Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

Auto-Organización de Redes de Dispositivos Móviles

Tesis que presenta:

J. Guadalupe Olascuaga Cabrera

para obtener el grado de:

Maestro en Ciencias

en la especialidad de:

Ingeniería Eléctrica

Directores de Tesis

Dr. Luis Ernesto López Mellado
Dr. Félix Francisco Ramos Corchado

Guadalajara, Jalisco, Agosto de 2009.

Auto-Organización de Redes de Dispositivos Móviles

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
J. Guadalupe Olascuaga Cabrera
Ingeniero en Sistemas Computacionales
Instituto Tecnológico de la Costa Grande 2002-2007

Becario de conacyt, expediente no. 212765

Directores de Tesis
Dr. Luis Ernesto López Mellado
Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, Agosto de 2009.

Dedicatoria

Dedico esta tesis a:
mi padre Balbino Olascuaga Castro †
mi madre Mariana Cabrera Bravo
mis hermanos Adalberto, Cristian, Jacob
mis hermanas Balby Ruby, Esmeralda
por todo el apoyo que me brindaron.

Agradecimientos

A toda mi familia y amigos por el apoyo que me brindaron.

A toda la gente que preguntaba: ¿Tu que haces? ¿Ya trabajas? ¿A qué te dedicas? ¿Cuando terminas?

A mis asesores por guiarme en el desarrollo de esta tesis.

A Maryna y Anita por sus correciones.

Al CINVESTAV por darme la oportunidad de estudiar.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT).

Abstract

This paper presents a cluster-based self-organizing algorithm for creating automatically ad-hoc networks including diverse wireless devices such as PDA, cell phones, laptops, wireless sensors, etc. In the proposed strategy each mobile device is represented by a multi-role agent, which generates a backbone among the devices within the environment based only on local interactions. During the operation each agent may act as leader, gateway or member according to both its neighborhood and its residual energy for creating and mantaining connected the network; this role switching allows the backbone reconfiguration when the nodes leave or arrive to the network yielding emergent behavior. After the network formation every agent varies the time interval and power of transmission allowing energy saving. Performance analysis via simulation is presented.

Resumen

Esta tesis presenta un algoritmo de auto-organización basado en grupos para crear redes tipo ad-hoc automáticamente incluyendo diversos dispositivos inalámbricos tal como PDA, celulares, laptops, consolas, sensores inalámbricos, etc. En la estrategia propuesta cada dispositivo móvil es representado por un agente multi-rol , el cual genera un backbone entre los dispositivos dentro del ambiente basado sólo con interacciones locales. Durante la operación de agrupación cada agente puede actuar como *líder, conexión* o *miembro* de acuerdo a su vecindario y la energía residual para crear y mantener en todo momento la red conectada; El cambio de rol permite la re-configuración del backbone cuando los agentes salen ó llegan a la red permitiendo un comportamiento emergente. Después de la formación de red cada agente varía el intervalo de tiempo y potencia de transmisión permitiendo un ahorro de energía. Es presentado el análisis de desempeño vía simulación con el simulador NS-2.

Índice general

Índice de figuras

Introducción

Las aplicaciones de redes inalámbricas han incrementado en los últimos años y consecuentemente han recibido la atención de la comunidad investigadora, especialmente redes ad-hoc en las cuales todos los dispositivos tienen un comportamiento similar, estos dispositivos transmiten mensajes comportándose como routers para comunicar con nodos vecinos.

Una *red inalámbrica ad-hoc* es una red inalámbrica descentralizada, la red es *ad-hoc* porque cada nodo puede retransmitir mensajes entre los dispositivos que la conforman.

Una red móvil ad hoc (*MANET*) es un tipo de red inalámbrica ad-hoc, la *MANET* es una red auto-configurada de dispositivos móviles conectados por enlaces inalámbricos, cada dispositivo en una *MANET* puede moverse libremente en cualquier dirección, y por lo tanto sus enlaces cambiarán conforme pasa el tiempo, cada dispositivo debe tener la capacidad de almacenar información sobre los vecinos y retransmitir mensajes entre los dispositivos de la red.

Hay muchas aplicaciones en redes tipo ad-hoc, por ejemplo, colocación de sensores inalámbricos para monitoreo en campos de agricultura, colocación de dispositivos en campo de batallas para predecir los movimientos de tropas.

El objetivo de este tipo de redes es mantener los nodos conectados durante la llegada o salida de dispositivos en la red, así como minimizar el consumo de energía.

La naturaleza de estas redes hace difícil la configuración, ya que el movimiento de los dispositivos es dinámico; así, la topología de red es constantemente cambiante. Esto motiva a investigar métodos efectivos para crear y mantener las redes inalambricas.

Para este propósito has sido usadas técnicas de auto-organización inspiradas en la naturaleza, es decir, inspirados en el comportamiento de bandadas de aves, escuela de peces y colonia de hormigas. Las entidades participantes establecen una estructura organizacional que no requiere coordinación central, las entidades interactúan directamente entre ellas y contínuamente reaccionan a cambios en su ambiente local. Los procedimientos que implementan este comportamiento deben satisfacer muchas propiedades tal como: *comportamiento emergente*, *escalabilidad*, *adaptabilidad* y *robustez contra fallas*.

El *comportamiento emergente* es comúnmente definido como aquél que no se atribuye

a algún agente individual, más bien como un resultado global de la coordinación de los dispositivos. Si el comportamiento es predecible y explicable, entonces no será tratado como comportamiento emergente. Incluso la emergencia es definida como la acción de simples reglas producen resultados complejos, es decir, las reglas aplicadas a los individuos pueden ser muy simples, pero el comportamiento del grupo resulta ser complejo e impredecible. Existen experimentos en la literatura que demuestran esta situación.

Muchos trabajos pueden ser encontrados en la literatura; estos serán revisados en el siguiente capitulo. La mayoría de estos trabajos aplican estrategias de auto-organización en redes con nodos estáticos; otras aproximaciones tratan con nodos móviles pero asumen que los nodos tienen las mismas restricciones tal como rangos de transmisión. Los trabajos que consideran rangos de transmisión para agrupación no usan técnicas de auto-organización.

En este trabajo presentamos un algoritmo para crear y mantener automáticamente redes ad-hoc. Cada nodo es presentado por un agente multi-rol, el cual genera un backbone entre los dispositivos dentro del ambiente basado únicamente en interacciones locales. El procedimiento permite la re-configuración de backbone cuando los nodos salen o llegan a la red causando un comportamiento emergente. Después que la red es formada cada agente varía el intervalo de tiempo en transmisión así como varía la potencia de transmisión de los dispositivos permitiendo un ahorro en la energía.

Esta tesis está organizada como sigue: El capítulo 1 presenta una revisión de las estrategias de agrupación en redes de dispositivos móviles, el capítulo 2 describe el algoritmo de auto-organización propuesto, el capítulo 3 muestra el desempeño del algoritmo en diferentes escenarios, el capítulo 4 presenta la inclusión del algoritmo y configuración en el simulador NS-2, y por último en el capítulo 5 se presentan las conclusiones y trabajo futuro.

Capítulo 1

Estrategias de configuración en dispositivos móviles

Resumen: En este capítulo se hace una revisión de las técnicas de agrupación y configuración en dispositivos móviles, también se presenta una breve introducción e investigaciones más recientes en esta área, desde los principios y paradigmas de diseño hasta aplicaciones en redes inalámbricas móviles ad-hoc.

1.1. Introducción

La auto-organización describe un fenómeno de la naturaleza. Un ejemplo de comportamiento auto-organizado es la organización de peces (*figura 1.1*), en este ejemplo, las entidades participantes establecen una estructura organizacional que no requiere coordinación central, las entidades interactúan directamente entre ellas y contínuamente reaccionan a cambios en su ambiente local.

En este trabajo de investigación nos centraremos en el área de comunicaciones y redes de computadoras, el objetivo es utilizar el principio de auto-organización para configurar una red ad-hoc de manera que sea adaptable, escalable, tolerante a fallas y además que haga un uso eficiente de la energía sin perder conectividad.

En general usaremos las siguientes difiniciones:

Definición 1.1 (Auto-Organización). Es la emergencia de grandes sistemas con estructura adaptiva y funcionalidad con simples interacciones locales entre entidades individuales [BPEL05].

Definición 1.2 (Auto-Organización). Es un proceso en el cual un patrón en el nivel global de un sistema emerge solamente de numerosas interacciones locales entre los componentes a bajo nivel del sistema [Dre06].

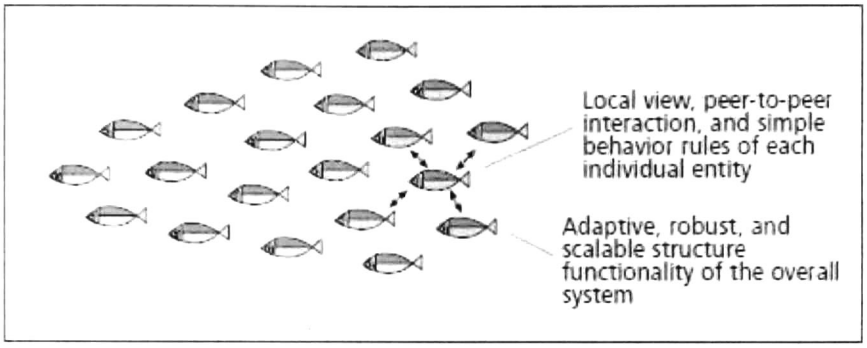

Figura 1.1: Auto-organización de peces

Definición 1.3 (Sistema Multi-agente). Es un conjunto de agentes situados en un ambiente común, interactuando e intentando alcanzar un objetivo [JO07].

Definición 1.4 (Comportamiento emergente). Es aquél que no se puede predecir a través de un nivel más simple que el sistema en conjunto.

1.2. Principios y paradigmas de diseño

Un sistema es organizado si tiene cierta *estructura* y *funcionalidad*. La *estructura* se refiere a que las entidades están organizadas de una manera particular que regula su interacción. *Funcionalidad* significa que el sistema en conjunto cumple un cierto objetivo.

Para diseñar un sistema que sea auto-organizado, los autores en [BPEL05] proponen cuatro paradigmas: 1) Diseño local de comportamiento, es decir, que las reglas locales generen propiedades globales; 2) Explotar la coordinación implícita, significa que los nodos detecten y analizen los paquetes que transmiten sus vecinos; 3) Minimizar el tiempo de estado de información y 4) Diseñar el protocolo que se adapte a los cambios del ambiente.

Cuando se desarrolla una estructura de red de dispositivos móviles, ésta debería tener un alto grado de auto-organización. Sería ideal tener una metodología que describa este desarrollo paso a paso, sin embargo, claramente es difícil definir tal proceso en detalle para funciones de red auto-organizadas. En la *figura 1.2* se hace un primer intento de tal proceso. Esta figura integra los 4 paradigmas de diseño y los aplica interactivamente para obtener un protocolo que implementa la función de auto-organización deseada.

En [Mil07] son presentadas algunas investigaciones actuales sobre auto-organización así como algunos modelos de auto-organización en redes de sensores inalámbricos basados en modelos biológicos, sociales, económicos, etc.

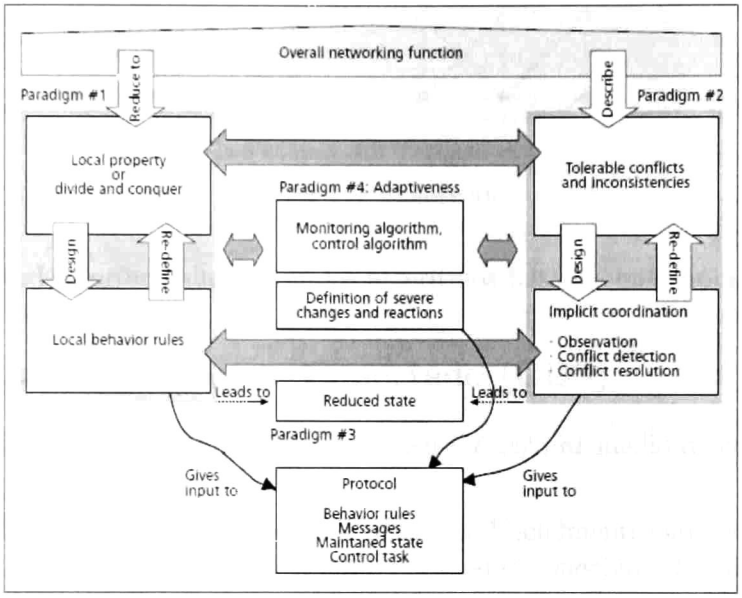

Figura 1.2: Paradigmas de diseño

Otros trabajos dividen en diferentes categorías y propiedades sobre redes ad-hoc y cómo se aplica la auto-organización [Dre06], así como la forma en que se realiza la clasificación de los métodos. En [HG03] la auto-organización está basada en interacciones dinámicas y definen la habilidad de una aplicación para capturar y explotar las características de las interacciones entre un grupo de usuarios móviles. En [Her07] se muestra un caso de estudio sobre colocación de replicas de información auto-organizadas en diferentes servidores estáticos. En este trabajo, el objetivo es distribuir las replicas de información para que el usuario tenga acceso de una forma rápida.

En [YMM06] es presentado un survey sobre la administración de la arquitectura en red de sensores inalámbricos (WSN). De la misma manera, también son discutidos algunos problemas de diseño y requerimientos para la administración de una arquitectura más eficiente.

1.3. Métodos en auto-organización

La mayoría de los algoritmos de auto-organización pueden ser clasificados en 3 principales categorías: Conjunto dominante conectado (CDS) [CLL04], multipoint relay (MPR) y basados en grupos. Sin embargo existen otros métodos para hacer auto-organización, por ejemplo los algoritmos de *árbol de expansión* [JK06]. En este trabajo nos enfocaremos solamente en los

Figura 1.3: Coloreado

tres primeros métodos, principalmente en los algoritmos basados en grupos.

1.3.1. Conjunto Dominante Conectado (CDS)

Los conjuntos dominantes conectados se pueden formar por medio de clusterheads y por nodos gateways.

El algoritmo CDS consiste de dos fases de marcado. Inicialmente cada vértice es marcado con F para indicar que no pertenece al conjunto dominante conectado. En la primera fase, un vértice se marca a sí mismo como T si cualquiera de dos de sus vecinos no están directamente conectados (no existe un enlace directo entre ellos); En esta fase son marcados todos los vértices que pueden ser potencialmente incluidos en un conjunto dominante conectado. En la segunda fase, un vértice v marcado con T cambia su marca a F si cualquiera de las siguientes condiciones se cumple:

1. $\exists u \in N(v)$ el cual es marcado T tal que $N[v] \subseteq N[u]$ y $id(v) < id(u)$; donde $N(v)$ es el vecindario de v.

2. $\exists u, w \in N(v)$ lo cual son ambos marcados con T donde $N[v] \subseteq N[u] \cup N[w]$ y $id(v) < min(id(u), id(w))$.

En la *figura 1.3* se ilustran las condiciones 1 y 2 respectivamente. Los vértices T son coloreados con negro y los vértices F son coloreados de blanco.

La complejidad computacional del proceso de encontrar el conjunto conectado mínimo es un problema NP-duro [DGH$^+$].

En [CLL04] los autores muestran varios algoritmos basados en CDS (*figura 1.4*). También muestran el uso de la estructura del grupo formado.

En [AJK06] se propone una forma de crear una infraestructura virtual para redes de sensores inalámbricos con CDS. Primero usan un algoritmo de agrupación basándose en el costo de comunicación; el algoritmo hace una partición de la red creando gateways. Estos gateways descubrirán los nodos que se localizan dentro de su rango de transmisión, un gateway es un nodo que comunica varios grupos, y después que los grupos son creados se usa un

Figura 1.4: Estructuración con la estrategia CDS

algoritmo CDS, cada nodo intercambia información con sus vecinos y cada uno conocerá todos los vecinos a dos-saltos. En este trabajo se hace uso eficiente de la energía, sin embargo, no usa rangos de transmisión variable, además no toma en cuenta la movilidad de los nodos, es decir, que no hacen reestructuración.

