (2012). Threshold based antnet algorithm for dynamic traffic routing of road networks. *Egyptian Informatics Journal*, 13(2):111–121.

Gibbens, R. J., Kelly, F. P., and Key, P. B. (1988). Dynamic alternative routing-modelling and behaviour. In *Proceedings of the 12th International Teletraffic Congress*, pages 1019–1025. Elsevier Science Publishers BV (North-Holland)(June 1988).

Goss, S., Beckers, R., Deneubourg, J.-L., Aron, S., Pasteels, J., and Hughes, R. N. (1990). How trail laying and trail following can solve foraging problems for ant colonies. *Behavioural Mechanisms of Food Selection*, pages 661–678.

Gunes, M., Sorges, U., and Bouazizi, I. (2002). Ara-the ant-colony based routing algorithm for manets. In *Parallel Processing Workshops, 2002. Proceedings. International Conference on*, pages 79–85.

Heegaard, P. E. and Wittner, O. J. (2010). Overhead reduction in a distributed path management system. *Computer Networks*, 54(6):1019–1041.

Jain, S., Kokate, S., Thakur, P., and Takalkar, S. (2012). A study of congestion aware adaptive routing protocols in manet. *Computer Engineering and Intelligent Systems*, 3(4):64.

Johnson, D. B., Maltz, D. A., and Broch, J. (2001). Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Ad Hoc Networking*, pages 139–172.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Techn. Rep. TR06, Erciyes Univ. Press, Erciyes*.

Kennedy, J. F., Kennedy, J., and Eberhart, R. C. (2001). *Swarm intelligence*. Morgan Kaufmann Pub.

Lalbakhsh, P., Zaeri, B., Lalbakhsh, A., and Fesharaki, M. (2010). Antnet with reward-penalty reinforcement learning. In *Computational Intelligence, Communication Systems and Networks (CICSyN), 2010 Second International Conference on*, pages 17–21.

Liang, S., Zincir-Heywood, A., and Heywood, M. (2006). Adding more intelligence to the network routing problem: Antnet and ga-agents. *Applied Soft Computing*, 6(3):244 – 257.

Ludwig, S. A. and Moallem, A. (2011). Swarm intelligence approaches for grid load balancing. *Journal of Grid Computing*, 9:279–301.

Lü, Y., Zhao, G.-z., Su, F.-j., and Li, X.-r. (2004). Adaptive swarm-based routing in communication networks. *Journal of Zhejiang University Science*, 5(7):867–872.

Mullen, R., Monekosso, D., Barman, S., and Remagnino, P. (2009). A review of ant algorithms. *Expert Systems with Applications*, 36(6):9608–9617.

Pan, J. and Jain, R. (2008). A survey of network simulation tools: Current status and future developments.

Scheidler, A., Merkle, D., and Middendorf, M. (2008). Congestion control in ant like moving agent systems. *The International Federation for Information Processing*, 268:33–43.

Schoonderwoerd, R., Holland, O., Bruten, J., and Rothkrantz, L. (1997). Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169–207.

Shahzad, M., Zahid, S., and Farooq, M. (2008). A scalable formal framework for analyzing the behavior of nature-inspired routing protocols. In *Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X*, pages 1130–1139, Berlin, Heidelberg. Springer-Verlag.

Sim, K. M. and Sun, W. H. (2002). Multiple ant-colony optimization for network routing. In *Cyber Worlds, 2002. Proceedings. First International Symposium on*, pages 277–281. IEEE.

Sim, K. M. and Sun, W. H. (2003). Ant colony optimization for routing and load-balancing: survey and new directions. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 33(5):560–572.

Singh, G., Kumar, N., and Verma, A. K. (2012). Ant colony algorithms in manets: A review. *Journal of Network and Computer Applications*, 35(6):1964–1972.

Stallings, W. (2006). *Comunicaciones y redes de computadores*. 7 edition.

Stützle, T., Dorigo, M., et al. (2002). A short convergence proof for a class of ant colony optimization algorithms. *IEEE Trans. Evolutionary Computation*, 6(4):358–365.

Subramanian, D., Druschel, P., and Chen, J. (1997). Ants and reinforcement learning: A case study in routing in dynamic networks. In *International Joint Conference on Artificial Intelligence*, volume 15, pages 832–839.

Sutton, R. S. (1991). Reinforcement learning architectures for animats. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 288–296. MIT Press.

Tekiner, F. and Ghassemlooy, Z. (2005). Improved antnet routing algorithm for packet switching. *Mediterranean Journal of Computer Networks*, pages 69–76.

Varga, A. (2011). *OMNeT++ User Manual*.

Verstraete, V., Strobbe, M., Van Breusegem, E., Coppens, J., Pickavet, M., and Demeester, P. (2006). Antnet: Aco routing algorithm in practice.

Wang, J. and Lee, S. (2009). A performance comparison of swarm intelligence inspired routing algorithms for manets. In *Computational Science and Its Applications–ICCSA 2009*, volume 5593 of *ICCSA '09*, pages 432–442. Springer, Berlin, Heidelberg.

Wang, J., Osagie, E., Thulasiraman, P., and Thulasiram, R. K. (2009). Hopnet: A hybrid ant colony optimization routing algorithm for mobile ad hoc network. *Ad Hoc Netw.*, 7(4):690–705.

Wedde, H. F. and Farooq, M. (2006). A comprehensive review of nature inspired routing algorithms for fixed telecommunication networks. *Journal of Systems Architecture*, 52(8–9):461–484. Nature-Inspired Applications and Systems.

Wedde, H. F., Farooq, M., Pannenbaecker, T., Vogel, B., Mueller, C., Meth, J., and Jeruschkat, R. (2005). Beeadhoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 153–160. ACM.

Wedde, H. F., Farooq, M., and Zhang, Y. (2004). Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. *Ant Colony Optimization and Swarm Intelligence*, 3172:83–94.

Weingartner, E., Vom Lehn, H., and Wehrle, K. (2009). A performance comparison of recent network simulators. In *Communications, 2009. ICC'09. IEEE International Conference on*, pages 1–5. IEEE.

Zhong, W. and Evans, D. (2002). When ants attack: Security issues for stigmergic systems. Master's thesis, University of Virginia.