FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

Tesis de Magíster en Ingeniería Electrónica y de Computadores

SIMULACIÓN DE PROTOCOLOS DE RED EFICIENTES EN ENERGÍA PARA REDES INALÁMBRICAS DE SENSORES.

Presentado por:

Carlos Eduardo Silva Martínez

Director:

Néstor Misael Peña Traslaviña Ph.D.

Bogotá, Agosto de 2007.

AGRADECIMIENTOS

A Dios porque nos ha dado la capacidad de aprender y perseverar ante situaciones adversas.

Al profesor Néstor Peña porque su apoyo y la confianza depositada en mí fueron determinantes para culminar exitosamente este proyecto.

A mi esposa Jeannette y mis hijos Luisa María y Manuel Eduardo por ser mis motivadores constantes y por su comprensión al permitirme utilizar gran parte de nuestro tiempo para dedicarlo al proyecto.

A mis padres y hermanos que me animaron constantemente y que impidieron que me diera por vencido.

A mis familiares y amigos porque la confianza que han depositado en mí ha sido un motivador para seguir adelante.

A mis compañeros de Colombia Móvil porque su soporte ha sido valioso para atender de manera simultánea el proyecto y mis labores dentro de la compañía.

Resumen.

La gran cantidad de aplicaciones que se han propuesto para las redes inalámbricas de sensores (WSN) y los retos que estas aplicaciones representan para el diseño de sistemas de comunicación eficientes, han generado un gran movimiento investigativo en torno a los protocolos de comunicación para estas redes. En el presente documento se presentan los resultados obtenidos en la investigación realizada sobre el estado del arte de los protocolos de nivel de red eficientes en energía utilizados en las redes inalámbricas de sensores. Se empieza con una introducción general a las redes inalámbricas de sensores, su arquitectura, sus principales características y aplicaciones siempre teniendo en cuenta el consumo de energía como aspecto principal, para posteriormente enfocarse en el estudio de los protocolos de enrutamiento y en el análisis de los resultados obtenidos a través de la simulación del protocolo IPOW el cual se implementó completamente en ns-2. Finalmente se validan los resultados de la simulación de IPOW con los resultados obtenidos para otros protocolos de enrutamiento para redes inalámbricas de sensores reportados en la literatura.

CONTENIDO

Capítulo 1. Introducción.

A pesar de los continuos avances tecnológicos y la abundante investigación en torno a las redes inalámbricas de sensores, la optimización en el consumo de energía en estas redes sigue siendo un aspecto que restringe importantemente su vida útil por lo que se sigue manteniendo como un tema de investigación con un elevado dinamismo tanto a nivel académico como industrial.

En el presente documento se presentan los resultados de la investigación realizada sobre el consumo de energía en redes inalámbricas de sensores (WSN), que empieza con la revisión del estado del arte e incluye el estudio de los diferentes aspectos desde los que se aborda la optimización en el consumo de energía en dichas redes, centrándose en los algoritmos de enrutamiento, para concluir con el desarrollo de módulos software que permitan simular el funcionamiento de los principales protocolos de enrutamiento diseñados para redes inalámbricas de sensores, y a partir de dichas simulaciones obtener conclusiones referentes al desempeño de los algoritmos al compararlos con resultados reportados en la literatura sobre el tema.

En el primer capítulo se hace una introducción a las redes inalámbricas de sensores, sus componentes, características y principales aplicaciones. En el segundo capítulo se revisa el tema de consumo de energía enfocado inicialmente a los módulos constitutivos del sensor y posteriormente a su incidencia en diferentes aspectos del funcionamiento de la red tomando como referencia para el análisis la pila de protocolos OSI. En el tercer capítulo se estudia el funcionamiento de los protocolos de control de acceso al medio y de enrutamiento eficientes en consumo de energía que se han propuesto para las redes inalámbricas de sensores. En el cuarto capítulo se describen las principales características del simulador utilizado en la investigación y los desarrollos que se requirieron para realizar las simulaciones, así como los escenarios y parámetros utilizados en la simulación. En la parte final de este mismo capítulo se presentan los resultados de la simulación y se comparan con resultados reportados en la literatura. En el capítulo número cinco se presentan las conclusiones obtenidas a partir de la investigación.

1.1 Componentes de una red inalámbrica de sensores.

Una red inalámbrica de sensores ad hoc se define como un sistema autónomo de nodos sensores que, además de captar algún tipo de información del medio en el que se han inmerso, actúan como enrutadores conectados de forma inalámbrica para transmitir hacia un destino predefinido la información captada del medio por los sensores.

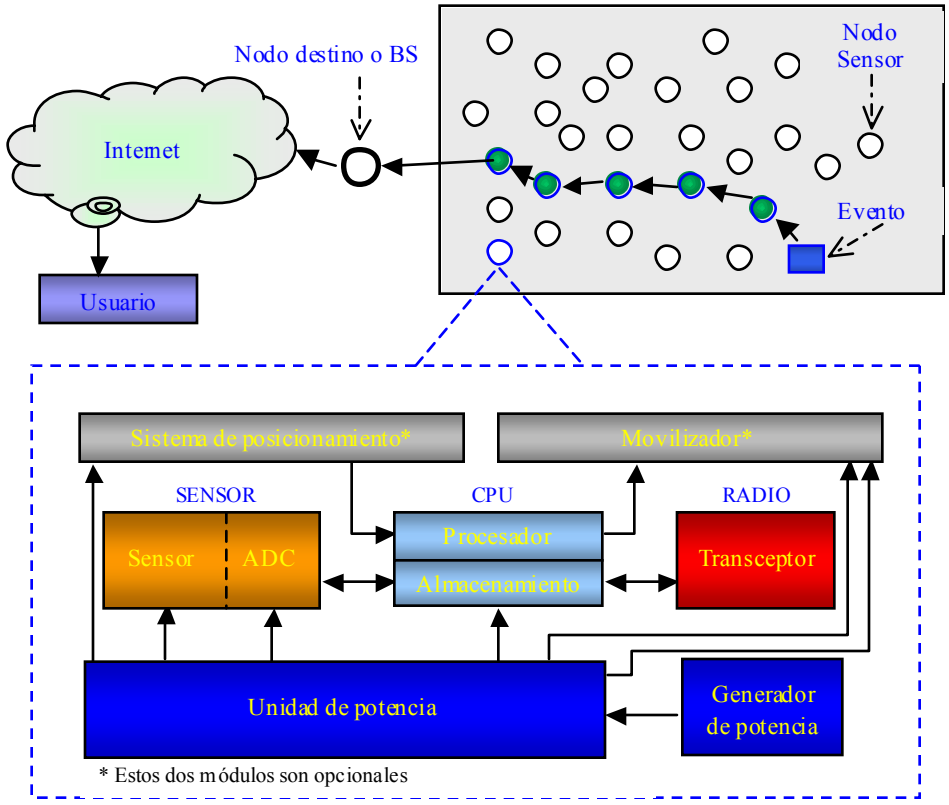

Fig. 1. Esquema de una red de sensores y módulos componentes de un sensor (tomado de [1])

En la parte superior de la Fig. 1 tomada de [1] se puede observar el esquema general de una red inalámbrica de sensores para monitoreo de ambientes en la que a partir de la detección, por parte de un sensor, de un evento en alguna parte del medio a monitorear, se generan una serie de procesos que culminan con la transmisión de información sobre dicho evento hacia un nodo predefinido como nodo destino (sink) o estación base (BT). Este último a su vez se encarga de concentrar información enviada desde los sensores y hacer las veces de pasarela de información hacia una red diferente, como la Internet por ejemplo, a la que se puede conectar el usuario final de la información obtenida por la red inalámbrica de sensores. A

pesar de que existen configuraciones de redes inalámbricas de sensores en las que se tienen múltiples destinos o nodos móviles, en el presente trabajo sólo tendremos en cuenta redes con nodos fijos y con un único destino.

En la parte inferior de la Fig. 1 se muestra el esquema funcional de un nodo sensor en el que se pueden observar los módulos constitutivos del nodo: la CPU que controla el funcionamiento de los demás módulos; el generador de potencia que alimenta la unidad de potencia encargada de administrar la distribución de energía a los demás módulos; el sensor propiamente dicho que se encarga de detectar los eventos en el medio y de realizar la conversión de analógico a digital necesaria para que la CPU pueda almacenar la información y procesarla; el módulo de radio encargado de transmitir y recibir información hacia y desde otros nodos; y dos módulos que no siempre se requieren en los nodos: el movilizador y el sistema de posicionamiento que brindan movimiento al sensor dentro del medio y entregan información sobre la ubicación del nodo respectivamente.

1.2 Aplicaciones

Las aplicaciones de redes inalámbricas de sensores son diversas y abarcan ámbitos como el militar, el de atención de desastres y emergencias; así como el monitoreo de ambientes [1], [2] y [3]. En [4] definen la siguiente expresión como una forma de especificar la gran cantidad de aplicaciones que este tipo de redes pueden tener:

Unidad de sensado + CPU + Unidad de radio = Miles de aplicaciones potenciales

A continuación enumeramos algunos de los ejemplos más representativos de las aplicaciones de las redes inalámbricas de sensores:

- Aplicaciones militares
    - Vigilancia de sitios
    - Monitoreo de campos minados
    - Seguimiento

- Emergencias
    - Prevención de incendios
    - Cuidado de la salud
    - Atención de desastres
- Monitoreo de ambientes
    - Monitoreo de fauna
    - Monitoreo del clima
    - Monitoreo de terrenos
    - Monitoreo de fábricas
    - Monitoreo de tráfico
- Seguimiento de sensores
    - Seguimiento de personas
    - Control de inventario

El tipo de aplicación es un factor determinante en cuanto a las restricciones que deben tenerse en cuenta para el diseño de los protocolos a utilizar, pues afectan aspectos como necesidades de sincronización, tolerancia a retardos, tipos de topologías y prioridad en el consumo de energía.

1.3 Características

Las redes inalámbricas de sensores son un tipo de redes ad hoc que presentan algunas características y aplicaciones que las hacen particulares y que como tal implican requerimientos que las redes adhoc tradicionales no presentan. A continuación se enumeran y definen sus principales características [5]:

- Comunicación inalámbrica a través de múltiples saltos: la comunicación desde la fuente hacia el destino, que generalmente está fuera del alcance de muchos nodos, se realiza mediante múltiples enlaces inalámbricos a través de otros nodos de la red, lo cual a su vez representa menor consumo de energía para la transmisión hacia el destino comparado con la transmisión directa de fuente a destino.

- Operación eficiente en energía: se requiere de técnicas que permitan el uso eficiente de la energía, escasa debida los pequeños espacios disponibles para la batería, y necesarios para soportar la operación de la red por períodos de tiempo del orden de varios meses sin posibilidad de recarga debido a los ambientes muchas veces hostiles en los que se despliegan los nodos.
- Auto configuración: los nodos deben ser capaces de adaptar sus parámetros de operación para soportar el dinamismo del ambiente representado en la aparición de fallas y obstáculos, así como en el ingreso y salida de nodos de la red.
- Colaboración y procesamiento en red: la naturaleza de algunas aplicaciones implica la interacción de grupos de sensores para detectar un evento, realizar procesamiento conjunto de señales como es el caso de la agregación de datos, o para el agrupamiento de nodos con el objetivo de ahorrar energía.

1.4 Proyectos de investigación

Desde la implementación de los primeros sensores con capacidad de comunicación inalámbrica construidos con base en dispositivos electrónicos genéricos (Common Off The Shelf –COTS-), han existido diferentes grupos de investigación tanto académicos como industriales de los que han resultado plataformas hardware, software y algoritmos diseñados específicamente para las redes inalámbricas de sensores [6] y de los cuales se enumeran a continuación algunos de los más importantes:

El proyecto Wireless Integrated Network Sensors (WINS) [6] de la Universidad de California en Los Angeles (UCLA) que se inició en 1993 y que utiliza radios de baja potencia y tecnología CMOS para implementar los demás módulos del nodo. Aunque este proyecto ya terminó, la misma universidad continúa con la investigación sobre las redes inalámbricas de sensores [7].

El proyecto Smart Dust [8] en la Universidad de California en Berkeley (UCB) que utiliza nodos de tamaño milimétrico en los que la comunicación se realiza a través de

comunicaciones ópticas. Este proyecto ya finalizó pero la universidad continúa con la investigación sobre las redes inalámbricas de sensores [9]

El proyecto Ultra Low Power Wireless Sensor [10] del Instituto Tecnológico de Massachussets (MIT) que utiliza comunicación por saltos múltiples a través de radios con potencias del orden de decenas de micro Watios y que soportan velocidades de transmisión de entre 1 bit por segundo y 1 Megabit por segundo.

El proyecto micro-Adaptative Multidomain Power-Aware Sensors (μAMPS) [11], también de Instituto Tecnológico de Massachussets en el que la investigación se enfoca hacia las técnicas de almacenamiento de potencia, comunicación y procesamiento de señales a través de la utilización de soluciones de hardware programables.

El proyecto PicoRadio [12] de el Berkley Wireless Research Center que involucra áreas de investigación en circuitos de RF, antenas y circuitos digitales de bajo voltaje operados por baterías solares.

El proyecto GNOMES [13] de la Universidad de Rice en el que se centran en el diseño y desarrollo de una plataforma hardware y software de bajo costo para redes de sensores heterogéneas.

El proyecto Autonomous Networks Research Group [14] de la Universidad del Sur de California que se enfoca en la investigación y desarrollo de algoritmos para configuración y procesamiento de información en redes inalámbricas de sensores.

Proyecto de investigación en redes de sensores inalámbricos del Grupo de Electrónica y Sistemas de Telecomunicaciones (GEST) de la Universidad de los Andes.

En lo referente a plataformas hardware, además de los fabricados con base a dispositivos electrónicos genéricos se han desarrollado plataformas como [4]: weC, Rene, Dot, Mica y Spec entre otros.

Capítulo 2. Consumo de energía en redes de sensores

Debido a los avances en dispositivos MEMs [2], los sensores han llegado a ser elementos de pequeños tamaños y bajos costos que involucran todas las funcionalidades previamente enunciadas y requeridas para trabajar de manera autónoma. Características como el pequeño tamaño y su funcionamiento autónomo hacen que se reduzca el espacio para las baterías y que estas no puedan fácilmente reemplazarse por lo que en el diseño de los nodos sensores se da mucha importancia a la optimización en el consumo de energía en cada uno de sus módulos constitutivos. Medidas similares se toman a nivel del sistema por lo que el diseño de la red de sensores involucra estrategias de ahorro de energía en prácticamente todos los niveles de la pila de protocolos OSI.

A diferencia de dispositivos como los celulares, en los que el consumo de potencia es del orden de cientos de miliamperios y los tiempos de vida de las baterías antes de una recarga son de unos cuantos días, en los nodos sensores el consumo de potencia es del orden de los microamperios y los tiempos de vida de las baterías deben ser del orden de varios meses y sin posibilidad de recarga de las mismas. Además, en redes basadas en infraestructura como las celulares, la gestión de energía se basa también en el aumento de potencia de las estaciones base fijas que tienen la posibilidad de alimentación de energía permanente, situación que no es posible con las redes inalámbricas de sensores.

Como se mencionó previamente, el presente trabajo está orientado al estudio del consumo de energía en las redes inalámbricas de sensores por lo que debemos revisar las técnicas utilizadas para administración de la energía que minimizan los posibles efectos adversos en el desempeño de la red representados en latencias elevadas y capacidades muy bajas para los requerimientos de las aplicaciones. Para esta revisión es primordial conocer como está distribuido el consumo de energía tanto en la red como en los mismos componentes de los nodos sensores, para así mismo comprender los diversos enfoques y técnicas utilizadas para reducir el consumo.

2.1 Consumo de energía en los módulos del nodo sensor

Los módulos que componen el sensor y que se presentan en la Fig. 1, pueden trabajar en diversos modos de operación y cada modo representa unos niveles de consumo de energía distintos lo que se ha determinado a través de medidas [2].

Modo de operación	Consumo de corriente en Sensor (mA)	Consumo de corriente en CPU (mA)	Consumo de corriente en radio (mA)	Consumo de corriente Total (mA)
Transmisión	0,52	8	25,4	**33,92**
Recepción	0,52	8	12,5	**21,02**
Inactivo (sleep)	<0,001	<0,001	0,001	**<0,003**

Tabla 1. Consumo de corriente en módulos de un nodo sensor

En la tabla 1 obtenida a partir de valores reportados en [6] se presentan los consumos de corriente en los diferentes modos de operación para el módulo de radio con un chip CC1000 de CrossBow; para el módulo de procesamiento de un nodo Mica2 y para el módulo de sensado. Para este último tomamos el valor promedio porque el consumo de los sensores varía dependiendo del tipo de sensor como se puede observar en la tabla 2 [6].

Tipo de sensor	Corriente (mA)
Magnetómetro	0,65
Luz	0,2
Temperatura	0,6
Presión	0,65
Promedio	**0,525**

Tabla 2. Consumo de corriente en distintos tipos de sensores

De lo anterior se evidencia que el módulo con mayor consumo de energía es el de radio, seguido por el de CPU y el de sensado. En modo transmisión el módulo de radio consume el 74,8% y el de CPU el 23,5% mientras que en recepción el de radio consume el 59,4% y el de CPU el 38,0%. Aun cuando el módulo de radio no esté transmitiendo ni recibiendo, si no se inactiva consumirá la misma corriente que si estuviera recibiendo e igual sucede con el módulo de procesamiento. Dada la elevada participación del módulo de radio en el

consumo de energía del nodo sensor, la inactivación de su módulo radio se convierte en una de las más importantes estrategias a utilizar para la optimización del consumo de energía. Además de los consumos en cada modo, también es importante el tiempo que tarda cada módulo en cambiar de estado inactivo a activo (tiempo para despertar), pero el más influyente en el desempeño es el que le toma al procesador.

La vida útil del nodo sensor está dada por el consumo de energía de los distintos módulos que los constituyen y por la energía total disponible en su batería. La capacidad de una batería se expresa en Amperios por hora (Ah), que es una métrica comúnmente utilizada por los fabricantes para especificar la capacidad teórica total de una batería. En [2] utilizan la expresión $T = C/I$ para calcular el tiempo de vida teórico de una batería; aquí T es el tiempo de vida de la batería, C es su capacidad máxima en *Ah*, e I es la corriente de descarga en *A*. De esta forma, una batería de 1000 mAh podría alimentar durante 10 horas a un dispositivo que consuma 100 mA. En [4] se presenta un breve análisis de los tipos de baterías que se utilizan en las redes inalámbricas de sensores.

En [2] proponen tres modelos diferentes de batería que dependen de la forma en que disminuye la corriente que esta tiene almacenada. El primero es un modelo líneal en el que la capacidad máxima disminuye constantemente en función del tiempo y la cantidad de corriente utilizada y no depende de la velocidad de descarga. El otro modelo considera el efecto de la tasa de descarga por lo que la capacidad máxima teórica se ve disminuida por un factor *k* que depende de la velocidad de descarga. El tercer modelo considera el efecto de relajación que exhiben las baterías, por el que al mantener una elevada tasa de descarga por un tiempo, la batería llegará al fin de su vida a pesar de que aún disponga de materiales activos; mientras que si la corriente de descarga se elimina o se disminuye durante el período de descarga el agotamiento de la batería se realizará en cuanto se agoten sus ingredientes activos. Los mismos autores de [2] detallan en [15] las medidas y montajes de laboratorio en los cuales se basaron para proponer los tres modelos de batería resumidos anteriormente.

2.2 Consumo de energía en la red

Para lograr el ahorro de energía durante la interacción entre los nodos de la red, se utilizan técnicas y estrategias en diversos niveles funcionales de la capa de protocolos. Estas técnicas incluyen la agregación y fusión de datos, la disminución de ciclos útiles, la transmisión por múltiples saltos, la dispersión de tráfico y el agrupamiento entre las más importantes.

La agregación de datos [1] saca provecho del hecho de que los módulos de cómputo consuman menos potencia que los de comunicaciones y se logra mediante la combinación en un nodo, de datos provenientes de diversas fuentes, eliminando la posible información redundante y utilizando funciones de minimización, maximización o promediado. Cuando se utilizan métodos de procesamiento digital de señales para la agregación, esta se conoce como fusión de datos [1] y le permite a un nodo generar una señal de salida precisa a través técnicas como beamforming para combinar las señales de entrada reduciendo el ruido que estas puedan tener.

La disminución de los ciclos útiles busca disminuir el consumo de energía en los nodos sensores mediante su desactivación o la de algunos de sus módulos, sin impactar significativamente el desempeño de la red según lo requiera una aplicación determinada.

La transmisión por múltiples saltos busca evitar la transmisión de señales sobre largas distancias. Para esto se reemplazan los saltos largos únicos por múltiples saltos entre los nodos fuente y destino, de forma tal que la suma de las potencias consumidas en los múltiples saltos sea menor que la potencia requerida por el salto único entre fuente y destino.

La dispersión de tráfico [1] apunta al aplanamiento del tráfico entre los nodos de la red para evitar que algunos nodos se utilicen más frecuentemente que otros en el re-envío de mensajes entre fuente y destino, y por lo tanto puedan generar tempranamente zonas descubiertas debidas al agotamiento de energía en grupos de nodos aislados. Si una misma

fuente desea transmitir repetidamente información hacia el destino, la dispersión de tráfico haría que para cada envío se utilicen nodos intermedios distintos para el re-envío de la información.

El agrupamiento (clustering) [1] es un esquema mediante el cual los nodos se organizan en grupos coordinados por un líder de grupo (cluster head) que controla los períodos de actividad e inactividad de los demás nodos, de forma tal que pueda recibir la información de cada nodo, aplicarle procesos de agregación o fusión de datos y enviarla al destino.

A continuación revisamos el consumo de energía en las redes inalámbricas de sensores desde las funciones de las primeras capas de la pila de protocolos OSI.

2.2.1 Nivel físico

A nivel físico la gran mayoría de nodos sensores utilizan la transmisión por radio, aunque algunos como SmartDust [8] proponen la utilización de comunicaciones ópticas. En nuestro trabajo de simulación tendremos en cuenta la transmisión por radio.

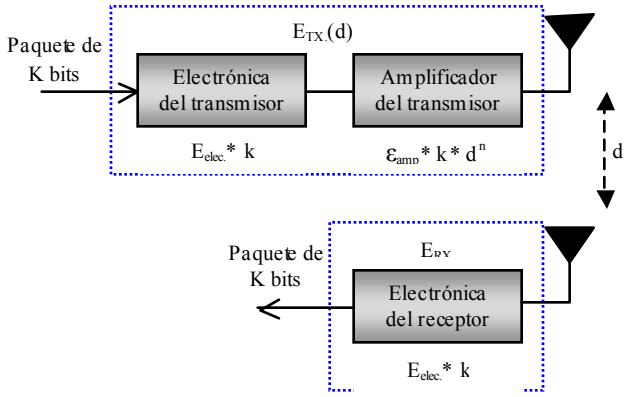

Fig. 2. Modelo de radio (tomado de [17])

La comunicación de información a través de radio frecuencias requiere la transmisión y recepción de señales a una determinada frecuencia, lo que implica un consumo de energía en los dispositivos electrónicos del transmisor y el receptor, así como el consumo por pérdidas de energía debidas al canal inalámbrico. El modelo propuesto en [17] que se puede

observar en la Fig. 2 es uno de los más utilizados para representar el consumo de energía en la comunicación por radio. Según dicho modelo, para transmitir k bits a través de una distancia d el módulo de radio consume

$$E_{TX}(k,d) = E_{TX-elec}(k) + E_{TX-amp}(k,d)$$
$$E_{TX}(k,d) = E_{elec} * k + \varepsilon_{amp} * k * d^2$$

Y para recibir el mismo mensaje el radio consume:

$$E_{RX}(k) = E_{RX-elec}(k) = E_{elec} * k$$

De tal forma que considerando el consumo de transmisión y recepción, el consumo de energía para transmitir k bits a una distancia d estará compuesta por una parte independiente de la distancia correspondiente al consumo en los dispositivos electrónicos y otra dependiente de la distancia:

$$E(k,d) = 2 * k * E_{elec} + k * \varepsilon_{amp} * d^2$$

Los valores de consumo de energía debidos a la electrónica son [17] $E_{elec.}$ = 50nJ/bit y ε_{amp} = 100 pJ/bit/m^2. Con estos valores es evidente que la parte más influyente en el consumo de potencia del módulo de radio es la que depende de la distancia por lo que en las redes inalámbricas de sensores se prefiere utilizar rutas a través de múltiples saltos realizadas a través de nodos con potencias de transmisión muy bajas en lugar de un único salto de larga distancia. Adicionalmente también es evidente que debe transmitirse la menor cantidad de bits posible, las medidas reportadas en [2] muestran que un incremento en la velocidad de transmisión de datos implica mayor consumo de energía.

Para la comunicaciones por radio las bandas de frecuencia más acordes a la restricciones que imponen las redes inalámbricas de sensores son las que están en las bandas de los 900 MHz y los 2,4 GHz [6]. Las técnicas de transmisión más utilizadas son las de espectro ensanchado por secuencia directa DSSS y los tipos de modulación dependen de la

plataforma de sensores que se utilice e incluyen técnicas como OOK para WeC y René, ASK para Mica, FSK para Mica2 y O-QPSK para Telos. En la tabla 3 se presentan los valores típicos de los principales parámetros de nivel físico de una red inalámbrica de sensores.

Parámetro	Rango de valores típicos
Velocidad de tx	entre 10 y 250 Kbps
Frecuencia de operación radio	entre 800 y 2400 MHz
Consumo de potencia en recepción	entre 12 y 71 mW
Radio de cobertura de un nodo	entre 1 y 300 m
Consumo de potencia en transmisión	entre 38 y 75 mW
Tiempo para despertar (wakeup)	entre 6 y 10000 us
Tipo de técnica de transmisión	DSSS
Tipo de modulación	OOK, ASK, FSK, BPSK, O-QPSK

Tabla 3. Valores típicos de parámetros físicos

A nivel de capa física, en [18] presentan los resultados del diseño e implementación de una red inalámbrica de sensores para monitoreo de cultivos en la que se basan en las especificaciones del estándar 802.15.4 utilizando DS-SS a una frecuencia de 2,4 GHz con modulación O-QPSK.

2.2.2 Nivel de enlace

Las redes de sensores difieren de las redes inalámbricas de voz y datos para las que se han diseñado los protocolos MAC más conocidos y utilizados actualmente, razón por la que se ha requerido el diseño de protocolos MAC específicos para redes inalámbricas de sensores. Las principales diferencias entre estas redes y las redes inalámbricas de voz y datos, que pueden afectar el desempeño de un protocolo de control de acceso al medio son [19]: los nodos funcionan con baterías no reemplazables, los nodos deben organizar ellos mismos la red de comunicaciones debido a que su despliegue no se realiza de manera planeada sino de una forma ad-hoc, la densidad de nodos es muy variable pues en algunos sectores hay muchos nodos y en otros hay muy pocos, la gran mayoría del tráfico en redes de sensores se genera por eventos y puede ser extremadamente variable.

Según [19] las principales fuentes de desperdicio de energía que se pueden controlar en el nivel MAC son: la colisión, la escucha inútil (idle listening), la escucha casual (overhearing) y el control de encabezados de paquetes (control packet overhead). La preponderancia de uno de estos fenómenos sobre los demás en el consumo de energía dependen del tipo de aplicación en la que se esté utilizando la red.

La colisión representa gasto adicional de energía debido a las retransmisiones que deben hacerse, por lo que es crucial atacar el problema del terminal oculto que se puede observar en la Fig. 3 y que se presenta cuando de tres nodos vecinos 1, 2 y 3, hay dos, nodo 1 y nodo 3, que están fuera del alcance (están ocultos) uno del otro y ambos intentan transmitir simultáneamente hacia el nodo 2 generando alta probabilidad de colisión.