En [HV07] se definen tres principales fases en tiempo de vida de la batería por nodo: *nacimiento*, *vida de trabajo* y *muerte*, el objetivo es analizar el comportamiento del CDS y otras estrategias en cada fase; Durante la fase de *nacimiento*, los nodos llegan progresivamente durante la configuración inicial o cuando más nodos son agregados, la fase de *vida de trabajo* empieza tan pronto como la estructura organizada se estabiliza, la tercera y última fase es llamada *muerte*, y ésta se da cuando uno o muchos nodos desaparecen y es necesario el proceso de reorganización. También se estudia el comportamiento de diferentes métodos de auto-organización en cada fase. Cuatro protocolos son estudiados (MPR, MPR-DS, CDS-rule k, CDS-MIS), este trabajo solo hace la comparación entre los métodos, pero al igual que en [AJK06] tampoco utilizan reagrupación y rangos de transmisión variables.

1.3.2. Multipoint Relay (MPR)

El objetivo de MPR es reducir la inundación de paquetes broadcast en una red mediante la reducción de las retransmisiones locales; cada nodo envía un mensaje *hello* periódicamente

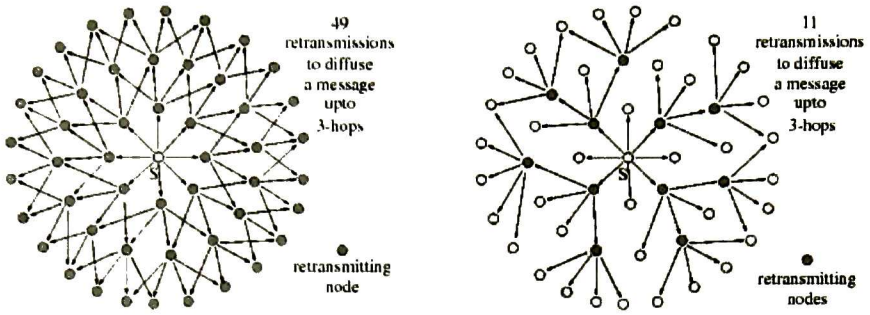

Figura 1.5: Estructuración con el algoritmo MPR

a todos sus vecinos a un-salto, un nodo obtiene conocimiento de sus vecinos a uno y dos-saltos después de recibir los mensajes *hello*, los vecinos de dos-saltos se obtienen por medio de vecinos de un-salto (existen variaciones en el algoritmo, en la *figura 1.5* se muestra un MPR a tres-saltos). Basados en esta información cada nodo u selecciona algunos nodos para retransmitir paquetes en su vecindario a dos-saltos, los nodos seleccionados son llamados Multipoint relays (MPRs).

Algunos investigadores han mezclado estrategias para aprovechar las ventajas que cada una de ellas posee y de este modo obtener mejores resultados. En [LSM06] los autores presentan un algoritmo broadcast basado en MPR para producir CDS de tamaño pequeño en la red, los nodos pueden estar en uno de cada cuatro estados dominantes: *dominator, dominate, connector* y *white-node*. El algoritmo trabaja en dos fases; en la primera es construido un conjunto independiente máximo (MIS) en la red, de este modo los nodos en MIS son llamados *dominators*. Los nodos *dominators* forman el conjunto dominante (DS) y en la segunda fase toman el rol de *gateways*. Después calculan un conjunto MPR para cubrir las vecinos a dos-saltos. Finalmente se usa un método de podado para eliminar gateways redundantes en CDS. Los resultados de simulación muestran un CDS más pequeño que otros métodos, pero los autores no consideran el mantenimiento de un CDS en ambientes móviles y los nodos no pueden variar la potencia de transmisión.

Desafortunadamente encontrar un conjunto MultiPoint Relay de tamaño pequeño está probado que es un problema NP-duro [QVL02] al igual que CDS.

1.3.3. Basados en Agrupación (Clustering)

Uno de los métodos más usados se basa en la agrupación de nodos. Estos algoritmos asignan un rol por nodo, los cuales realizan diferentes tareas en el grupo. Por ejemplo el rol líder o Clúster-Head (CH) se encarga de atender la comunicación en el grupo, por tanto el CH es el nodo que más consume energía. Un algoritmo basado en grupos busca minimizar el

consumo de energía en la red haciendo un cambio de rol entre los nodos.

Algunas Jerarquías de Roles

Los algoritmos de agrupación proveen una forma efectiva de prolongar el tiempo de vida de una red de sensores inalámbricos, en [Wor08] es propuesto un protocolo de agrupación distribuido con el fin de reducir el consumo de energía. Este trabajo se enfoca principalmente en balancear el tráfico y la energía a través de la red y resolver el problema de hot-spot en redes de sensores inalámbricos (El problema hot-spot describe cómo los nodos alrededor de una estación base gastan rápidamente su energía debido al alto tráfico que pasa a través de ellos desde los nodos de fuera), este proceso se lleva a cabo de acuerdo al número de vecinos y energía residual del nodo. La propuesta de este trabajo considera que existe una estación base que sincroniza a todos los sensores de la red, y cada nodo conoce el número de sensores de toda la red; del mismo modo, no considera la reagrupación cuando un nodo sale del sistema, no utiliza potencias de transmisión variable y los nodos tienen las mismas restricciones.

Otros trabajos toman en cuenta el número de nodos por cada rol que manejan, por ejemplo, algunas estrategias reducen el número de nodos CHs que comunican la red. En [CLC06] los nodos son divididos usando 3 niveles aplicando la regla 20/80 para determinar la tasa de CHs respecto a nodos miembros. Esta regla es bien conocida y fue originada por Vilfredo Pareto, quien descubrió un fenómeno común: cerca del 80 % de la riqueza en varias ciudades fue controlada por una minoría del 20 % de la gente. Los nodos de más bajo nivel son administrados y organizados por los nodos de nivel más alto. Los nodos deben intercambiar mensajes para seleccionar un clúster-head y de este modo formar los grupos. La propuesta de este trabajo es hacer un uso eficiente de energía, pero no toma en cuenta la reorganización en caso de que un nodo se pierda, los nodos son estacionarios y tienen un rango de cobertura estático; tampoco definen cómo determinan el nivel del nodo.

En [WCJcW07] se describe una estrategia de rotación de clúster head por etapas maximizando el tiempo de vida del grupo; en cada etapa, el nodo con más energía residual será seleccionado como CH y éste determinará el siguiente CH para la próxima etapa, pero hacer un cambio de rol implica gasto de energía, de esta forma, en este trabajo buscan minimizar al máximo el cambio de rol para disminuir el consumo de energía.

Otros algoritmos usan agrupación a 2-saltos, es decir, que un nodo miembro se conecta con su líder por medio de un nodo vecino. En [LGF08] se presenta un algoritmo de auto-organización basado en grupos. El algoritmo elige el CH en base a un peso (valor) que se calcula con los parámetros k-densidad, energía residual y la movilidad del nodo, el algoritmo trabaja en 2 fases: en la primera el nodo que tenga un mejor peso en su vecindario a 2-saltos será elegido como CH. Cada grupo tiene un rango entre dos valores $Thresh_{Upper}$ y $Thresh_{Lower}$. La segunda fase se encarga de agregar los nodos de los grupos que no cumplen

con el $Thresh_{Lower}$ a los grupos que no alcanzan $Thresh_{Upper}$. El objetivo es maximizar el tiempo de vida de los grupos, pero no muestran comunicación entre CH, asumen que los nodos son estables en tiempo de agrupación y tampoco usan rangos de transmisión variable.

En [DE06] se propone un algoritmo para mantener una arquitectura de anillo dirigida y construir un árbol de expansión mínimo entre los CHs, al mismo tiempo estos son clasificados en *backbone* y *leaf*. Este algoritmo funciona de 2 formas: basado en saltos y posición. En la primera, los saltos entre los CHs son tomados en cuenta para construir el árbol de expansión mínimo; en la segunda forma, el árbol de expansión mínimo es formado en base a la posición de los CHs, pero esto último requiere que el nodo esté equipado con GPS; en este trabajo buscan minimizar el retardo de ruteo de paquetes y que la formación del *backbone* sea tolerante a fallas, pero no hacen uso eficiente de la energía.

Los mismos autores que el trabajo anterior y Deniz Cokuslu en [DEC06] proponen un algoritmo distribuido para encontrar los grupos en Mobile Ad Hoc Networks (MANET) sin hacer uso del árbol de expansión mínimo, los nodos toman dos roles *leader, member* y cinco estados diferentes *IDLE, WT-INFO, WT-ACK, LDRWT-CONN* y *IDLE-WT-CONN*. El nodo con el máximo id tomará el rol *leader*; el objetivo es reducir la complejidad en los mensajes y balancear el número de nodos en cada grupo, pero no muestran cómo hacen la comunicación entre los diferentes grupos y la solución de movilidad ni muerte (agotamiento de energía) de un nodo leader.

Usando member, gateway y leader

En [OXZ04] se propone un protocolo de agrupación ligero y una infraestructura de comunicación multi-salto libre de colisiones y adaptativo. En éste, los nodos despiertan aleatoriamente y si no escuchan mensajes del leader se convierten en leader; si ya existe un leader se convierten en nodos member y cuando un nodo escucha a dos leaders, el nodo cambia su rol a gateway. Los leaders usan un canal de frecuencia predeterminada para transmitir paquetes anunciando su liderazgo y con el fin de reducir la interferencia en la comunicación seleccionan un canal de frecuencia aleatoriamente de un gran pool de frecuencias. Después de que los nodos son organizados buscan alcanzar: reducir el consumo de energía, adaptabilidad y un sistema multi-salto para la comunicación entre nodos. Sin embargo, el proceso necesita que la red sea muy densa para que exista una organización completa, es decir, asegurar comunicación en toda la red. En esta propuesta no toman en cuenta la movilidad de los nodos, no usan el cambio de rol, los nodos son estáticos y la propuesta no es tolerante a fallas.

Otro esquema de auto-organización es presentado en [LVBD07] y [LVB07], en estos se propone FISCO (a Fully Integrated Scheme of self-Configuration and self-Organization for WSN). Esta propuesta clasifica los nodos en tres roles: leader, gateway y member. El leader es el encargado de toda la comunicación, así como la asignación de direcciones en su vecindario

a un-salto, los leaders no están directamente conectados, un gateway los interconecta obteniendo comunicación a dos-saltos; un member se conecta sólo con un leader y éste controla la comunicación del nodo member; a cada nodo con rol de leader es asignado un pool de direcciones, así el espacio de direcciones es distribuido entre todos los leaders. Este trabajo asume que los nodos son aleatoriamente encendidos y distribuidos en el ambiente, los enlaces de comunicación son simétricos, su objetivo es proveer un esquema simple, de bajo costo en energía, escalable y una estructura para la comunicación y asignación de direcciones, pero no contemplan la movilidad de los nodos además de que los rangos de transmisión son estáticos.

En [/ H07] se adopta una estrategia basada en consultas de primer orden, se usa un lenguaje variante de *datalog*. Todo el protocolo es definido por medio de consultas de primer orden y se basan en procesamiento local eliminando miles de líneas de código. Son verificadas algunas propiedades como seguridad, pero el método es el mismo descrito anteriormente, por lo que la solución presenta los mismos problemas antes mencionados.

Otros autores representan los dispositivos como agentes. En [FKR+06] es presentado un proceso de auto-organización de agentes con asignación de roles dinámicos, con lo que se obtiene una adaptación del sistema multi-agente a través de la emergencia. De esta manera, la organización es construida de acuerdo al intercambio de mensajes entre los agentes, descentralizando la decisión con un algoritmo muy simple el cual toma en cuenta las restricciones de energía y número de vecinos. El algoritmo está compuesto de tres etapas; construir un modelo gráfico que describe el comportamiento del sistema, trasladar del modelo gráfico a un sistema de especificación formal y usar el modelo obtenido para verificar algunas propiedades del sistema ya organizado como son seguridad y vivacidad. El enfoque es muy parecido al descrito en [LVBD07] y en [LVB07]. El objetivo es minimizar el gasto de energía producido por técnicas de inundación. La estructura organizacional básica considera: un agente representativo (r) (ó clúster-head) el cual administra la comunicación en el grupo; algunos agentes conexión (c), los cuales conocen a varios agentes representativos y pueden pertenecer a muchos grupos; y algunos agentes ordinarios o simples miembros que solo se ocupan de sus propias tareas. Por ejemplo dados dos agentes miembros (a) y (b), con la estructura organizacional formada, la comunicación entre los dos agentes digamos es $((a;r);*[(r;c);(c;r)];(r;b))$. Este algoritmo es propuesto en [JO07] a través del modelo MWAC (Multi-Wireless-Agent-Communication) para la administración de comunicación abierta en sistemas multi-agentes embebidos en el contexto de comunicaciones inalámbricas y se hace una descripción formal del cambio de roles en el sistema.

1.4. Algoritmos que usan potencia de transmisión variable

Todos los trabajos revisados anteriormente usan diferentes métodos, reglas, métricas, etc. para que un sistema sea auto-organizado; pero los rangos de transmisión de los nodos son estáticos. Usar rangos de transmisión variable implica crear nuevas reglas para que el sistema se organice y obtener un mayor ahorro de energía. Usando rangos de transmisión variable los nodos pueden transmitir paquetes a cada destino usando la mínima energía necesaria. No importa qué tipo de algoritmo de agrupación se esté usando, los CHs son inevitablemente más activos que los demás nodos, por tanto su energía se reduce drásticamente, por estos motivos, se busca usar rangos de transmisión variable y aumentar los tiempos de vida de los clúster-head.

En [VM06] proponen un algoritmo distribuido para diseñar una red robusta y eficiente en energía, los autores presentan un modelo de red que asume que cada sensor puede seleccionar de dos a tres rangos de transmisión; las potencias de transmisión son: *whisperer, speaker* y *shouter*. *Whisperer* es el más bajo y *shouter* el más alto en potencia de transmisión. El principio fundamental del algoritmo es que cada nodo seleccione la potencia de transmisión más pequeña posible mientras mantenga la conectividad. Usando esta aproximación se muestra que la red obtenida es mejor que las redes homogéneas (rangos de transmisión estáticos) y más robustas respecto a la falla de nodos; sin embargo, en este artículo asumen que el ambiente es libre de obstrucción y que cada sensor conoce sus coordenadas.

En el articulo [LL05] un protocolo de agrupación eficiente en consumo de energía es propuesto en redes de sensores inalámbricos, por lo tanto, se espera que el protocolo resuelva problemas a través de varias capas de la red. Los autores sugieren 2 protocolos en este trabajo, el primero cambiará los rangos de transmisión dinámicamente para construir eficientemente una red de sensores inalámbricos basado en grupos. El segundo protocolo puede utilizar la energía residual de los nodos para cambiar el rol del clúster-head y consecuentemente distribuir la carga de tráfico en toda la red. El primer protocolo aumenta o reduce el rango de transmisión para mantener un cierto número de vecinos en el rango de Dl (mínimo) y Dh (máximo) vecinos. El segundo protocolo de elección de clúster-head tiene dos fases, la *fase inicial* que crea una partición la red y selecciona los CHs de acuerdo a la energía residual; y *reagrupación*, que cambia su rol con un nodo que tenga la máxima energía residual. En este trabajo los autores consideran que todos los nodos en la red son homogéneos respecto a restricciones de energía y cada nodo puede variar su rango de transmisión, los resultados muestran que este protocolo puede mejorar el tiempo de vida del grupo y, en consecuencia, de la red; la desventaja es que el método sólo trabaja bien para redes de sensores estáticos y su fase de reagrupación es débil, ya que si falla un cluster-head nadie puede llevar el control de la energía residual de los vecinos y seleccionar el siguiente.

Otra aproximación es evaluar el comportamiento de un sistema auto-organizado. El trabajo en [KA06] tiene como objetivo construir un simulador para analizar el proceso de auto-organización. El primer simulador encuentra la potencia de transmisión mínima que es necesaria para asegurar la conectividad de la red, este simulador proporciona una topología de la red para usarla en otro simulador y medir qué tan rápido se sincroniza cierto valor a través de la red, los experimentos con el simulador de conectividad muestran buenos resultados y la red se estabiliza rápidamente, pero necesita que la red sea lo suficientemente densa para asegurar una conectividad completa y los ambientes de simulación para los experimentos no son complicados, y tampoco toman en cuenta la llegada y salida de nuevos nodos.

Capítulo 2

Auto-organización de dispositivos móviles

Resumen: En este capítulo se presenta un algoritmo de auto-organización basado en grupos para crear automáticamente redes tipo ad-hoc que incluya diversos dispositivos inalámbricos tales como PDA, celulares, consolas, laptops, sensores inalámbricos, etc.