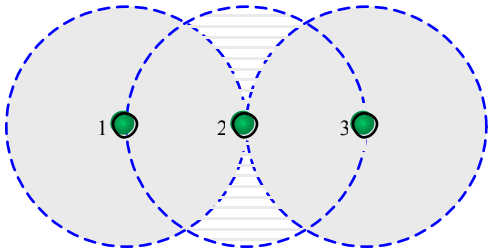

Fig. 3. Problema del terminal oculto

La escucha inútil se presenta cuando el módulo radio está escuchando el canal en espera de posibles datos y representa un desperdicio de energía cuando no hay nada que transmitir; la escucha casual se presenta cuando un nodo recibe paquetes que están dirigidos hacia otros nodos; y el control de encabezados de paquetes representa gastos de energía en datos que no llevan información sobre los fenómenos sensados.

La eficiencia en energía de los protocolos MAC se logra mediante la reducción del consumo en los nodos utilizando mensajes de nivel MAC para intercambiar información que permita inactivar los módulos radio durante el mayor tiempo que sea posible (bajos ciclos de payload) y sin afectar notoriamente el desempeño de la red en lo referente a latencia y capacidad.

En la sección 3.1 se detallan las principales características de los protocolos de nivel de acceso al medio diseñados específicamente para redes inalámbricas de sensores y que buscan el ahorro de energía mediante la minimización de los efectos de los tres factores enunciados previamente.

2.2.3 Nivel de red

Al disminuir el tamaño de los nodos sensores que integran todas las funcionalidades requeridas para operar, y debido a los ambientes en que se despliegan y a las aplicaciones en que se utilizan, las redes inalámbricas de sensores presentan elevadas restricciones en cuanto a capacidad de procesamiento, almacenamiento de datos, y consumo de energía a pesar de los avances tecnológicos alcanzados.

Las características de propagación por radio y las limitaciones en energía, inversamente proporcionales al pequeño tamaño de los nodos sensores, inciden directamente en la operación de estas redes, haciendo que la transmisión de datos hacia el destino se deba realizar a través de múltiples saltos de comunicación entre los nodos hasta llegar al destino, en lugar de hacerse de forma directa desde cada sensor hacia el destino lo que requeriría de altas potencias de transmisión. Este escenario de transmisión a través de múltiples saltos conlleva a la necesidad de algoritmos de enrutamiento que, además de permitir la transferencia de la información, utilicen de manera eficiente los escasos recursos de procesamiento y energía disponibles en los nodos.

En las redes tradicionales los nodos poseen unas direcciones globales únicas, pero en las redes inalámbricas de sensores dicho esquema sería demasiado costoso, pues debido a la gran cantidad de nodos que pueden conformar la red y al pequeño tamaño de la información que generalmente se requiere enviar desde los nodos, las direcciones pueden representar la mayor parte de la información a transmitir generando una muy baja eficiencia.

Así como el consumo de energía, las limitadas capacidades de cómputo de los nodos sensores restringen notoriamente la complejidad de los protocolos de enrutamiento que se pueden utilizar satisfactoriamente por lo que, al igual que con los protocolos de nivel MAC, se han propuesto protocolos de enrutamiento específicos para redes inalámbricas de sensores que permitan el descubrimiento de rutas y el reenvío de los datos hacia el destino.

Los protocolos de enrutamiento que buscan eficiencia en el consumo de energía, tratan de maximizar la vida útil de la red balanceando el consumo de energía entre todos sus nodos, seleccionando los nodos con mayor energía disponible, los que requieren menor potencia de transmisión, los caminos más cortos o una combinación de las alternativas anteriores [6]. La mayoría de protocolos de enrutamiento para redes de sensores asumen que los nodos son estacionarios y es lo que asumiremos en el presente documento.

Capítulo 3. Protocolos eficientes en energía en redes de sensores

A diferencia de las redes inalámbricas de sensores, en las redes móviles y ad hoc el consumo de energía es un factor de diseño importante, pero no es la consideración primaria porque en estas últimas las baterías pueden reemplazarse fácilmente.

Como se ha mencionado previamente, la eficiencia en el consumo de energía se debe procurar en todos los niveles de la pila de protocolos. Un protocolo eficiente en el aspecto de gestión de energía debe minimizar el consumo de energía del módulo radio de los sensores, a la vez que minimiza su impacto sobre el desempeño de la red. En el diseño de protocolos que minimicen el consumo de energía debe tenerse en cuenta: el tipo de aplicación para la que se desplegará la red, la topología y su dinámica, la capacidad de los nodos, las políticas de agregación o fusión de datos, además del manejo de posibles fallas en los nodos [20].

Los protocolos de nivel MAC y los protocolos de enrutamiento son complementarios en lo referente al ahorro de energía; por lo que los ahorros en energía que se logren en nivel MAC y los que se puedan lograr en el nivel de red se suman generando un ahorro total de energía por lo que el efecto de la utilización de un protocolo

A continuación se hace una revisión de los protocolos de los niveles dos y tres que se han diseñado específicamente para redes inalámbricas de sensores.

3.1 Protocolos de control de acceso al medio

A continuación se presentan los principales protocolos eficientes de energía que existen en el nivel MAC.

S-MAC. Sensor MAC [19] es un protocolo basado en contención en el que la reducción en el consumo de energía se logra mediante el control de las cuatro principales fuentes de desperdicio de energía enunciadas previamente. La escucha inútil se evita mediante un

esquema de escucha e inactivación periódicas para reducir el consumo de energía formando grupos virtuales de vecinos para sincronizarse [21] y disminuir el ciclo útil. Para evitar el problema de escucha casual se utiliza señalización común (in-channel signaling) con la que se controla que unos nodos estén inactivos mientras alguno está transmitiendo. Para minimizar el control de los encabezados se utiliza el paso de mensajes que reduce la latencia percibida por las aplicaciones.

El protocolo usa un ciclo de activad/inactividad de baja resolución para permitir que los nodos estén inactivos la mayor parte del tiempo. A diferencia de PAMAS, S-MAC sólo utiliza un canal que se comparte entre la transmisión de datos y la señalización.

El mecanismo de contención de S-MAC es el mismo utilizado en las redes 802.11, es decir, se logra mediante la utilización de paquetes RTS y CTS, el nodo que primero envíe un paquete RTS gana el acceso al medio y el receptor debe responder con un CTS y una vez que se inicia la transmisión los dos nodos no participan en la inactivación periódica hasta que no se haya terminado la transmisión [21].

El funcionamiento del protocolo S-MAC está constituido por dos fases: la fase de agendamiento y la de transmisión de datos.

En la primera fase el protocolo garantiza el mantenimiento de los nodos en unos ciclos útiles bajos de entre el 1 y el 10% [19]. Todos los nodos tienen libertad de elección sobre sus agendas de activad/inactividad y las comparten con sus vecinos quienes las almacenan en una tabla de agendas para que sea posible la comunicación entre todos los nodos, de tal forma que los nodos programan sus transmisiones cuando los destinos de su información están escuchando. Para prevenir errores de temporización debidos a las fluctuaciones de largo plazo del reloj, cada nodo difunde periódicamente su agenda en un paquete SYNC con lo que se hace una sincronización sencilla del reloj. Para minimizar costos por control de encabezados, S-MAC fomenta entre los nodos vecinos la adopción de agendas idénticas. Adicionalmente, los nodos periódicamente escuchan una trama completa para descubrir nodos con diferentes agendas.

El proceso de agendamiento se realiza en tres pasos [21]:

1. Un nodo empieza escuchando por una cierta cantidad de tiempo después de la cual, si no escucha la agenda de ningún otro nodo, el selecciona aleatoriamente un tiempo después del cuál se inactivará e inmediatamente difunde su agenda. Este nodo cumple el papel de sincronizador.
2. Si antes de seleccionar su propia agenda, un nodo escucha la de un vecino, él sigue la del nodo vecino por lo que cumple el papel de seguidor. Este nodo espera por un tiempo aleatorio antes de difundir su agenda.
3. Si un nodo recibe la agenda de un vecino pero ya había seleccionado una agenda propia, dicho nodo adoptará las dos agendas de forma tal que se activará en los tiempos especificados por ambas agendas. Antes de inactivarse, el nodo difundirá la agenda que escogió como propia.

Para la fase de transmisión de datos, S-MAC utiliza la evasión de colisiones mediante el sensado físico y el sensado virtual de portadora [19]. Para indicar la duración de cada transmisión, el protocolo agrega a cada paquete un campo de duración de transmisión, de tal forma que si un nodo recibe un paquete de la fuente o el destino, él sabe cuanto tiempo debe permanecer en silencio para no interrumpir la transmisión a la vez que se ahorra energía pues se inactiva por el tiempo de silencio.

Para disminuir la latencia que se puede generar por los bajos ciclos útiles, S-MAC utiliza la audición adaptiva en la que, en lugar de esperar hasta el siguiente intervalo programado para escuchar, los vecinos se activan inmediatamente después de que la transmisión termina.

Según el modelo analítico de [3], el límite mínimo de consumo de energía en S-MAC se obtiene cuando el ciclo útil es del 20%

PAMAS Power Aware Multi-Access protocol with Singalling for Ad Hoc Networks [22]. Está basado en el protocolo MACA, y como se puede deducir de su nombre, no se diseñó específicamente para redes de sensores sino en general para redes Ad Hoc. El protocolo

basa la optimización de energía en minimizar la escucha casual [19] desactivando los nodos cuando sus vecinos están transmitiendo. PAMAS al igual que MACA utiliza un canal para datos y otro para control. Una vez que un nodo se activa, él sondea el canal de control para saber si hay alguna transmisión en curso. Si algún vecino responde al sondeo, el nodo volverá a inactivarse por un tiempo predeterminado.

En PAMAS los nodos pueden estar en uno de seis estados posibles [22]: Idle, AwaitCTS, BEB (Binary Exponential Backoff), Await Packet, Receive Packet y Transmit Packet. Un nodo está en estado idle cuando no está transmitiendo ni recibiendo un paquete, bien sea porque no tiene nada para transmitir o porque, aunque si lo tiene, no puede transmitir porque un vecino está recibiendo información. Cuando el nodo desea transmitir un paquete, el envía un mensaje RTS e ingresa al estado AwaitCTS. Si el CTS esperado no llega, el nodo pasa al estado BEB, pero si recibe el CTS esperado el nodo entra al estado Transmit Packet y comienza la transmisión. El receptor de la transmisión, una vez que envía el CTS ingresa al estado Await Packet y si no recibe la transmisión esperada dentro de un tiempo de ida y vuelta (roundtrip) el nodo receptor vuelve al estado Idle, pero si el paquete empieza a llegar el transmite un tono de ocupado sobre el canal de señalización e ingresa al estado Receive Packet.

Los mensajes RTS, CTS y tono de ocupado se envían sobre el canal de control, por lo que se evitan interrupciones en la transmisión de datos y se permite que los nodos vecinos responda, también sobre el canal de control, sin interrumpir su transmisión de datos.

En [22] se prueba la capacidad de ahorro de energía de PAMAS mediante simulaciones en tres tipos de redes distintas: una red de topología aleatoria, una red de topología lineal y una red de topología completamente interconectada. Para la red completamente interconectada la reducción en consumo de energía reportada es de casi el 50%, mientras que en la red lineal los ahorros están entre el 10% y el 20%. En la topología aleatoria los ahorros resultantes en las simulaciones estuvieron en los rangos del 20% al 30% y del 60% al 70% dependiendo de si la red consistía de pocos nodos (dispersa) o de muchos nodos (densa) respectivamente. Los porcentajes los computan comparando PAMAS sin ahorro de energía

y PAMAS con ahorro de energía. También en [22] se determinan unos límites teóricos para el ahorro de energía dependiendo del tipo de red y su carga.

En [23] se define un mapa de energía como la información de la cantidad de energía disponible en cada parte de la red. En el artículo se describen los resultados de la simulación en ns-2 de la aplicación de mapas de energía construidos con base en probabilidades, esto permite actualizar la información de energía disponible en cada punto de la red sin necesidad de que los nodos envíen permanentemente esta información al nodo principal, lo que minimiza el consumo de energía. Para esto cada nodo debe enviar, mediante mensajes de nivel MAC, información que permita predecir el consumo de energía del mismo y sólo se vuelve a enviar información de energía disponible cuando el error entre el nivel que se predijo y el nivel real supera un umbral predeterminado. Los resultados de la simulación se comparan con modelos sencillos en los que el mapa se construye mediante la información de energía disponible enviada desde cada nodo cuando se presenta un cambio notorio.

MAC IEEE 802.15.4. [24] El control de acceso al medio del protocolo 802.15.4 establece dos mecanismos para acceso al canal, uno basado en contención y otro sin contención. El primero controla el acceso al canal mediante el algoritmo CSMA-CA y está orientado a redes con comunicación uno a uno, mientras que en el segundo el control del acceso lo realiza el coordinador de PAN y está orientado a redes con topología en estrella. Para los intereses del presente trabajo sólo tendremos en cuenta el mecanismo basado en contención que tiene aplicación en las redes inalámbricas de sensores.

Para el control de acceso al medio, el estándar define dos mecanismos: CSMA-CA ranurado y CSMA-CA no ranurado (slotted y no slotted). El no ranurado se utiliza para redes en las que no está habilitada la señalización. En estas, cada vez que un dispositivo desea transmitir tramas de datos o comandos MAC, debe esperar por un período aleatorio de tiempo. Si el canal se encuentra libre, en cuanto termine el período de back off, el dispositivo podrá transmitir sus datos. Si el canal está ocupado, en cuanto termine el back off aleatorio, el dispositivo deberá esperar por otro período aleatorio antes de tratar de

acceder al canal nuevamente. Las trama de reconocimiento no deben utilizar el mecanismo de CSMA-CA.

Las redes en las que está habilitada la señalización utilizan el mecanismo CSMA-CA ranurado para el acceso al canal, las ranuras o tiempos de back off se alinean con el inicio de la transmisión de señalización. Cada vez que un dispositivo desea transmitir tramas de datos durante el período de acceso a contención (CAP), el deberá localizar el límite de la siguiente ranura de bakcoff y luego esperar por un número aleatorio de ranuras de backoff. Si después de esto el canal está ocupado, el dispositiva podrá empezar su transmisión sobre el límite de la siguiente ranura de backoff disponible.

El estándar define la posibilidad de utilizar opcionalmente una estructura de supertrama que está limitada por señalizaciones de red, que es transmitida por el coordinador y que está dividida en 16 ranuras de igual tamaño. La trama de señalización se transmite en la primera ranura de cada supertrama y se utiliza para sincronizar los dispositivos, identificar la PAN y describir la estructura de las supertramas.

La supertrama puede tener porciones activas e inactivas. En las inactivas el coordinador no debe interactuar son su PAN y puede pasar a un modo de bajo consumo. Las porciones activas pueden utilizarse para aplicaciones que exigen bajos retardos o anchos de banda específicos para datos. A cada parte de una porción activa que se utilice para una aplicación de las descritas anteriormente se denomina ranura de tiempo garantizada (GTS). los GTS forman el período libre de contención (CFP) que siempre aparecen al final de la supertrama activa empezando un límite de ranura inmediatamente a continuación de un período de acceso a contención (CAP). Un GTS puede ocupar más de una ranura y el coordinador de PAN puede asignar hasta siete GTS.

El estándar establece la posibilidad de topologías jerárquicas organizadas por grupos (clusters) en las que se especifican unos líderes de grupo (CLH) entre los cuales se encuentra el coordinador de la PAN. Este tipo de topología es útil para algunos esquemas de enrutamiento utilizados en redes inalámbricas de sensores.

Para intentar una transmisión, un dispositivo debe mantener tres variables: número de backoffs (NB), longitud de la ventana de contención (CW) y exponente de backoff (BE). NB es el número de veces que el algoritmo CSMA-CA debió esperar mientras el dispositivo intentaba su transmisión; este valor debe inicializarse en 0 antes de cada nuevo intento de transmisión. CW define el número de períodos de backoff que deben permanecer sin actividad del canal antes de que la transmisión pueda comenzar; este valor debe inicializarse a 2 antes de cada intento de transmisión o cada vez que el canal esté ocupado. La variable CW sólo se utiliza para la versión ranurada de CSMA-CA. BE está relacionado al número de períodos de backoff que debe esperar un dispositivo antes de intentar acceder al canal.

El reconocimiento de recepción exitosa y la validación de datos o tramas de comando MAC pueden opcionalmente confirmarse mediante mensajes de reconocimiento (ack). Si se utilizan los reconocimientos, los dispositivos que envían la información esperan por la recepción de estos, y si no la reciben deberán reintentar la transmisión varias veces. Los ciclos útiles de funcionamiento en 802.15.4 deben ser menores del 1%.

En la literatura también encontramos artículos que describen otros protocolos de nivel MAC diseñados específicamente para redes de sensores y que, teniendo en cuenta el enfoque de nuestra investigación, sólo enunciaremos pero se pueden consultar en las siguientes referencias: T-MAC [25], CSMAC [26], DMAC [27] y WiseMAC [28] entre otros.

3.2 Protocolos de enrutamiento

En [5] realizan una evaluación y clasificación de los protocolos de enrutamiento para redes inalámbricas de sensores que se puede observar en la Fig. 4 tomada de la misma referencia. En ella los protocolos se clasifican según la estructura de red como: protocolos de enrutamiento basado en redes planas, basado en redes jerárquicas y protocolos adaptivos. La otra gran clasificación la hacen según la operación del protocolo en: protocolos de enrutamiento basado en negociación, basado en múltiples trayectos, basado en consultas y

basados en localización. Vale la pena aclarar que esta clasificación no implica exclusividad, es decir, un protocolo puede pertenecer a varias clases.

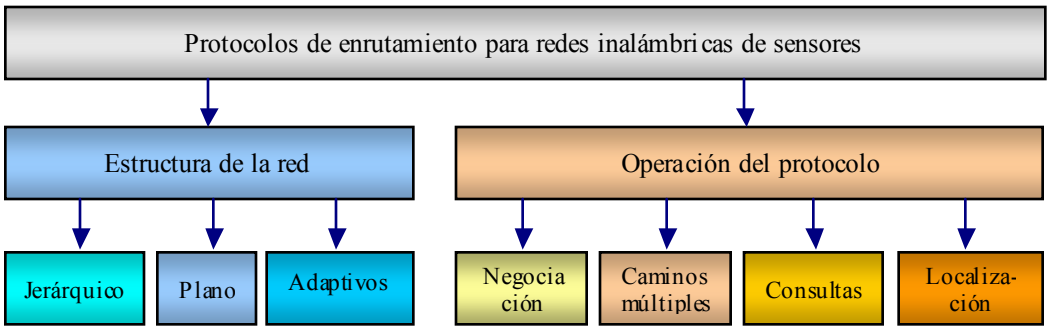

Fig. 4. Clasificación de protocolos de enrutamiento para redes de sensores (tomado de [1])

3.2.1 Protocolos jerárquicos

Este tipo de protocolos basan su funcionamiento en la formación de grupos (clusters) de nodos dentro de la red. Los grupos se forman con base en la energía disponible en los nodos y en la proximidad con el líder de grupo (cluster head) que hace las veces de retransmisor de información proveniente de los nodos de su grupo hacia el nodo destino.

Seguidamente se describen los protocolos jerárquicos de mayor utilización en redes de sensores.

LEACH Low Energy Adaptive Cluster Hierarchy. En [17] proponen el algoritmo LEACH, en el que se da especial importancia al ahorro de energía y la tolerancia a fallas en los nodos, se presume que los nodos son homogéneos y que el nodo destino está en una posición fija. LEACH es un algoritmo basado en agrupamiento (clustering) en el que los grupos se forman con base en la intensidad de la señal recibida y los líderes de grupo (cluster heads) se rotan aleatoriamente entre los nodos para dispersar la carga de energía. Los líderes recogen los datos provenientes de los nodos vecinos a través de una agenda fija (TDMA) y sobre estos datos realizan agregación para finalmente enviarlos al nodo destino (sink) o estación base.

Se utiliza la fusión o agregación de datos local (en cada cluster) para construir pequeños conjuntos de información significativa obtenida a partir de la gran cantidad de información procedente de los sensores, dando importancia a las señales comunes y restándosela al ruido no correlacionado. Esto, hecho localmente, implica una gran disminución en la cantidad de información a transmitir hacia el nodo destino.

El número óptimo de líderes de grupo, y por tanto de grupos, en la red es del 5% del total de nodos [16].

La operación de LEACH [17] se divide en rondas, cada ronda comienza con una fase de configuración en la que se organizan los grupos, seguida de una fase de estado estacionario en la que se realiza la transferencia de datos hacia el destino. Para minimizar el encabezado, la fase de estado estacionario debe ser grande comparada con la de configuración.

Las fases del algoritmo son cuatro [17], la primera es la fase de anunciación (advertisement) en la que cada nodo decide si será un líder de grupo para el ciclo actual. La decisión se basa en un porcentaje sugerido y predefinido de líderes para la red y en el número de veces en que el nodo ha sido un líder de grupo. La fórmula mediante la que se toma la decisión es la siguiente [13]:

$$T(n) = \begin{cases} \dfrac{P}{1 - P * \left(r \bmod \dfrac{1}{P} \right)}, si\ n \in G \\ 0, \rightarrow en\ cualquier\ otro\ caso \end{cases}$$

El nodo n escoge un número aleatorio entre 0 y 1, si el número seleccionado es menor que el umbral T(n) definido por la fórmula anterior, el nodo será un líder de grupo en la ronda actual. P es el porcentaje deseado de líderes de grupo, r es el número de ronda actual y G es el grupo de nodos que no han sido líderes de grupo en las últimas 1/P rondas. De esta forma cada nodo será líder de grupo en alguna de las 1/P rondas. Durante la ronda r = 0 cada nodo tiene una probabilidad P de llegar a ser líder de grupo. Los nodos que son líderes de grupo en r = 0 no pueden volver a serlo en las siguientes 1/P rondas. A medida que avanzan las

rondas, aumenta la probabilidad de que los nodos que aún no han sido líderes de grupo lleguen a serlo. Después de (1/P) – 1 rondas, T = 1 para cualquier nodo que aún no haya sido líder de grupo, y después de 1/P rondas todos los nodos son elegibles una vez más como líderes de grupo. En la fase de anunciación cada líder de grupo transmite su anuncio mediante el protocolo CSMA-MAC por lo que los nodos que no son líderes deben permanecer encendidos para recibir los anuncios de sus respectivos líderes. Con base en la intensidad de la señal recibida de los líderes, los demás nodos deciden a que grupo pertenecerán en esta ronda y si hay empates se realiza una selección aleatoria.

La segunda fase es la de configuración del grupo que comienza una vez que los nodos han decidido a que grupo pertenecen. Entonces los nodos deben informar al líder mediante CSMA-MAC que ellos serán miembros de su grupo y todos los líderes deberán tener encendidos sus receptores durante esta fase.

La tercera fase es la de creación de una agenda para comunicación entre nodos y líderes de grupo. Una vez que el líder ha recibido todos los mensajes de nodos a los que les gustaría pertenecer al grupo, el líder crea una agenda TDMA para que cada nodo sepa cuando puede transmitir hacia él. Esta agenda se difunde hacia todos los nodos del grupo.

La cuarta y última fase es la de transmisión de datos y que empieza una vez que los líderes han creado la agenda TDMA por lo que los nodos pueden empezar la transmisión hacia el líder. El líder debe mantener su receptor activo durante toda esta fase, si todos los nodos tienen algo que enviar, ellos estarán apagados siempre y sólo se encenderán para transmitir durante el período de tiempo asignado a cada uno, para lo que utilizarán la mínima cantidad de energía requerida obtenida con base en la intensidad de señal recibida en la primera fase mediante el anuncio del líder de grupo. Una vez que todos los nodos han transmitido su información hacia el líder, este realiza funciones de procesamiento de señales para comprimir los datos en una única señal que será enviada hacia el destino mediante un seudo código CDMA distinto para cada líder. Esta fase es la de estado estacionario y después de cierto tiempo determinado previamente debe comenzar la siguiente ronda con una nueva elección de los líderes de grupo y con estos anunciando hacia todos los nodos de su grupo.

PEGASIS y PEGASIS Jerárquico. Power Efficient Gathering in Sensor Information Systems [29]. Protocolo basado en LEACH en el que se busca mejorar su desempeño mediante la construcción de cadenas lineales de nodos en lugar de agrupaciones. Los líderes de grupo se seleccionan aleatoriamente y cada nodo se comunica únicamente con un vecino inmediato formando una cadena dirigida hacia el líder, el cual finalmente envía la información recibida de sus nodos hacia el destino. La elección del vecino en la cadena se hace utilizando la intensidad de la señal recibida para medir la distancia hacia todos los nodos vecinos y ajustando luego la intensidad de la señal para que sólo pueda escucharla el vecino más cercano. En cada salto se realiza agregación de datos entre los datos recibidos y los datos del nodo receptor antes de transmitir hacia el siguiente nodo en la cadena.

El uso de cadenas en PEGASIS en lugar de los grupos de LEACH, representa una disminución en la señalización requerida para la conformación de grupos. Esta disminución se representa en una mejora de entre el 100 y el 300% con respecto a LEACH en lo referente a consumo de energía, pero a costas de un elevado aumento en los retardos para los nodos alejados del líder, debidos a la existencia de un único transmisor hacia el destino[16].

En PEGASIS jerárquico se busca la disminución de los retardos mediante la transmisión simultánea de datos en diferentes cadenas. Para evitar la interferencia se permite la transmisión simultánea entre nodos alejados espacialmente o se hace utilizando codificación de señal como CDMA. Cuando se utiliza CDMA, se realiza mediante la construcción de un árbol jerárquico en el que hay transmisión simultánea desde nodos en posiciones pares hacia nodos vecinos en posiciones impares a sus derechas. Los nodos receptores pasan a formar un nivel jerárquico superior en el que se realiza una vez más, transmisión simultánea de datos desde los nodos en posiciones pares hacia sus vecinos en posiciones impares a la derecha y este esquema se repite sucesivamente con el asenso de los nodos receptores al siguiente nivel jerárquico y con transmisión simultánea de datos, hasta que la información llega al nodo líder y este la envía hacia el destino.

TEEN y APTEEN Treshold sensitive Energy Efficient sensor Network protocol [16]. Son las siglas de protocolo para redes de sensores eficiente en energía y sensible a umbral. El protocolo se basa en la conformación de grupos jerárquicos de dos niveles en los que el agrupamiento se hace con base en la cercanía entre los nodos en el primer nivel y de los grupos en el segundo. Con el fin de minimizar las transmisiones desde los nodos hacia el líder de grupo, los líderes de grupo difunden hacia los nodos dos umbrales: uno difícil (hard) y uno suave (fácil). El umbral difícil hace que los sensores transmitan información hacia el líder únicamente cuando la señal sensada excede dicho umbral. Una vez excedido el umbral difícil, el sensor sólo volverá a transmitir hacia el líder si la señal sensada sobrepasa el umbral fácil.

TEEN es un protocolo pensado para su utilización en redes que respondan apropiadamente a cambios inesperados en los atributos sensados en que la rapidez de respuesta es muy importante. TEEN no es recomendable para aplicaciones que requieren reportes periódicos de información.

El protocolo APTEEN o TEEN adaptativo es una variación de TEEN que persigue funcionar tanto en ambientes reactivos como en ambientes con necesidad de envío periódico de información. APTEEN cambia la periodicidad o los valores de los dos umbrales de acuerdo a las necesidades del usuario y al tipo de aplicación.

Los líderes de grupo, además de difundir los valores de los dos umbrales, también difunden atributos que son parámetros físicos que son de interés para el usuario, la agenda de transmisión TDMA para asignar una ranura (slot) para cada nodo y una cuenta de tiempo que es el máximo período de tiempo entre dos reportes sucesivos enviados por un nodo. APTEEN incluye tres tipos de consultas: histórica para analizar datos pasados; de una sola vez para tomar “fotos de la red”; y persistentes para monitorear un evento durante un período de tiempo.

Enrutamiento con base en energía para redes de sensores basadas en agrupamiento. Es un algoritmo de enrutamiento jerárquico basado en una arquitectura de tres niveles. Se

organizan grupos en los que los líderes de grupo conocidos como pasarelas son los únicos que se comunican con el destino, para lo cual deben conocer la posición de los sensores, deben mantener los diferentes estados de operación de los mismos para recolectar sus datos y cuentan con menos restricciones de energía que los sensores. La pasarela del grupo utiliza un control de acceso al medio TDMA mediante el cual informa a cada sensor en que intervalos debe escuchar la transmisión de otros sensores y en que intervalo debe transmitir su propia información.