2.1. Introducción

En años recientes, las aplicaciones de redes inalámbricas se han venido incrementado y consecuentemente han recibido la atención de la comunidad de investigadores. Especialmente las redes tipo ad-hoc en las cuales todos los nodos tienen un comportamiento similar; estos retransmiten mensajes (actuando como routers) para comunicarse con nodos vecinos. Existen varias aplicaciones de redes ad-hoc, por ejemplo, despliegue de sensores para monitoreo de campos de agricultura, redes de juegos con dispositivos móviles, colocación de dispositivos en campos de batalla para predecir movimientos de tropas, etc.

Los objetivos de este tipo de redes son: mantener la conectividad entre los nodos de la red, considerando la llegada o salida de dispositivos a la red; minimizar el consumo de energía.

La naturaleza dinámica de estas redes hace difícil configurarlas, principalmente dado que la localización de los dispositivos es dinámica, esto quiere decir que la topología de la red es altamente cambiante. Ésta es la principal motivación de la investigación de métodos efectivos para crear y mantener este tipo de redes. Para este propósito, técnicas de auto-organización han sido usadas [Mil07] [Dre06].

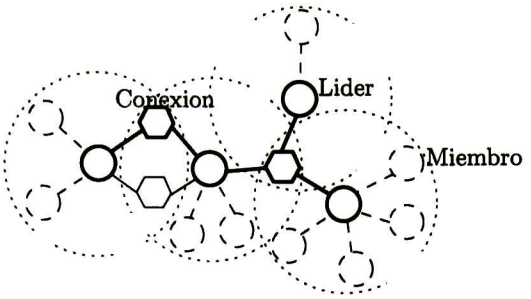

Figura 2.1: Estructura de grupos

2.2. Planteamiento y propuesta

Se propone un algoritmo basado en la auto-organización para crear automáticamente y administrar redes tipo ad-hoc. Cada nodo es representado por un agente multi-rol el cual genera un *backbone* entre los dispositivos dentro del ambiente, basado sólo en interacciones locales basadas en difusiones (broadcast).

El procedimiento propuesto permite la reconfiguración del *backbone* cuando los nodos llegan o salen de la red generando un comportamiento complejo emergente. Una vez que la red es formada, cada agente varía el intervalo de tiempo de comunicación por difusión y la potencia de transmisión permitiendo un ahorro en el consumo de la energía.

2.3. Estructura de la red auto-organizada

2.3.1. Estructura basada en grupos

La estructura de la red es creada por el principio de auto-organización, el cual asegura un bajo número de intercambio de mensajes. Básicamente la estructura está formada por un conjunto de grupos que incluyen elementos comunes que permiten la comunicación entre ellos. Los elementos de cada grupo pueden adoptar uno de tres diferentes roles *líder, conexión* y *miembro* en función del estado de la red. Cada grupo se compone de un agente jugando el rol de *líder*, uno o más agentes jugando el rol de miembro y uno o más agentes jugando el rol de *conexión*. El *líder* hace posible la comunicación entre miembros de su grupo o miembros de diferentes grupos. Un agente *conexión* es responsable de comunicar miembros de diferentes grupos a través de los agentes *líder* de cada grupo. Los miembros son conectados a un único *líder*. Los agentes *miembro* se encargan únicamente de sus propias tareas y no retransmiten los mensajes que reciben (*figura 2.1*).

2.3.2. Modelado del nodo

Como se mencionó anteriormente, los nodos son modelados como agentes.

- Cada agente tiene un identificador único ID, por ejemplo su dirección IP.
- Los agentes únicamente conocen sus vecinos a un-salto.
- Los agentes pueden moverse, llegar ó dejar el sistema (red) en cualquier momento.
- Los rangos de transmisión pueden variar entre los agentes.
- Cada agente podrá recibir mensajes donde la dirección destino no es la suya (overhearing), esta técnica será usada para recuperar información útil para el nodo y así evitar envió de mensajes broadcast.
- La comunicación entre agentes será bidireccional (dúplex).
- Cada agente mantiene: una tabla de vecinos actualizada y un *score* que es igual a la suma del número de vecinos n y la energía residual e ($score = n + e$), el *score* será usado para seleccionar el rol *líder*.
- Los agentes desconocen su posición geográfica.

El formato de mensajes que los agentes usan para comunicarse será el siguiente.

id	Type	Role	Score	P_{TX}

Donde:

1. *Id* es el identificador del agente.
2. *Type* es el tipo de mensaje que se envía.
3. *Role* es el rol que el agente actualmente esté jugando.
4. *Score* es una evaluación del nodo para la asignación de rol.
5. P_{TX} es la potencia de transmisión del mensaje.

Cada mensaje (paquete) a enviar llevará estos campos además de otra información necesaria, tal como la dirección destino, tamaño del paquete y demás. Pero sólo la información del paquete anterior es necesaria para actualizar la tabla de vecinos, la cual contiene exactamente los mismos campos antes mencionados.

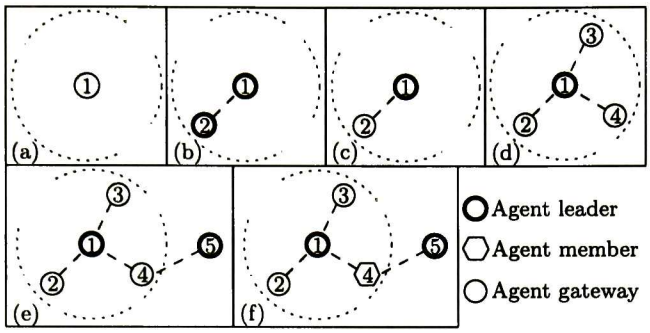

Figura 2.2: Proceso de agrupación

2.4. Algoritmo de auto-organización

2.4.1. Estrategia general

El algoritmo propuesto está basado en estrategias de auto-organización que administra la asignación de tres diferentes roles que jugará el agente los cuales, como ya se mencionó, son: *líder, conexión* y *miembro*. En realidad se tiene la ejecución de un algoritmo distribuido por los agentes que conforman la red.

Al inicio un agente no tiene vecinos y por lo tanto no tiene asignado algún rol; cuando el agente detecta vecinos el primer comportamiento que intenta adoptar es *líder*; cuando mas de un agente quiere tomar el rol de *líder*, el conflicto se resuelve por medio de un procedimiento de *elección de líder*. Los agentes que detectan más de un agente jugando el rol de *líder* en su vecindario se convertirán automáticamente en agentes *conexión* los cuales comunicaran los diferentes grupos. Finalemente otros agentes tomarán el rol de *miembro* cuando tienen un *líder* como vecino, de este modo se forman los grupos.

Cuando un nuevo agente llega a la red su rol es asignado de acuerdo al vecindario, por ejemplo, si existe un agente *líder* en el grupo, entonces el agente que llega tomará el rol de *miembro*. Si el agente detecta uno o más agentes como vecinos, y ellos no tienen rol de *líder*, el agente se convierte en *líder* y el grupo se forma con los agentes antes detectados.

Durante la formación del grupo o cuando los agentes se mueven o cambian de rol, se puede dar el caso donde dos o más agentes tienen el rol de *líder* en un mismo grupo, en este caso se ejecuta el procedimiento de *elección de agente líder*, el proceso se muestra en la *figura 2.2.*

El tiempo de vida de las baterías usadas en este tipo de dispositivos (agentes) para implementar estas redes, debe ser considerada en nuestro algoritmo; ya que ésta es la fuente de posibles cambios de rol de los agentes que conforman los grupos, la razón principal de estos cambios es que los agentes que juegan el rol de *líder* y *conexión* inevitablemente gastan

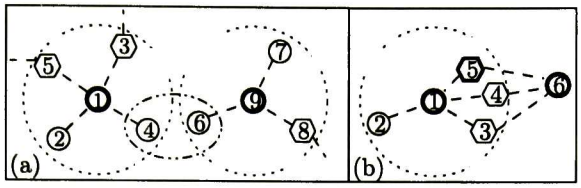

Figura 2.3: Segmentación y Agentes Conexión Duplicados

más energía que los agentes jugando el rol *miembro*, de este modo un agente decidirá cambiar su rol cuando el nivel de energía llegue a un nivel crítico.

Cuando los agentes usan las reglas de auto-organización descritas anteriormente, crean la organización permitiendo comunicación entre diferentes grupos de la red, este proceso y sus propiedades han sido formalmente probados en [FKR+06].

Código 2.1: Algoritmo de auto-organización

```
if numVecinos ≠ 0 then
        if liderNum = 0 then
                Role = lider;
        else if Role = lider then
                Eleccion_lider();
        else if liderNum = 1 then
                Role = miembro;
        else
                Role = conexion;
        end if
else
        Role = ninguno;
end if
```

En la agrupación formada anteriormente por el algoritmo, puede surgir una segmentación debido a una mala organización de los grupos (*figura 2.3a*). Una posible solución sería reiniciar el proceso de auto-organización cuando esto se presenta, sin embargo esta solución no garantiza que la siguiente organización no presente el mismo problema, y estaríamos gastando demasiada energía con el proceso, es por eso que necesitamos hacer más estudios para encontrar un método eficiente y seguro sin gastar demasiada energía para resolver el problema, (entre más densa esté la red, la probabilidad de segmentación disminuye).

2.4.2. Conservación de la energía

Reduciendo el consumo de energía

Para aumentar el tiempo de vida de la red, se debe de reducir el consumo de energía en el proceso de auto-organización. Nuestro enfoque toma en cuenta las capacidades de hardware de cambiar la potencia de transmisión de mensajes y analiza la energía necesaria en función del rol que se esté jugando y tratar de reducir el consumo.

Los agentes que juegan el rol de *conexión* y *líder* como ya se mencionó antes, gastan más energía que los demás agentes; los agentes *líder* además de ocuparse de sus propias tareas, necesitan administrar una tabla que almacena sus vecinos y comunica a todos los agentes miembro que mantiene conectados. Estas tareas incluyen la transmisión de mensajes broadcast para reconocer nuevos vecinos y la retransmisión de los mensajes entre los diferentes grupos.Los agentes *miembro* necesitan energía sólo para comunicar con su agente *líder* y las tareas asignadas en el grupo.

Nuestra aproximación toma ventaja de estos diferentes requerimientos de energía necesaria por rol. El algoritmo propuesto además de ajustar el consumo de energía en la red con el rol jugado por el agente, usa una estrategia para mantener actualizada la tabla de ruteo.

Después del proceso de agrupación, se puede dar el caso donde varios agentes *conexión* conectan los mismos agentes *líder* como se muestra en la *figura 2.3b*. Dado que solo es necesario un agente conexión para comunicar dos grupos, cuando esta situación es detectada, es lanzado un protocolo de selección de líder para elegir a un agente *conexión* y los demás agentes conexión recibirán un mensaje para que disminuyan su potencia y pasen a ser simples miembros. Cuando el agente *conexión* seleccionado consuma toda su energía, avisará a sus *líderes* para que seleccionen otro *conexión* en su lugar, y de esta manera, no se pierda la comunicación teniendo agentes *conexión* redundantes.

Energía en función del rol

Tomar en cuenta el rol que juega el agente como estrategia nos permite obtener un significante ahorro de energía, principalmente porque los nodos no necesitan la máxima potencia de transmisión para jugar su rol.

Los agentes que juegan el rol *miembro* necesitan energía sólo para comunicarse con su agente *líder* (*figura 2.4*).

La potencia de transmisión requerida en agentes miembro es calculada de la siguiente forma:

- La potencia de transmisión ideal puede ser calculada como una función de la atenuación

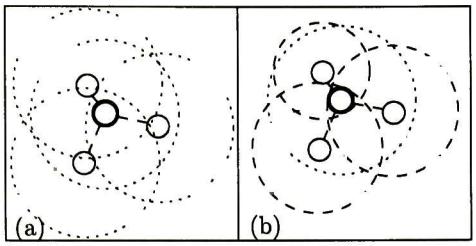

Figura 2.4: Reducción de potencia en agentes miembro

de la señal, y debe satisfacer las siguientes condiciones. Las fórmulas se refieren a un agente transmisor `A` y un receptor `B`.

La potencia de transmisión debe compensar la atenuación impuesta por la propagación de la señal del transmisor al receptor [CMd+07], garantizando que la fuerza de la señal recibida es más alta que la mínima fuerza de la señal para reconocer un paquete (mensaje) ($RX_{threshold}$), Es decir, si la potencia de la señal recibida es más baja que $RX_{threshold}$ los agentes serán incapaces de decodificar el mensaje. La atenuación que sufre la señal ($G_{A\to B}$) es asimétrica, y es calculada por la fuerza de la señal transmitida (P_{TX}) y la señal recibida (P_{RX}).

$$G_{A\to B} = \frac{P_{RX}}{P_{TX}} \tag{2.1}$$

De este modo, la potencia de transmisión ideal debe satisfacer la ecuación:

$$R_{TXmin} = \frac{RX_{threshold}}{G_{A\to B}} \tag{2.2}$$

Que es equivalente a:

$$R_{TXmin} = \frac{RX_{threshold} * P_{TX}}{P_{RX}} \tag{2.3}$$

Otro factor que influye en la fuerza de la señal, es el ruido causado por señales presentes en el ambiente. Con el fin de diferenciar los datos del ruido, la potencia de recepción del dato debe ser más alta que el nivel de potencia de ruido (N_B) por un cierto *umbral* llamado $SNR_{threshold}$; como se describe en la siguiente ecuación:

$$P_{TXmin} \geq \frac{SNR_{threshold} * N_B}{G_{A\to B}} \tag{2.4}$$

En resumen, la potencia de transmisión ideal debe satisfacer las ecuaciones 2.2 y 2.4 al mismo tiempo, se calcula con la siguiente formula.

$$P_{TXideal} = max(\frac{RX_{threshold}}{G_{A \to B}}, \frac{SNR_{threshold} * N_B}{G_{A \to B}}) \tag{2.5}$$

Los agentes líder necesitan energía para ocuparse de sus propias tareas, para mantener su tabla de vecinos actualizada y para comunicar sus agentes *miembro* y *conexión*. Cuando la energía del agente *líder* es menor que un nivel α el agente *líder* reducirá su potencia de transmisión P_{TX} hasta un 50 % y mantendrá su rol de líder, mientras mantenga por lo menos 2 agentes *conexión* en comunicación (esta estrategia puede variar dependiendo del ambiente de aplicación), y de este modo asegurar la conexión con la red y reducir el efecto del proceso de auto-organización en la red completa, y como consecuencia, obtener un ahorro en el consumo de la energía. Sin embargo, si la energía del agente *líder* es menor que un nivel β (donde $\beta < \alpha$), el agente *líder* cambiará su rol a *miembro*, ya que no tendrá la suficiente energía para mantenerse en el rol y atender las tareas asignadas.

Los agentes *conexión* necesitarán energía para comunicar los agentes *líder* de diferentes grupos en la red y para llevar a cabo sus propias tareas.

El resultado de adaptar la energía que necesitan los diferentes roles que juegan los agentes, reduce la energía a través de toda la red. La estrategia completa reduce los efectos del cambio de rol, el resultado es el incremento de vida de la red.

Manteniendo la tabla de ruteo actualizada

La idea es reducir el intervalo de transmisión broadcast si la tabla de vecinos cambia constantemente; de otra manera el intervalo será incrementado, cabe mencionar que esta estrategia no afecta el reconocimiento de nuevos agentes en el vecindario, ya que los agentes que llegan a la red son los que buscarán por vecinos.

Los agentes líder transmiten mensajes broadcast para mantener actualizada la tabla de vecinos, existen investigaciones en la literatura sobre métodos que reducen el envió de mensajes broadcast, dado que este tipo de mensajes lleva a un consumo de energía importante. Nuestra estrategia controla el intervalo (T) de envío de mensajes de manera adaptiva, de esta forma, si la red es estable el intervalo de transmisión es alto, por el contrario, si la red no es estable el intervalo de transmisión es bajo (en la *figura 2.5* se muestra gráficamente el proceso). El intervalo T se adapta tomando en cuenta:

Cuando el agente ya tiene un número N de vecinos aumentará su intervalo broadcast a $T_1 = A_1 + T$ donde A_1 es una variable de incremento.