Los sensores pueden operar en modo activo o en modos de baja potencia en que los módulos de procesamiento y los de transmisión pueden estar apagados de manera independiente por lo que cada nodo puede estar en uno de cuatro posibles estados: sólo sensando, sólo retransmitiendo, sensando y retransmitiendo o inactivo. Para el enrutamiento entre los nodos se utiliza una función de costo que depende del consumo de energía, la optimización del retardo y otras métricas de desempeño.

SOP Self Organizing Protocol [1]. Se define un conjunto de nodos que actúan como enrutadores en una estructura jerárquica y conforman un backbone de la red a través del cual los sensores se comunican con el nodo destino. El algoritmo auto organizacional para los nodos enrutadores crea tablas de enrutamiento y consta de cuatro fases:

- Fase de descubrimiento: en que se descubren los nodos que están en la vecindad de cada sensor
- Fase de organización: en la que se forman y mezclan los grupos en forma jerárquica. A cada nodo se le asigna una dirección dentro de la jerarquía y se le crean tablas de enrutamiento, además de la construcción de árboles de difusión que abarcan todos los nodos.
- Fase de mantenimiento: se actualizan las tablas de enrutamiento y los niveles de energía de forma tal que cada nodo informa a sus vecinos su tabla de enrutamiento y su nivel de energía. Se utiliza el algoritmo de ciclos locales de Markov (LML) para mantener los árboles de difusión.
- Fase de reorganización: se reorganizan los grupos en casos de fallas de los nodos.

MECN y **SMECN** Minimum Energy Communication Network [1]. En MECN para cada nodo se identifica una subred o zona de retransmisión que consiste de nodos vecinos hacia los cuales, desde el punto de vista de energía, es más conveniente transmitir que hacerlo directamente hacia el nodo destino o sitio maestro. La idea principal del protocolo [16] es encontrar una subred con menor cantidad de nodos y que requiere menos potencia para la transmisión entre dos de sus nodos, de forma tal que se encuentran rutas globales de mínima potencia. El protocolo es auto configurable por lo que se puede adaptar dinámicamente a fallas en los nodos o al despliegue de nuevos sensores.

SMECN es MECN pequeño, es una extensión de MECN en que las subredes que se constituyen para la retransmisión son más pequeñas que las de MECN en términos de nodos, lo que a su vez implica que se utilizarán menos saltos para transmisión con el consecuente ahorro de energía.

El consumo de potencia total sobre un trayecto $r = (u, u_1, \ldots, u_{k-1}, v)$ entre los nodos u y v que incluye $k-1$ nodos intermedios es [1]

$$C(r) = \sum_{i=0}^{k-1} \left(p(u_i, u_{i+1}) + c \right)$$

aquí $u=u_0$ y $v=u_k$, y la potencia necesaria para transmitir datos en este protocolo es [1] $p(u,v) = t \cdot d(u,v)^n$, n es el exponente de pérdidas de trayecto de modelos de radio propagación para ambientes externos ($n \geq 2$), y $d(u,v)$ es la distancia entre los nodos u y v.

3.2.2 Protocolos planos

En estos protocolos todos los nodos desempeñan el mismo papel dentro de la red.

SAR Sequential Assignment Routing [16]. Es un protocolo para múltiples trayectos que, además de procurar ser eficiente desde el punto de vista de energía, busca ser tolerante a fallas en la red. Su funcionamiento se basa en la creación de árboles con raíz en los nodos que son vecinos inmediatos al destino con base en métricas de calidad del servicio (QoS), recursos de energía de cada trayecto y nivel de prioridad de cada paquete. Aunque desde el

punto de vista de eficiencia de energía, SAR es mejor que el algoritmo de mínima energía, es muy pesado desde el punto de vista de encabezados debido al mantenimiento de las tablas y los estados en cada uno de los nodos.

Directed difussion [30]. Todos los datos generados por los sensores se nombran por medio de pares valor y atributo. La idea principal radica en la combinación de datos provenientes de diferentes fuentes mediante la utilización de agregación en cada nodo. La comunicación únicamente se da entre nodos vecinos, y es orientado a consultas por demanda, es decir, sólo se envía información si el nodo destino lo solicita, lo que se logra mediante el uso de un mensaje conocido como *interés* que contiene una lista de pares consistentes de un atributo y su valor, además de algunos campos de gradiente. Los atributos que comúnmente se utilizan son: nombre del objeto, intervalo de tiempo, duración de un evento, área geográfica, etc.

El *interés* [30] describe una tarea que se requiere realizar en la red, se difunde desde el nodo destino hacia sus vecinos desde los cuales se vuelve a difundir hasta llegar a todos los nodos. Cada nodo compara la información sensada con los valores almacenados localmente a partir del interés y responde al destino cuando encuentra coincidencias. Para la respuesta cada nodo puede hacer agregación y almacenamiento temporal intermedio (caching).

Un gradiente [16] es el enlace a través del cual un nodo recibe el interés de un vecino, y este enlace es el que se utilizará como ruta de comunicación hacia la fuente, de forma tal que el camino a seguir por los datos sensados será el mismo que utilizó el interés para llegar hasta el nodo pero en sentido opuesto. Se pueden establecer varios caminos entre fuente y destinos pero uno de ellos será utilizado por el destino mediante el refuerzo que se consigue mediante el envío del mensaje original del interés por parte del nodo destino, haciendo que la fuente envíe datos cada vez más frecuentemente sobre ese mismo camino.

Debido a su filosofía de funcionamiento, este algoritmo no es recomendable para utilizar con aplicaciones que requieren entrega constante de información como el caso de

monitoreo de ambientes en los que en lugar de esperar una consulta, los nodos transmiten información hacia el destino cuando se detecta algún evento.

Gradient-based routing [1]. Es una variante de directed diffusion en la que cada nodo descubre y almacena el número de saltos hacia el destino con base en el interés que mantiene el número de saltos mientras se difunde por la red. Se denomina altura del nodo a la cantidad mínima de saltos que hay entre un nodo y el destino o sumidero, y el gradiente es la diferencia entre la altura de un nodo y la altura de sus vecinos. Para el re-envío de paquetes se utiliza el enlace con el mayor gradiente, es decir aquel que requiera menos saltos para llegar hasta el destino. El protocolo se complementa con técnicas de dispersión de tráfico y agregación de datos. La dispersión se puede realizar de tres fomras distintas:

1. Estocástica: cuando dos enlaces tienen el mismo gradiente, la selección de la ruta se hace aleatoriamente.
2. basada en energía: cuando la energía cae por debajo de un umbral específico, se incrementa la altura del nodo.
3. basada en flujo de datos: cuando un enlace está participando en otra transmisión, se evita la utilización del enlace en otras sesiones.

CADR Constrained anisotropic diffusion routing [1]. Se propuso como una forma general de directed difusión en la que se consultan los sensores y se enrutan los datos de forma tal que se maximice la ganancia de información y se minimicen la latencia y el ancho de banda. Para lograrlo sólo se activan los sensores cercanos a un evento particular y dinámicamente se ajustan las rutas de datos. Además de los costos de comunicación, se evalúan localmente en cada nodo las ganancias de información. La utilidad de la información se modela utilizando la teoría estándar de estimación. Utiliza otra técnica a la que se le conoce como IDSQ Information-driven sensor querying (consulta de sensores impulsada por información), se basa en un protocolo en el que el nodo que hace la consulta puede determinar cual es el nodo que le puede brindar la información más útil a la vez que balancea los costos de energía. IDSQ se debe utilizar en conjunto con otro procedimiento porque no define como enrutar la información hacia el destino. El protocolo de enrutamiento utilizado normalmente es CADR para el cual los resultados de información

han demostrado que es más eficiente en energía que Directed Difussion [5], [16]. Estos resultados se ajustan al hecho de que en CADR las consultas se difunden de manera no isotrópica, alcanzando a algunos nodos de la red mientras que en Directed Difussion las consultas se difunden por toda la red lo que generará gastos de energía en más nodos de los necesarios.

ACQUIRE ACtive QUery forwarding In SensoR nEtworks [1]. Emisión de consultas activas en redes de sensores propone un mecanismo para la consulta de información que, al igual que COUGAR, toma a la red como una base de datos distribuida y está diseñado para realizar consultas complejas que pueden contener varias subconsultas. Las consultas surgen en el nodo destino que las envía hacia sus nodos vecinos para que cada uno de estos las responda parcialmente según la información de que disponga. Si la respuesta aún no está completa o está desactualizada, estos nodos re-envían la consulta hacia los nodos que estén a una distancia menor a *d* saltos y una vez la consulta esté completamente resuelta, se devuelve la información hacia el nodo destino por el camino inverso (al de la consulta) que presente el menor número de saltos. La selección del siguiente nodo a consultar se hace aleatoriamente o está basada en máximo potencial de satisfacción de la consulta.

El desempeño del protocolo en cuanto a consumo de energía depende de manrea importante del valor del parámetro *d*, si este es igual al tamaño de la red, el protocolo se comportará de manera similar al protocolo de inundación (flooding). Por el contrario, si *d* es muy pequeño la consulta deberá viajar muchos más saltos para poder responder al destino de manera satisfactoria. Aunque mediante modelamiento matemático se ha determinado el valor óptimo de *d* para una red con grillas compuestas por 5 nodos, no existe validación de resultados con base en simulaciones.

Energy aware routing [1]. Es una variación del protocolo de difusión directa en la que en lugar de tener un único trayecto óptimo hacia el destino, con base en probabilidades, ocasionalmente se usan trayectos sub óptimos para evitar agotar la energía de los nodos del trayecto óptimo. El protocolo inicia una conexión a través del desborde localizado que se utiliza para descubrir las rutas entre todos los pares fuente-destino así como sus costos con

lo que se construyen las tablas de enrutamiento. Los trayectos de mayor costo se descartan y se construye una tabla de enrutamiento seleccionando los nodos de forma proporcional a su costo. Posteriormente se utilizan las tablas para enviar los datos hacia el destino con una probabilidad que es inversamente proporcional al costo del nodo. El desborde localizado se utiliza por el nodo destino para mantener vivos los trayectos.

3.2.3 Protocolos adaptivos

En los protocolos adaptivos, la toma de decisiones de enrutamiento se controla con base en información de las condiciones actuales de la red como por ejemplo la definición de rutas de acuerdo a la energía disponible en los nodos.

SPIN Sensor Protocol for Information via Negotiation [16]. Es una familia de protocolos basados en un sistema de difusión (broadcast) para diseminación de datos que incorpora la negociación antes de cualquier envío de información por lo cual se elimina la información redundante y sólo se transmite información útil. Está basado en el desborde controlado, en el que se manejan problemas como implosión (causada por mensajes duplicados enviados hacia el mismo nodo), traslape (dos o más nodos monitorean la misma región y envían paquetes muy similares al mismo nodo) e invisibilidad de recursos [1]. Entre las principales características de SPIN está el uso de meta datos en lugar de paquetes de datos completos, lo que implica que se transmiten paquetes de datos mucho más pequeños que los que se transmitirían normalmente, debido a la utilización de un descriptor con mapeo uno a uno entre datos y meta datos. Desde el punto de vista de energía, SPIN utiliza la información de nivel de energía restante en los nodos para adaptar su funcionamiento.

En SPIN se manejan tres tipos de mensajes: ADV que se utiliza para informar sobre nuevos datos, REQ para solicitar datos, y DATA que incluye el mensaje en si mismo. Se utilizan tiempos aleatorios para los paquetes REQ buscando prevenir el traslape y se negocia utilizando ADV y REQ para reducir el problema de desborde. El funcionamiento de SPIN permite que cada nodo pueda comportarse como un destino. Cuando un nodo obtiene un nuevo dato envía un mensaje de ADV que contiene meta-datos; si a un nodo vecino le

interesan los datos descritos en el mensaje de ADV, debe enviar un mensaje REQ y el nodo fuente le enviará el mensaje tipo DATA con la información solicitada convirtiéndolo en nodo destino. El nodo vecino repite este mismo esquema con sus demás vecinos de forma tal que el área completa recibirá una copia de los datos.

Hay varias versiones de SPIN, las principales son SPIN-1 que no involucra la eficiencia en el consumo de energía, y SPIN-2 que es una extensión de la primera que incorpora un mecanismo de información de recursos basado en umbrales.

En SPIN-2 un nodo solicitará datos solamente si dispone de la energía necesaria para realizar la operación completa con el nodo que envió el ADV, es decir si puede enviar el REQ y recibir el DATA sin caer por debajo de un umbral de energía mínima especificado previamente. Las decisiones de enrutamiento de SPIN se basan en información local lo que lo hace atractivo para aplicaciones que involucran movilidad en los nodos ya que estos escenarios no le representan exigencias elevadas de memoria o procesamiento.

Maximum Lifetime Data Gathering [16]. En este algoritmo se modela la configuración de rutas como un problema de recolección de datos para máxima duración de la red y se presenta un algoritmo de tiempo polinomial. La duración "T" o tiempo de vida de la red se define como el número de ciclos (rounds) o lecturas periódicas de datos que se realizan a los sensores hasta que el primer sensor agote su energía. Se define una agenda de recolección de datos en la que se especifica como capturar y enrutar datos hacia el destino.

Para cada ciclo de recolección de la agenda se especifica un árbol con raíz en el destino y que abarca todos los nodos de la red. La vida o duración de la red depende del tiempo para el cual tiene validez la agenda, por lo que este tiempo debe maximizarse. Como parte de la propuesta se define la agregación de datos que se hace a través del algoritmo MLDA Maximum Lifetime Data Aggregation a la vez que se configuran rutas para maximizar la vida de la red.

Una agenda S con T ciclos induce un flujo de red G. El flujo de red que, sujeto a restricciones de energía en los nodos, maximiza la vida de la red es un flujo óptimo admisible. Con base en este flujo óptimo se construye una agenda de red.

Minimum Cost Forwarding. [16] Este protocolo busca el trayecto de mínimo consumo de energía en la red y que además, sea sencillo y escalable dado que el destino es siempre el mismo nodo. Para esto define el costo de un trayecto con base en retrazo, throughput y consumo de energía. El protocolo consta de dos fases, en la primera, la fase de configuración, se encuentra el valor del costo en cada nodo, empezando por el destino hasta incluir todos los nodos de la red. El nodo destino difunde un mensaje con el costo fijado a cero mientras cada nodo fija su costo mínimo hacia el destino como infinito. Tras la recepción del mensaje difundido desde el destino, cada nodo revisa si el costo estimado en el mensaje más el del enlace sobre el cual se recibe, es menor que el estimado actual. Si es menor se actualizan el estimado actual y el del mensaje difundido desde el destino. Si el mensaje difundido que se recibió se actualizó, este se re-envía en caso contrario se descarta. En la segunda fase el nodo que desea iniciar una transmisión (nodo fuente), difunde los datos hacia sus vecinos. Cada nodo que recibe el mensaje de difusión suma su costo de transmisión hacia el destino con el costo del paquete y revisa el costo restante en el paquete, y si este costo no es mayor que el requerido para llegar al destino, se descarta el paquete; de lo contrario el paquete se re-envía hacia los nodos vecinos.

El ajuste del costo no se hace a través del desborde (flooding), sino a través de un algoritmo de back off que permite reducir en un factor de 50 el número de mensajes que se intercambia comparado con un algoritmo de desborde a la vez que se mantiene el mismo valor de costo para los nodos.

Energy-Aware QoS Routing Protocol [16]. El protocolo surgió como respuesta a las necesidades de tráfico de tiempo real como el emitido por sensores de video o imágenes. Para esto busca los trayectos que cumplan con un nivel de retrazo dado en una transmisión, y selecciona el de menor costo y eficiencia en energía. la función de costo en cada nodo depende de las reservas de energía, energía de transmisión, rata de errores y otros

parámetros de comunicación. Para soportar simultáneamente tráfico de tipo mejor esfuerzo y tráfico de tiempo real utiliza un modelo de encolamiento basado en clases que se controla a través del parámetro *r* que representa la cantidad de ancho de banda dedicado a tráfico de tiempo real y tráfico sin estas restricciones sobre un enlace en caso de congestión. El protocolo encuentra una lista de caminos de menor costo mediante una versión extendida del algoritmo de Dijkstra y de esta lista selecciona el camino que mejor cumpla los requerimientos de retardo extremo a extremo.

3.2.4 Protocolos basados en negociación

En los protocolos basados en negociación se utilizan descriptores de datos de alto nivel denominados meta-datos con el fin de eliminar, a través de negociaciones previas a la transmisión del mensaje, el envío de datos duplicados o redundantes [1].

La familia de protocolos SPIN, descrita en la sección anterior, es un claro ejemplo de este tipo de protocolos.

3.2.5 Protocolos basados en múltiples caminos

Los protocolos basados en múltiples caminos ofrecen tolerancia a fallas a través de la disponibilidad de al menos un camino alterno entre fuente y destino de la información. Desde el punto de vista de consumo de energía no son atractivos pues para mantener la información sobre los caminos se requiere el envío periódico de paquetes para todas sus rutas.

Maximum Lifetime Energy Routing [2]. En este algoritmo se definen los costos para un enlace como una función de la energía disponible en el nodo y la energía requerida para la transmisión sobre dicho enlace. El camino de menor costo hacia el destino será aquel que tenga la mayor energía disponible entre todos los caminos posibles. En [16] se definen dos algoritmos de este tipo cuya diferencia radica en la definición de costo del enlace entre los nodos *i* y *j*. Si $\underline{E}_i$ es la energía disponible o energía residual en el nodo *i*, y e_{ij} es la energía

consumida en la transmisión entre el nodo *i* y el nodo *j*, el costo del enlace entre los nodos *i* y *j* para los dos algoritmos son [16]:

$$c_{ij} = \frac{1}{\underline{E}_i - e_{ij}} \text{ y } c_{ij} = \frac{e_{ij}}{\underline{E}_i}$$

Para fines comparativos se define el algoritmo Minimum Transmitted Energy MTE en el cual el costo del enlace entre los nodos i y j es e_{ij} .

Hierarchical Power-aware Routing in Sensor Networks [1]: busca mejorar la confiabilidad de la red mediante la utilización de múltiples rutas entre los nodos fuente y destino. Los nodos se organizan en grupos de acuerdo a su proximidad geográfica, estableciendo zonas que se tratan de manera individual como una entidad. Cada zona debe decidir como enrutará un mensaje a través de las otras zonas de tal forma que se maximice la vida de los nodos. Los mensajes se enrutan a través del camino que tenga el máximo entre todos los mínimos de potencia disponible, configurando un camino denominado camino max-min. El algoritmo inicialmente busca el camino con el consumo de potencia mínimo (P_{min}) utilizando para esto el algoritmo de Dijkstra. Seguidamente busca un camino que maximice la potencia residual mínima en la red. La idea del algoritmo es optimizar ambos criterios de solución. Según [5] la información a enviar se divide en subpaquetes que se envían por caminos separados de forma tal que en el destino se pueda reconstruir la información enviada a pesar de que haya pérdida de algunos subpaquetes.

En [5] se menciona a **Directed Difussion Multipath** como una versión para caminos múltiples del protocolo Directed Difussion.

3.2.6 Protocolos basados en consultas

En los protocolos basados en consultas los nodos destinatarios de la información (sinks) propagan por la red una consulta de datos, de forma tal que el nodo que contiene los datos solicitados en la consulta devuelve la información siguiendo de manera inversa la ruta de la

consulta. Estos protocolos buscan optimizar el uso de energía mediante la consulta selectiva a subconjuntos de nodos dentro de una red de sensores, además de la utilización de técnicas de agregación de datos. De manera general, el proceso se inicia en el nodo destino con la emisión de consultas hacia un subconjunto específico de sensores que devuelven los datos hacia el destino. A continuación se describen brevemente los principales protocolos basados en consultas.

Flooding and Gossiping[16]. En el mecanismo de desbordamiento (flooding), cada nodo que recibe un paquete de datos lo re-envía hacia sus vecinos y este proceso continúa hasta que el paquete llega a su destino o se alcanza el máximo número de saltos especificado para el paquete. El mecanismo de murmullos (gossiping) es una versión mejorada de desbordamiento en la que el nodo que recibe un paquete selecciona aleatoriamente a qué nodos lo re-enviará y este proceso se realiza de manera sucesiva.

Diffusion based routing algorithm [31]. Este es un algoritmo de enrutamiento que usa difusión o dispersión del tráfico para aplanar la vida de los sensores en una red. El algoritmo se basa en localización de cada nodo, nivel de potencia disponible y carga sobre el nodo. En [31] se comparan los resultados del porcentaje de vida con respecto al 100% de utilidad con otros algoritmos como Leach [17], Pegasis [29] y Minimum Transmission Energy (MTE) [16], pero la comparación se hace con base en resultados expresados en los artículos respectivos en los que los escenarios de implementación o simulación no son iguales para todos por lo que los resultados no son representativos.

En este algoritmo, los nodos deciden localmente los vecinos a través de los cuales se comunicarán, con base en información de localización, potencia y carga. La primera selección de vecinos se debe hacer con base en información de potencia obtenida a partir de la intensidad de la señal. Después de que se han realizado algunas transmisiones con este criterio, los nodos deberán intercambiar mensajes de sincronización que comuniquen energía disponible y carga, para con base en esta información tomar las nuevas decisiones de rutas para transmisión de mensajes hacia el destino. Los mensajes de sincronización informando a un nodo fuente que debe buscar otro nodo vecino para el re-envío de

información se emiten como mensajes de excepción únicamente cuando se cumple una cualquiera de tres condiciones. La primera es que los niveles de energía del receptor sean menores que los de la fuente. La segunda es que los niveles de energía del receptor hayan caído por debajo de un umbral determinado previamente, por lo que dicho nodo no retransmitirá información de ningún otro, sólo la propia. La tercera condición es que las colas del nodo estén llenas de forma tal que no puede manejar más solicitudes.

Si se utiliza la transmisión directa entre cada nodo y el nodo destino, los nodos que están más alejados del destino agotarán sus energías más rápidamente. Si se utiliza un algoritmo de difusión sencilla basado únicamente en información espacial (Minimum Energy Transmission Routing), cada nodo haría una lista de sus vecinos y sus preferencias con base en distancias y empezaría a transmitir hacia los nodos que estén en la ruta del destino. En este caso, los nodos que están más cerca del destino agotarán sus energías más rápidamente que los alejados ya que actuarán como retransmisores de todos los demás nodos.

Rumor routing [1]. Es una variación de difusión directa, en la que en lugar de dispersar consultas a toda la red, se trata de hacerla hacia un nodo que conozca el camino hacia el evento de interés, lo cual se logra mediante el almacenamiento en tablas de eventos para cada nodo que se dispersan a través de paquetes de larga vida conocidos como agentes. Cuando un nodo detecta un evento lo adiciona a su tabla local y genera un agente. Los agentes viajan por la red para propagar la información sobre eventos locales a nodos distantes. Así cuando un nodo genera una consulta sobre un evento, los nodos que saben la ruta pueden responder a la consulta inspeccionando su tabla de eventos. En este algoritmo se mantiene sólo una ruta entre fuente y destino, a diferencia de directed diffusion en el que los datos se pueden enrutar a través de varios trayectos a bajas velocidades. Este protocolo es eficiente cuando el número de eventos es bajo ya que para un gran número de eventos se eleva incontrolablemente el costo de mantener los eventos y las tablas de eventos en cada nodo y no todos los eventos serán del interés del nodo destino.

e3D Energy-Efficient Routing Algorithm for Wireless Sensor Networks [32]. En este artículo exponen el algoritmo e3D que está basado completamente en el propuesto en [31]

por el autor de este último artículo, por lo que son muy similares, la diferencia de e3D está en que se hacen variaciones sobre las condiciones que debe cumplir un nodo receptor para emitir un mensaje de excepción. El receptor analiza si su energía es menor que la del transmisor, sólo cuando la energía de que dispone está por debajo de un umbral especificado en 50% que es el óptimo obtenido empíricamente. Para la condición de que la energía del nodo esté por debajo de un umbral a partir del cual no retransmitirá mensajes de otros nodos, el umbral propuesto está entre el 5% y el 10%. Se realizan comparaciones con otros algoritmos como LEACH [17], Minimum Transmisión Energy [16], difusión básica, difusión ideal y agrupamiento ideal. Los algoritmos ideales, aunque no se pueden implementar, se utilizan como límites superiores de desempeño, y la conclusión a la que se llega en [31] es que, a diferencia de los demás algoritmos, e3D se comporta de una manera muy cercana a la de los algoritmos ideales.

COUGAR[16]. Es un protocolo basado en consultas en el que la red se toma como una base de datos distribuida para la que se utilizan consultas declarativas con el fin de abstraer procesamiento de consultas de las funciones del nivel de red. Adicionalmente utiliza agregación de datos en red y requiere, para todos los nodos, de la introducción de un nivel de consultas que se ubica en la pila de protocolos entre el nivel de red y el de aplicación. El nodo destino es responsable del plan de consultas y de su envío a los nodos que considere importantes, además debe seleccionar un nodo líder que está encargado de realizar la agregación y la transmisión de datos hacia el destino.

Los inconvenientes de COUGAR están asociados a los costos energéticos que implican la introducción del nivel adicional en todos los nodos, la sincronización necesaria para las consultas y el mantenimiento dinámico que requiere el nodo líder para ejercer sus funciones de manera confiable.

3.2.7 Protocolos basados en localización

En este tipo de protocolos, el enrutamiento de la información se basa en la localización se los sensores dentro de la red. La localización se puede lograr por medio de GPS o por

triangulación de señales, siendo la segunda la opción más viable en redes de sensores, pues la primera implica altos costos de los dispositivos y elevados consumos de energía debidos a los receptores satelitales del GPS lo cual no los hace atractivos para las redes inalámbricas de sensores.

GAF Geographic Adaptive Fidelity [16]. Es un algoritmo de enrutamiento basado en la posición de los nodos que se creó inicialmente para redes móviles ad hoc pero puede utilizarse en redes de sensores. Con base en información geográfica obtenida a través de un GPS, los nodos constituyen una grilla virtual y dentro de cada cuadrícula sólo un nodo estará activo por un determinado período de tiempo mientras los demás nodos de la misma grilla permanecen inactivos. El nodo activo en cada grilla deberá monitorear y enviar la información hacia el nodo destino de tal forma que no se afecte el nivel de fidelidad del enrutamiento. En GAF se definen tres estados para los nodos [1]: descubrimiento en el que se determina qué nodos pertenecen a la grilla, activo en el que se presenta la participación activa de los nodos en el enrutamiento, y durmiente en el que se apaga el módulo radio. Aunque GAF puede considerarse también como jerárquico en que los grupos se conforman de acuerdo a la ubicación geográfica de los nodos, los líderes de grupo no realizan ningún tipo de agregación.

En [33] se propone un esquema de agrupamiento (clustering), pero a diferencia de GAF [16] el tamaño de los clusters no es el óptimo único sino que varía dependiendo del tráfico del área y por lo tanto de la cercanía de esta al nodo destino. Al hacer la comparación con otros métodos de gestión de topología como GAF, el método propuesto en [33] permite el 50% de ahorro en energía (de 2.1 $x10^{-3}$J/s a $9x10^{-4}$J/s)

GEAR Geographic and Energy Aware Routing [1]. Selecciona los vecinos que usará como enrutadores hacia el destino mediante una heurística que utiliza información geográfica e información de energía. El algoritmo busca restringir el número de intereses en el método de difusión directa al enviarlo hacia una región específica de la red en lugar de hacerlo a toda la red.

Cada nodo mantiene dos valores estimados de costo, uno estimado y otro aprendido. El primero se obtiene como una combinación de la energía residual y de la distancia hacia el destino. El segundo es un refinamiento del costo estimado que tiene en cuenta enrutamiento alrededor de vacíos. Un vacío se presenta cuando un nodo no tiene ningún vecino que esté más cerca que él del destino. Como la única diferencia entre los dos costos es el enrutamiento alrededor de vacíos, si no los hay el costo aprendido es igual al costo estimado.