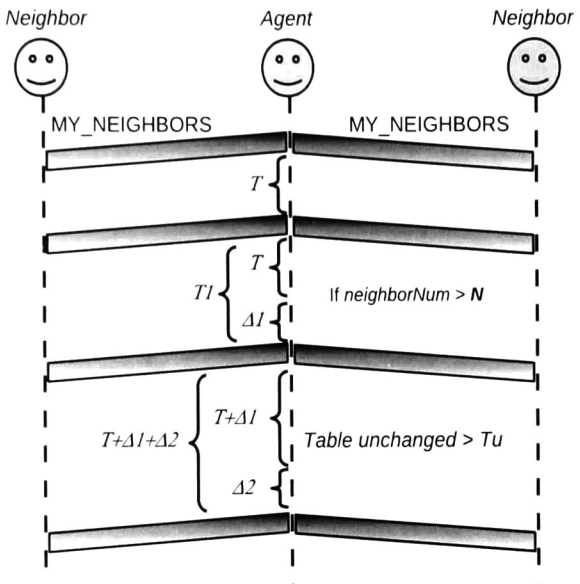

Figura 2.5: Intervalos de transmisión broadcast

- La estructuración tiende a establecerse después de un tiempo, es decir, los agentes seleccionan su rol después de solucionar los conflictos y no lo cambian durante un tiempo t_u, este tiempo depende del tipo de ambiente que se está tratando, después de este tiempo el agente incrementará el intervalo T_1 más un incremento A_2 (A_1 y A_2 pueden tener el mismo valor), es decir, $T_2 = A_2 + T_1$.

Capítulo 3

Simulación

Resumen: En este capítulo se mostrarán los resultados de agrupación del algoritmo en diferentes ambientes variando el número de nodos, dimensiones del ambiente, rangos de cobertura en la trasmisión de los agentes, etc. También se presentará un análisis del consumo de energía total en la red, comparando entre el algoritmo que usa las potencias de transmisión estáticas y el algoritmo con potencias variables.

3.1. Introducción

El comportamiento y desempeño del algoritmo propuesto ha sido analizado vía simulación usando NS-2 versión 2.33 [DAR89], éste es un simulador orientado a eventos dirigido a investigación de redes. En esta sección se presentarán ejemplos de pruebas de simulación.

3.2. Consumo de energía en agentes

3.2.1. Reducción de la potencia

El agente líder es el nodo con mayor demanda en la red, por esta razón es necesario implementar técnicas efectivas para hacer un uso eficiente de energía y prolongar la vida de la red completa, como se describió en el capítulo anterior; el agente líder tiene la capacidad de reducir la potencia de transmisión cuando el nivel de energía llega a un nivel α, cabe mencionar que el reducir la potencia del agente depende del ambiente, en las simulaciones que se realizaron se reduce la potencia hasta un 30 % mientras mantenga conectados por lo menos 2 nodos conexión.

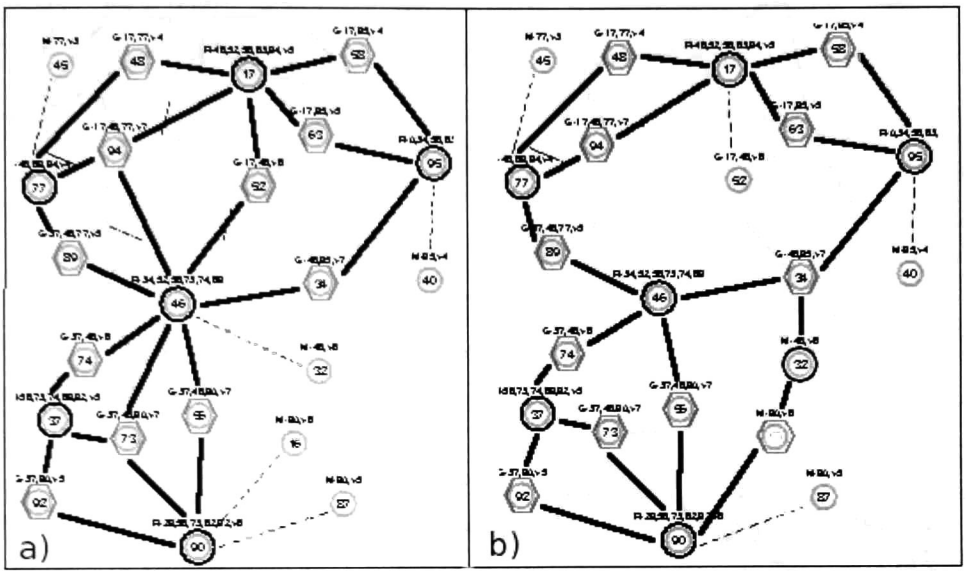

Figura 3.1: Reducción de potencia de transmisión

La *figura 3.1a* muestra un caso donde el agente 46 que es líder reduce su potencia de transmisión perdiendo a los agentes 52, 94, 32 y 73, el agente 52 simplemente se cambia a agente miembro ya que pierde comunicación con uno de sus líderes. El agente 32 pierde su único líder, y como no hay otro agente líder en su vecindario, cambia su rol a líder causando que el agente 16 cambie a agente conexión. Después que cada agente se reorganiza al perder un líder, la red vuelve a crear conexión como se muestra en la *figura 3.1b*, pero este proceso no siempre tiene éxito ya que la red puede reorganizarse en segmentos.

3.2.2. Agotamiento de energía

Cuando la energía de un agente llega al mínimo se dice que el agente muere, es decir, el agente deja su rol y todos los agente vecinos deben reorganizarse para volver a conectar la red y cubrir el trabajo que dejó el agente muerto. En las figuras 3.2 y 3.3 se muestra el proceso de perder un agente líder.

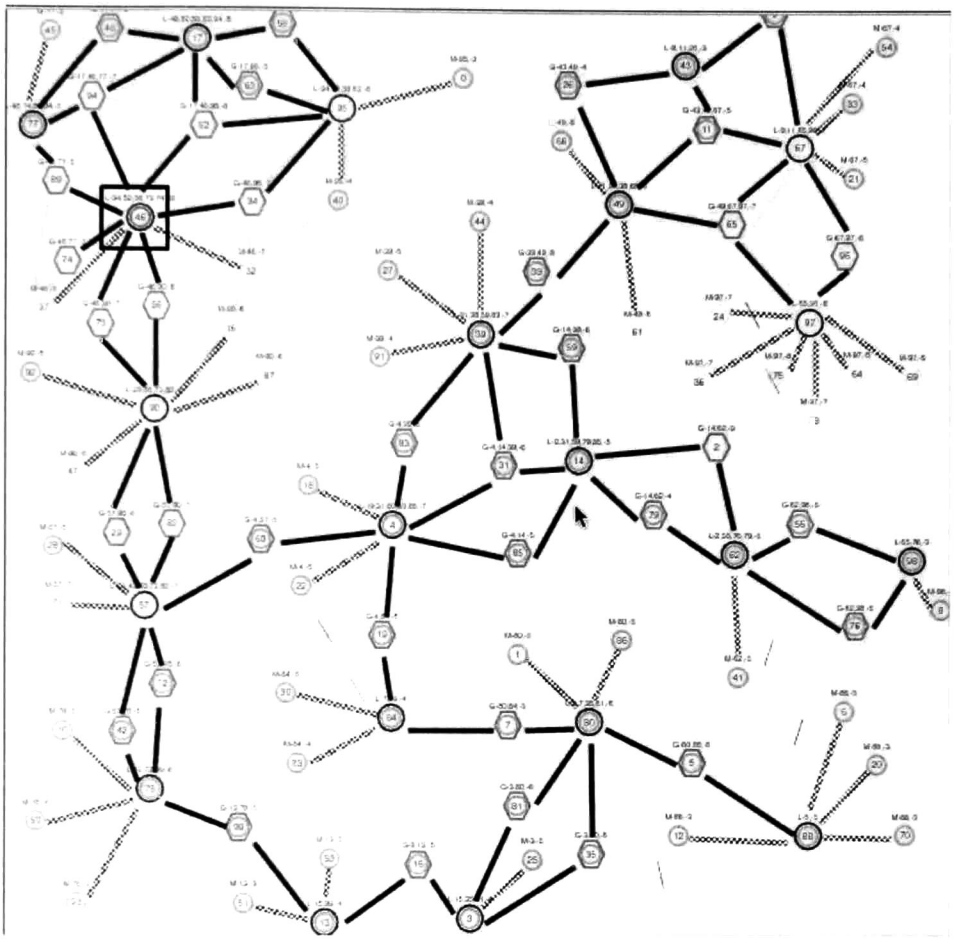

Figura 3.2: Agotamiento de energía en un nodo

En la figura 3.2 se muestra una red formada con 100 agentes, el agente líder 46 tiene 9 agentes vecinos, de este modo el agente líder 46 es el primer agente que gasta toda su energía, después que este agente muere, los vecinos deben reorganizarse cambiando su rol.

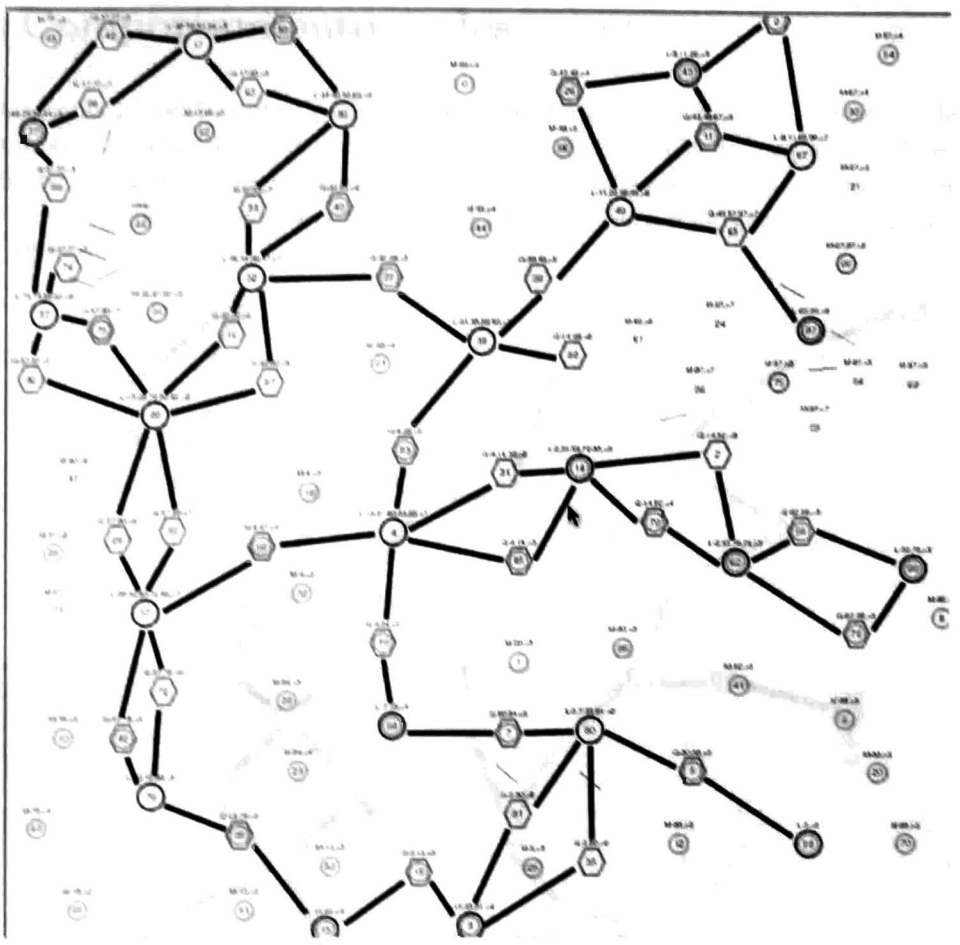

Figura 3.3: Reagrupación después del agotamiento de energía en un nodo

La *figura 3.3* muestra el agente 46 en modo dormido (sleep) porque no tiene energía para mantenerse despierto, los agentes vecinos reciben el mensaje del agente líder informando que ya no tiene energía para mantenerse, el agente 32 y 37 que jugaban el rol miembro ahora se convierten en líderes, los agentes 16, 87 y 92 se convierten en agentes conexión, el agente 56 y 52 toman el rol de miembro para conectarse con agentes líderes. La figura 3.3 muestra el backbone formado después de la reorganización.

3.3. Comportamiento en los agentes

Como sabemos la auto-organización es más que control distribuido e interacción local, más bien, se refiere a la relación entre el comportamiento de entidades individuales y la estructura resultante del sistema en conjunto, este fenómeno es llamado comportamiento emergente. Las figuras *3.4* y *3.5* muestran cómo con el simple hecho de mover un nodo la estructura cambia dramáticamente.

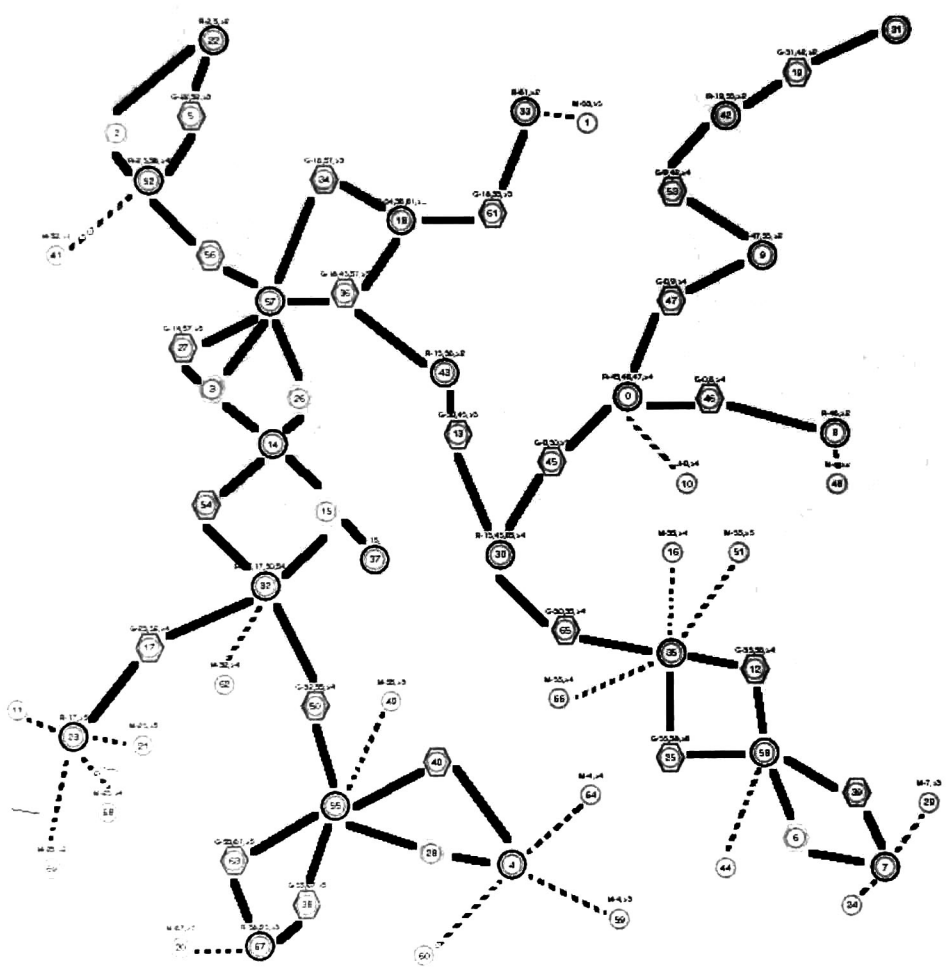

Figura 3.4: Ambiente con 70 agentes

Podemos ver la figura 3.4 que conecta todos los agentes en el ambiente, y la figura 3.5 muestra la re-organización de los agentes cuando se mueve al agente 48 hacia abajo, con este

simple movimiento la estructura se crea diferente y en 4 segmentos.