GPSR Greedy Perimeter Stateless Routing [34]. Es un algoritmo basado en la localización de los nodos que está compuesto por dos métodos de reenvío de paquetes: el reenvío voraz que se utiliza en la mayoría de ocasiones posibles, y el reenvío perimetral que se utiliza en regiones en la que no es posible utilizar el primer tipo de reenvío.

a. Reenvío voráz: El originador de los paquetes los marca a todos con la localización de los destinos hacia los que están dirigidos por lo que cualquier nodo que vaya a participar en el reenvío de los paquetes podrá hacer la selección localmente óptima del siguiente salto del paquete. Si un nodo conoce las posiciones de los vecinos que están dentro de su rango de cobertura de radio, la selección localmente óptima del siguiente salto es el vecino que está geográficamente más cerca del destino especificado en el paquete, y la aplicación sucesiva de esta política de re-envío se mantiene hasta que se alcance el destino.

La información sobre ubicación de los vecinos dentro del radio de cobertura se realiza mediante la utilización de un algoritmo de señalización con el que cada nodo transmite periódicamente una señal a la dirección MAC de *broadcast* incluyendo su identificador y posición. Esta posición se codifica en forma de coordenadas x y y. Si se define el intervalo B como el intervalo de tiempo entre el cual se realiza la señalización, para evitar sincronización en la señalización y colisión entre señales de los vecinos, se agita o desvía la transmisión de señales en un 50% de B. De esta forma el valor medio del intervalo de transmisión entre señales es B y está uniformemente distribuido entre $[0.5B, 1.5B]$.

Si un nodo no ha recibido señalización de un vecino después de un tiempo T, él asume que dicho vecino falló o está fuera de rango por lo que lo excluye de su tabla de vecinos. En [34] el tiempo de espera para eliminar un nodo de la tabla de vecinos es de $4.5B$, tres veces el tiempo del intervalo de señalización con la máxima desviación.

La selección adecuada del intervalo de señalización B, que depende del rango de cobertura radio de los nodos y de la movilidad de los mismos, es un aspecto muy importante en el desempeño del protocolo, pues si este valor es muy alto existirán en las tablas de vecinos "falsos" vecinos debido a la desactualización de las tablas, y si es muy bajo se consumirá energía innecesaria enviando información repetida sobre los nodos vecinos.
Este mecanismo de señalización se puede traducir en tráfico de protocolo proactivo como el que evitan los protocolos AODV y DSR, por lo que para minimizar su costo en GPSR se incluye la posición del nodo local que hace el envío en todos los paquetes de datos que se reenvían, y se ejecuta en modo promiscuo las interfaces de red de todos los nodos. De esta forma cada nodo recibe una copia de todos los paquetes enviados para todos los nodos dentro del área de cobertura con lo que cada paquete de datos lleva a su vez información de señalización que permite la actualización de las tablas. Así cada vez que un nodo recibe un paquete de datos puede reiniciar su temporizador de señalización reduciendo el tráfico de señalización sobre todo en áreas en las que se presenta reenvío constante de paquetes de datos.

b. Reenvío perimetral: se utiliza excepcionalmente cuando al llegar un paquete hasta un nodo, este nodo no encuentra dentro de sus vecinos (cobertura radio) ningún nodo que esté más cerca que él del destino, es decir este nodo es un máximo local en cercanía al destino. Antes de describir el funcionamiento del reenvío perimetral debemos especificar algunas definiciones de grafos que se requiere tener claras para comprender tal funcionamiento.

Grafo unitario: son grafos cuyos bordes están limitados por un umbral mínimo de distancia entre sus vértices. En el presente trabajo asumimos que el radio de cobertura de un nodo es la región circular ideal limitada por el alcance de transmisión del nodo, por lo que la red se puede representar mediante un grafo unitario.

Grafo plano: o grafo planar es un gráfico en el que no hay cruces entre los bordes.
Vértice: es un punto en un grafo, lo que equivale a un nodo de la red
Borde: es la conexión entre dos vértices del grafo, en términos de la red es el enlace existente entre dos nodos vecinos.
Cara: es una región poligonal constituida por los bordes de un grafo.

El reenvío perimetral, que opera correctamente sobre grafos planos se rige por la regla de la mano derecha [34 con la que se bordea el área vacía apreciada por el nodo hasta el que llegó el paquete en modo de reenvío voraz y que se puede observar en la Fig 5. La regla de la mano derecha establece que, cuando un paquete arriba a un nodo *a* desde otro nodo *c*, el siguiente borde por el que debe reenviar el paquete será el primero que se encuentre en sentido contrario a las manecillas del reloj y así se aplica sucesivamente tomando como referencia el borde por el que se recibió el paquete en cada reenvío.

La aplicación de la regla de la mano derecha sobre una cara interna (el interior de una región poligonal cerrada) realiza un reenvío de los paquetes, o recorrido de la cara, en el sentido de las manecillas del reloj, mientras que la misma regla aplicada a la región exterior de la cara realiza el recorrido en el sentido opuesto a las manecillas del reloj.

Las redes de sensores que estamos considerando en el presente trabajo son redes de sensores homogéneos y como tal, todos tienen la misma potencia de transmisión lo que implica que sus radios de cobertura son iguales. En términos de grafos, entre dos sensores o nodos existirá un borde si la distancia entre los nodos es menor o igual que su radio de cobertura. Teniendo en cuenta lo anteriormente dicho, una red de sensores la podemos considerar como un grafo unitario.

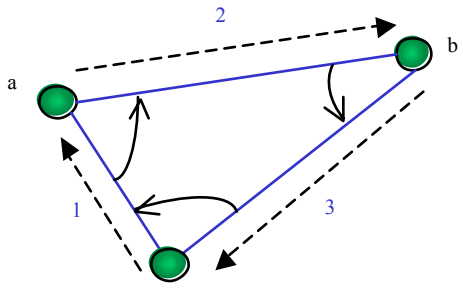

Fig. 5. Regla de la mano derecha. Tomada de [34]

Para cambiar de modo de reenvío voraz a perimetral se requiere de una heurística que fuerce a la regla de la mano derecha a encontrar perímetros que encierren vacíos en regiones en las que los bordes del grafo se cruzan. Los autores de [34] han propuesto previamente la heurística de *no cruce* con la cual se encuentra el 99.5% de las $n(n-1)$ rutas existentes entre n nodos, pero incluso con este nivel de aciertos aún se presentan casos de rutas existentes que la heurística no encuentra debido a que por su funcionamiento la heurística de *no cruce* elimina ciegamente el segundo enlace que encuentre en un par de bordes que se cruzan y este borde eliminado puede partir la red en dos, de forma tal que el algoritmo no encontrará rutas que atraviesen esta partición.

Para superar esta situación en [34] proponen un algoritmo distinto que produzca una red sin cruce de enlaces (grafo plano) mediante la eliminación de bordes del gráfico siempre y cuando estos bordes no sean parte del Relative Neighborhood Graph (RNG) o del Gabriel Graph (GG).

Para el caso de RNG se pueden eliminar bordes que no pertenezcan al RNG, es decir, tomando como referencia la Fig. 6 solo se elimina un borde *ab* siempre y cuando exista al menos un nodo *c* dentro del rango de cobertura de los dos nodos *a* y *b* a los que une el borde, con lo que la red no se partirá. Expresado matemáticamente, un borde *ab* existe en un RNG si $\vee\, c \neq a,b : d(a,b) \leq \max[d(a,c), d(b,c)]$

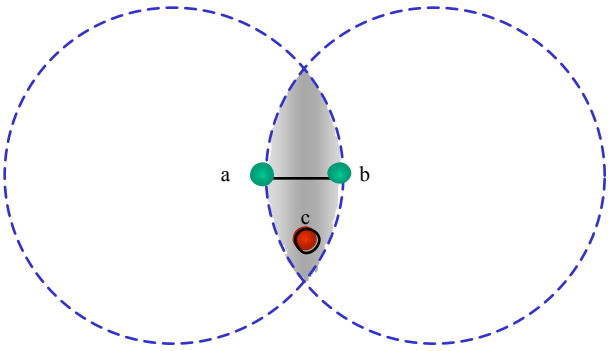

Fig. 6. Relative Neighborhood Graph. Tomada de [34]

De igual forma, para el caso de GG, que se puede observar en la Fig. 7 se puede eliminar un borde *ab* sin partir la red siempre y cuando exista al menos un nodo *c* dentro del círculo con radio *ab/2* y centrado en el punto medio del borde *ab*. Expresado matemáticamente, un borde *ab* existe en un GG si $\vee c \neq a,b : d^2(a,b) < [d^2(a,c), d^2(b,c)]$. A partir de Fig. 6 y Fig. 7 se puede concluir que en GG se eliminan menos bordes que en RNG.

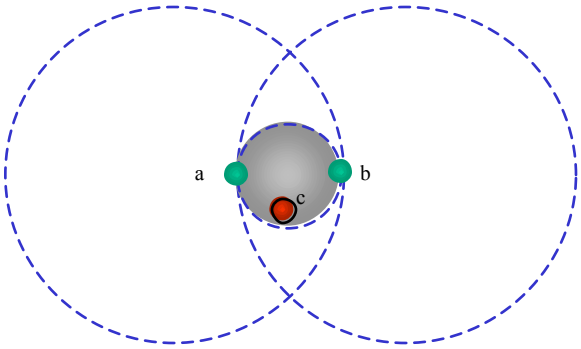

Fig. 7. Gabriel Graph. Tomada de [34]

c. Combinación de reenvío voraz y perimetral. El reenvío voraz se realiza sobre el grafo de red completo mientras que el reenvío perimetral se realiza sobre el grafo aplanado (RNG o GG) de red en el que no es posible el reenvío voraz. El modo perimetral se realiza sobre caras adyacentes del grafo buscando siempre ir de una cara hacia otra que esté más cerca del destino.

En la Fig. 8 se describe el proceso de envío en modo perimetral en el que cuando un paquete destinado al nodo *D* entra en modo perimetral en un nodo *x*, GPRS inicia el reenvío del paquete sobre el primer borde que se encuentre en el sentido contrario a las manecillas del reloj a partir de la línea *xD*, esto determina la primera cara sobre la cual se reenviará el paquete y luego lo reenvía progresivamente sobre las caras del grafo planar que están más cercanas al destino las cuales son atravesadas por la línea *xD*. La travesía utiliza la regla de la mano derecha sobre cada cara para alcanzar un borde que cruce la línea *xD*. En ese borde la travesía se mueve hacia la cara adyacente cruzada por la línea *xD*.

Cuando un paquete entra en modo perimetral, GPSR registra en el paquete la localización *Lp*, el nodo en el cual falló el reenvío voraz. Esta información puede ser útil para

determinar en posteriores saltos la localización en la que el paquete puede retornar a modo voraz. Cada vez que GPSR reenvía un paquete sobre una nueva cara, registra en *Lf* el punto de la línea *xD* que se comparte entre la cara nueva y la previa. Finalmente GPSR registra en *e0* del paquete el primer borde que cruza el paquete en la nueva cara.

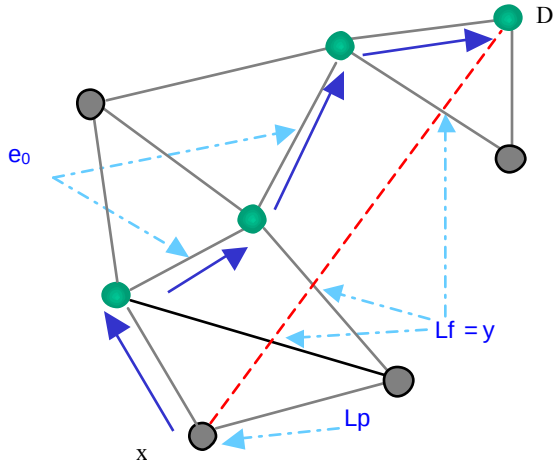

Fig. 8 Reenvío perimetral de paquetes en GPSR. Tomada de [34]

Tras la recepción de un paquete en modo perimetral, lo primero que hace GPSR es comparar la localización *Lp* con la localización del nodo del que está recibiendo el paquete, si la distancia de este nodo a *D* es menor que la distancia entre *Lp* y *D* GPSR retorna el paquete a modo voraz, el modo perimetral solo está pensado para recuperarse de un máximo local. En el ejemplo de la figura se realiza el reenvío perimetral hasta llegar al destino sólo para ilustrar el procedimiento del modo perimetral.

Es posible que el destino *D* no sea alcanzable porque está desconectado del grafo, pueden darse dos casos; que el nodo *D* esté en una cara interior o que esté en una cara exterior. GPSR reenviará el paquete hasta que alcance la cara interior o exterior según corresponda, en esta momento el paquete girará alrededor de toda la cara sin encontrar una intersección entre los bordes y la línea *xD* a un punto más cercano de *D* que *Lf*. Cuando el paquete atraviesa por segunda vez el primer borde que tocó sobre esta cara GPSR nota la repetición pues esta información está almacenada en el campo e_0 del paquete por lo que descarta este paquete porque el destino es inalcanzable. La travesía del grafo en modo perimetral a un

destino alcanzable nunca envía un paquete a través del mismo enlace (borde) dos veces en la misma dirección.

El algoritmo GPSR tal como se propone en [34] requiere hacer las siguientes:

- Cada nodo conoce su localización geográfica
- La conectividad de radio es bidireccional
- El modelo de radio utilizado es el ideal con cobertura circular
- Los nodos se mueven dentro de un plano (z=0)
- Para cada paquete se conoce la localización de la fuente y el destino

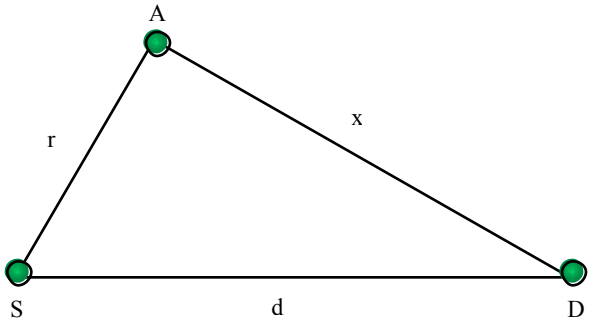

Fig. 9 Selección del mejor vecino en el protocolo Power Progress. Tomada de [35]

Power Progress [35]. Es un protocolo basado en la localización de los nodos, que utiliza además información de potencia como métrica de costo para la toma de decisiones de enrutamiento. Al igual que GPSR, Power Progress utiliza los modos de operación voráz y perimetral, siendo igual el esquema de funcionamiento general en el que el modo voráz es el modo por defecto y el perimetral es un modo de operación excepcional. La diferencia con GPSR radica en que en el modo voráz de Power Progress, la decisión sobre el siguiente nodo al que se reenviará el paquete no depende de la cercanía geográfica al destino sino de una relación entre potencia y progreso.

En [35] describen el algoritmo Power Progress con base en la Fig. 9, en la que $r = |SA|$, $d = |SD|$ y $x = |AD|$, con $x < d$. El algoritmo se basa en la noción de progreso proporcional el cual se mide como la potencia utilizada en un salto para lograr un determinado progreso hacia el destino. En este algoritmo el costo es función de la potencia requerida para la transmisión de mensajes entre fuente y destino. Utilizando el modelo de energía de [35] que

es el mismo utilizado en [17] y que describimos en la Fig. 2 de la sección 2.2.1, la potencia requerida para enviar un paquete de *S* hasta *A* es proporcional a la α potencia de la distancia y al consumo de energía en los dispositivos electrónicos del módulo de radio, expresado matemáticamente la potencia requerida para enviar un paquete desde *S* hasta *A* es: $r^{\alpha} + c$ y la porción de progreso lograda con este envío es $d - x$; continuando de esta forma, para llegar de *S* hasta *D* se requeriría reenviar el paquete a través de *n* saltos con $n = d/(d-x)$ y el costo total del envío del paquete entre fuente y envío sería $(r^{\alpha} + c_e) \cdot d/(d-x) = (r^{\alpha} + c_e) \cdot n$.

Para el algoritmo Power Progress, en cada salto o reenvío del paquete se debe utilizar el vecino que minimice la relación potencia/progreso. Con base en la Fig. 9 podemos describir el reenvío de paquetes en este algoritmo, allí el nodo *A* hacia el que se reenviará el paquete será aquel que minimice la relación potencia progreso $(r^{\alpha} + c_e)/(d-x)$. De esta forma el vecino seleccionado minimiza la potencia utilizada por unidad de avance realizado hacia el destino, es decir, se seleccionará el nodo que con la mínima potencia permita el mayor avance hacia el destino.

IPOW Iterative Power Progress [35]. Es una versión mejorada de Power Progress propuesta por los mismos autores de este. En el algoritmo IPOW el nodo *S* inicialmente selecciona el siguiente nodo al que transmitirá un paquete, en la misma forma en que lo hace el algoritmo Power Progress, es decir que utilizando la Fig. 9 seleccionará el nodo *A* que minimiza la relación potencia progreso $(r^{\alpha} + c_e)/(d-x)$; pero en lugar de que *S* entregue el paquete directamente a *A*, IPOW busca un nodo intermedio *B* que sea vecino de *S* y de *A* Fig. 10, que esté más cerca del destino *D* que *S*, y que satisfaga el requerimiento de que la potencia para transmitir desde *S* hasta *B* más la potencia para transmitir de *B* hasta *A* sea la mínima y sea menor que la potencia requerida para transmitir directamente desde *S* hasta *A*, esto es:

- Potencia (|*SB*|) + Potencia (|*BA*|) = min [Potencia (|*SB*|) + Potencia (|*BA*|)], y

- Potencia (|*SB*|) + Potencia (|*BA*|) < Potencia (|*SA*|).

Si se encuentra un nodo *B* que cumpla con estos requerimientos, este nodo *B* reemplaza a *A* como vecino seleccionado y la búsqueda por un nodo intermedio aún mejor se continúa de manera iterativa hasta que no sea posible encontrar un mejor vecino. Entonces el nodo *S* reenviará el mensaje al nodo seleccionado y este a su vez volverá a aplicar el mismo proceso de búsqueda de un nodo *A*.

La parte iterativa de IPOW se puede ilustrar mediante la asignación de costos a los enlaces, de tal forma que en lugar de transmitir un paquete directamente desde el nodo *S* hacia el nodo preseleccionado *A*, lo que puede tener un costo de 1000 unidades, busca una ruta alterna *SBA* en la que la suma de los costos sea menor, 800 unidades en el ejemplo de la Fig. 9, y las iteraciones en búsqueda de rutas seguirán para el enlace *BA* en búsqueda de rutas aún menores 300 unidades continuando con el mismo ejemplo.

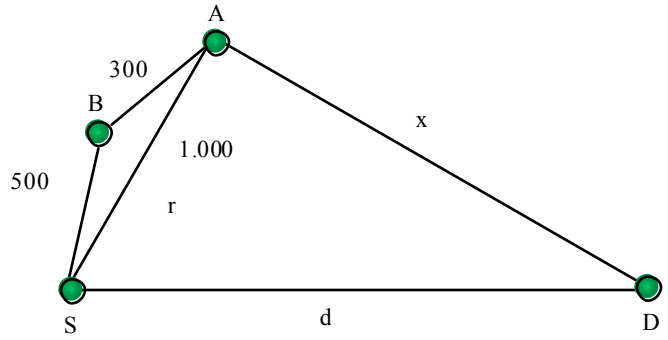

Fig. 10 Selección del mejor vecino en el protocolo Iterative Power Progress. Tomada de [35]

Como lo hemos mencionado previamente, el modelo de energía utilizado normalmente en la literatura consultada [17], [36] y en el presente trabajo, tiene en cuenta el consumo debido a la potencia requerida para transmitir una señal a una distancia *d*, y el consumo debido a la electrónica tanto del transmisor como del receptor, es decir, una parte del consumo es dependiente de la distancia y otra parte es independiente de esta. Siendo así existirá una distancia mínima por debajo de la cual no es posible reducir el consumo. Esta distancia es [36]:

$$d_{\min} = \sqrt[k]{\frac{C_e}{k-1}}$$

La distancia mínima para los valores de $\alpha = 4$ y $C_e = 100.000$ reportados en [36] y usados por nosotros será entonces de 13,5 m. Por esta razón en la programación de nuestro algoritmo hemos incluido como condición que no se busque vecinos intermedios para la transmisión si la distancia entre dos nodos vecinos *S* y *A* es menor que 13,5 m.

El seudo código para el algoritmo IPOW y que utilizaremos en nuestra implementación es [35]:

Entradas: Nodo fuente actual ***S***, nodo destino ***D***, y distancia ***d*** = |***DS***|

Encontrar un vecino ***A*** que cumpla con: **Potencia (|*SA*|)/(*d* - *x*) es mínimo**

Hacer ***M*** = ***A***

Repetir

Hacer **Min = Potencia (|*SM*|)**;

Hacer ***N*** = ***M***

Para cada vecino ***B*** de ***S*** (Que esté más cerca de ***D*** que ***S***)

Si ***B*** es vecino de ***M*** entonces

Si {**Potencia (|*SB*|)** + **Potencia (|*BM*|)**} < **Min**

Hacer **Min** = **Potencia (|*SB*|)** + **Potencia (|*BM*|)**

Hacer M = B

Fin

Fin

Hasta que ***M*** = ***N***

(***S*** reenvía el mensaje a ***M***)

Las métricas de potencia se pueden reemplazar por métricas basadas en costo, o por combinaciones de las dos (potencia y costo) con las que se construyen otros algoritmos basados en progreso tal como se describe en [35]. Según la misma referencia, las métricas de desempeño utilizadas para los algoritmos basados en potencia son: la cantidad total de

potencia requerida para enrutar exitosamente un mensaje desde la fuente hasta el destino y la rata de éxitos alcanzada en la búsqueda de rutas entre fuente y destino.

3.2.8 Conclusiones sobre los protocolos de enrutamiento

A partir de la revisión bibliográfica que hemos realizado durante nuestra investigación, se ha hecho evidente la gran cantidad de protocolos de enrutamiento existentes para redes de sensores, tanto diseñados específicamente para estas redes, como los extendidos desde redes ad-hoc para su utilización en redes de sensores. No obstante esto, la investigación sobre el tema sigue presentando un alto dinamismo; incluso en los protocolos más reconocidos, aún hay aspectos que representan oportunidades de mejora y que se convierten en puntos de partida para nuevos proyectos de investigación.

La decisión sobre la conveniencia de utilizar uno u otro protocolo de enrutamiento depende de diversos factores que van desde el tipo de aplicación, de la forma en que se despliegan los nodos en el medio, hasta las particularidades de los sensores y del medio en que estos están inmersos. En el análisis de energía de los protocolos de enrutamiento realizado en [37] se compara el comportamiento en lo referente a energía, de los protocolos de comunicación directa entre nodos y nodo destino y los de mínima energía que utilizan varios saltos entre nodos hasta el destino. El resultado reportado es que el tipo de protocolo con mejor comportamiento depende de la circuitería utilizada para transmisión y de la topología de la red. Si la circuitería de transmisión consume mucha energía comparada con la del amplificador y/o las distancias de transmisión son cortas, los protocolos de comunicación directa son más eficientes que los de enrutamiento de mínima energía. En cuanto al agrupamiento (clustering) convencional, aunque parece eficiente en energía, su comportamiento depende de que los líderes de grupo (cluster heads) sean nodos con buenas reservas de energía, y de la adecuada selección de parámetros para el agrupamiento.

Capítulo 4. Simulación de protocolos de nivel de red

4.1 Simulador utilizado

Para nuestras simulaciones utilizamos ns-2 versión 2.29 [38]. ns-2 es uno de los simuladores de red más conocidos en la comunidad académica y científica. Es un simulador por eventos discretos orientado a objetos, libremente distribuido y de código abierto que se ejecuta sobre el sistema operativo Linux y que permite la simulación tanto de redes alambradas como de redes inalámbricas [39] mediante la utilización de generadores de tráfico, protocolos de nivel mac, protocolos de enrutamiento y modelos de propagación, entre otros módulos que tiene incluidos en su distribución oficial y que se pueden desarrollar.

El núcleo de ns-2 está escrito en C++ y la interfase con el usuario en OTcl que es una extensión orientada a objetos de Tcl, estos dos lenguajes permiten una adecuada velocidad de ejecución a la vez que facilitan la configuración de parámetros de simulación por parte del usuario. En la Fig. 11 se presenta el esquema general de funcionamiento de ns-2; las entradas al simulador se realizan a través de scripts oTcl en los que se describen los parámetros de configuración de la simulación; el interprete oTcl se encarga de realizar las acciones descritas en el script mediante interacciones con la biblioteca de funciones de ns-2; y a partir de estas interacciones se generan archivos de trazas con la información resultante de la simulación.

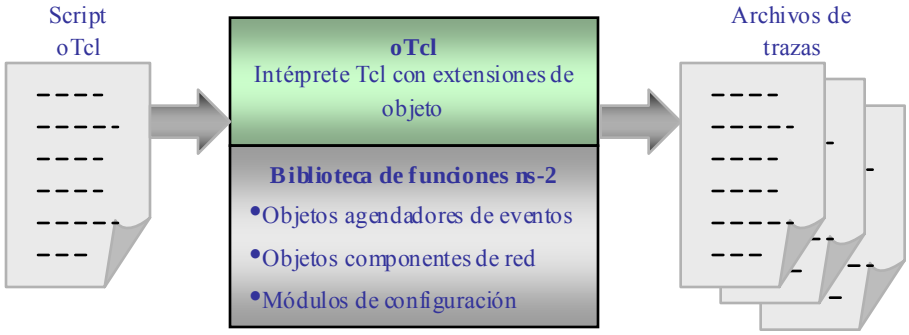

Fig. 11 Arquitectura de ns-2. Tomada de [40]

El uso de ns-2 está ampliamente difundido tanto a nivel investigativo como educativo, lo que se demuestra con la gran cantidad de reportes en que aparece en la literatura, llegando a convertirse en un estándar de facto para la simulación de redes.

A pesar de que ns-2 se ha utilizado constantemente en investigaciones sobre redes inalámbricas de sensores y que existen extensiones desarrolladas para protocolos de enrutamiento como LEACH, PEGASIS y GPSR, la versión 2.29 incluye relativamente pocos módulos específicos para sensores. A nivel MAC incluye S-MAC y 802.15.4; y a nivel de red sólo incluye el protocolo de enrutamiento Directed Difussion.

4.2 Implementación del protocolo IPOW

Como ns-2 se desarrolló para ejecutarse sobre el sistema operativo Linux, para nuestro proyecto debimos instalar ns-2 sobre Cygwin [40] que es un emulador de Linux sobre Windows.

El aprendizaje de ns-2 es un proceso que requiere el estudio detallado de la extensa documentación existente entre la que el manual de ns-2 [39] es un referente con buenos niveles de profundidad incluyendo detalles de la implementación del simulador. Sin embargo, es recomendable iniciar el proceso de aprendizaje a través de tutoriales como el de Marc Greiss [41] y el ns by example [42] en los que se describe paso a paso el proceso de generación de scripts de simulación y la utilización progresiva de los módulos que incluye el simulador.

Para realizar la implementación de módulos sobre ns-2 debe conocerse a buen nivel de detalle el funcionamiento del simulador y su arquitectura. En [43] se describe el proceso de implementación de protocolos de enrutamiento unicast en ns-2.

La implementación del protocolo parte del conocimiento de la estructura de clases de ns-2 [39] para decidir la estructura de herencia que le permitirá al nuevo protocolo interactuar de

manera adecuada con los módulos ya existentes en el simulador de acuerdo al nivel en el que el protocolo debe ubicarse que para nuestro caso es el nivel de red.

En ns-2 un agente [39] representa un punto extremo en el que los paquetes de nivel de red se construyen o se consumen y se utilizan para la implementación de protocolos en varios niveles de la pila OSI. Por eso para la implementación del agente IPOW debemos heredar de la clase **Agent**.

Para el funcionamiento normal de un protocolo se requiere el envío periódico de paquetes de control por lo que también debemos tener en cuenta la clase **Timer**. Otra clase importante para la implementación es la clase **Trace** que permite la generación de archivos de registro o trazas que se constituyen en la base para el análisis de los resultados de funcionamiento del protocolo.

La unidad básica de información en ns-2 es el paquete de datos por lo que la clase **packet** también es de interés en nuestra implementación. El intercambio de paquetes entre los nodos requiere la toma de decisiones sobre el destino del paquete dependiendo de si el destino es el nodo actual o es un nodo diferente. Para el primer caso ns-2 define la clase **PortClassifier** que permite el paso del paquete para protocolos de nivel superior dentro del nodo.

El enlace entre los scripts de simulación en oTcl y la implementación se logra heredando de la clase **TclClass**.

El primer paso de la implementación consiste en la creación de una carpeta con el mismo nombre del protocolo a implementar, IPOW que debe estar ubicada dentro del directorio base de ns-2 (/home/ns-allinone-2.29/ns-2.29/ipow/). Dentro de esta carpeta se deben crear los archivos que determinan el comportamiento del protocolo. Los archivos que se crearon son:

ipow_packet.h: en este se definen las estructuras de datos requeridas para el intercambio de paquetes del protocolo IPOW.

ipow.h: en este archivo se define el agente de enrutamiento, los temporizadores y las clases que permiten definir los datos y comportamiento del protocolo.

ipow.cc: en el que se realiza la implementación de los métodos definidos en la clase ipow.h y la conexión entre C++ y oTcl.

ipow_neighbor.h: en este archivo se definen las clases y métodos correspondientes a la definición de la lista de vecinos, y todo lo relacionado con el enrutamiento propiamente dicho.

ipow_neighbor.cc: archivo en el que se realiza implementación de los métodos definidos en las clases ipow_neighbor.

ipow_sinklist.h: en este archivo se encuentra la clase en la que se definen la lista de nodos de destino que, aunque en el proceso siempre fue uno sólo pueden ser múltiples destinos lo que permite flexibilidad en las aplicaciones sobre las que se puede utilizar el protocolo.

ipow_sinklist.cc: en este se realiza la implementación de los métodos definidos en las clases ipow_sinklist.

En el anexo A se incluye el código en C++ que se desarrolló para la implementación del protocolo IPOW en ns-2 y que está constituido por los archivos que acabamos de describir.

Además de la implementación de las clases, se requirieron algunas modificaciones en archivos de ns-2 para integrar en él el código desarrollado.

Se agregó el protocolo ipow en el archivo **packet.h** de la carpeta **common** (/home/ns-allinone-2.29/ns-2.29/common/packet.h) lo cual se realiza incluyendo la sentencia

PT_IPOW en **enum packet_t** antes de la sentencia: **PT_NTYPE**, y la sentencia **name_[PT_IPOW]="ipow"** dentro del método **p_info()**.

Se incluyó ipow en la cola de prioridades, lo que se logra agregando las sentencias **case PT_IPOW**: **recvHighPriority(p, h); break;** en el archivo **cmu-trace.cc** de la carpeta **trace** (/home/ns-allinone-2.29/ns-2.29/queue/priqueue.cc)

Para integrar ipow con las funcionalidades de generación de trazas de ns-2 se debe agregar las siguientes líneas en el archivo **cmu-trace.cc** de la carpeta **trace** (/home/ns-allinone-2.29/ns-2.29/trace/cmu-trace.cc): **#include <ipow/ipow_packet.h>**; también se debe agregar la sentencia **case PT_IPOW: break;** en el método **format(Packet* p, const char *why)**

Para incluir nuestro tipo de paquete ipow en Tcl, debemos agregar la línea **IPOW** en **foreach prot** del archivo **ns-packet.tcl** (/home/ns-allinone-2.29/ns-2.29/tcl/lib/ns-packet.tcl). Para definir los valores por defecto de los parámetros de nuestro protocolo, debemos hacerlo en el archivo **ns-lib.tcl** de la misma carpeta.

Una vez que se han generado las clases necesarias para el protocolo de enrutamiento y se han realizado las modificaciones anteriormente descritas, sólo debe modificarse el archivo **Makefile** (/home/ns-allinone-2.29/ns-2.29/Makefile) para que permita compilar los archivos en los que se encuentra definido el nuevo protocolo y este sea funcional dentro del simulador. Para esto debe agregarse en él: **ipow/ipow.o ipow/ipow_neighbor.o ipow/sinklist.o** y escribir **Makefile** en la interfase de comandos para que empiece la compilación.

4.3 Escenarios de simulación

Además de la implementación del protocolo se debieron construir los scripts necesarios para realizar las simulaciones que nuestro proyecto requería. Dichos scripts se incluyen en el anexo B y se describen brevemente a continuación.

wireless-ipow.tcl: Programa principal, que incluye la configuración de parámetros físicos de la red, y de la simulación, además de la generación de archivos de trazas.

ipow.tcl: en él se realiza la creación de los nodos y del agente de enrutamiento

cbr100.tcl: se utiliza al nivel de aplicación para la definición del tipo de tráfico a utilizar, velocidad de transmisión, tamaño de paquetes, etc.

Grid-deploy10x10_rdm.tcl: script en el que se define la topología: tamaño del área, posición aleatoria de los nodos dentro del área geográfica.

Además de estos scripts, utilizamos versiones modificadas de los mismos para la simulación del protocolo GPSR.

Para la simulación utilizamos los mismos escenarios que en [44] y [45] para poder comparar el desempeño de los algoritmos con resultados reportados en las mismas referencias. Se definen cinco topologías de áreas distintas y con diferentes cantidades de nodos en las que los nodos se distribuyen aleatoriamente dentro del área por medio de una distribución uniforme.

Escenario	Número de nodos	Nodo fuente	Dimensiones de la región (m)	Densidad promedio de nodos (vecinos por nodo)
1	50	49	160 x 160	5,68
2	100	99	230 x 230	6,90
3	150	149	277 x 277	6,60
4	200	199	320 x 320	6,88
5	250	249	360 x 360	6,48

Tabla 4. Topologías utilizadas

Para la ubicación de los nodos en la topología se utiliza una distribución aleatoria uniforme entre 0 y la longitud de un lado del cuadrado, excepto para el nodo fuente cuya ubicación se hace aleatoriamente pero siguiendo una distribución uniforme entre 0 y 70 m. A partir de esta topología se generan cinco escenarios distintos en los que se modifica el número de

nodos entre 50 y 250 en saltos de 50, y la longitud del cuadrado entre 160 m y 360 m tal como se puede observar en la tabla 4.

En los experimentos realizados con estos escenarios, la fuente de información es el nodo número n-1 (49, 99, 149, 199 y 249) y el destino es el nodo 0. En las figuras Fig. 12 hasta Fig. 16 se pueden observar 5 escenarios de la topología aleatoria.

En la tabla 5 se listan los parámetros de nivel físico que se utilizaron en las simulaciones y que son iguales a los de [44] y [45] para tener igualdad de condiciones en las comparaciones. Estos parámetros corresponden a una tarjeta WaveLan de Lucent.

Para la propagación se utilizó el modelo de propagación de dos rayos (TwoRayGround) porque es el adecuado para distancias mayores a la distancia de crossover según se especifica en [39] como la distancia máxima para la cual el modelo de propagación en espacio libre se puede utilizar y desde la que debe utilizarse el modelo de dos rayos terrestres. Para la frecuencia de 914 MHz que utilizamos en nuestros experimentos, la distancia de crossover es de 9,57 m, y nuestro rango de radio es de 40 m.

Parámetro ns-2	**Valor**
Energía inicial	80 J
Consumo de potencia en transmisión	0,660 W
Consumo de potencia en recepción	0,396 W
Consumo de potencia en estado ocioso (idle)	0,035 W
Frecuencia de transmisión	914 MHz
Umbral de recepción (sensitividad)	$3{,}652 \times 10^{-10}$ W
Umbral para evitar colisiones	$1{,}559 \times 10^{-11}$ W
Umbral de colisiones	10 dB
Exponente de pérdidas por trayecto	4
Altura de antenas tx y rx	0,5 m
Rango de radio	40 m

Tabla 5. Parámetros de nivel físico

El modelo de energía de ns-2 [39] toma como referencia la energía inicial y la va decrementando con cada evento por un valor igual al producto de la potencia de transmisión por el tiempo de transmisión, o al producto de potencia de recepción por el tiempo de recepción según corresponda el evento.

Para los niveles físico y MAC utilizamos el IEEE 802.11 para realizar la simulación en igualdad de condiciones de las referencias [44] y [45] con las que se realizan las comparaciones, pero a manera de comparación también se pueden utilizar IEEE 802.15.4 o S-MAC.

Las antenas utilizadas son antenas omnidireccionales de ganancia unidad. Y las pérdidas *L* también se consideran de valor unitario.

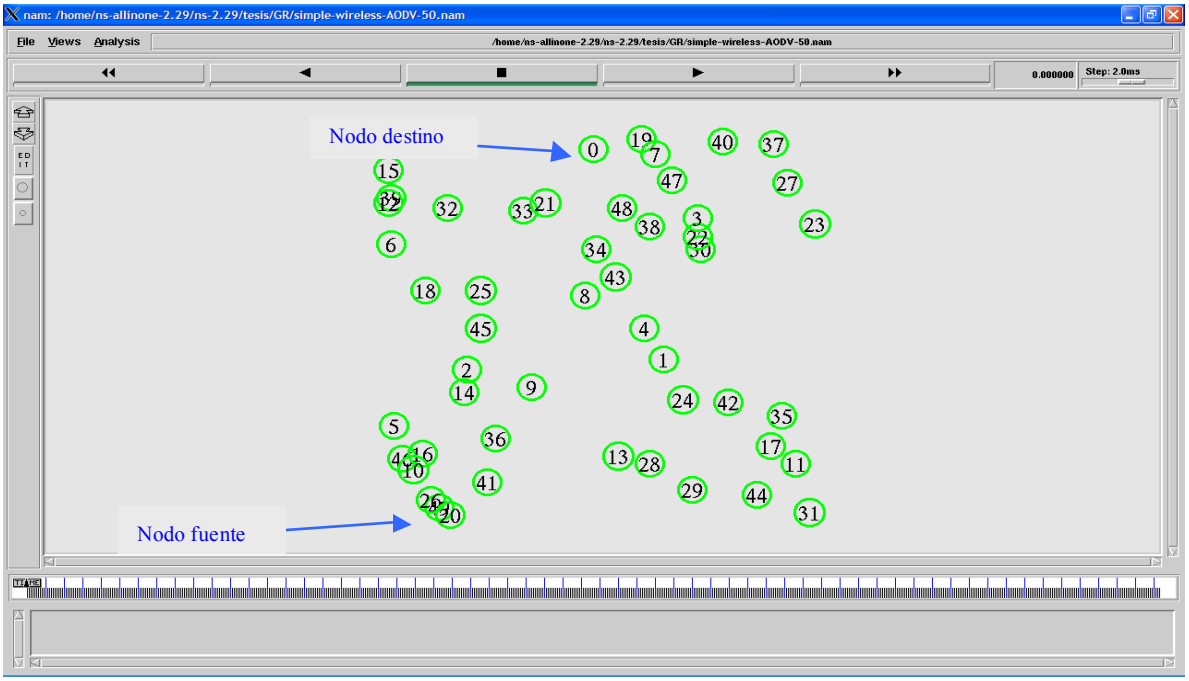

Fig. 12. Escenario 1, 50 nodos distribuidos aleatoriamente.

En el nivel de red, utilizamos los algoritmos de enrutamiento IPOW y GPSR. La simulación de GPSR se hizo con base en el código para ns-2 reportado en [46] para el cual se realizaron los ajustes necesarios para su compilación y ejecución en la versión 2.29 del simulador.

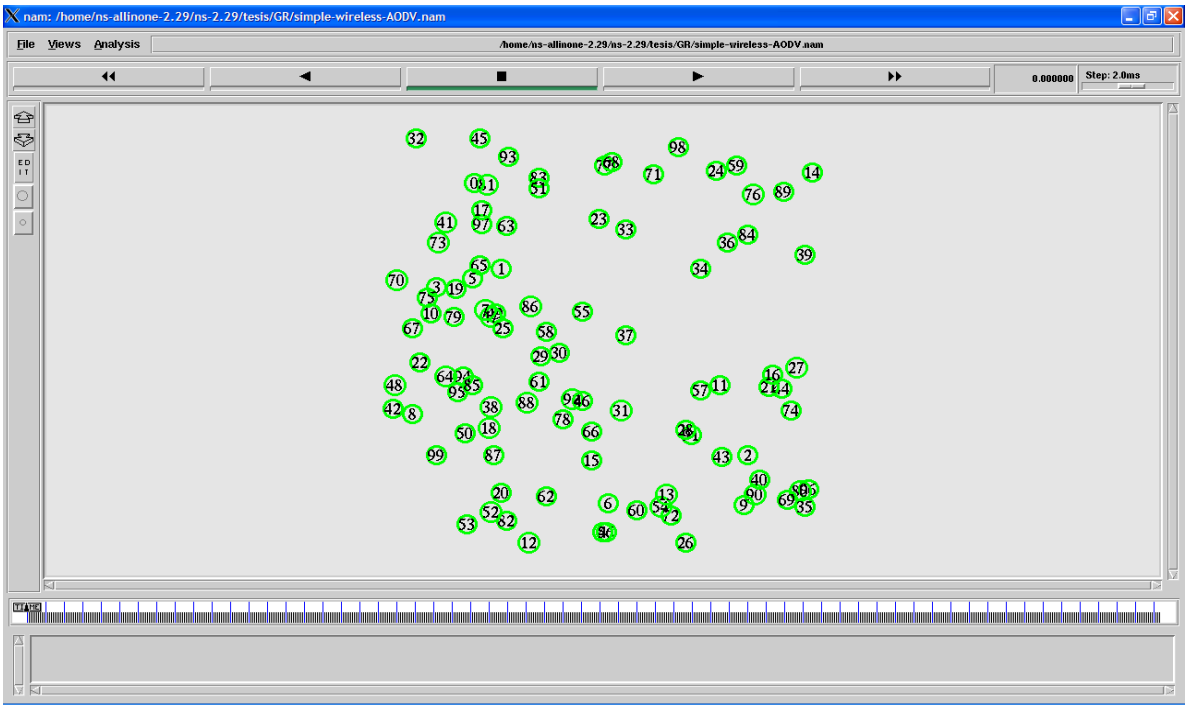

Fig. 13. Escenario 2, 100 nodos distribuidos aleatoriamente.

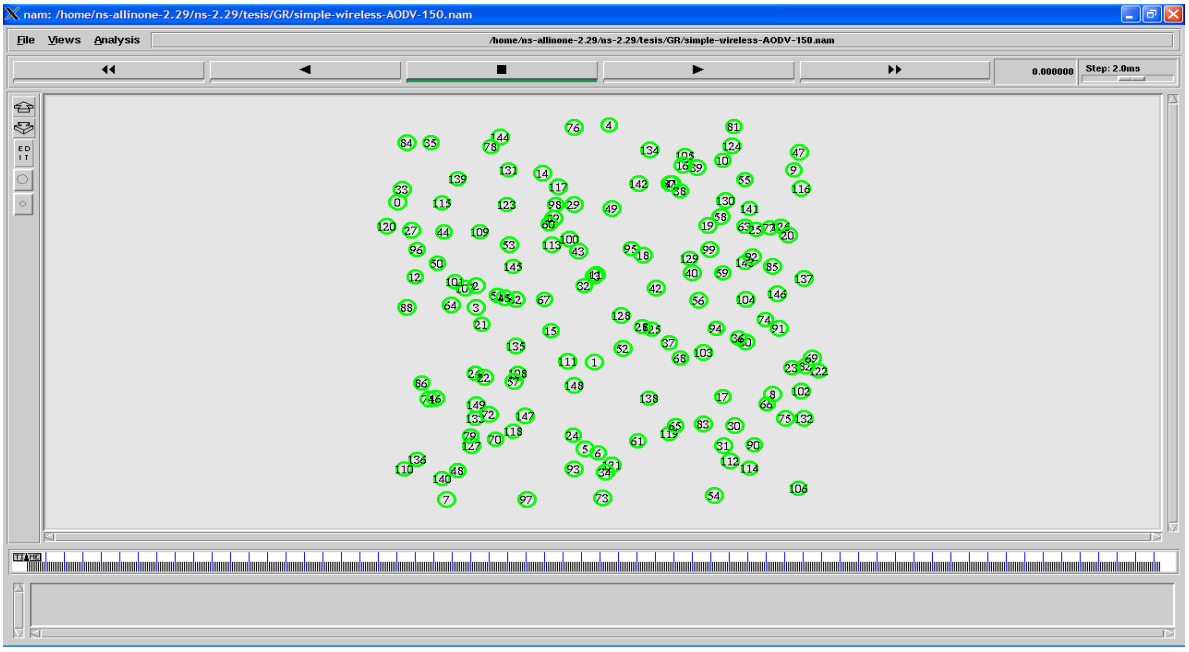

Fig. 14. Escenario 3, 150 nodos distribuidos aleatoriamente.

Para IPOW utilizamos un período de envío de paquetes Hello de 5 segundos siguiendo la recomendación de [35]. A nivel de aplicación utilizamos paquetes UDP de 32 bytes generados por fuentes de rata constante de bits (CBR), se envía un paquete cada dos

segundos desde el nodo fuente definido en cada escenario, hacia el nodo destino que en todos los escenarios es el nodo 0. En las figuras desde la Fig. 12 hasta la Fig. 16 se presentan las topologías generadas para cada uno de los escenarios.

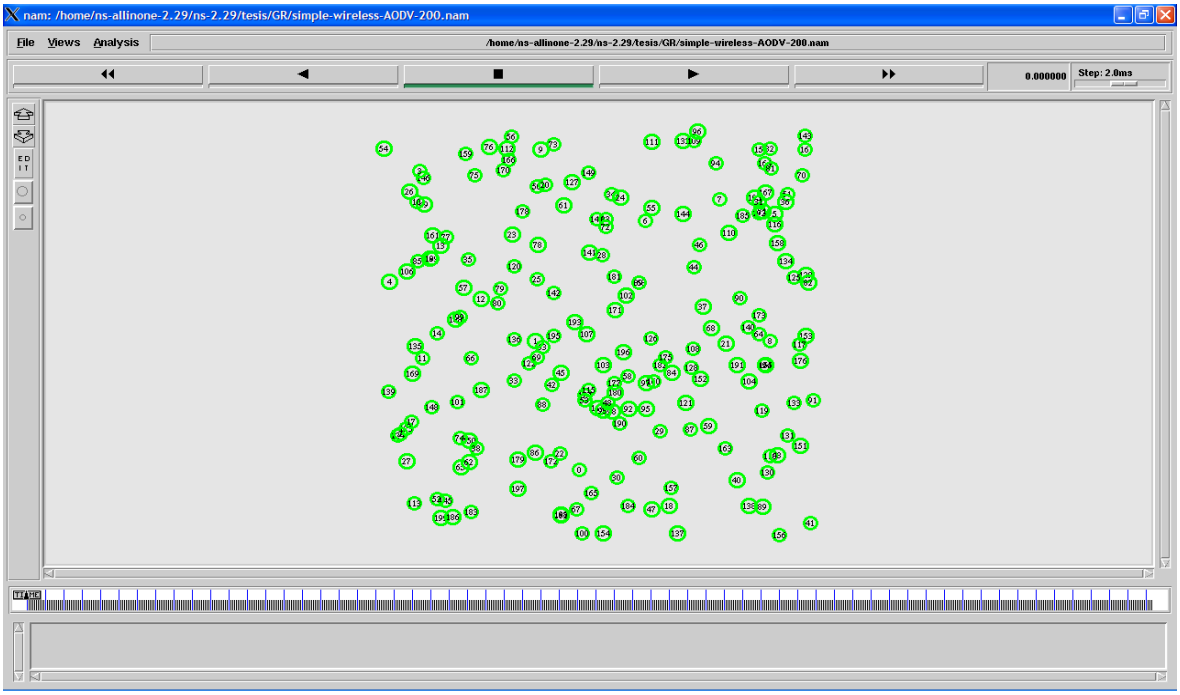

Fig. 15. Escenario 4, 200 nodos distribuidos aleatoriamente.

Para cada uno de estos escenarios se mantuvieron los mismos parámetros de simulación y se simularon en condiciones idénticas de funcionamiento los protocolos IPOW y GPSR. Cada simulación tuvo una duración de 70 segundos.

Es muy importante resaltar que para las simulaciones hacemos algunas suposiciones que dependen tanto de los modelos utilizados en el simulador, como de las características propias de los algoritmos utilizados. Las suposiciones son:

- En todos los escenarios los nodos están desplegados sobre una superficie plana (z = 0),
- El radio de cobertura de los nodos es circular ideal según lo determina la distancia del modelo de reflexión de dos rayos terrestres de ns-2 [39],

- Los nodos permanecen fijos durante la simulación en las posiciones establecidas aleatoriamente para cada topología. Esto para mantener igualdad de condiciones con los experimentos realizados en [44] y [45], pues la generación de movimiento en los nodos es sencilla en ns-2 y puede incluirse en las simulaciones modificando los scripts.
- Todas las simulaciones se realizan con una única fuente y un único destino, lo cual se hace para igualdad de condiciones con las referencias [44] y [45] pero tanto el protocolo GPSR como el protocolo IPOW implementado permiten la utilización de múltiples fuentes y múltiples destinos.
- Asumimos que la red es homogénea, es decir, todos los nodos presentan idénticas características.
- No tenemos en cuenta errores en la transmisión.

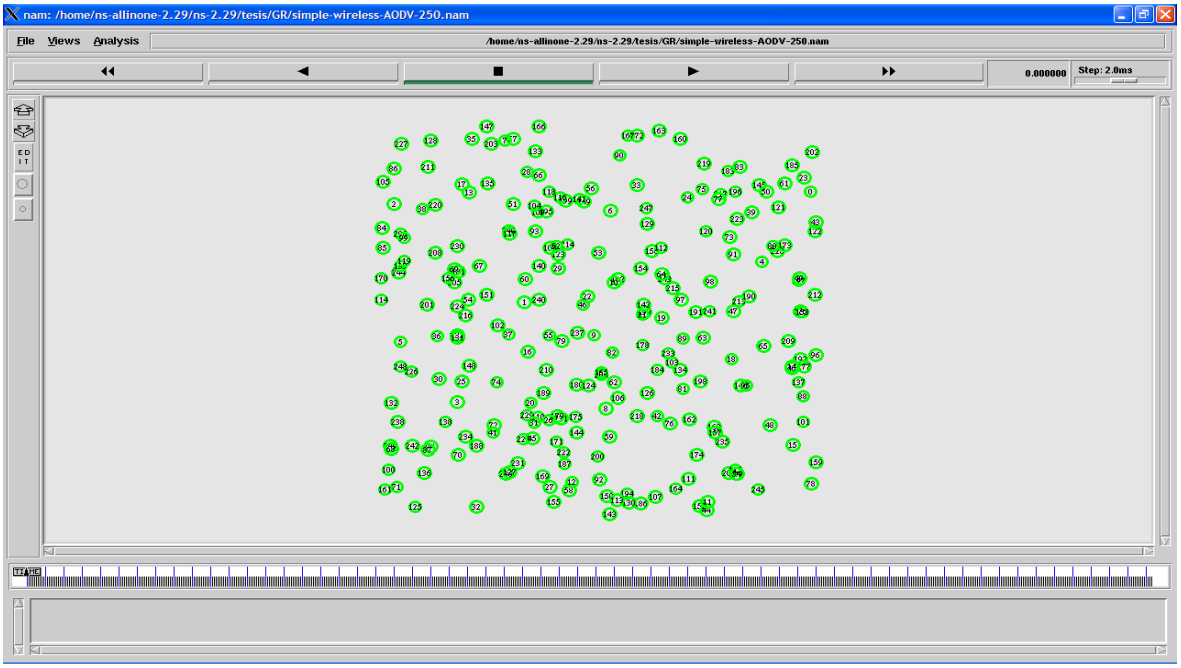

Fig. 16. Escenario 5, 250 nodos distribuidos aleatoriamente.

Para la determinación de las rutas en IPOW se utiliza el modelo de energía utilizado en [17] y [36] en el que el consumo de energía depende de la α-ésima potencia de la distancia y de la energía consumida por los dispositivos electrónicos de transmisión y recepción.

4.4 Resultados obtenidos

Para analizar los resultados de la simulación y validarlos mediante comparaciones con los reportes encontrados en la literatura, después de cada ejecución procesamos los archivos de trazas obtenidos para obtener las siguientes métricas:

Energía total disipada en la red: [45]: Indica la suma de energía consumida en todos los nodos de la red.

Promedio de energía disipada [44]: indica la cantidad media de energía consumida por los nodos para hacer que un paquete de información llegue al destino. Se obtiene dividiendo la cantidad total de energía disipada en la red sobre el total de nodos de la red, y este resultado se divide por la cantidad total de paquetes recibidos en el destino durante la ejecución de la simulación.

Retardo promedio extremo a extremo [44]: indica el retardo promedio que sufre un paquete al ir del nodo fuente al nodo destino. Se obtiene restando el tiempo de arribo del paquete y el tiempo de envío del mismo.

Número de campo	Valor ejemplo	Descripción del campo
1	r	Es una letra indicando el tipo de evento que se registra. puede tener los valores: r, s, f, D indicando recibido, enviado, re-enviado y descartado respectivamente. También puede ser una M para indicar la posición del nodo o movimientos del mismo.
2	50.446175719	Tiempo del evento
3	_35_	Número de nodo
4	RTR	Tipo de traza . Puede ser MAC, AGT, RTR, IFQ
5	- - -	
6	618	Número de secuencia del paquete
7	IPOW	Tipo de paquete, puede ser: TCP, ACK, IPOW, GPSR
8	29	Tamaño del paquete en bytes
9	[0 ffffffff 0 800]	Información del nivel MAC: tiempo para enviar el paquete sobre el canal inalámbrico, id de MAC en nodo envío, id de MAC en nodo recepción, el último segmento es el tipo MAC, 800 es ETHERTYPE_IP
10	[energy 79.961305]	Nivel de energía en el nodo
11	-------	
12	[0:255 -1:255 32 0]	Direcciones IP del nodo fuente y del nodo destino, TTL del paquete

Tabla 6. Formato de la traza utilizada en las simulaciones.

```
r 55.490762377 _36_ RTR  --- 674 gpsr 29 [0 ffffffff 1a 800] [energy 79.903056] ------- [26:255 -1:255 32 0]
s 55.496528795 _49_ AGT  --- 675 cbr 32 [0 0 0 0] [energy 79.932622] ------- [49:0 0:0 32 0]
r 55.496528795 _49_ RTR  --- 675 cbr 32 [0 0 0 0] [energy 79.932622] ------- [49:0 0:0 32 0]
```

Fig. 17. Parte de un archivo de traza obtenido en la simulación.

El formato de los registros de la traza entregada por ns-2 a partir de las simulaciones y que utilizamos para obtener las métricas se presenta en la tabla 6. La traza genera, además de registros en el formato presentado, otros registros de solo seis campos en los que se especifica al comienzo de la traza la identificación de cada nodo y su posición dentro del área; también existen en el archivo de traza registros de tan sólo cuatro campos en los que se especifica para cada nodo el valor de la energía en determinados instantes de tiempo.
Un ejemplo de los registros de trazas generadas durante nuestros experimentos se presenta en la Fig. 17, en ella se puede observar el tipo de evento, el tiempo del evento, el número de nodo, el tipo de traza, el número de secuencia del paquete, el tipo de paquete, la longitud del paquete, información de MAC, información de energía disponible y por último información del protocolo IP según lo descrito en la Tabla 6.

Para el procesamiento de las trazas se utilizó un programa desarrollado en Acces que permitió la graficación de los resultados a través de la obtención del promedio de los datos obtenidos a partir de 10 simulaciones realizadas para cada protocolo en cada uno de los cinco escenarios.