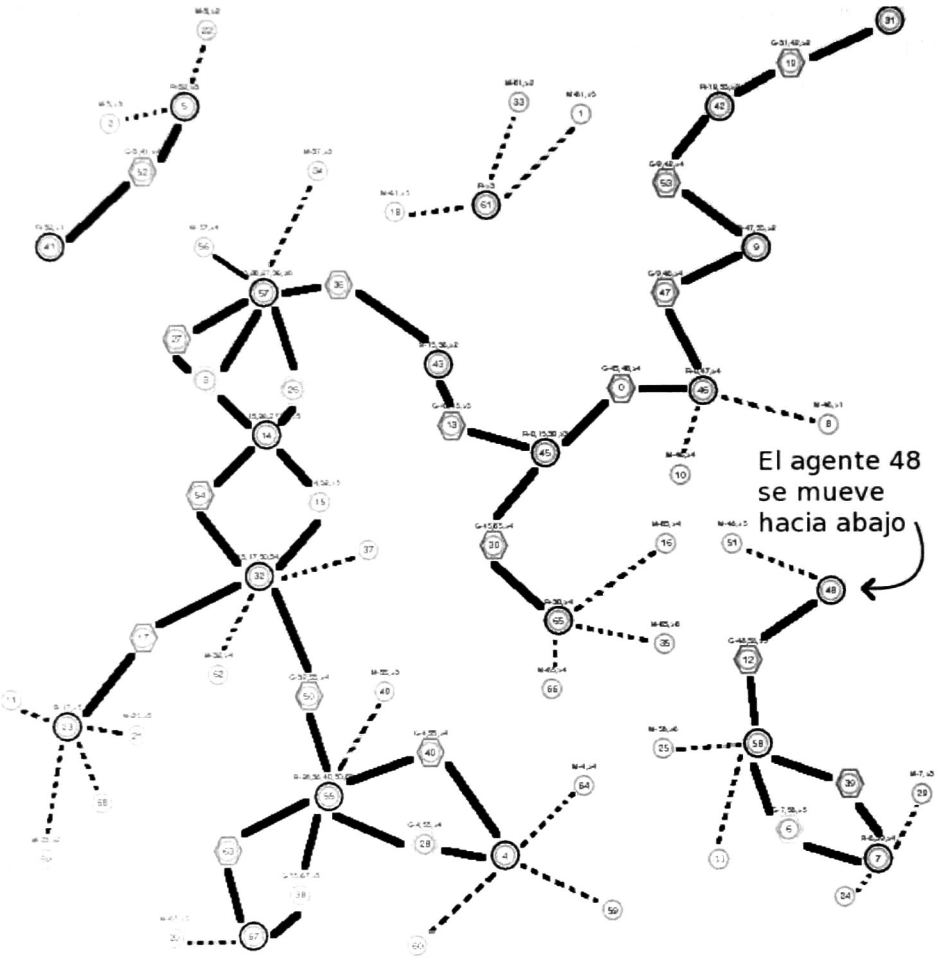

Figura 3.5: Moviendo un agente en el ambiente

De esta forma probamos que es impredecible saber cómo se comportará el sistema en conjunto, no podemos determinar qué estructura se formará después de mover un solo nodo.

3.4. Escenarios de simulación

3.4.1. Escenario 1

Consideremos un escenario donde una red de sensores inalámbricos tiene que ser organizada (*figura 3.6*), la simulación consiste de 100 agentes móviles distribuidos dentro de un ambiente de 100x100 metros. El tiempo de simulación será de 100 segundos.

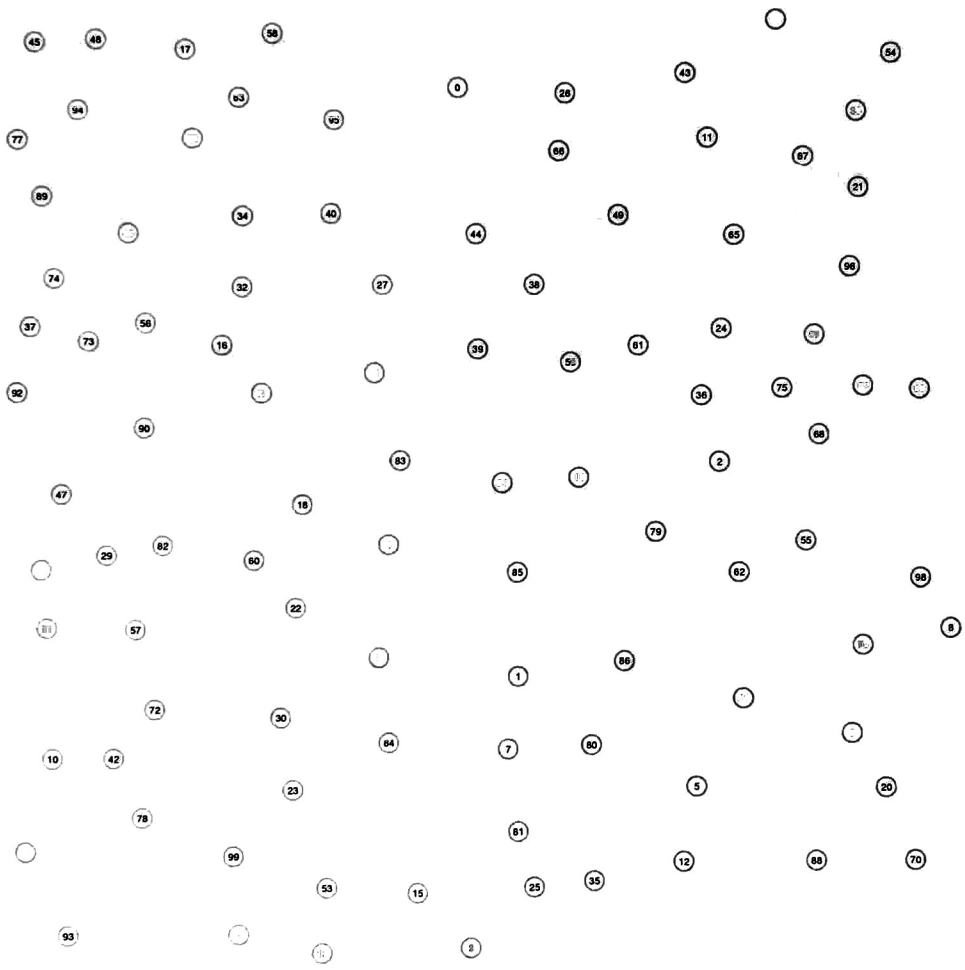

Figura 3.6: Simulación 1 - Ambiente inicial

Por simplicidad en la simulación asumimos que los agentes tienen las mismas restricciones y los agentes son encendidos simultáneamente (aunque esto no es estrictamente requerido).

La configuración inicial de los agentes es como sigue:

- La interfaz de red es 802.15.4
- La energía inicial de cada agente es de 1 joule.
- Las tareas de recibir y transmitir mensajes consumen energía, el consumo será por defecto del simulador, al igual que cambiar de modo dormido a activo.
- El rango de transmisión máximo es de 15 metros.
- La potencia de transmisión a reducir en los agentes líder será de 10 metros.

Dos algoritmos serán probados, uno considerando potencia de transmisión estática y otro que considera tanto la potencia de transmisión como el periodo de transmisión variable.

Corriendo NS-2

La simulación en la *figura 3.7* muestra el backbone generado en el instante 15 que conecta toda la red, las líneas gruesas indican enlace entre agente *líder* y *conexión* y la línea punteada indica la conexión entre agentes *miembro* y su *líder*. Este comportamiento se obtiene usando las dos versiones del algoritmo, es decir, la estructuración de los agentes no cambia aún modificando estos parámetros, ya que si la estructura cambiara, no estaríamos cumpliendo el objetivo.

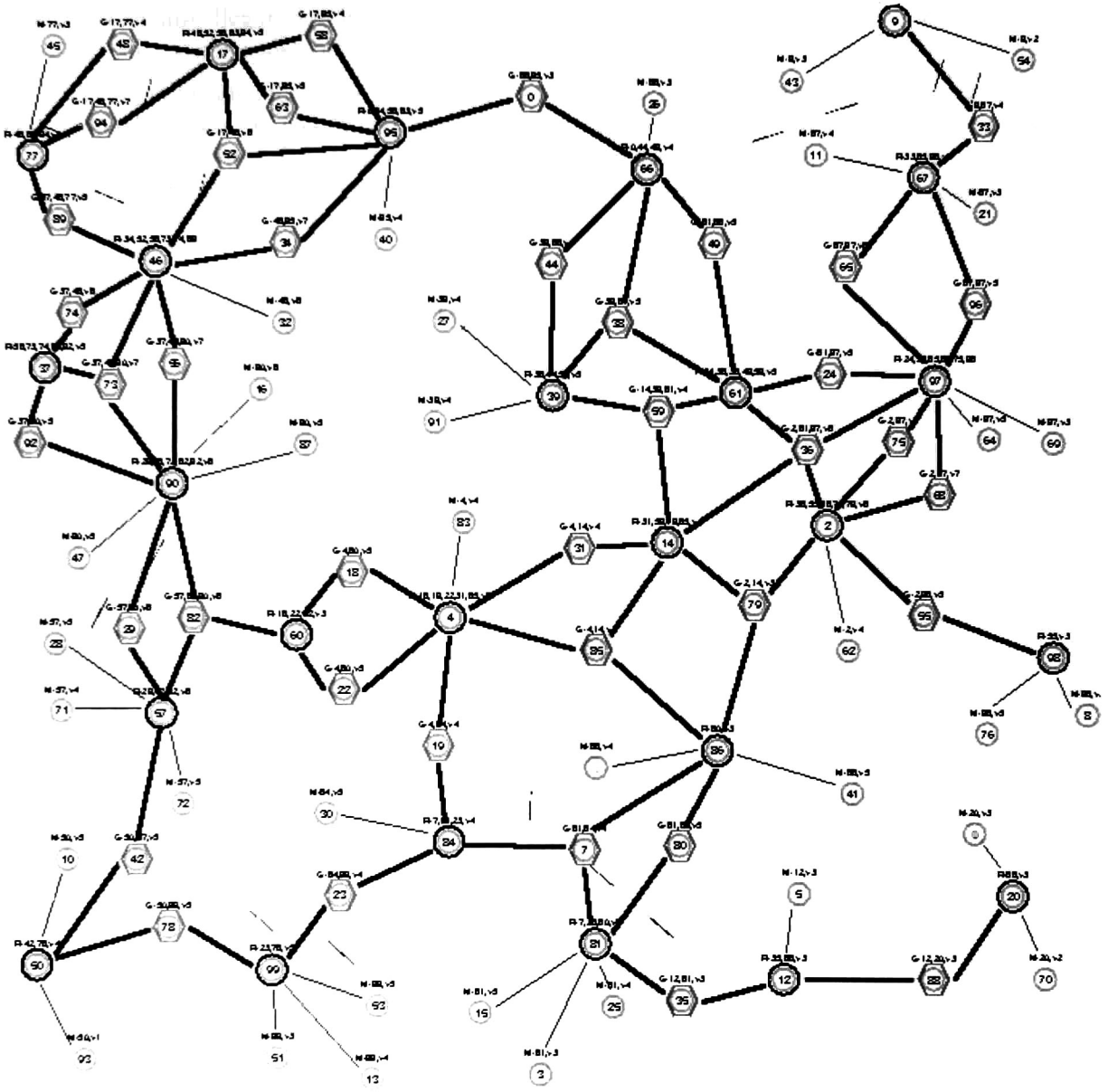

Figura 3.7: Simulación 1 - Agentes organizados

En ambientes donde los agentes son encendidos al mismo tiempo, la red puede ser creada en segmentos; ya que al inicio, todos serán agentes *líder* y después tendrán que solucionar los conflictos, esto lleva a un incremento en consumo de energía, al contrario en ambientes

donde los agentes llegan a la red gradualmente, la estructuración se forma sin problemas de segmentación, y por tanto permite un ahorro en energía.

Una de las propiedades comunes de los procesos de auto-organización es la emergencia, esta propiedad define el comportamiento complejo que se obtiene con simples reglas básicas, el comportamiento que se da en la red no puede ser predecible, incluso cuando un agente es removido del ambiente y corremos la simulación, como resultado obtenemos una organización muy diferente.

Analizando el consumo de la energía

Ahora vamos a comparar el consumo de la energía en toda la red usando las dos versiones del protocolo. Es claro que en aplicaciones reales el lapso de tiempo es demasiado pequeño, para propósito de simulación y comparación de resultados, el tiempo es bastante bueno.

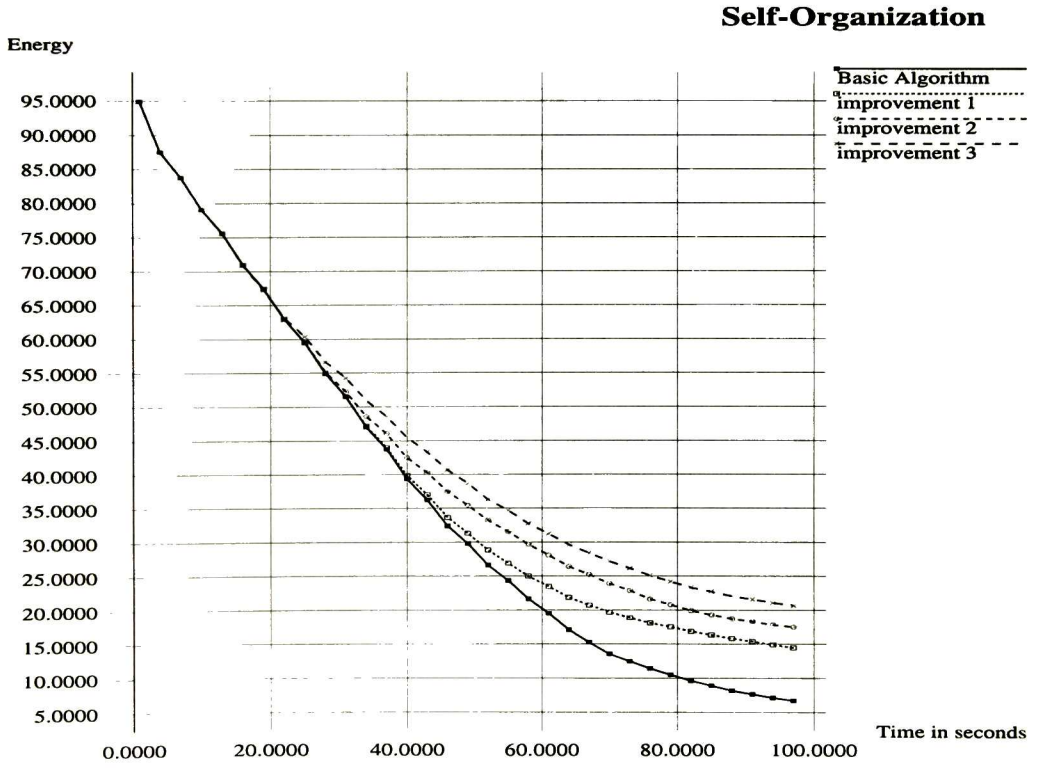

Figura 3.8: Simulación 1 Gráfica

El comportamiento de la energía de toda la red puede verse en la *figura 3.8.* Se puede

notar que el instante 20 la energía consumida es igual para todas las curvas, ya que éste es el tiempo que lleva a la red estabilizarse, este instante depende de la aplicación, por ejemplo, si en el ambiente los agentes llegan gradualmente, el tiempo que toma a los agentes estabilizarse es casi instantáneo, pues no hay conflictos entre agentes. Después de este tiempo las curvas en la gráfica difieren una de la otra.

- La curva de más abajo corresponde al consumo de energía usando el algoritmo más simple.
- La siguiente curva hacia arriba muestra el comportamiento del segundo algoritmo cuando la potencia de transmisión es ajustada en los nodos miembros.
- La tercera curva muestra el desempeño de energía cuando los periodos de transmisión broadcast se incrementan y la potencia de transmisión son reducidos.
- Finalmente la curva de arriba muestra el comportamiento del consumo de energía cuando, además de las mejoras anteriores, la potencia de transmisión de los agentes *líder* es ajustada.

En esta gráfica el ahorro de energía en el instante 100 es cerca del 14 %, adicional a este tiempo la conectividad se conserva.

3.4.2. Escenario 2

Consideremos un escenario donde una red de sensores inalámbricos tiene que ser organizada, el ambiente es de 50x50 metros, colocados aleatoriamente 50 agentes inalámbricos (*figura 3.9*). El tiempo de simulación será de 100 segundos.

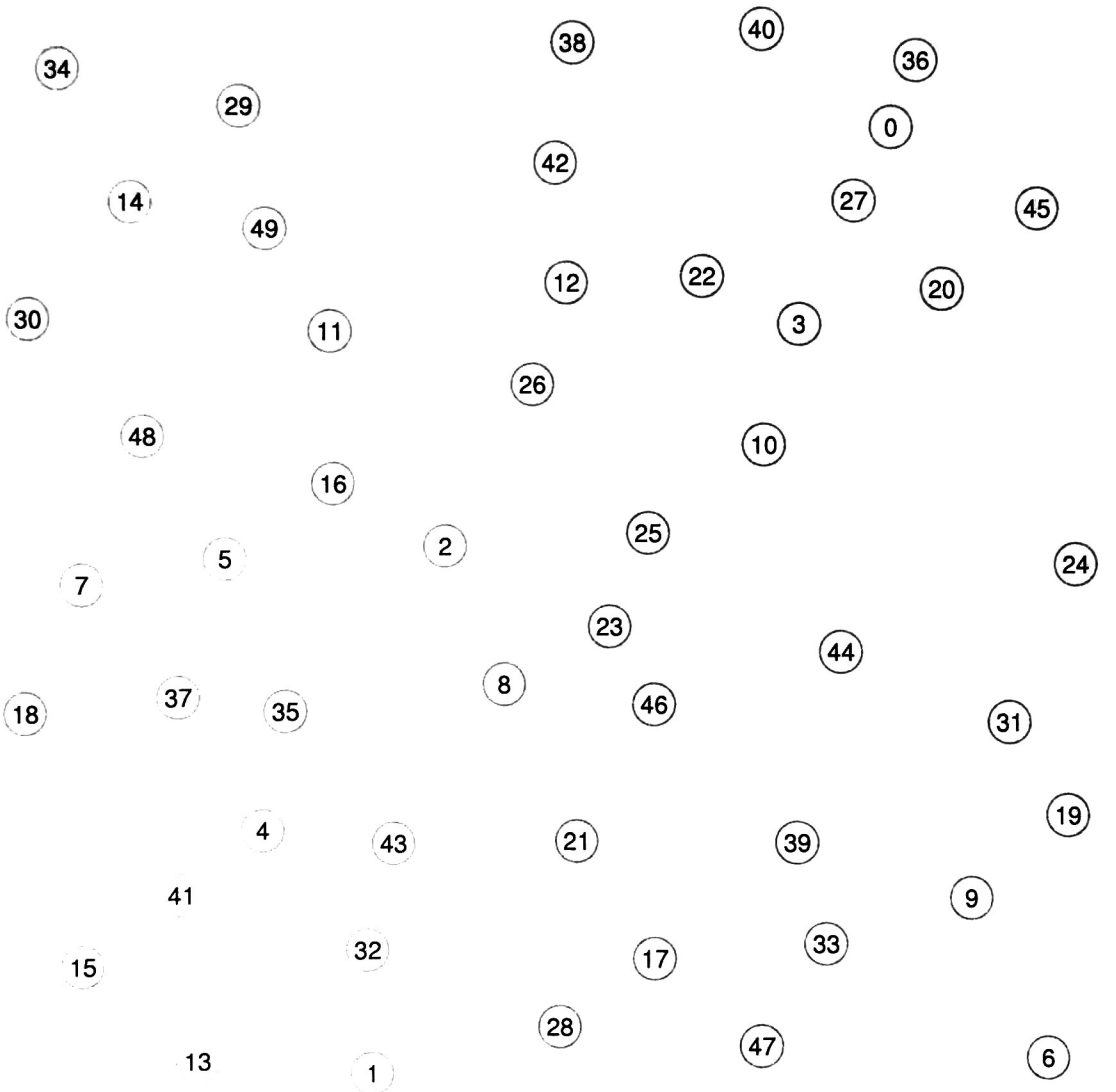

Figura 3.9: Simulación 2 - Ambiente inicial

Otras características de simulación son las siguientes:

- La interfaz de red es 802.15.4
- La energía inicial de cada agente es de 2 joules.
- Las tareas de recibir y transmitir mensajes consumen energía, el consumo será por defecto del simulador, al igual que cambiar de modo dormido a activo.
- El rango de transmisión máximo es de 10 metros.
- El rango de transmisión a reducir en los agentes líder será de 9 metros.

Corriendo NS-2

En la *figura 3.10* se muestra el ambiente en el instante 10 cuando el backbone ya está formado conectando la red completa. La estructuración que se muestra en la figura al igual que la simulación anterior, no modifica el backbone al modificar las potencias de transmisión.

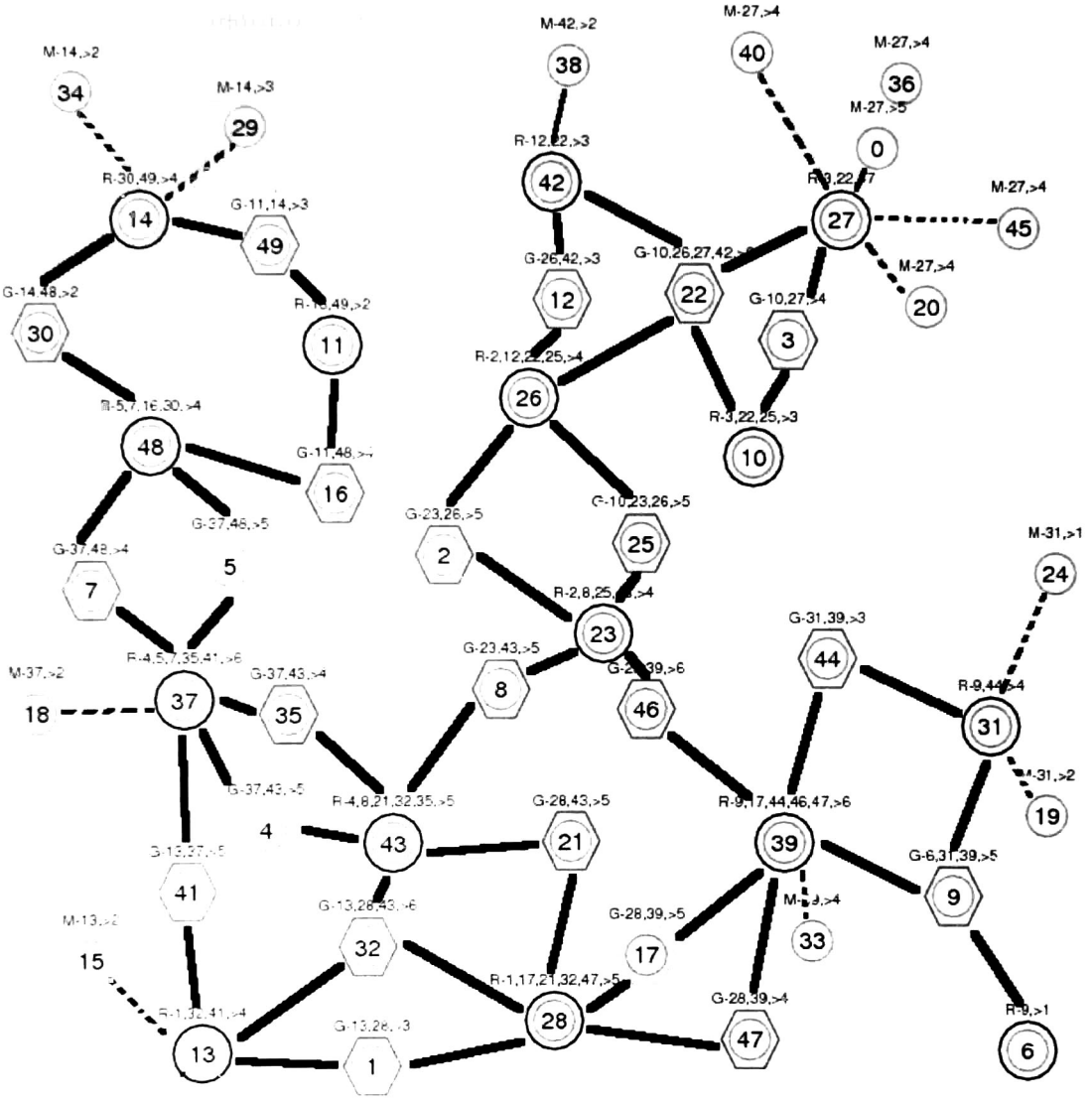

Figura 3.10: Simulación 2 - Agentes organizados

En la figura podemos ver cómo los agentes conexión 4, 5 y 17 han pasado a modo dormido (sleep) ya que existe otro agente conexión haciendo el trabajo de comunicar los agentes líderes, por ejemplo; el agente 5 conecta los mismos agentes lideres que el agente 7, los agentes líderes se dan cuenta de esto y seleccionan al agente con mejor *score*, en caso que tengan el mismo seleccionarán el que tiene mayor *id*. Este método se lleva a cabo para ahorrar la energía, entre más agentes se encuentren en el ambiente existirán mas agentes conexión en modo dormido.

Analizando el consumo de energía

La energía consumida por cada uno de los algoritmos se puede ver en la gráfica *3.11*. La curva sólida corresponde al algoritmo básico, y la curva punteada corresponde al algoritmo que implementa los métodos de ahorro de energía.

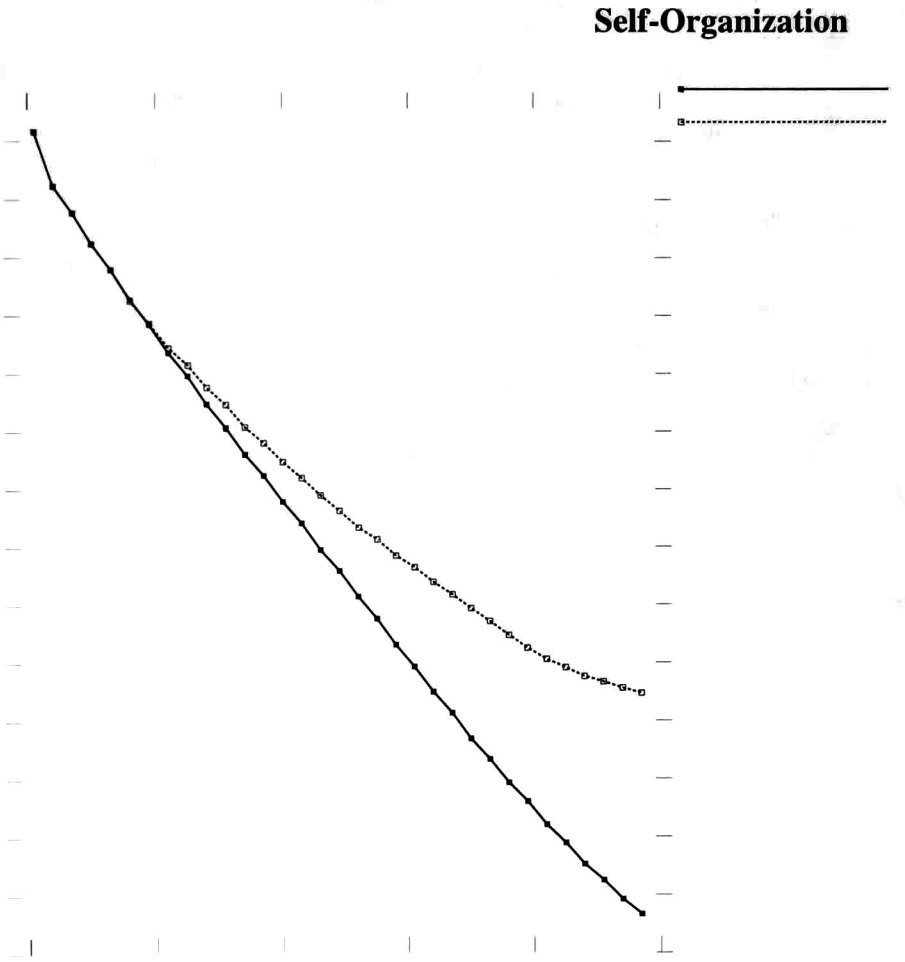

Figura 3.11: Simulación 2 - Gráfica

El ahorro de energía que se obtuvo en este ambiente es casi del 30 %, un poco más que en el ambiente anterior.

3.4.3. Escenario 3

El ambiente que presentamos ahora tiene una dimensión de 100x100 metros, con un tiempo de simulación de 100 segundos, colocados de forma aleatoria 80 agentes (*figura 3.12*).

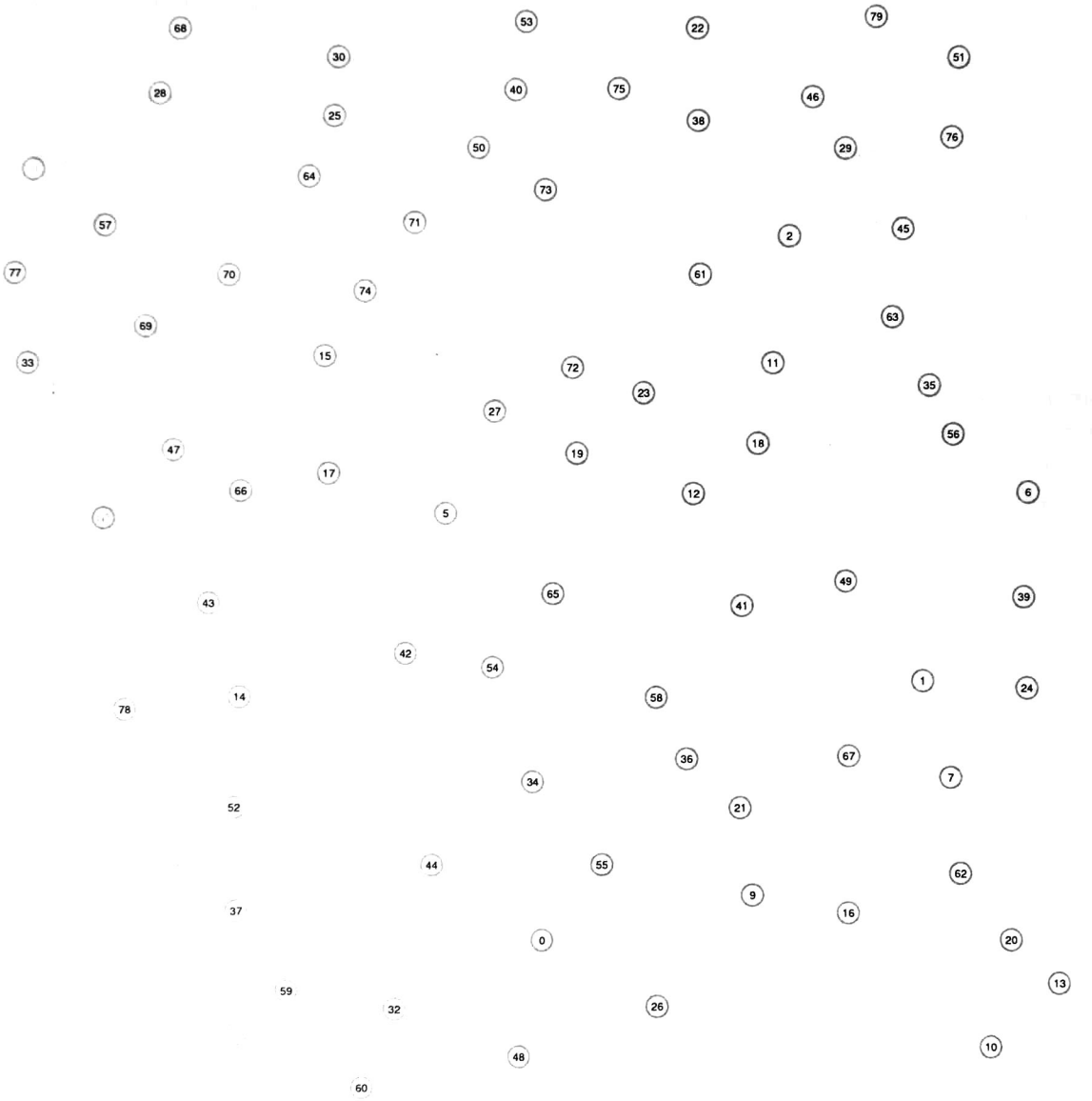

Figura 3.12: Simulación 3 - Ambiente inicial

Las características de simulación son las siguientes:

- Interfaz de red: 802.15.4.
- Energía inicial: 2 joules.
- Rango de transmisión: 15 metros.
- Mínimo rango de transmisión usado por los agentes líder que intentan reducir el consumo de energía: 9 metros.
- Rango de transmisión de agentes conexión en modo dormido: 5 metros.