La energía total disipada se obtuvo revisando para cada uno de los nodos al final de la simulación, el campo de energía disponible; haciendo la diferencia de energía inicial menos energía disponible, y sumando este resultado para todos los nodos.

La cantidad total de paquetes recibidos en el destino se obtiene filtrando las líneas tipo “r” que se registraron para el nodo destino (nodo 0). Para obtener la energía media disipada se dividen los dos datos obtenidos previamente y se vuelve a dividir sobre el número de nodos de la red.

El retardo promedio extremo a extremo se obtiene extrayendo de la traza las líneas “s” y “r” tipo CBR, y para las líneas con el mismo número de secuencia del paquete se hace la diferencia de tiempos y este resultado se divide entre el número de paquetes recibidos para promediarlo.

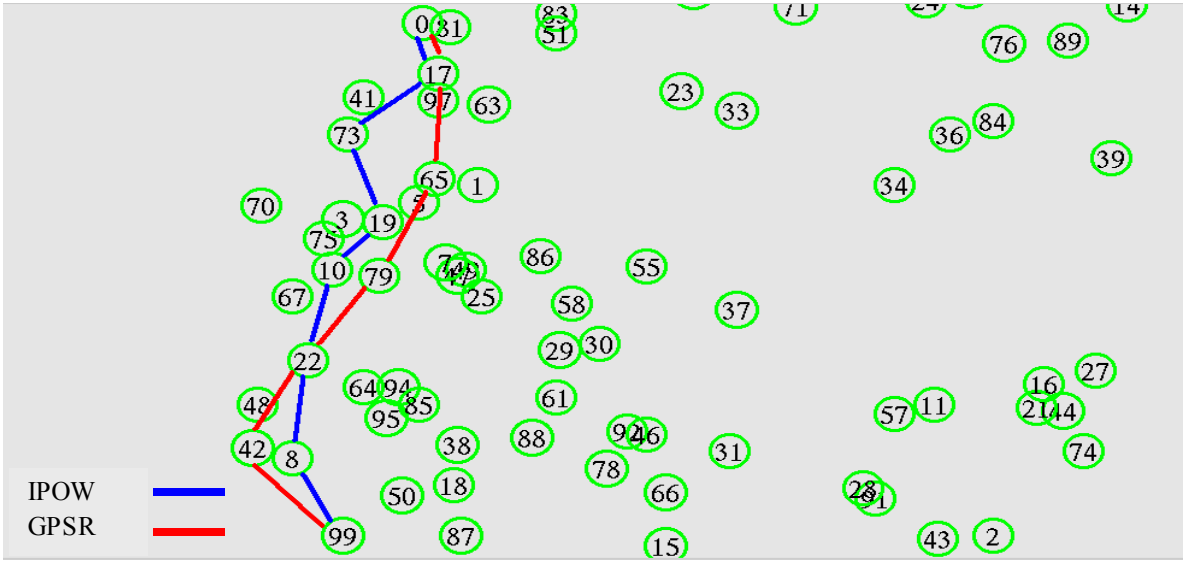

Fig. 18. Ruta de los paquetes según GPSR e IPOW en escenario 2.

En la Fig. 18 podemos ver que la ruta que siguen los paquetes en los algoritmos GPSR e IPOW para el escenario 2 es muy similar, aunque debemos recordar que GPSR re-envía el paquete hacia el nodo que le de el mayor avance geográfico, mientras que IPOW lo hace hacia el nodo que le de una mejor relación costo energético a progreso hacia el destino. GPSR utiliza 6 saltos para llegar a la fuente mientras que IPOW utiliza siete, esto se debe a la naturaleza de IPOW por lo que, siempre que encuentra un vecino con avance hacia el destino, el algoritmo intenta conseguir un nodo intermediario alterno que le permita llegar al vecino seleccionado a través de un enlace no directo que represente menores costos de energía (ver Fig. 10) y saltos adicionales.

Las métricas obtenidas a partir de la simulación se presentan en la Fig. 19, la Fig 20 y la Fig 21. Como se mencionó previamente, los datos graficados se obtuvieron promediando los datos obtenidos de 10 simulaciones para cada protocolo en cada escenario.

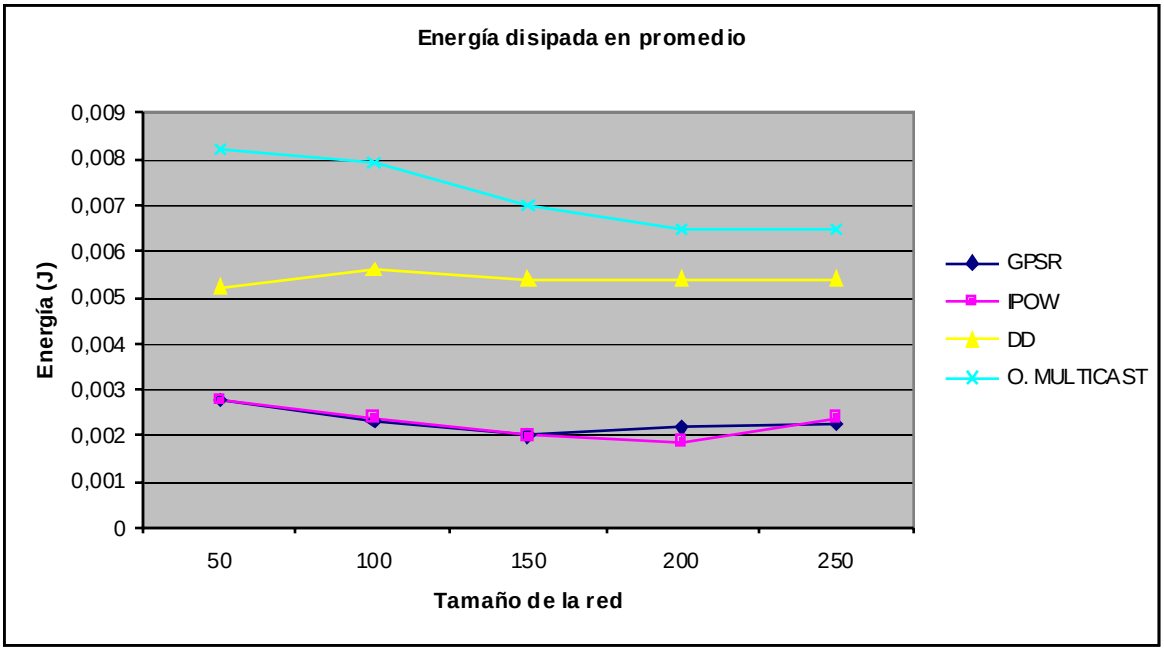

Fig. 19. Energía disipada en promedio para los cinco escenarios.

En la Fig. 19 se pueden observar los resultados del consumo promedio de energía en la red para los distintos tamaños de la misma para los dos algoritmos simulados, y su comparación con los resultados reportados en [44] para los algoritmos Directed Difussion (DD) y Omniscient Multicast. Se seleccionó esta referencia para comparación con nuestros resultados porque en ella se incluye el algoritmo Directed Difussion (DD) que es el único algoritmo de enrutamiento diseñado específicamente para redes inalámbricas de sensores que está incluido en la implementación oficial de ns-2, y nos parece muy importante realizar las comparaciones con un algoritmo que se haya diseñado específicamente para redes inalámbricas de sensores, pues en la literatura [47], [48] y [49] hemos encontrado que algunas comparaciones se hacen contra algoritmos como AODV o DSR que se diseñaron para redes Adhoc y por lo tanto no consideran aspectos importantes de diseño de las redes inalámbricas de sensores por lo que la comparación no es adecuada.

Para las comparaciones procuramos que las condiciones de simulación con respecto a las reportadas en las referencias [44] y [45] fueran idénticas para generar confianza en los resultados, evitando lo encontrado en reportes como los de [31] en que las comparaciones se realizan entre protocolos simulados bajo distintas condiciones de funcionamiento.

En la Fig. 19 es importante destacar que tanto IPOW como GPSR presentan un mejor desempeño en consumo de energía que los otros dos algoritmos. También se observa que para los distintos tamaños de red el comportamiento energético de IPOW y GPSR son muy similares, siendo aparentemente GPSR un poco mejor que IPOW lo cual no es consistente con lo esperado pues según lo reportado en [36] se presume que IPOW es el algoritmo de enrutamiento geográfico que mejor desempeño presenta en consumo energético.

Para aclarar lo anterior se debe tener en cuenta, que la energía media consumida depende directamente de la energía total consumida. Retomando la Fig. 18, revisamos la energía consumida en los nodos pertenecientes a la ruta seleccionada por cada algoritmo y la sumamos para finalmente comparar sus valores según se observa en la tabla 7. La ruta seleccionada por IPOW es: 99, 8, 22, 10, 19, 73, 17, 0; y la ruta seleccionada por GPSR es: 99, 42, 22, 79, 65, 17, 0.

En la tabla 7 se observa para cada protocolo, la energía consumida en cada salto por cada uno de los nodos participantes en el envío de un paquete de fuente a destino según los reportes del archivo de trazas de ns-2 en los que la potencia de transmisión es constante. Debido a esto, IPOW consumiría más energía pues en este caso hace un salto más que GPSR, y en general siempre hará más, pues parte de su estrategia de ahorro de energía está en reemplazar saltos directos entre dos nodos vecinos por múltiples saltos.

Saltos IPOW	**Energía consumida IPOW (J)**	**Saltos GPSR**	**Energía consumida GPSR (J)**
99 – 8	$3{,}38 \times 10^{-3}$	99 - 42	$3{,}38 \times 10^{-3}$
8 – 22	$3{,}38 \times 10^{-3}$	42 – 22	$3{,}38 \times 10^{-3}$
22 - 10	$3{,}38 \times 10^{-3}$	22 – 79	$3{,}38 \times 10^{-3}$
10 – 19	$3{,}38 \times 10^{-3}$	79 – 65	$3{,}38 \times 10^{-3}$
19 – 73	$3{,}38 \times 10^{-3}$	65 – 17	$3{,}38 \times 10^{-3}$
73 – 17	$3{,}38 \times 10^{-3}$	17 - 0	$3{,}38 \times 10^{-3}$
17 - 0	$3{,}38 \times 10^{-3}$		
Consumo total por paquete entregado	**$2{,}36 \times 10^{-2}$**		**$2{,}02 \times 10^{-2}$**

Tabla 7. Consumo de energía salto a salto en esenario 2

Volviendo a la Fig. 19, se debe aclarar que el desempeño en consumo de energía de IPOW no se refleja en ella debido a que, como se mencionó previamente, ns-2 impone una

potencia de transmisión constante para todos los nodos al inicio de la simulación por lo que aunque IPOW seleccione una ruta con menor costo energético hacia un vecino preseleccionado, el consumo de energía reportado por las trazas a través de esos saltos alternos será incluso mayor que el consumo a través del salto directo hacia el vecino preseleccionado pues se realizan más saltos. Para manejar esta situación hemos incluido en ns-2 un archivo de trazas adicional para IPOW que nos muestre, para todos los nodos participantes en la ruta generada por IPOW, la energía que se consumiría en los nodos de la ruta por cada salto si tuviéramos control de potencia de transmisión. En la tabla 8 se pueden ver los resultados de este proceso aplicado al mismo escenario de la tabla 7, con lo que la energía consumida en IPOW por los nodos participantes en el reenvío del mensaje es menor que la de GPSR en un orden de magnitud. El mismo procedimiento se puede aplicar en todos los escenarios y calcularse su impacto en la energía total y promedio consumida por la red lo cual se presenta en la Fig. 20, con lo que se confirma que IPOW presenta el mejor desempeño de los cuatro algoritmos en lo referente a consumo de energía.

Saltos IPOW	**Energía consumida IPOW (J)**	**Saltos GPSR**	**Energía consumida GPSR (J)**
99 – 8	$3{,}70 \times 10^{-4}$	99 - 42	$3{,}38 \times 10^{-3}$
8 – 22	$3{,}86 \times 10^{-4}$	42 – 22	$3{,}38 \times 10^{-3}$
22 - 10	$1{,}72 \times 10^{-4}$	22 – 79	$3{,}38 \times 10^{-3}$
10 – 19	$1{,}19 \times 10^{-4}$	79 – 65	$3{,}38 \times 10^{-3}$
19 – 73	$1{,}47 \times 10^{-4}$	65 – 17	$3{,}38 \times 10^{-3}$
73 – 17	$9{,}67 \times 10^{-4}$	17 - 0	$3{,}38 \times 10^{-3}$
17 - 0	$3{,}91 \times 10^{-5}$		
Consumo total por paquete entregado	$\mathbf{2{,}20 \times 10^{-3}}$		$\mathbf{2{,}02 \times 10^{-2}}$

Tabla 8. Consumo de energía salto a salto corregido para IPOW en escenario 2

En la Fig. 21 se presentan los resultados obtenidos por los algoritmos simulados en cuanto a retardo extremo a extremo, y su comparación con los valores reportados en [44]. Los dos algoritmos simulados se comportan mejor que los reportados para los tamaños de red menores de 200 nodos. Esto puede explicarse por el hecho de que los procesos utilizados para aplanar los grafos implican un mayor tiempo de procesamiento del que se requiere para avanzar en modo voraz, y a medida que aumenta el área de despliegue de los sensores en el enrutamiento geográfico se hace más probable la utilización del modo perimetral que involucra estos esquemas de aplanamiento de la red para evadir vacíos o zonas en las que

no hay nodos vecinos que permitan un mayor avance hacia el destino comparado con el del último nodo que recibió el paquete.

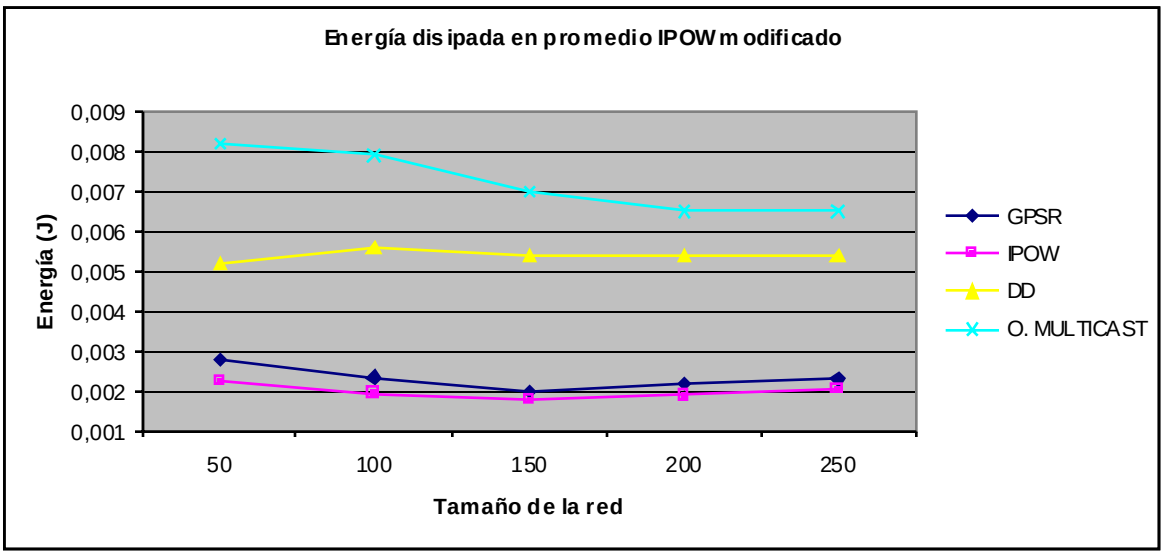

Fig. 20 Energía disipada en promedio con modificación a partir de traza de energía de IPOW.

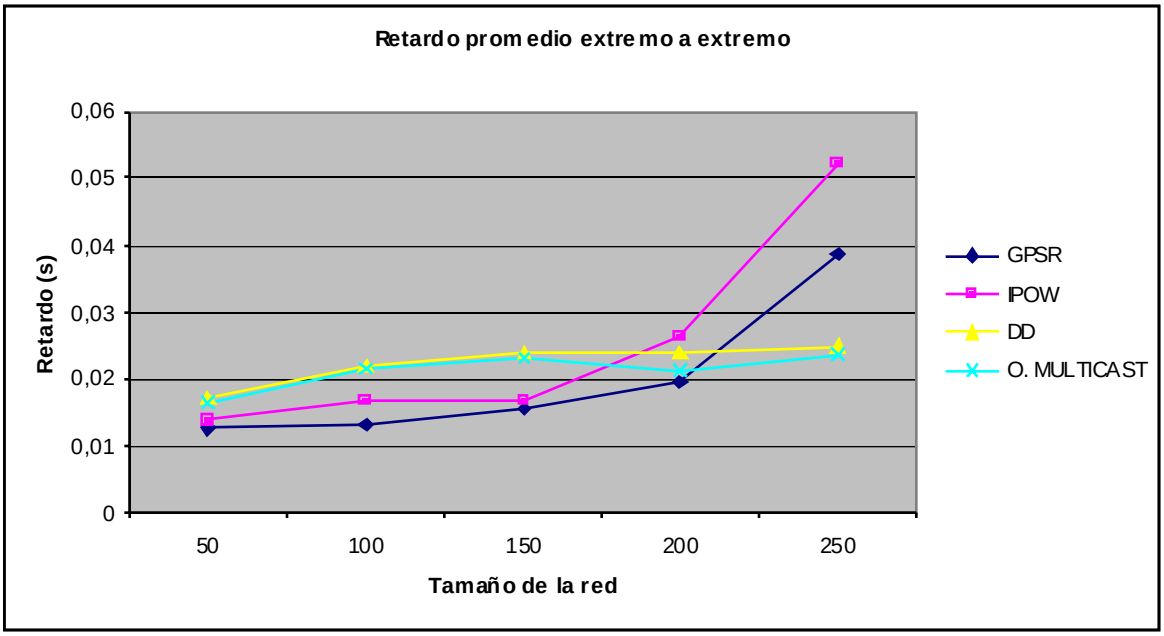

Fig. 21. Retardo promedio extremo a extremo para los cinco escenarios.

Los valores de energía media disipada obtenidos para los dos algoritmos que simulamos son similares a aquellos reportados en [45] para el algoritmo allí propuesto. En cuanto a retardo promedio extremo a extremo, nuestros valores son mejores que los reportados en [45].

Capítulo 5 Conclusiones

A partir de la revisión bibliográfica que hemos realizado durante nuestra investigación, se ha hecho evidente la gran cantidad de protocolos de enrutamiento existentes para redes de sensores, tanto diseñados específicamente para estas redes, como los extendidos desde redes ad-hoc para su utilización en redes de sensores. No obstante esto, la investigación sobre el tema sigue presentando un alto dinamismo; incluso en los protocolos más reconocidos, aún hay aspectos que representan oportunidades de mejora y que se convierten en puntos de partida para nuevos proyectos de investigación.

En el proyecto se consiguió extender las posibilidades del simulador ns-2 mediante el desarrollo de los módulos software requeridos para implementar un nuevo agente de enrutamiento. El agente de enrutamiento implementado está basado en IPOW, un algoritmo geográfico que iterativamente busca rutas más económicas en términos de consumo de energía con base en información local.

Los resultados obtenidos en la simulación demostraron la efectividad del algoritmo implementado en términos de consumo de energía, lo cual se validó con respecto a reportes existentes en la literatura y al algoritmo GPSR que también se simuló.

A diferencia de reportes encontrados generalmente en la literatura sobre el tema, nuestra validación de resultados se realiza contra un protocolo diseñado específicamente para sensores, y sobre las mismas condiciones de operación en la simulación (escenarios y parámetros de configuración) lo que genera confianza en los resultados obtenidos.

El estado, definido como la memoria necesaria para tomar las decisiones de enrutamiento en cada nodo, en los algoritmos basados en localización como son GPSR e IPOW es muy pequeño pues estos únicamente requieren información local para la toma de decisiones de enrutamiento. Esto puede considerarse como una importante característica muy deseable cuando los nodos sensores presentan elevadas restricciones en capacidad de procesamiento y de almacenamiento.

La elección de un protocolo eficiente desde el punto de vista de energía en redes de sensores, está altamente influenciado por los detalles específicos de cada red. El desempeño de los protocolos depende de aspectos como la aplicación y sus requerimientos de retardo, calidad de servicio y disponibilidad; de la topología de la red, y el área geográfica en que esta se implementó; de la densidad de nodos y la forma en que los nodos están ubicados.

Con respecto a los protocolos de enrutamiento adaptivos y aquellos basados en negociación, la investigación está dirigida a encontrar esquemas estandarizados y eficientes para nombrar los datos [16]. En lo referente a los protocolos de enrutamiento basados en el agrupamiento (clustering), la investigación apunta hacia formas eficientes de establecer los grupos de tal forma que se permita el ahorro de energía y la optimización del desempeño de la red; aunque también se continúan investigando los aspectos de agregación y fusión de datos [16]. Para los protocolos basados en localización, el campo de mayor investigación es el de la utilización inteligente de la información de localización con el fin de mejorar la eficiencia en energía[16].

ANEXO A. CÓDIGO DESARROLLADO EN C++

A.1 ipow_packet.h

```
/* Copyright (C) 2005 State University of New York, at Binghamton
 * All rights reserved.
 *
 * NOTICE: This software is provided "as is", without any warranty,
 * including any implied warranty for merchantability or fitness for a
 * particular purpose.  Under no circumstances shall SUNY Binghamton
 * or its faculty, staff, students or agents be liable for any use of,
 * misuse of, or inability to use this software, including incidental
 * and consequential damages.

 * License is hereby given to use, modify, and redistribute this
 * software, in whole or in part, for any commercial or non-commercial
 * purpose, provided that the user agrees to the terms of this
 * copyright notice, including disclaimer of warranty, and provided
 * that this copyright notice, including disclaimer of warranty, is
 * preserved in the source code and documentation of anything derived
 * from this software.  Any redistributor of this software or anything
 * derived from this software assumes responsibility for ensuring that
 * any parties to whom such a redistribution is made are fully aware of
 * the terms of this license and disclaimer.
 *
 */

/*
 * IPOW code for ns-2 version 2.29
 * Based in GPSR implementation of Ke Liu
 * CS Dept., State University of New York, at Binghamton
 *

 ********
 ********
 * Carlos Eduardo Silva Martínez
 * Maestría en Ingeniería Eléctrónica y de Computadores
 *
 *****
 * Universidad de los Andes
 * Bogotá Colombia 2007
 ********
 ********
 */
```

```
/* ipow_packet.h : the definition the ipow routing protocol packet header
 *
 * hdr_hello : hello message header, broadcast to one hop neighbors
 * hdr_query : query message header, flood the data sink info into networks
 * hdr_data  : not a really data packet header, it is just used to
 *           be attached to any data packet created by higher layer apps
 *           The reason to use this type of header is decided by the
 *           design of the GPSR: each data packet has to carry the
 *           location info of its data sink, maintain the routing protocol
 *           stateless.
 */


#ifndef IPOW_PACKET_H_
#define IPOW_PACKET_H_

#include "packet.h"
#include <math.h>

#define SINK_TRACE_FILE "sink_trace.tr"
#define NB_TRACE_FILE "ipownb_trace.tr"

#define IPOW_CURRENT Scheduler::instance().clock()
#define INFINITE_DELAY 5000000000000.0

#define IPOWTYPE_HELLO  0x01   //hello msg
#define IPOWTYPE_QUERY  0x02   //query msg from the sink
#define IPOWTYPE_DATA   0x04   //the CBR data msg

#define IPOW_MODE_IPOW  0x01   //basic ipow forwarding mode
#define IPOW_MODE_PERI  0x02   //perimeter routing mode

#define HDR_IPOW(p)   ((struct hdr_ipow*)hdr_ipow::access(p))
#define HDR_IPOW_HELLO(p) ((struct hdr_ipow_hello*)hdr_ipow::access(p))
#define HDR_IPOW_QUERY(p) ((struct hdr_ipow_query*)hdr_ipow::access(p))
#define HDR_IPOW_DATA(p) ((struct hdr_ipow_data*)hdr_ipow::access(p))

struct hdr_ipow {
  u_int8_t type_;

  static int offset_;
  inline static int& offset() {return offset_;}
  inline static struct hdr_ipow* access(const Packet *p){
    return (struct hdr_ipow*) p->access(offset_);
  }
};
struct hdr_ipow_hello {
```

```
  u_int8_t type_;
  float x_;       //My geo info
  float y_;
  inline int size(){
    int sz =
      sizeof(u_int8_t) +
      2*sizeof(float);
    return sz;
  }
};

struct hdr_ipow_query {
  u_int8_t type_;
  float x_;       //The sink geo info
  float y_;
  float ts_;      //time stampe
  int hops_;
  u_int8_t seqno_;     //query sequence number
  inline int size(){
    int sz =
      2*sizeof(u_int8_t) +
      3*sizeof(float) +
      sizeof(int);
    return sz;
  }
};

struct hdr_ipow_data {
  u_int8_t type_;
  u_int8_t mode_;      //basic ipow forwarding or Perimeter Routing.
                       //if doesn´t found a neighbor use perimeter routing

  float sx_;      //the geo info of src
  float sy_;
  float dx_;      //the geo info of dst
  float dy_;
  float ts_;      //the originating time stamp
  inline int size(){
    int sz =
      2*sizeof(u_int8_t) +
      5*sizeof(float);
    return sz;
  }
};


union hdr_all_ipow {
```

```
  hdr_ipow       gh;
  hdr_ipow_hello ghh;
  hdr_ipow_query gqh;
  hdr_ipow_data  gdh;
};

#endif
```

A.2 ipow.h

```
/* Copyright (C) 2005 State University of New York, at Binghamton
 * All rights reserved.
 *
 * NOTICE: This software is provided "as is", without any warranty,
 * including any implied warranty for merchantability or fitness for a
 * particular purpose.  Under no circumstances shall SUNY Binghamton
 * or its faculty, staff, students or agents be liable for any use of,
 * misuse of, or inability to use this software, including incidental
 * and consequential damages.
 * License is hereby given to use, modify, and redistribute this
 * software, in whole or in part, for any commercial or non-commercial
 * purpose, provided that the user agrees to the terms of this
 * copyright notice, including disclaimer of warranty, and provided
 * that this copyright notice, including disclaimer of warranty, is
 * preserved in the source code and documentation of anything derived
 * from this software.  Any redistributor of this software or anything
 * derived from this software assumes responsibility for ensuring that
 * any parties to whom such a redistribution is made are fully aware of
 * the terms of this license and disclaimer.
 */
/*
 * IPOW code for ns-2 version 2.29
 * Based in GPSR implementation of Ke Liu
 * CS Dept., State University of New York, at Binghamton
 *

 ********
 ********
 * Carlos Eduardo Silva Martínez
 * Maestría en Ingeniería Eléctrónica y de Computadores
 *
 *****
 * Universidad de los Andes
 * Bogotá Colombia 2007
 ********
 ********
```

```
 *
 */

/* ipow.h : The head file for the IPOW routing agent, defining the
 *          routing agent, methods (behaviors) of the routing
 *
 * Note: the routing table (local neighborhood) information is kept
 *       in another class ipow_neighbor which is defined in
 *       ipow_neighbor{.h, .cc}. So the planarizing and next hop deciding
 *       is made there, not in this file
 *
 */

#ifndef IPOW_ROUTING_H_
#define IPOW_ROUTING_H_

#include "config.h"
#include "agent.h"
#include "ip.h"
#include "address.h"
#include "timer-handler.h"
#include "mobilenode.h"
#include "tools/random.h"
#include "packet.h"
#include "trace.h"
#include "classifier-port.h"
#include "cmu-trace.h"

#include "ipow_packet.h"
#include "ipow_neighbor.h"
#include "ipow_sinklist.h"

class IPOWAgent;

/*
 * Hello timer which is used to fire the hello message periodically
 */
class IPOWHelloTimer : public TimerHandler {
public:
  IPOWHelloTimer(IPOWAgent *a) : TimerHandler() {a_=a;}
protected:
  virtual void expire(Event *e);
  IPOWAgent *a_;
};
```

```
/*
 * The Query Timer which is used by the data sink to fire the
 * data query. It is not a part of the design of the IPOW routing.
 * Since the information of the data sink mostly is not able to be
 * obtained directly (mostly, by DHT: distributed hash table), I
 * just let the data sink to trigger the routing, like a common
 * On-Demond routing.
 */
class IPOWQueryTimer : public TimerHandler {
public:
  IPOWQueryTimer(IPOWAgent *a) : TimerHandler() {a_=a;}
protected:
  virtual void expire(Event *e);
  IPOWAgent *a_;
};

class IPOWAgent : public Agent {
private:

  friend class IPOWHelloTimer;
  friend class IPOWQueryTimer;

  MobileNode *node_;          //the attached mobile node
  PortClassifier *port_dmux_;  //for the higher layer app de-multiplexing

  nsaddr_t my_id_;            //node id (address), which is NOT necessary
  double my_x_;               //node location info, fatal info
  double my_y_;               //     obtained from the attached node

  Sinks *sink_list_;              //for multiple sinks

  IPOWNeighbors *nblist_;             //neighbor list: routing table implemenation
                          //and planarizing implementation
  struct hop_entry *hoplist_;  //temporal hops list to go to neighbor A
                          //selected for Power Progress

  int recv_counter_;
  u_int8_t query_counter_;

  IPOWHelloTimer hello_timer_;
  IPOWQueryTimer query_timer_;

  int alfa_;                //alpha to calculate the power
  int constante_;                   //Ce constant to calculate the power
```

```
  int planar_type_;             //1=GG planarize, 0=RNG planarize
  double hello_period_;
  double query_period_;

  void turnon();                //set to be alive
  void turnoff();               //set to be dead
  void startSink();
  void startSink(double);

  void GetLocation(double*, double*); //called at initial phase
  virtual void getLoc();

  void hellomsg();
  void query(nsaddr_t);

  void recvHello(Packet*);
  void recvQuery(Packet*);

  void sinkRecv(Packet*);
  void forwardData(Packet*);

  RNG randSend_;

  /**
   * The below variables and functions are used for
   * localization algorithms
   */
  double localized_x_;
  double localized_y_;
  void dvhop();

protected:
  Trace *tracetarget;           //for routing agent special trace
  void trace(char *fmt,...);    //  Not necessary

  void hellotout();             //called by timer::expire(Event*)
  void querytout();

public:
  IPOWAgent();

  int command(int, const char*const*);
  void recv(Packet*, Handler*);         //inherited virtual function

};

#endif
```