  Las tareas de recibir y transmitir mensajes consumen energía, el consumo será por defecto del simulador, al igual que cambiar de modo dormido a activo.

Corriendo NS-2

En la *figura 3.13* se puede observar la organización de los agentes en el instante 40 formando un backbone que conecta todos los agentes en el ambiente. En este escenario los agentes conexión que ahorran energía son más que en el escenario anterior, en la figura 3.13 se pueden ver 9 agentes conexión (60, 25, 40, 53, 29, 45, 11, 13, 59) en modo ahorro de energía (sleep).

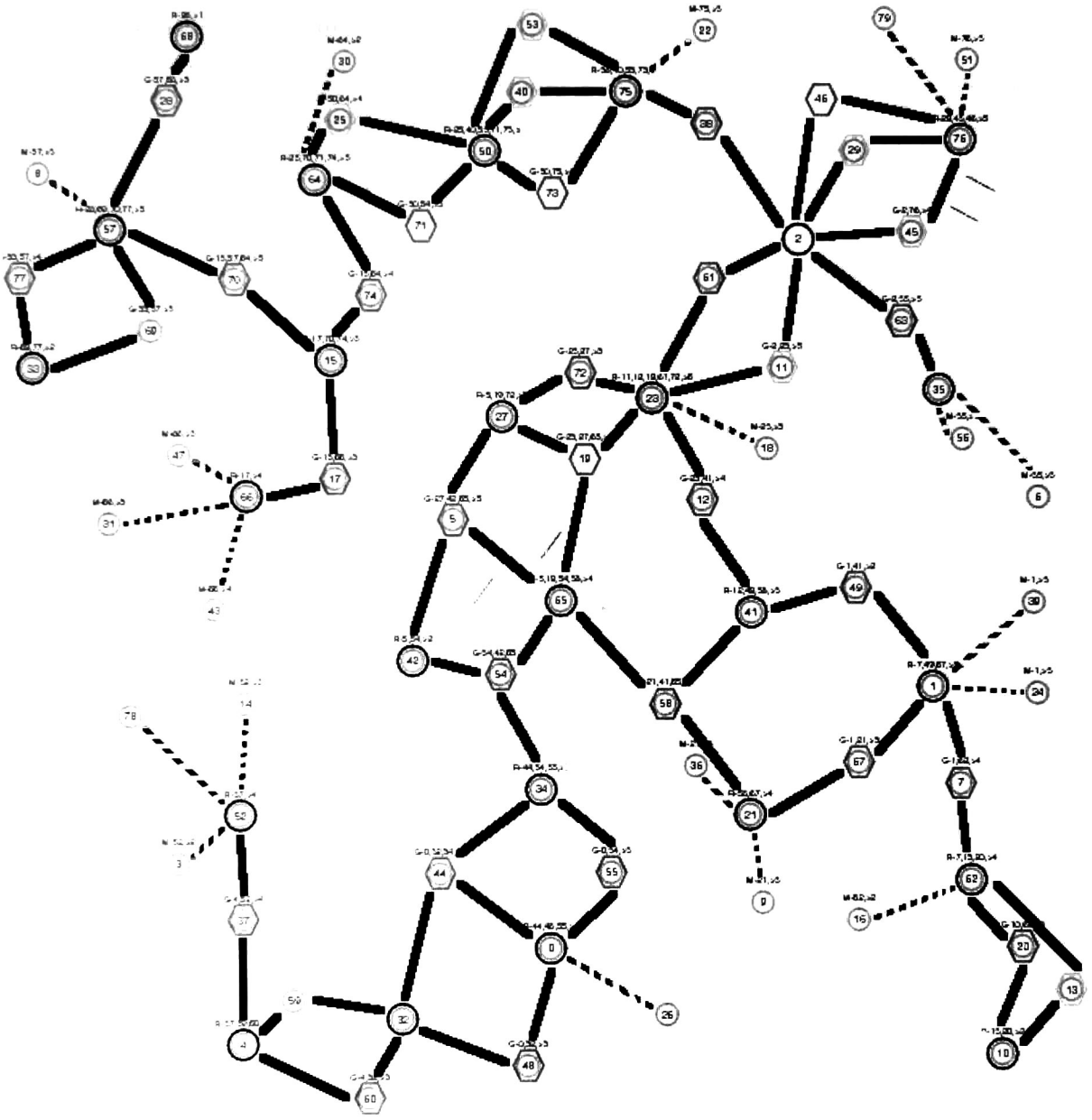

Figura 3.13: Simulación 3 Agentes organizados

Analizando el consumo de energía

La energía total consumida por los agentes se muestra en la *gráfica 3.14*, la curva sólida corresponde al algoritmo básico, podemos ver que en el instante 100 la curva llega hasta 40 joules de energía, mientras la curva punteada que corresponde al algoritmo con los métodos de ahorro de energía está cerca de 80 joules, en esta simulación se obtiene un 25% en el ahorro de la energía.

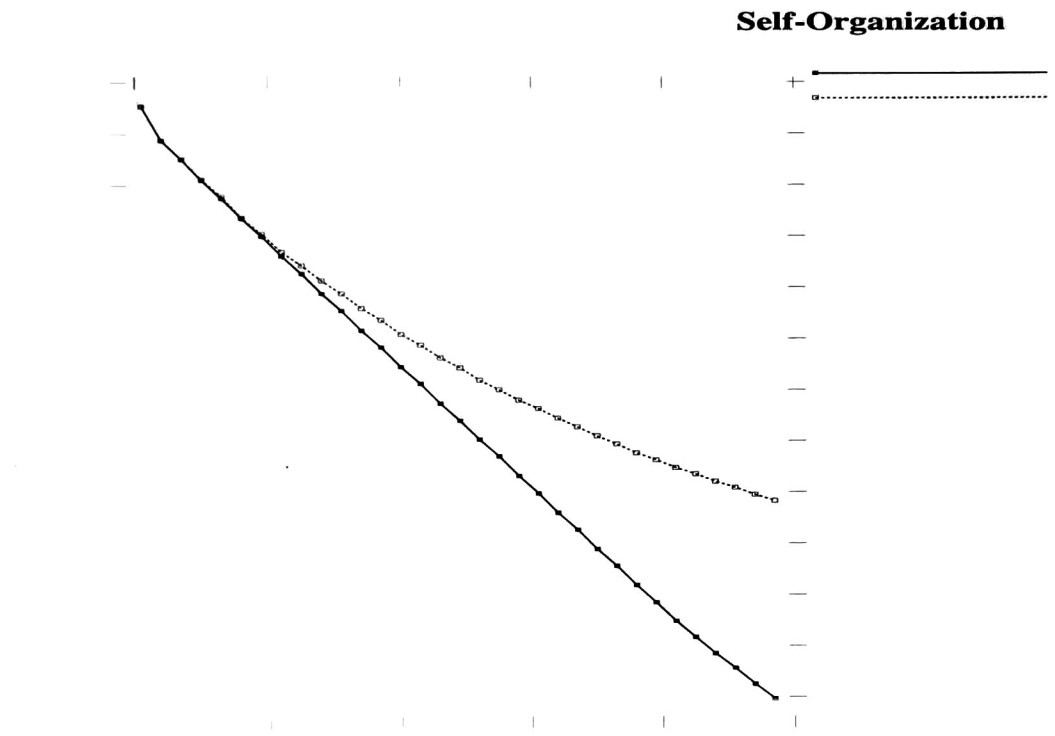

Figura 3.14: Simulación 3 - Gráfica

Aplicando los métodos de ahorro en la energía estamos obteniendo un ahorro considerable sin perder conexión alguna entre los agentes, es decir, al correr los dos algoritmos nos da como resultado exactamente la misma estructura backbone.

Capítulo 4

Configuración del simulador NS-2

4.1. Objetivo

Dar una introducción del simulador NS-2 [DAR89], así como el proceso de configuración para agregar funcionalidad al simulador. Se implementará un protocolo llamado *Soa.*

4.2. Introducción

NS-2 provee soporte para simulación de protocolos TCP, ruteo y multicast tanto en redes alámbricas como inalámbricas, en este trabajo nos referimos a NS-2 en la versión actual 2.33 y trabajamos sobre redes inalámbricas. La simulación fue realizada en fedora 10 (*Linux*), los lenguajes de programación necesarios para la simulación son TCL y C++

4.3. Definición del agente

El protocolo creado será agregado a cada agente en la simulación para formar la organización entre ellos. Primeramente creamos una nueva carpeta llamada SOA dentro del directorio base de NS-2, este directorio tendrá 5 archivos.

- soa.h: Éste es el archivo cabecera donde se definirán los relojes necesarios y agentes de envió y recepción de mensajes.
- soa.cc: En este archivo se implementa todo lo definido en el archivo de cabecera *Soa.h.*

  pkt.h: Aquí se definen los tipos de paquetes que serán enviados por los agentes.

- table.h: Archivo de cabecera donde se define la tabla de vecinos.
- table.cc: En este archivo se implementa cada uno de los métodos definidos en table.h para la manipulación de vecinos.
- definitions.h: Se definen variables y constantes usadas por los agentes.

Para implementar un protocolo en NS-2 debemos crear un agente que herede de la clase *Agent*, ésta es la clase principal que tenemos que codificar con el fin de implementar nuestro algoritmo. Además, esta clase ofrece un enlace con la interfaz Tcl, de este modo seremos capaces de controlar nuestro algoritmo a través de los archivos de simulación escritos en Tcl. Nuestro agente mantendrá un estado interno y una tabla de vecinos, esta tabla la tratamos como una nueva clase *table.h*.

Nuestro protocolo debe definir un nuevo tipo de paquete, que representará el formato de control de paquetes con el cual se comunicará el agente. Para que un agente envíe mensajes durante períodos de tiempo, se debe implementar una clase *PktTimer* que herede de la clase *TimerHandler*. Para obtener resultados, aparte de la simulación, necesitamos usar la clase *Trace* para crear archivos de bitácora y gráficas de resultados, es decir, obtener archivos donde se indica lo que está pasando en la simulación.

4.3.1. Tipos de paquete

Creamos un nuevo archivo llamado *SOA/pkt.h*, aquí pondremos las estructuras de datos que vamos a utilizar.

```
/* pkt.h */

#ifndef _PKT_H
#define _PKT_H
#include <packet.h>
#include <list>

#define HDR_SOA_PKT(p) hdr_soa::access(p)
using namespace std;

struct hdr_soa {

    nsaddr_t  src_;
    u_int8_t  seq_num_;
    short type_;
```

```
    short role_;
    double score_;
    inline nsaddr_t& src() { return src_; }
    inline u_int8_t& seq_num() { return seq_num_; }
    inline short& type() { return type_; }
    inline short& role() { return role_; }
    inline double& score() { return score_; }

    static int offset_;
    inline static int& offset() { return offset_; }
    inline static hdr_soa* access(const Packet* p) {
        return (hdr_soa*)p->access(offset_);
    }
};

#endif /* _PKT_H */
```

Declaramos una estructura *hdr_soa* que representa el nuevo tipo de paquete que estamos definiendo. Para ligar nuestro paquete definido a la interfaz Tcl lo haremos en *SOA/soa.cc* como sigue:

```
int hdr_soa::offset_;

static class SoaHeaderClass : public PacketHeaderClass {
    public:
    SoaHeaderClass() : PacketHeaderClass("PacketHeader/Soa"
    sizeof(hdr_soa)) {
        bind_offset(&hdr_soa::offset_);
    }
} class_rtProtoSoa_hdr;
```

4.3.2. Programación del algoritmo

Después que hemos definido nuestro paquete, empezamos a programar nuestro agente, dentro de *SOA/soa.h* definimos una nueva clase llamada *Soa* que contiene los atributos y funciones que se necesitan. Para comunicar con los agentes vecinos, el algoritmo envía un mensaje broadcast periódicamente. Para el envío de mensajes necesitamos una clase que herede de *TimerHandler* como se muestra en el código siguiente:

```
#ifndef _SOA_H
#define _SOA_H
```

```
#include "pkt.h"
#include "table.h"

#define CURRENT_TIME Scheduler::instance().clock()
#define JITTER (Random::uniform()*0.5)
#define LOW_ENERGY (me_->initialenergy() / 100 * 50)
#define MIN_ENERGY (me_->initialenergy() / 100 * 15)

class Soa;
/* Timers */
class PktTimer : public TimerHandler {
    public:
        PktTimer(Soa* agent) : TimerHandler() {
            agent_ = agent;
        }
    protected:
        Soa* agent_;
        virtual void expire(Event* e);
};

/* Agent */
class Soa : public Agent {
    /* Friends */
    friend class PktTimer;
    static bool ban_create;

    nsaddr_t addr_;
    double INTERVAL;
    int    ROLE;
    double SCORE;

    void get_node();

    protected:
        //
    public:
        //
};
#endif  /* _SOA_H */
```

Ahora debemos ligar el agente como lo hicimos con el paquete, el objetivo es poder instanciar nuestro agente desde Tcl. Para hacerlo debemos heredar de la clase *TclClass* como se describe en el siguiente código, en el archivo *SOA/soa.cc.*

```
static class SoaClass : public TclClass {
    public:
        SoaClass() : TclClass("Agent/Soa") {}
        TclObject* create(int argc  const char*const* argv) {
            assert(argc == 5);
            return (new Soa((nsaddr_t)Address::instance().
            str2addr(argv[4])));
        }
} class_rtProtoSoa;
```

Lo que tenemos que codificar en *SOA/soa.cc* acerca de los timers es el método expire();

```
void PktTimer::expire(Event* e) {
    agent_->send_soa_pkt(MY_NEIGHBORS, IP_BROADCAST);
    agent_->reset_soa_pktTimer();
}
```

Para llamar el constructor de la clase base pasamos PT_SOA como argumento, esta constante será definida después y se usará para identificar el envío y recepción de paquetes.

El código Tcl de abajo muestra cómo ejecutar el comando de impresión de tabla, esta operación puede ser ejecutada en un cierto tiempo en el archivo de simulación. Se asume que *ns_*contiene una instancia del simulador y *node_* es un agente creado por *ns_*, pasamos *255* como argumento; ya que éste es el número de puerto al que el agente está ligado.

```
#simulation.tcl

$ns_ at 15.0  [$node_(0)_agent_255]_print_table"
```

4.3.3. Administración de la tabla de vecinos

Para manejar una tabla de vecinos vamos a crear una clase que encapsule la funcionalidad que necesitamos. Por cada entrada en la tabla de vecinos debemos almacenar la *dirección* del agente vecino, *rol* que juega, *score* actualizado y una lista de vecinos por agente. El siguiente código corresponde a *SOA/table.h.*

```
#ifndef _TABLE_H
#define _TABLE_H
```

```
#include <trace.h>

class Soa_table {
    table table_;
    bool compare_list(list<nsaddr_t> l1  list<nsaddr_t> l2);

    public:
        void print(Trace*);
        void print_table();
        void clear();
        void rm_entry(nsaddr_t);
        void add_entry(nsaddr_t, Entry);
        bool lookup(nsaddr_t  Entry*);
        u_int32_t size();

        const char* neighbors(int role);
        const char* neighbors();
        const char* idsToChar(std::vector<nsaddr_t>& v);
        int num_neighbors(int role);
        int num_neighbors();
        nsaddr_t get_leader();
        list<nsaddr_t> get_neighbors();
        list<nsaddr_t> get_neighbors(int role);
};
#endif  /* _TABLE_H */
```

La implementación de la tabla se codifica en *SOA/table.cc*, solo se muestra la función de imprimir la tabla en el archivo de salida:

```
void Soa_table::print(Trace* out) {
    sprintf(out->pt_->buffer()  "P\tneighbor\trole");
    out->pt_->dump();
    for (ti it = table_.begin(); it != table_.end(); it++) {
        sprintf(out->pt_->buffer()  "P\t%d\t%d", (*it).first
        ((Entry)it->second).role);
        out->pt_->dump();
    }
}
//
```

4.4. Cambios en el NS-2

Hemos implementado un agente para crear auto-organización dentro del NS-2, pero hay algunos cambios que necesitamos hacer en con el fin de integrar nuestro código dentro del simulador.