A.3 ipow.cc

```
/* Copyright (C) 2005 State University of New York, at Binghamton
 * All rights reserved.
 *
 * NOTICE: This software is provided "as is", without any warranty,
 * including any implied warranty for merchantability or fitness for a
 * particular purpose.  Under no circumstances shall SUNY Binghamton
 * or its faculty, staff, students or agents be liable for any use of,
 * misuse of, or inability to use this software, including incidental
 * and consequential damages.

 * License is hereby given to use, modify, and redistribute this
 * software, in whole or in part, for any commercial or non-commercial
 * purpose, provided that the user agrees to the terms of this
 * copyright notice, including disclaimer of warranty, and provided
 * that this copyright notice, including disclaimer of warranty, is
 * preserved in the source code and documentation of anything derived
 * from this software.  Any redistributor of this software or anything
 * derived from this software assumes responsibility for ensuring that
 * any parties to whom such a redistribution is made are fully aware of
 * the terms of this license and disclaimer.
 *
 */

/*
 * IPOW code for ns-2 version 2.29
 * Based in GPSR implementation of Ke Liu
 * CS Dept., State University of New York, at Binghamton
 *

 ********
 ********
 * Carlos Eduardo Silva Martínez
 * Maestría en Ingeniería Eléctrónica y de Computadores
 *
 *****
 * Universidad de los Andes
 * Bogotá Colombia 2007
 ********
 ********

 *
 */

/* ipow.cc : the definition of the ipow routing agent class */
```

```
#include "ipow.h"
int hdr_ipow::offset_;

static class IPOWHeaderClass : public PacketHeaderClass{
public:
  IPOWHeaderClass() : PacketHeaderClass("PacketHeader/IPOW",
                                        sizeof(hdr_all_ipow)){
    bind_offset(&hdr_ipow::offset_);
  }
}class_ipowhdr;

static class IPOWAgentClass : public TclClass {
public:
  IPOWAgentClass() : TclClass("Agent/IPOW"){}
  TclObject *create(int, const char*const*){
    return (new IPOWAgent());
  }
}class_ipow;

void
IPOWHelloTimer::expire(Event *e){
  a_->hellotout();
}

void
IPOWQueryTimer::expire(Event *e){
  a_->querytout();
}

void
IPOWAgent::hellotout(){
  hellomsg();
  hello_timer_.resched(hello_period_);
}

void
IPOWAgent::startSink(){
  if(sink_list_->new_sink(my_id_, my_x_, my_y_,
                          my_id_, 0, query_counter_))
    querytout();
}

void
IPOWAgent::startSink(double gp){
  query_period_ = gp;
  startSink();
}
```

```
void
IPOWAgent::querytout(){
  query(my_id_);
  query_counter_++;
  query_timer_.resched(query_period_);
}

void
IPOWAgent::getLoc(){
  GetLocation(&my_x_, &my_y_);
}

void
IPOWAgent::GetLocation(double *x, double *y){
  double pos_x_, pos_y_, pos_z_;
  node_->getLoc(&pos_x_, &pos_y_, &pos_z_);
  *x=pos_x_;
  *y=pos_y_;
}

IPOWAgent::IPOWAgent() : Agent(PT_IPOW),
                         hello_timer_(this), query_timer_(this),
                         my_id_(-1), my_x_(0.0), my_y_(0.0),
                         recv_counter_(0), query_counter_(0),
                         query_period_(INFINITE_DELAY)
{
  bind("planar_type_", &planar_type_);
  bind("hello_period_", &hello_period_);
  bind("alfa_", &alfa_);                  //To bind alpha to calculate the power
  bind("constante_", &constante_);   //to bind Ce to calculate the power

  sink_list_ = new Sinks();
  nblist_ = new IPOWNeighbors();
  hoplist_= NULL;                         //Start with de empty list of temporal hops

  for(int i=0; i<5; i++)
    randSend_.reset_next_substream();
}

void
IPOWAgent::turnon(){
  getLoc();
  nblist_->myinfo(my_id_, my_x_, my_y_);
  hello_timer_.resched(randSend_.uniform(0.0, 0.5));
}
```

```
void
IPOWAgent::turnoff(){
  hello_timer_.resched(INFINITE_DELAY);
  query_timer_.resched(INFINITE_DELAY);
}

void
IPOWAgent::hellomsg(){
  if(my_id_ < 0) return;

  Packet *p = allocpkt();
  struct hdr_cmn *cmh = HDR_CMN(p);
  struct hdr_ip *iph = HDR_IP(p);
  struct hdr_ipow_hello *ghh = HDR_IPOW_HELLO(p);

  cmh->next_hop_ = IP_BROADCAST;
  cmh->last_hop_ = my_id_;
  cmh->addr_type_ = NS_AF_INET;
  cmh->ptype() = PT_IPOW;
  cmh->size() = IP_HDR_LEN + ghh->size();

  iph->daddr() = IP_BROADCAST;
  iph->saddr() = my_id_;
  iph->sport() = RT_PORT;
  iph->dport() = RT_PORT;
  iph->ttl_ = IP_DEF_TTL;

  ghh->type_ = IPOWTYPE_HELLO;
  ghh->x_ = (float)my_x_;
  ghh->y_ = (float)my_y_;

  send(p, 0);
}

void
IPOWAgent::query(nsaddr_t id){
  if(my_id_ < 0) return;

  Packet *p = allocpkt();

  struct hdr_cmn *cmh = HDR_CMN(p);
  struct hdr_ip *iph = HDR_IP(p);
  struct hdr_ipow_query *gqh = HDR_IPOW_QUERY(p);

  cmh->next_hop_ = IP_BROADCAST;
  cmh->last_hop_ = my_id_;
  cmh->addr_type_ = NS_AF_INET;
```

```
  cmh->ptype() = PT_IPOW;
  cmh->size() = IP_HDR_LEN + gqh->size();

  iph->daddr() = IP_BROADCAST;
  iph->saddr() = id;
  iph->sport() = RT_PORT;
  iph->dport() = RT_PORT;
  iph->ttl_ = IP_DEF_TTL;

  gqh->type_ = IPOWTYPE_QUERY;
  double tempx, tempy;
  int hops;
  sink_list_->getLocbyID(id, tempx, tempy, hops);
  if(tempx >= 0.0){
    gqh->x_ = (float)tempx;
    gqh->y_ = (float)tempy;
    gqh->hops_ = hops;
  }else {
    Packet::free(p);
    return;
  }
  gqh->ts_ = (float)IPOW_CURRENT;
  gqh->seqno_ = query_counter_;

  send(p, 0);
}

void
IPOWAgent::recvHello(Packet *p){
  struct hdr_cmn *cmh = HDR_CMN(p);
  struct hdr_ipow_hello *ghh = HDR_IPOW_HELLO(p);

  nblist_->newNB(cmh->last_hop_, (double)ghh->x_, (double)ghh->y_);
  //  trace("%d recv Hello from %d", my_id_, cmh->last_hop_);
}

void
IPOWAgent::recvQuery(Packet *p){
  struct hdr_cmn *cmh = HDR_CMN(p);
  struct hdr_ip *iph = HDR_IP(p);
  struct hdr_ipow_query *gqh = HDR_IPOW_QUERY(p);

  if(sink_list_->new_sink(iph->saddr(), gqh->x_, gqh->y_,
                          cmh->last_hop_, 1+gqh->hops_, gqh->seqno_))
    query(iph->saddr());
  //  trace("%d recv Query from %d ", my_id_, iph->saddr());
}
```

```
void
IPOWAgent::sinkRecv(Packet *p){
  FILE *fp = fopen(SINK_TRACE_FILE, "a+");
  struct hdr_cmn *cmh = HDR_CMN(p);
  struct hdr_ip *iph = HDR_IP(p);
  //  struct hdr_ipow_data *gdh = HDR_IPOW_DATA(p);

  fprintf(fp, "%2.f\t%d\t%d\n", IPOW_CURRENT,
          iph->saddr(), cmh->num_forwards());
  fclose(fp);
}
void
IPOWAgent::forwardData(Packet *p){
  struct hdr_cmn *cmh = HDR_CMN(p);
  struct hdr_ip *iph = HDR_IP(p);

  if(cmh->direction() == hdr_cmn::UP &&
     ((nsaddr_t)iph->daddr() == IP_BROADCAST ||
      iph->daddr() == my_id_)){
      sinkRecv(p);
      printf("receive\n");
      port_dmux_->recv(p, 0);
      return;
  }
  else {
    nsaddr_t nexthop;
    if (hoplist_ == NULL){     //Temporal hops list is empty and have to search a new hop

        struct hdr_ipow_data *gdh=HDR_IPOW_DATA(p);

        double dx = gdh->dx_;
        double dy = gdh->dy_;

        if(gdh->mode_ == IPOW_MODE_IPOW){
          //nexthop = nblist_->ipow_nexthop(dx, dy, alfa_, constante_);
          hoplist_ = nblist_->ipow_nexthop(dx, dy, alfa_, constante_);
          nexthop= hoplist_->id_;
          hoplist_ = nblist_->remove_hop(hoplist_,nexthop);

          if(nexthop == -1){
               nexthop = nblist_->peri_nexthop(planar_type_, -1,
                                               gdh->sx_, gdh->sy_,
                                               gdh->dx_, gdh->dy_);
               gdh->sx_ = my_x_;
               gdh->sy_ = my_y_;
               gdh->mode_ = IPOW_MODE_PERI;
          }
```

```
        }
        else {
         double sddis = nblist_->getdis(gdh->sx_, gdh->sy_, gdh->dx_, gdh->dy_);
         double mydis = nblist_->getdis(my_x_, my_y_, gdh->dx_, gdh->dy_);
         if(mydis < sddis){
             //switch back to ipow forwarding mode
             //nexthop = nblist_->ipow_nexthop(dx, dy, alfa_, constante_);
           hoplist_ = nblist_->ipow_nexthop(dx, dy, alfa_, constante_);
           nexthop= hoplist_->id_;
           hoplist_ = nblist_->remove_hop(hoplist_,nexthop);

             gdh->mode_ = IPOW_MODE_IPOW;

            if(nexthop == -1){
              nexthop =
              nblist_->peri_nexthop(planar_type_, -1,
                                    gdh->sx_, gdh->sy_,
                                    gdh->dx_, gdh->dy_);
            gdh->sx_ = my_x_;
              gdh->sy_ = my_y_;
              gdh->mode_ = IPOW_MODE_PERI;
            }
         }
         else{                          //still perimeter routing mode
             nexthop =
                nblist_->peri_nexthop(planar_type_, cmh->last_hop_,
                                    gdh->sx_, gdh->sy_, gdh->dx_, gdh->dy_);
         }
        }
      }
      else{                                     //If temporal hops list isn´t empty have to do the hop
        nexthop = hoplist_->id_;
        hoplist_ = nblist_->remove_hop(hoplist_,nexthop);
      }

      cmh->direction() = hdr_cmn::DOWN;
      cmh->addr_type() = NS_AF_INET;
      cmh->last_hop_ = my_id_;
      cmh->next_hop_ = nexthop;
      FILE *fp=fopen("powerHOPS.tr","a+"); //The energy trace to correct IPOW energy
      //fprintf(fp, "%s\n%d\t%s\t%d\n", "Source node: ", my_id_, "Sink node: ",nexthop);
            fprintf(fp, "%s\t%d\t%s\t%d\n", "Source node: ", my_id_, "Sink node: ",nexthop);
      fclose(fp);
      send(p, 0);
    }
}
```

```
void
IPOWAgent::recv(Packet *p, Handler *h){
  struct hdr_cmn *cmh = HDR_CMN(p);
  struct hdr_ip *iph = HDR_IP(p);

  if(iph->saddr() == my_id_){                          //a packet generated by myself
    if(cmh->num_forwards() == 0){
      struct hdr_ipow_data *gdh = HDR_IPOW_DATA(p);
      cmh->size() += IP_HDR_LEN + gdh->size();

      gdh->type_ = IPOWTYPE_DATA;
      gdh->mode_ = IPOW_MODE_IPOW;
      gdh->sx_ = (float)my_x_;
      gdh->sy_ = (float)my_y_;
      double tempx, tempy;
      int hops;
      sink_list_->getLocbyID(iph->daddr(), tempx, tempy, hops);
      if(tempx >= 0.0){
          gdh->dx_ = (float)tempx;
          gdh->dy_ = (float)tempy;
      }
      else {
          drop(p, "NoSink");
          return;
      }
      gdh->ts_ = (float)IPOW_CURRENT;
    }
    else if(cmh->num_forwards() > 0){        //routing loop
      if(cmh->ptype() != PT_IPOW)
          drop(p, DROP_RTR_ROUTE_LOOP);
      else Packet::free(p);
      return;
    }
  }

  if(cmh->ptype() == PT_IPOW){
    struct hdr_ipow *gh = HDR_IPOW(p);
    switch(gh->type_){
    case IPOWTYPE_HELLO:
      recvHello(p);
      break;
    case IPOWTYPE_QUERY:
      recvQuery(p);
      break;
    default:
      printf("Error with gf packet type.\n");
      exit(1);
```

```
    }
  } else {
    iph->ttl_--;
    if(iph->ttl_ == 0){
      drop(p, DROP_RTR_TTL);
      return;
    }
    forwardData(p);
  }

}

void
IPOWAgent::trace(char *fmt, ...){
  va_list ap;
  if(!tracetarget)
    return;
  va_start(ap, fmt);
  vsprintf(tracetarget->pt_->buffer(), fmt, ap);
  tracetarget->pt_->dump();
  va_end(ap);
}

int
IPOWAgent::command(int argc, const char*const* argv){
  if(argc==2){
    if(strcasecmp(argv[1], "getloc")==0){
      getLoc();
      return TCL_OK;
    }

    if(strcasecmp(argv[1], "turnon")==0){
      turnon();
      return TCL_OK;
    }

    if(strcasecmp(argv[1], "turnoff")==0){
      turnoff();
      return TCL_OK;
    }

    if(strcasecmp(argv[1], "startSink")==0){
      startSink();
      return TCL_OK;
    }

    if(strcasecmp(argv[1], "neighborlist")==0){
```

```
    nblist_->dump();
    return TCL_OK;
   }
   if(strcasecmp(argv[1], "sinklist")==0){
    sink_list_->dump();
    return TCL_OK;
   }
  }

  if(argc==3){
   if(strcasecmp(argv[1], "startSink")==0){
    startSink(atof(argv[2]));
    return TCL_OK;
   }

   if(strcasecmp(argv[1], "addr")==0){
    my_id_ = Address::instance().str2addr(argv[2]);
    return TCL_OK;
   }

   TclObject *obj;
   if ((obj = TclObject::lookup (argv[2])) == 0){
    fprintf (stderr, "%s: %s lookup of %s failed\n", __FILE__, argv[1],
             argv[2]);
    return (TCL_ERROR);
   }
   if (strcasecmp (argv[1], "node") == 0) {
    node_ = (MobileNode*) obj;
    return (TCL_OK);
   }
   else if (strcasecmp (argv[1], "port-dmux") == 0) {
    port_dmux_ = (PortClassifier*) obj; //(NsObject *) obj;
    return (TCL_OK);
   } else if(strcasecmp(argv[1], "tracetarget")==0){
    tracetarget = (Trace *)obj;
    return TCL_OK;
   }

  }// if argc == 3

  return (Agent::command(argc, argv));
 }
```

A.4 ipow_neighbor.h

```
/* Copyright (C) 2005 State University of New York, at Binghamton
 * All rights reserved.
 *
 * NOTICE: This software is provided "as is", without any warranty,
 * including any implied warranty for merchantability or fitness for a
 * particular purpose.  Under no circumstances shall SUNY Binghamton
 * or its faculty, staff, students or agents be liable for any use of,
 * misuse of, or inability to use this software, including incidental
 * and consequential damages.

 * License is hereby given to use, modify, and redistribute this
 * software, in whole or in part, for any commercial or non-commercial
 * purpose, provided that the user agrees to the terms of this
 * copyright notice, including disclaimer of warranty, and provided
 * that this copyright notice, including disclaimer of warranty, is
 * preserved in the source code and documentation of anything derived
 * from this software.  Any redistributor of this software or anything
 * derived from this software assumes responsibility for ensuring that
 * any parties to whom such a redistribution is made are fully aware of
 * the terms of this license and disclaimer.
 */
/*
 * IPOW code for ns-2 version 2.29
 * Based in GPSR implementation of Ke Liu
 * CS Dept., State University of New York, at Binghamton
 *

 ********
 ********
 * Carlos Eduardo Silva Martínez
 * Maestría en Ingeniería Eléctrónica y de Computadores
 *
 *****
 * Universidad de los Andes
 * Bogotá Colombia 2007
 ********
 ********

 *
 */

/* ipow_neighbor.h : the head file for the definition of the neighbor
 *                   list for each node in the network
 * Note: Due to the network nature, the neighbor list of different nodes
 *       which are neighbors from a view of the transimission radio range
```

```
 *      may not consistent, which means for a pair of nodes, one may know
 *      the other is a neighbor, but the other one does not
 */


#ifndef IPOW_NEIGHBOR_H_
#define IPOW_NEIGHBOR_H_

#include "ipow_packet.h"

#define DEFAULT_IPOW_TIMEOUT   23.0
      //the default time out period is 23.0 sec (4.5 B; B = 5 sec.)
      //If the hello messge was not received during this period
      //the entry in the neighbor list may be deleted

#define INIFINITE_DISTANCE   1000000000.0 //bigger than radio range is enough

/* The structure used for neighbor entry
 */
struct ipow_neighbor {
  nsaddr_t id_;
  double x_;     //the geo info
  double y_;

  double ts_;              //the last time stamp of the hello msg from it
  struct ipow_neighbor *next_;
  struct ipow_neighbor *prev_;
};

struct hop_entry {
  nsaddr_t id_;
  struct hop_entry *next_;
};

class IPOWNeighbors {
private:
  //the neighbors list
  struct ipow_neighbor *head_;
  struct ipow_neighbor *tail_;
  int nbSize_; //number of neighbors


  nsaddr_t my_id_;     //my id
  double my_x_;        //my geo info
  double my_y_;
```

```
  //find the entry in neighbor list according to the provided id
  struct ipow_neighbor *getnb(nsaddr_t);

  //delete the entry in neighbors list according to the provided id
  void delnb(nsaddr_t);

  //delete the entry directly
  void delnb(struct ipow_neighbor *);

  //delete all the entries time out
  void delalltimeout();

  //for a given neighbor list (such as planarized neighbors), return size
  int num_of_neighbors(struct ipow_neighbor*);

  /* functions used for perimeter routing calculation */
  struct ipow_neighbor *gg_planarize();  //GG planarize
  struct ipow_neighbor *rng_planarize(); //RNG planarize
  void free_neighbors(struct ipow_neighbor*); //free the given neighbor list

  double angle(double, double, double, double);
                              //absolute angle of the given line
                              //relative angle calculation based

  int intersect(nsaddr_t, double, double, double, double);
  //check the 2 lines are intersected locally

public:
  IPOWNeighbors();
  ~IPOWNeighbors();
  //using to update location information of myself
  void myinfo(nsaddr_t, double, double);
  //return the number of neighbors
  int nbsize();

  //calculate the distance between (x1, y1, x2, y2)
  double getdis(double, double, double, double);

  //add a possible new neighbor
  void newNB(nsaddr_t, double, double);

  //decide the next hop based on the destination(x, y)
  struct hop_entry *ipow_nexthop(double, double, int, int);          //basic ipow forwarding
        mode
  struct hop_entry *iterative_power_progress(struct hop_entry *,ipow_neighbor*, int, int);
        //iterative power progress method
  nsaddr_t peri_nexthop(int, nsaddr_t, double, double, double, double); //perimeter mode
```

```
  //métodos correspondientes al manejo de la lista de saltos auxiliares
  struct hop_entry *new_hop(struct hop_entry *, nsaddr_t);
  struct hop_entry *remove_hop(struct hop_entry *, nsaddr_t);


  //dump the neighbor list
  void dump();
};

#endif
```