Tenemos que usar una constante para reconocer nuestro paquete dentro del simulador, definiremos PT_SOA dentro de *common/packet.h.*

Existe la definición de constantes en este archivo, donde todos los tipos de paquetes son agregados, agregamos PT_SOA como se muestra en el siguiente código.

```
/* common/packet.h */

typedef unsigned int packet_t;

static const packet_t PT_TCP = 0;
static const packet_t PT_UDP = 1;



static const packet_t PT_BLTRACE = 60;
static const packet_t PT_SOA = 61; // insertamos el nuevo paquete aquí
static packet_t       PT_NTYPE = 62; // This MUST be the LAST one
```

Dentro del mismo archivo localizamos la clase *p_info* y proporcionamos un nombre a nuestro agente.

```
/* common/packet.h */

class p_info {
public:
  p_info()
  {
    initName();
  }
  static void initName()
  {
    name_[PT_TCP]= "tcp";
    name_[PT_UDP]= "udp";
    name_[PT_CBR]= "cbr";
    name_[PT_SOA]= "Soa";   //definimos en esta lista
    name_[PT_NTYPE]= "undefined";
```

```
  }
};
```

El objeto *Trace* es usado para escribir información requerida sobre los paquetes enviados, recibidos y borrados. Para almacenar información respecto a nuestro paquete, implementamos la función *format_soa* dentro de la clase *CMUTrace*. Para esto editamos el archivo *trace/cmu-trace* y agregamos la función.

```
/* trace/cmu-trace.h */

class CMUTrace : public Trace {
  /* ... definitions      */
  private:
  /* ... */
  void format_aodv(Packet *p, int offset);
  void format_soa(Packet *p, int offset);//agregamos esta linea
};
```

La siguiente pieza de código muestra los diferentes tipos de trazas, tenemos que agregarla en *trace/cmu-trace.cc*.

```
void CMUTrace::format_soa(Packet *p, int offset)
{
  struct hdr_soa* ph = HDR_SOA_PKT(p);
  struct hdr_ip* ih  = HDR_IP(p);

  if (pt_->tagged()) {
    sprintf(pt_->buffer() + offset    -soa:Is %d -soa:Hs %d
-soa:It %d -soa:Ir %d"  ph->src()  ih->daddr()  ph->type()  ph->role())
  }
  else if (newtrace_) {
    sprintf(pt_->buffer() + offset,  -P soa -Ps %d -Pd %d
-Pt %d -Pr %d"  ph->src(), ih->daddr(), ph->type(), ph->role());
  }
  else {
    sprintf(pt_->buffer() + offset    [soa %d %d %d %d] ",
ph->src()  ih->daddr()  ph->type()  ph->role());
  }
}
```

Con el fin de poder llamar la función creada debemos modificar el archivo *trace/cmu-trace.cc* y cambiar la función format.

```
/* trace/cmu-trace cc */

void CMUTrace::format(Packet* p, const char *why)
{
  /*      */
  case PT_PING:
  break;

  case PT_SOA:
    format_soa(p, offset);
  break;

  default:
  /*      */
}
```

Ahora necesitamos hacer algunos cambios en archivos Tcl, en *tcl/lib/ns-packet.tcl* debemos localizar el siguiente código y agregar *Soa* a la lista.

```
#tcl/lib/ns-packet.tcl

foreach prot {
  Soa
  AODV  # routing protocol for ad-hoc networks
  ARP # Address Resolution Protocol, network wireless stack
  # ...
  HDLC  # High Level Data Link Control
} {
  add-packet-header $prot
}
```

Finalmente debemos modificar *tcl/lib/ns-lib.tcl*, tenemos que agregar una función para instanciar el nuevo agente, nuestro interés se centra en crear un agente inalámbrico con el algoritmo (*agente*) Soa.

```
#tcl/lib/ns-lib.tcl

Simulator instproc create-wireless-node args {
  # ..
  switch -exact $routingAgent_ {
    Soa {
      set ragent [$self create-soa-agent $node]
```

```
    }
    # ..
  }
  #
}
```

Después para crear el agente con la dirección del nodo codificamos lo siguiente.

```
#tcl/lib/ns-lib.tcl

Simulator instproc create-soa-agent { node } {
  # Create Soa agent
  set ragent [new Agent/Soa [$node node-addr]]
  $self at 0.0 "$ragent start"
  $node set ragent_ $ragent
  return $ragent
}
```

Ahora todo está implementado y necesitamos compilar, para hacer esto necesitamos editar el archivo, y agregar nuestros archivos objeto dentro de la variable OBJ_CC como se muestra enseguida.

```
#Makefile

OBJ_CC = \
tools/random.o tools/rng.o tools/ranvar.o common/misc.o \
#
SOA/soa.o SOA/table.o \
#
$(OBJ_STL)
```

Necesitamos ejecutar el comando *touch* al archivo *common/packet.cc* ya que modificamos el archivo *common/packet.h.* Después de esto podemos hacer *Make* y el algoritmo está listo para correrlo dentro de un archivo Tcl.

[ns-2.33]$ touch common/packet.cc [ns-2.33]$ make

4.5. Creando el archivo de simulación Tcl

El código siguiente muestra un ejemplo de cómo usar el agente agregado al simulador.

```
set val(chan)    Channel/WirelessChannel
```

```
set val(prop)    Propagation/TwoRayGround
set val(netif)   Phy/WirelessPhy/802_15_4
set val(mac)     Mac/802_15_4
set val(ifq)     Queue/DropTail/PriQueue   ;# tipo de cola
set val(ll)      LL                        ;# tipo de la capa de enlace
set val(ant)     Antenna/OmniAntenna       ;# modelo de la entena
set val(ifqlen)  50
set val(nn)      50                        ;# numero de agentes
set val(rp)      Soa                       ;# agregamos nuestro agente Soa
set val(x)   100
set val(y)   100

set val(tr) soa802_15_4.tr   ;#archivo de trazas
set val(nam)  soa802_15_4.nam ;#archivo para ver los nodos
set val(xg) soa802_15_4.xg   ;#archivo para guardar los datos

set stopTime     100              ;#tiempo

# Initialize Global Variables
set ns_   [new Simulator]
set tracefd       [open  /$val(tr) w]
$ns_ trace-all $tracefd
set namtrace      [open  /$val(nam) w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
set tracexg       [open  /$val(xg) w]

$ns_ puts-nam-traceall {nam4wpan}

Mac/802_15_4 wpanCmd verbose on
Mac/802_15_4 wpanNam namStatus on    ;# default = off

# Establecer la topografia
$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

set chan_1_ [new $val(chan)]

# configurar nodo
```

```
$ns_ node-config -adhocRouting $val(rp) \
  -llType $val(ll) \
  -macType $val(mac) \
  -ifqType $val(ifq) \
  -ifqLen $val(ifqlen) \
  -antType $val(ant) \
  -propType $val(prop) \
  -phyType $val(netif) \
  -topoInstance $topo \
  -agentTrace OFF \
  -routerTrace OFF \
  -macTrace ON \
  -movementTrace OFF \
          -energyModel "EnergyModel" \
          -initialEnergy 2 \
          -rxPower 0.3 \
          -txPower 0.3 \
  -channel $chan_1_

for {set i 0} {$i < $val(nn) } {incr i} {
  set node_($i) [$ns_ node]
  $node_($i) random-motion 0
        # deshabilitar el movimiento aleatorio
}

for {set i 0} {$i < $val(nn) } {incr i} {
  set phy_($i) [$node_($i) set netif_(0)]
        # creamos la capa f\´isica de cada nodo
}

# fuente donde se especifican las coordenadas de los nodos
source /n50t100xy50.scn ;

# define el tama\˜no del nodo
for {set i 0} {$i < $val(nn)} {incr i} {
  $ns_ initial_node_pos $node_($i) 2
}

# decirle a los nodos cuando termina
for {set i 0} {$i < $val(nn) } {incr i} {
```

```
    $ns_ at $stopTime "$node_($i) reset";
}

$ns_ at $stopTime "stop"
$ns_ at $stopTime "puts \"\nNS EXITING...\n\""
$ns_ at $stopTime "$ns_ halt"

proc stop {} {
    global ns_ tracefd tracexg namtrace appTime val env
    $ns_ flush-trace
    close $tracefd                    ;#cerramos los archivos
    close $namtrace
    close $tracexg
    exec nam $val(nam)
    exec xgraph -m -tk $val(xg) &
}

$ns_ run
```

Capítulo 5

Conclusión y trabajo futuro

5.1. Conclusión

El uso eficiente de energía es un problema muy importante en redes ad-hoc, existe mucho trabajo en la literatura acerca de esto, ya que la mayoría de los dispositivos inalámbricos son afectados por sus restricciones, tal como; energía y procesamiento, en esta tesis se ha demostrado que las estrategias de auto-organización mejoran el ahorro de energía durante la formación y mantenimiento de la red. El propósito del algoritmo basado en grupos es manejar variaciones en las potencias de transmisión y periodos de transmisión permitiendo un ahorro en la energía y consecuentemente alargar el tiempo de vida de la red. La simulación muestra mejoras substanciales con respecto al algoritmo básico que no implementa los métodos de ahorro de energía.

5.2. Trabajo futuro

El trabajo futuro de esta tesis trata con implementar características adicionales a la formación del algoritmo, tal como; algoritmos de ruteo, asignación de direcciones, asignación de ancho de banda (por ejemplo, cada grupo pude tener diferente ancho de banda dependiendo las necesidades). También estudiar otras posibilidades de reducir el consumo de energía, y buscar una solución optima para resolver el problema de segmentación en la red, es decir, buscar una solución al problema sin gastar demasiada energía en el proceso y que nos asegure que la nueva auto-organización no presentara el mismo problema.

Bibliografía

[/ H07] Springer Berlin / Heidelberg, editor. *Self-organization of Wireless Networks Through Declarative Local Communication (Extended Abstract)*, volume 4806/2007 of *Lecture Notes in Computer Science*. OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, OTM Academy Doctoral Consortium, MONET, OnToContent, ORM, PerSys, PPN, RDDS, SSWS, and SWWS 2007, 2007.

[AJK06] Reza Azarderakhsh, Amir H. Jahangir, and Manijeh Keshtgary. A new virtual backbone for wireless ad-hoc sensor networks with connected dominating set. *Third Annual Conference on Wireless On-demand Network Systems and Services*, pages 191–195, 2006.

[BPEL05] Christian Bettstetter, Christian Prehofer, and DoCoMo Euro-Laps. Self-organization in communication networks: principles and design paradigms. *Communications Magazine, IEEE*, 43(7):78–85, July 2005.

[CLC06] Yao-Chung Chang, Zhi-Sheng Lin, and Jiann-Liang Chen. Cluster based self-organization management protocols for wireless sensor networks. *Consumer Electronics, IEEE Transactions on*, 52(1):75–80, Feb. 2006.

[CLL04] Yuanzhu Peter Chen, Arthur L. Liestman, and Jiangchuan Liu. Clustering algorithms for ad hoc wireless networks. *Ad Hoc and Sensor Networks. Nova Science Publishers*, 2004.

[CMd+07] Luiz H.A. Correia, Daniel F. Macedo, Aldri L. dos Santos, Antonio A.F. Loureiro, and Jose Marcos S. Nogueira. Transmission power control techniques for wireless sensor networks. *Computer Networks*, 51(17):4765–4779, December 2007.

[DAR89] DARPA. The network simulator - ns-2, 1989.

[DE06] Orhan Dagdeviren and Kayhan Erciyes. A distributed backbone formation algorithm for mobile ad hoc networks. In Springer Berlin / Heidelberg, editor,

Parallel and Distributed Processing and Applications, volume 4330/2006 of *Lecture Notes in Computer Science*, pages 219–230. SpringerLink, November 2006.

[DEC06] Orhan Dagdeviren, Kayhan Erciyes, and Deniz Cokuslu. A merging clustering algorithm for mobile ad hoc networks. *International Conference on Computational Science and its Applications*, pages 681–690, May 2006.

[DGH+] Jean E. Dunbar, Jerrold W. Grossman, Johannes H. Hattingh, Stephen T. Hedetniemi, and Alice A. Mcrae. On weakly-connected domination in graphs.

[Dre06] Falko Dressler. Self-organization in ad hoc networks: Overview and classification. Technical report, UNIV. OF ERLANGEN, DEPT. OF COMPUTER SCIENCE 7, Feb 2006.

[FKR+06] Hind Fadil, Jean-Luc Koning, Félix F. Ramos, Jean-Paul Jamont, and Michel Occello. Graphically designing and formally checking self-organizations for wireless network systems. *ITSSA*, 2(3):297–302, 2006.

[Her07] Klaus Herrmann. Self-organizing replica placement - a case study on emergence. pages 13–22. IEEE Computer Society, July 2007.

[HG03] Klaus Herrmann and Kurt Geihs. Self-organization in mobile ad hoc networks based on the dynamics of interaction. *Berlin University of Technology*, pages 1–6, Mar 2003.

[HV07] K. Heurtefeux and F. Valois. Self-organization protocols: Behavior during the sensor networks life. pages 1–5, Sept. 2007.

[JK06] Kil-Woong Jang and Byung-Soon Kim. An adaptive self-organization protocol for wireless sensor networks. In Springer Berlin / Heidelberg, editor, *Euro-Par 2006 Parallel Processing*, volume 4128/2006 of *Lecture Notes in Computer Science*. 12th International Euro-Par Conference, November 2006.

[JO07] Jean-Paul Jamont and Michel Occello. A self-organization process for communication management in embedded multiagent systems. pages 55–58, Nov. 2007.

[KA06] T. Kirt and A. Anier. Self-organization in ad hoc wireless networks. pages 1–4, October 2006.

[LGF08] Mohamed Lehsaini, Herve Guyennet, and Mohammed Feham. A novel cluster-based self-organization algorithm for wireless sensor networks. pages 19–26, May 2008.

[LL05] Jain-Shing Liu and Chun-Hung Richard Lin. Energy-efficiency clustering protocol in wireless sensor networks. *Ad Hoc Networks*, pages 371–388, May 2005.

[LSM06] Ou Liang, Y.A. Sekercioglu, and Nallasamy Mani. Gateway multipoint relays-an mpr-based broadcast algorithm for ad hoc networks. In *International Conference on Communication systems, 2006. ICCS 2006. 10th IEEE Singapore*, pages 1–6, 30 2006-Nov. 1 2006.

[LVB07] Jia-Laing Lu, Fabrice Valois, and Dominique Barthel. Low-energy self-organization scheme for wireless ad hoc sensor networks. pages 138–145, Jan. 2007.

[LVBD07] Jia-Liang Lu, Fabrice Valois, Dominique Barthel, and Mischa Dohler. Fisco: A fully integrated scheme of self-configuration and self-organization for wsn. pages 3370–3375, March 2007.

[Mil07] Kevin L. Mills. A brief survey of self-organization in wireless sensor networks. *WIRELESS COMMUNICATION AND MOBILE COMPUTING*, pages 823–834, May 2007.

[OXZ04] S. Olariu, Q. Xu, and A.Y. Zomaya. An energy-efficient self-organization protocol for wireless sensor networks. pages 55–60, Dec. 2004.

[QVL02] Amir Qayyum, Laurent Viennot, and Anis Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences, HICSS*, pages 3866–3875, Jan. 2002.

[VM06] Abhinay Venuturumilli and Ali Minai. Obtaining robust wireless sensor networks through self-organization of heterogeneous connectivity. *International Conference on Complex Systems (ICCS)*, pages 1–9, 2006.

[WCJcW07] Yong Wu, Zhong Chen, Qi Jing, and Yong cai Wang. Leno: Least rotation near-optimal cluster head rotation strategy in wireless sensor networks. pages 195–201, May 2007.

[Wor08] World Academy of Science, Engineering and Technology. *An energy-efficient Distributed Unequal Clustering Protocol for Wireless Sensor Networks*, December 2008.

[YMM06] Mengjie Yu, Hala Mokhtar, and Madjid Merabti. A survey of network management architecture in wireless sensor network. *Liverpool John Moores University*, May 2006.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Auto-Organización de Redes de Dispositivos Móviles

del (la) C.

J. Guadalupe OLASCUAGA CABRERA

el día 26 de Agosto de 2009.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Mario Angel Siller González
Pico
Investigador CINVESTAV 2A
CINVESTAV Unidad Guadalajara

Dr. Andrés Méndez Vásquez
Investigador CINVESTAV 2A
CINVESTAV