A.5 ipow_neighbor.cc

```
/* Copyright (C) 2005 State University of New York, at Binghamton
 * All rights reserved.
 *
 * NOTICE: This software is provided "as is", without any warranty,
 * including any implied warranty for merchantability or fitness for a
 * particular purpose.  Under no circumstances shall SUNY Binghamton
 * or its faculty, staff, students or agents be liable for any use of,
 * misuse of, or inability to use this software, including incidental
 * and consequential damages.

 * License is hereby given to use, modify, and redistribute this
 * software, in whole or in part, for any commercial or non-commercial
 * purpose, provided that the user agrees to the terms of this
 * copyright notice, including disclaimer of warranty, and provided
 * that this copyright notice, including disclaimer of warranty, is
 * preserved in the source code and documentation of anything derived
 * from this software.  Any redistributor of this software or anything
 * derived from this software assumes responsibility for ensuring that
 * any parties to whom such a redistribution is made are fully aware of
 * the terms of this license and disclaimer.
 *
 */
/*
 * IPOW code for ns-2 version 2.29
 * Based in GPSR implementation of Ke Liu
 * CS Dept., State University of New York, at Binghamton
 *

 ********
 ********
 * Carlos Eduardo Silva Martínez
 * Maestría en Ingeniería Eléctrónica y de Computadores
 *
```

```
 *****
 * Universidad de los Andes
 * Bogotá Colombia 2007
 ********
 ********

 *
 */

/* ipow_neighbor.cc: the implementation of routing table */


#include "ipow_neighbor.h"
#include "math.h"

#define PI 3.141593

#define MAX(a, b) (a>=b?a:b)
#define MIN(a, b) (a>=b?b:a)

IPOWNeighbors::IPOWNeighbors(){
  my_id_ = -1;
  my_x_ = 0.0;
  my_y_ = 0.0;

  head_ = tail_ = NULL;
  nbSize_ = 0;
}

IPOWNeighbors::~IPOWNeighbors(){
  struct ipow_neighbor *temp = head_;
  while(temp){
    temp = temp->next_;
    free(head_);
    head_ = temp;
  }
}

double
IPOWNeighbors::getdis(double ax, double ay, double bx, double by){
  double tempx = ax - bx;
  double tempy = ay - by;

  tempx = tempx * tempx;
  tempy = tempy * tempy;
  double result = sqrt(tempx + tempy);
  return result;
```

```
}

int
IPOWNeighbors::nbsize(){
  return nbSize_;
}

void
IPOWNeighbors::myinfo(nsaddr_t mid, double mx, double my){
  my_id_ = mid;
  my_x_ = mx;
  my_y_ = my;
}

struct ipow_neighbor*
IPOWNeighbors::getnb(nsaddr_t nid){
  struct ipow_neighbor *temp = head_;
  while(temp){
    if(temp->id_ == nid){
      if((IPOW_CURRENT - temp->ts_) < DEFAULT_IPOW_TIMEOUT)
          return temp;
      else {
          delnb(temp);              //if this entry expire, delete it and return NULL
          return NULL;
      }
      return temp;
    }
    temp = temp->next_;
  }
  return NULL;
}

void
IPOWNeighbors::newNB(nsaddr_t nid, double nx, double ny){
  struct ipow_neighbor *temp = getnb(nid);

  if(temp==NULL){                       //it is a new neighbor
    temp=(struct ipow_neighbor*)malloc(sizeof(struct ipow_neighbor));
    temp->id_ = nid;
    temp->x_ = nx;
    temp->y_ = ny;
    temp->ts_ = IPOW_CURRENT;
    temp->next_ = temp->prev_ = NULL;


    if(tail_ == NULL){          //the list now is empty
      head_ = tail_  = temp;
```

```
    nbSize_ = 1;
   }
   else {                                  //now the neighbors list is not empty
    tail_->next_ = temp;
    temp->prev_ = tail_;
    tail_ = temp;
    nbSize_++;
   }
  }
  else {                                   //it is a already known neighbor
   temp->ts_ = IPOW_CURRENT;
   temp->x_ = nx;                          //the updating of location is allowed
   temp->y_ = ny;
  }
}

void
IPOWNeighbors::delnb(nsaddr_t nid){
  struct ipow_neighbor *temp = getnb(nid);
  if(temp==NULL) return;
  else delnb(temp);
}

void
IPOWNeighbors::delnb(struct ipow_neighbor *nb){
  struct ipow_neighbor *preffix = nb->prev_;

  if(preffix == NULL){
   head_ = nb->next_;
   nb->next_ = NULL;

   if(head_ == NULL)
    tail_ = NULL;
   else head_->prev_ = NULL;

   free(nb);
  }
  else {
   preffix->next_ = nb->next_;
   nb->prev_ = NULL;
   if(preffix->next_ == NULL)
    tail_ = preffix;
   else (preffix->next_)->prev_ = preffix;
   free(nb);
  }
  nbSize_--;
```

```
}

void
IPOWNeighbors::delalltimeout(){
  struct ipow_neighbor *temp = head_;
  struct ipow_neighbor *dd;
  while(temp){
    if((IPOW_CURRENT - temp->ts_) >= DEFAULT_IPOW_TIMEOUT){
      dd = temp;
      temp = temp->next_;
      delnb(dd);
    }
    else temp = temp->next_;
  }

}

struct hop_entry
*IPOWNeighbors::ipow_nexthop(double dx, double dy, int alfa_, int constante_){

  struct ipow_neighbor *temp = head_;
  struct ipow_neighbor *tempA= NULL;
  struct hop_entry *hoplist_=NULL;                    //temporal hops list

  double x, r, temp_power;
  double temp_powerTx, temp_enerTx; //To adjustable Power Tx, Tx energy by hop -e
//  double timeTx = 0.00084085, energyTx = 0;       //Tx time and dissipated energy -e
  double timeTx = 0.00085872, energyTx = 0;       //Tx time and dissipated energy -e
  //To set d (the distance between S and D)
  double d =getdis(my_x_, my_y_, dx, dy);
  nsaddr_t nexthop = -1;                          //Store final addres
  nsaddr_t nexthop_temp = -1;                         //Store results temporally

  //To find the neighbor that has a minimum power(|SA|)/(d-x)
    double min_power=100000000.0;
  //Check all neighbors of source node S
    while(temp){
      x= fabs(getdis(temp->x_, temp->y_, dx, dy)); //Distance between A and D
      r = fabs(getdis(my_x_, my_y_, temp->x_, temp->y_)); //Distance between S and A
      if ((d - x)>0){
        temp_power = (pow(r, alfa_) + constante_) / (d - x); //To calculate the power-progress
ratio
        if (temp_power < min_power) { //Search for minimum power-progress ratio
          min_power = temp_power;
              temp_powerTx = min_power * (d-x);       //Tx power -e
              temp_enerTx = temp_powerTx * timeTx; //Tx energy by hop -e
          nexthop=temp->id_;
```

```
        tempA = temp;
       }
          energyTx = energyTx + temp_enerTx;                //Total dissipated energy in hops
        FILE *fp=fopen("powerTx.tr","a+");
//        fprintf(fp, "%d\t%d\t%f\n", my_id_, temp->id_, min_power);
                fprintf(fp, "%d\t%d\t%f\t%f\t%f\t%f\t%f\t%f\t%d\n", my_id_, temp->id_, d,
x, r, temp_power, temp_enerTx,energyTx, nexthop);
        fclose(fp);

      }
     temp=temp->next_;
   }
  hoplist_=new_hop(hoplist_, nexthop);

  if (nexthop != -1)                                    //Call to iterative iterative power
progress
      hoplist_=iterative_power_progress(hoplist_,tempA, alfa_, constante_);

  return hoplist_;
}

struct hop_entry
*IPOWNeighbors::iterative_power_progress(struct hop_entry
                              *hoplist_, ipow_neighbor *M, int alfa_, int constante_){
    struct ipow_neighbor *N=NULL;
    struct ipow_neighbor *tempR=NULL;
    struct ipow_neighbor *tempM=M;                      //New
    struct ipow_neighbor *temp=NULL;                    //Neighbor list of surce node S
    double rSB, rBM, powerSB, powerBM, Min;



    do{
      rSB= fabs(getdis(my_x_, my_y_,M->x_,M->y_));//Distance between source node S
and next hop node A
      Min = pow(rSB, alfa_) + constante_;//To calculate the power
      N = M;
      temp=head_;
      tempR=NULL;
      while(temp){                                      //For each B neighbor of S
       if (num_of_neighbors(M)!=0){                     //Validate if hops list of A is empty
         while (M && (M != temp->next_))
           M=M->next_;
         if (M!=NULL){                                  //It is neighbor of S and M
           rSB= fabs(getdis(my_x_, my_y_,temp->x_,temp->y_));   //Distance between
source node S and temporal node B
           rBM= fabs(getdis(temp->x_,temp->y_,M->x_,M->y_));   //Distance between node
B and node M
```

```
            powerSB=pow(rSB, alfa_) + constante_;
            powerBM=pow(rBM, alfa_) + constante_;
            if ((powerSB + powerBM) < Min) {
              Min=powerSB + powerBM;
              M=temp;
              tempR=M;
            }
          }
        }
        temp=temp->next_;
      }

      if (tempR != NULL)                                  //Change the hop
        hoplist_=new_hop(hoplist_, tempR->id_);

    }while (M == N);

  return hoplist_;
}

//To GG planarize
struct ipow_neighbor *
IPOWNeighbors::gg_planarize(){
    struct ipow_neighbor *temp, *result, *index;
    index = head_;
    result = NULL;

    while(index){
      double midpx = my_x_ + (index->x_ - my_x_)/2.0;
      double midpy = my_y_ + (index->y_ - my_y_)/2.0;
      double mdis = getdis(my_x_, my_y_, midpx, midpy);

      temp = head_;
      while(temp){
          if(temp->id_ != index->id_){
            double tempdis = getdis(midpx, midpy, temp->x_, temp->y_);
            if(tempdis < mdis) break;
          }
          temp=temp->next_;
      }

      if(temp==NULL){
          temp = (struct ipow_neighbor*)malloc(sizeof(struct ipow_neighbor));
          temp->id_ = index->id_;
          temp->x_ = index->x_;
          temp->y_ = index->y_;
          temp->next_ = result;
```

```
            temp->prev_ = NULL;
            if(result) result->prev_ = temp;
            result = temp;
        }

        index=index->next_;
    }

    return result;
}

//To RNG planarize
struct ipow_neighbor *
IPOWNeighbors::rng_planarize(){
    struct ipow_neighbor *temp, *result, *index;
    index = head_;
    result = NULL;

    while(index){
        double mdis = getdis(my_x_, my_y_, index->x_, index->y_);

        temp = head_;
        while(temp){
            if(temp->id_ != index->id_){
                double tempdis1 = getdis(my_x_, my_y_, temp->x_, temp->y_);
                double tempdis2 = getdis(index->x_, index->y_, temp->x_, temp->y_);
                if(tempdis1 < mdis && tempdis2 < mdis) break;
            }
            temp=temp->next_;
        }

        if(temp==NULL){
            temp = (struct ipow_neighbor*)malloc(sizeof(struct ipow_neighbor));
            temp->id_ = index->id_;
            temp->x_ = index->x_;
            temp->y_ = index->y_;
            temp->next_ = result;
            temp->prev_ = NULL;
            if(result) result->prev_ = temp;
            result = temp;
        }

        index=index->next_;
    }
    return result;
}
```

```
double
IPOWNeighbors::angle(double x1, double y1, double x2, double y2){
  double line_len = sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1));

  double sin_theta, cos_theta;
  double theta;

  if(line_len == 0.0){
    printf("2 nodes are the same\n");
    return -1.0;
  }

  sin_theta = (y2-y1)/line_len;
  cos_theta = (x2-x1)/line_len;

  theta = acos(cos_theta);

  if(sin_theta<0){
    theta = 2*PI - theta;
  }

  return theta;
}

/* To check the line from me to theother, and the line from source
 * and destination is intersecting each other or not
 * Note: 2 line segments intersects each other if they have a common
 *       point, BUT here, if the common point is the end point,
 *       we don't count it.
 */

int
IPOWNeighbors::intersect(nsaddr_t theother, double sx, double sy,
                         double dx, double dy){

  struct ipow_neighbor *other = getnb(theother);

  if(other==NULL){
    printf("Wrong the other node\n");
    exit(1);
  }

  double x1 = my_x_;
  double y1 = my_y_;
  double x2 = other->x_;
  double y2 = other->y_;
  double x3 = sx;
```

```
  double y3 = sy;
  double x4 = dx;
  double y4 = dy;

  double a1 = y2 - y1;
  double b1 = x1 - x2;
  double c1 = x2*y1 - x1*y2;

  double a2 = y4 - y3;
  double b2 = x3 - x4;
  double c2 = x4*y3 - x3*y4;

  double denom = a1*b2 - a2*b1;

  double x, y;          //the result;

  if(denom == 0) {
    return 0;           //parallel lines;
  }

  x = (b1*c2 - b2*c1)/denom;
  y = (a2*c1 - a1*c2)/denom;

  if(x > MIN(x1, x2) && x < MAX(x1, x2) &&
     x > MIN(x3, x4) && x < MAX(x3, x4))
    return 1;
  else return 0;
}

int
IPOWNeighbors::num_of_neighbors(struct ipow_neighbor *nblist){
  struct ipow_neighbor *temp = nblist;
  int counter = 0;
  while(temp){
    counter++;
    temp = temp->next_;
  }
  return counter;
}

void
IPOWNeighbors::free_neighbors(struct ipow_neighbor *nblist){
  struct ipow_neighbor *temp, *head;
  head = nblist;
  while(head){
    temp = head;
    head = head->next_;
```

```
    free(temp);
  }
}

nsaddr_t
IPOWNeighbors::peri_nexthop(int type_, nsaddr_t last,
                            double sx, double sy,
                            double dx, double dy){
  struct ipow_neighbor *planar_neighbors, *temp;
  double alpha, minangle;
  nsaddr_t nexthop=-1;

  if(type_){                        //GG planarizing
    planar_neighbors = gg_planarize();
  }else {                           //RNG planarizing
    planar_neighbors = rng_planarize();
  }

  if(last>-1){
    struct ipow_neighbor *lastnb = getnb(last);
    if(lastnb == NULL) {
      printf("Wrong last nb %d->%d\n", last, my_id_);
      exit(1);
    }
    alpha = angle(my_x_, my_y_, lastnb->x_, lastnb->y_);
  }
  else
    alpha = angle(my_x_, my_y_, dx, dy);
  temp = planar_neighbors;

  while(temp){
    if(temp->id_ != last){
      double delta;
      delta = angle(my_x_, my_y_, temp->x_, temp->y_);
      delta = delta - alpha;
      if(delta < 0.0) {
          delta = 2*PI + delta;
      }

      if(delta < minangle){
          minangle = delta;
          nexthop = temp->id_;
      }
    }
    temp = temp->next_;
  }
```

```
  if(num_of_neighbors(planar_neighbors) > 1 &&
      intersect(nexthop, sx, sy, dx, dy)){
     free_neighbors(planar_neighbors);
     return peri_nexthop(type_, nexthop, sx, sy, dx, dy);
  }
  return nexthop;
}

void
IPOWNeighbors::dump(){
  delalltimeout();

  FILE *fp = fopen(NB_TRACE_FILE, "a+");

  struct ipow_neighbor *temp = head_;
  fprintf(fp, "%d:\t", my_id_);
  while(temp){
    fprintf(fp, "%d ", temp->id_);
    temp = temp->next_;
  }
  fprintf(fp,"\n");
  fclose(fp);
}

struct hop_entry
*IPOWNeighbors::new_hop(struct hop_entry *hoplist_, nsaddr_t id){
  struct hop_entry *temp = hoplist_;
  while(temp && (temp->id_ != id)){
    temp = temp->next_;
  }
  if (temp == NULL){
   temp = (struct hop_entry*)malloc(sizeof(struct hop_entry));
   temp->id_ = id;
   temp->next_ = hoplist_;
   hoplist_ = temp;
  }
  return hoplist_;
}

struct hop_entry
*IPOWNeighbors::remove_hop(struct hop_entry *tempH, nsaddr_t id){
  struct hop_entry *temp;
  struct hop_entry *p, *q;

  q = NULL;
  p = tempH;
  while(p){
```

```
    temp = p;
    if(temp->id_ == id){
      p = temp->next_;
      if(q){
            q->next_ = p;
      }
      else {
            tempH = p;
      }
      free(temp);
      return tempH;
    }

    q = p;
    p = p->next_;
  }
    return tempH;
}
```

A.6 ipow_sinklist.h

```
/* Copyright (C) 2005 State University of New York, at Binghamton
 * All rights reserved.
 *
 * NOTICE: This software is provided "as is", without any warranty,
 * including any implied warranty for merchantability or fitness for a
 * particular purpose.  Under no circumstances shall SUNY Binghamton
 * or its faculty, staff, students or agents be liable for any use of,
 * misuse of, or inability to use this software, including incidental
 * and consequential damages.

 * License is hereby given to use, modify, and redistribute this
 * software, in whole or in part, for any commercial or non-commercial
 * purpose, provided that the user agrees to the terms of this
 * copyright notice, including disclaimer of warranty, and provided
 * that this copyright notice, including disclaimer of warranty, is
 * preserved in the source code and documentation of anything derived
 * from this software.  Any redistributor of this software or anything
 * derived from this software assumes responsibility for ensuring that
 * any parties to whom such a redistribution is made are fully aware of
 * the terms of this license and disclaimer.
 *
 */
```

```
/*
 * IPOW code for ns-2 version 2.29
 * Based in GPSR implementation of Ke Liu
 * CS Dept., State University of New York, at Binghamton
 *

 ********
 ********
 * Carlos Eduardo Silva Martínez
 * Maestría en Ingeniería Eléctrónica y de Computadores
 *
 *****
 * Universidad de los Andes
 * Bogotá Colombia 2007
 ********
 ********

 *
 */

/* ipow_sinklist.h : the sink table which maintain a list of data sinks
 *                   it is used for multiple data sink, which is not
 *                   a part of the design of IPOW.
 *
 *                   Not each node needs to maintain a list of the data sinks
 *                   Only the data source nodes may need this list
 */


#ifndef IPOW_SINK_LIST_H_
#define IPOW_SINK_LIST_H_

#include "config.h"

struct sink_entry {
  nsaddr_t id_;
  double x_;
  double y_;
  nsaddr_t lasthop_;
  int hops_;
  int seqno_;
  struct sink_entry *next_;
};

class Sinks {
  struct sink_entry *sinklist_;
 public:
```

```
  Sinks();
  bool new_sink(nsaddr_t, double, double, nsaddr_t, int, int);
  bool remove_sink(nsaddr_t);
  void getLocbyID(nsaddr_t, double&, double&, int&);
  void dump();
};

#endif
```

A.7 ipow_sinklist.cc

```
/* Copyright (C) 2005 State University of New York, at Binghamton
 * All rights reserved.
 *
 * NOTICE: This software is provided "as is", without any warranty,
 * including any implied warranty for merchantability or fitness for a
 * particular purpose.  Under no circumstances shall SUNY Binghamton
 * or its faculty, staff, students or agents be liable for any use of,
 * misuse of, or inability to use this software, including incidental
 * and consequential damages.
 * License is hereby given to use, modify, and redistribute this
 * software, in whole or in part, for any commercial or non-commercial
 * purpose, provided that the user agrees to the terms of this
 * copyright notice, including disclaimer of warranty, and provided
 * that this copyright notice, including disclaimer of warranty, is
 * preserved in the source code and documentation of anything derived
 * from this software.  Any redistributor of this software or anything
 * derived from this software assumes responsibility for ensuring that
 * any parties to whom such a redistribution is made are fully aware of
 * the terms of this license and disclaimer.
 *
 */
/*
 * IPOW code for ns-2 version 2.29
 * Based in GPSR implementation of Ke Liu
 * CS Dept., State University of New York, at Binghamton
 *

 ********
 ********
 * Carlos Eduardo Silva Martínez
 * Maestría en Ingeniería Eléctrónica y de Computadores
 *
 *****
 * Universidad de los Andes
 * Bogotá Colombia 2007
```

```
 ********
 ********
 *
 */

/* ipow_sinklist.cc : the implementation of the sink list class */

#include "ipow_sinklist.h"

Sinks::Sinks(){
  sinklist_ = NULL;
}

bool
Sinks::new_sink(nsaddr_t id, double x, double y,
                    nsaddr_t lasthop, int hops, int seqno){
  struct sink_entry *temp = sinklist_;
  while(temp){
    if(temp->id_ == id){
      if(temp->x_ == x && temp->y_ == y &&
           ((seqno <= temp->seqno_) || (temp->hops_ <= hops)))
          return false;
      else {
          temp->x_ = x;
          temp->y_ = y;
          temp->lasthop_ = lasthop;
          temp->hops_ = hops;
          temp->seqno_ = seqno;
          return true;
      }
    }
    temp = temp->next_;
  }
  temp = (struct sink_entry*)malloc(sizeof(struct sink_entry));
  temp->id_ = id;
  temp->x_ = x;
  temp->y_ = y;
  temp->lasthop_ = lasthop;
  temp->hops_ = hops;
  temp->seqno_ = seqno;
  temp->next_ = sinklist_;
  sinklist_ = temp;
  return true;
}
```

```
bool
Sinks::remove_sink(nsaddr_t id){
  struct sink_entry *temp;
  struct sink_entry *p, *q;

  q = NULL;
  p = sinklist_;
  while(p){
    temp = p;
    if(temp->id_ == id){
      p = temp->next_;
      if(q){
          q->next_ = p;
      }
      else {
          sinklist_ = p;
      }
      free(temp);
      return true;
    }

    q = p;
    p = p->next_;
  }
  return false;
}

void
Sinks::getLocbyID(nsaddr_t id, double &x, double &y, int &hops){
  struct sink_entry *temp = sinklist_;
  while(temp){
    if(temp->id_ == id){
      x = temp->x_;
      y = temp->y_;
      hops = temp->hops_;
      return;
    }
    temp = temp->next_;
  }
  x = -1.0;
  y = -1.0;
}
```

```
void
Sinks::dump(){
  struct sink_entry *temp = sinklist_;
  FILE *fp = fopen("sinklist.tr", "a+");
  while(temp){
    fprintf(fp, "%d\t%f\t%f\t%d\t%d\t%d\n",
            temp->id_, temp->x_, temp->y_,
            temp->lasthop_, temp->hops_, temp->seqno_);
    temp = temp->next_;
  }
  fclose(fp);
}
```

ANEXO B. CÓDIGO EN oTcl PARA NS-2

B.1 wireless-ipow.tcl

```
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
# are met:
# 1. Redistributions of source code must retain the above copyright
#    notice, this list of conditions and the following disclaimer.
# 2. Redistributions in binary form must reproduce the above copyright
#    notice, this list of conditions and the following disclaimer in the
#    documentation and/or other materials provided with the distribution.
# 3. All advertising materials mentioning features or use of this software
#    must display the following acknowledgement:
#      This product includes software developed by the Computer Systems
#      Engineering Group at Lawrence Berkeley Laboratory.
# 4. Neither the name of the University nor of the Laboratory may be used
#    to endorse or promote products derived from this software without
#    specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS
IS'' AND
# ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE
# ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
BE LIABLE
# FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
# DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS
# OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY
# OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF
# SUCH DAMAGE.
#
#
```

```
# Ported from CMU/Monarch's code, nov'98 -Padma.

########
## Carlos Eduardo Silva Martínez
## Maeestría en Ingeniería Eléctrónica y de Computadores
#####
## Universidad de los Andes
## Bogotá Colombia 2007
########

## In this script is implemented the main program that includes the
## simulation parameters such as: propagaion model, physical layer
## parameters configuration, energy model, traces, etc.
##



#
# ============================================================
# Default Script Options
#
# ============================================================
set opt(chan)           Channel/WirelessChannel
set opt(prop)           Propagation/TwoRayGround
set opt(netif)          Phy/WirelessPhy
set opt(mac)            Mac/802_11
set opt(ifq)            Queue/DropTail/PriQueue    ;# for dsdv
set opt(ll)                 LL
set opt(ant)        Antenna/OmniAntenna

set opt(x)                  160             ;# X dimension of the topography: 160, 230,
277, 320 y 360
set opt(y)                  160             ;# Y dimension of the topography: 160, 230,
277, 320 y 360
set opt(cp)                 "/home/ns-allinone-2.29/ns-2.29/tesis/sc/cbr100.tcl"
set opt(sc)                 "/home/ns-allinone-2.29/ns-2.29/tesis/sc/grid-
deploy10x10_rdm.tcl"

set opt(ifqlen)         50              ;# max packet in ifq
set opt(nn)                 50              ;# number of nodes: 50, 150, 200 y 250
set opt(seed)           0.0
set opt(stop)           75.0            ;# simulation time
set opt(tr)                 trace_IPOW.tr       ;# trace file
set opt(nam)        nam.out.tr
set opt(rp)         IPOW            ;# routing protocol script (dsr or dsdv)
```

```
set opt(lm)             "off"           ;# log movement
set opt(initialenergy)  80.0            ;#Nuevos
set opt(energyModel)    EnergyModel     ;#Nuevos
set opt(P_tx)           0.660   ;#Nuevos
set opt(P_rx)           0.395   ;#Nuevos
set opt(P_idle)         0.035   ;#Nuevos

#
======================================================================
=====

LL set mindelay_                50us
LL set delay_           25us
LL set bandwidth_               0       ;# not used

Agent/Null set sport_       0
Agent/Null set dport_       0

Agent/CBR set sport_        0
Agent/CBR set dport_            0

Agent/TCPSink set sport_    0
Agent/TCPSink set dport_    0

Agent/TCP set sport_        0
Agent/TCP set dport_        0
Agent/TCP set packetSize_   1460

Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1

# unity gain, omni-directional antennas
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_      0
Antenna/OmniAntenna set Y_      0
Antenna/OmniAntenna set Z_      0.5
Antenna/OmniAntenna set Gt_     1.0
Antenna/OmniAntenna set Gr_     1.0

# Initialize the SharedMedia interface with parameters to make
# it work like the 914MHz Lucent WaveLAN DSSS radio interface
Phy/WirelessPhy set CPThresh_       10.0
Phy/WirelessPhy set CSThresh_       1.559e-11
#Phy/WirelessPhy set RXThresh_      3.652e-10
Phy/WirelessPhy set RXThresh_               3.652e-11
Phy/WirelessPhy set Rb_                     2*1e6
Phy/WirelessPhy set freq_           914e+6
Phy/WirelessPhy set L_                      1.0
```

```
# The transimssion radio range
#Phy/WirelessPhy set Pt_      6.9872e-4    ;# ?m
Phy/WirelessPhy set Pt_                8.5872e-4    ;# 40m
#Phy/WirelessPhy set Pt_      1.33826e-3   ;# 50m
#Phy/WirelessPhy set Pt_      7.214e-3     ;# 100m
#Phy/WirelessPhy set Pt_      0.2818       ;# 250m
#
==========================================================
=====

# Agent/IPOW setting
Agent/IPOW set planar_type_          1   ;#1=GG planarize, 0=RNG planarize
Agent/IPOW set hello_period_  5.0 ;#Hello message period
Agent/IPOW set alfa_          4;   #variable alfa
Agent/IPOW set constante_     100000;# constante c

#
==========================================================
=====

proc usage { argv0 }  {
        puts "Usage: $argv0"
        puts "\tmandatory arguments:"
        puts "\t\t\[-x MAXX\] \[-y MAXY\]"
        puts "\toptional arguments:"
        puts "\t\t\[-cp conn pattern\] \[-sc scenario\] \[-nn nodes\]"
        puts "\t\t\[-seed seed\] \[-stop sec\] \[-tr tracefile\]\n"
}

proc getopt {argc argv} {
        global opt
        lappend optlist cp nn seed sc stop tr x y

        for {set i 0} {$i < $argc} {incr i} {
                set arg [lindex $argv $i]
                if {[string range $arg 0 0] != "-"} continue

                set name [string range $arg 1 end]
                set opt($name) [lindex $argv [expr $i+1]]
        }
}

proc log-movement {} {
   global logtimer ns_ ns

   set ns $ns_
```

```
    source /home/ns-allinone-2.29/ns-2.29/tcl/mobility/timer.tcl
    Class LogTimer -superclass Timer
    LogTimer instproc timeout {} {
        global opt node_;
        for {set i 0} {$i < $opt(nn)} {incr i} {
            $node_($i) log-movement
        }
        $self sched 0.1
    }

    set logtimer [new LogTimer]
    $logtimer sched 0.1
}

source /home/ns-allinone-2.29/ns-2.29/tcl/lib/ns-bsnode.tcl
source /home/ns-allinone-2.29/ns-2.29/tcl/mobility/com.tcl

# do the get opt again incase the routing protocol file added some more
# options to look for
getopt $argc $argv

if { $opt(x) == 0 || $opt(y) == 0 } {
        usage $argv0
        exit 1
}

if {$opt(seed) > 0} {
        puts "Seeding Random number generator with $opt(seed)\n"
        ns-random $opt(seed)
}

#
# Initialize Global Variables
#
set ns_         [new Simulator]
set chan        [new $opt(chan)]
set prop        [new $opt(prop)]
set topo        [new Topography]


set tracefd   [open $opt(tr) w]
$ns_ trace-all  $tracefd ;#Probar con trazas particulares no trace-all

set namfile [open $opt(nam) w]
#$ns_ namtrace-all $namfile
$ns_ namtrace-all-wireless $namfile opt(x) opt(y)
```

```
$topo load_flatgrid $opt(x) $opt(y)

$prop topography $topo

#
# Create God
#
set god_ [create-god $opt(nn)]

#Use new trace
$ns_ use-newtrace

#
#  Create the specified number of nodes $opt(nn) and "attach" them
#  the channel.
#  Each routing protocol script is expected to have defined a proc
#  create-mobile-node that builds a mobile node and inserts it into the
#  array global $node_($i)
#

$ns_ node-config -adhocRouting IPOW \
                 -llType $opt(ll) \
                 -macType $opt(mac) \
                 -ifqType $opt(ifq) \
                 -ifqLen $opt(ifqlen) \
                 -antType $opt(ant) \
                 -propType $opt(prop) \
                 -phyType $opt(netif) \
                 -channelType $opt(chan) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF \
                 -movementTrace OFF \
                -energyModel $opt(energyModel)\
                -initialEnergy 80.0 \
                -rxPower $opt(P_tx) \
                -txPower $opt(P_rx) \
                      -idlePower $opt(P_idle)

source ./ipow.tcl

for {set i 0} {$i < $opt(nn) } {incr i} {
   ipow-create-mobile-node $i
}

#
```

```
# Source the Connection and Movement scripts
#
if { $opt(cp) == "" } {
        puts "*** NOTE: no connection pattern specified."
      set opt(cp) "none"
} else {
        puts "Loading connection pattern..."
        source $opt(cp)
}

#
# Tell all the nodes when the simulation ends
#
for {set i 0} {$i < $opt(nn) } {incr i} {
   $ns_ at $opt(stop).000000001 "$node_($i) reset";
}
$ns_ at $opt(stop).00000001 "puts \"NS EXITING...\" ; $ns_ halt"


if { $opt(sc) == "" } {
        puts "*** NOTE: no scenario file specified."
      set opt(sc) "none"
} else {
        puts "Loading scenario file..."
        source $opt(sc)
        puts "Load complete..."
}

puts $tracefd "M 0.0 nn $opt(nn) x $opt(x) y $opt(y) rp $opt(rp)"
puts $tracefd "M 0.0 sc $opt(sc) cp $opt(cp) seed $opt(seed)"
puts $tracefd "M 0.0 prop $opt(prop) ant $opt(ant)"

puts "Starting Simulation..."

proc finish {} {
   global ns_ tracefd namfile
   $ns_ flush-trace
   close $tracefd
   close $namfile
   exit 0
}

$ns_ at $opt(stop) "finish"

$ns_ run
```

B.2 ipow.tcl

```
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
# are met:
# 1. Redistributions of source code must retain the above copyright
#    notice, this list of conditions and the following disclaimer.
# 2. Redistributions in binary form must reproduce the above copyright
#    notice, this list of conditions and the following disclaimer in the
#    documentation and/or other materials provided with the distribution.
# 3. All advertising materials mentioning features or use of this software
#    must display the following acknowledgement:
#      This product includes software developed by the Computer Systems
#      Engineering Group at Lawrence Berkeley Laboratory.
# 4. Neither the name of the University nor of the Laboratory may be used
#    to endorse or promote products derived from this software without
#    specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS
IS'' AND
# ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE
# ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
BE LIABLE
# FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
# DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS
# OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY
# OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF
# SUCH DAMAGE.
#
# Ported from CMU/Monarch's code, nov'98 -Padma.

########
## Carlos Eduardo Silva Martínez
## Maeestría en Ingeniería Eléctrónica y de Computadores
```

```
#####
## Universidad de los Andes
## Bogotá Colombia 2007
########

## In this script is implemented the node and routing agent creation
##


Agent/IPOW set sport_       0
Agent/IPOW set dport_       0

puts "IPOW configuration file"
#
==========================================================
=====

proc create-ipow-routing-agent { node id } {
   global ns_ ragent_ tracefd opt

   #
   #  Create the Routing Agent and attach it to port 255.
   #
   set ragent_($id) [new Agent/IPOW]
   set ragent $ragent_($id)

   set addr [$node node-addr]

   $ragent addr $addr
   $ragent node $node
   if [Simulator set mobile_ip_] {
         $ragent port-dmux [$node set dmux_]
   }
   $node addr $addr
   $node set ragent_ $ragent

         set T [new Trace/Generic]
         $T target [$ns_ set nullAgent_]
         $T attach $tracefd
         $T set src_ $id
         $ragent tracetarget $T

   $node attach $ragent [Node set rtagent_port_]
}

proc ipow-create-mobile-node { id args } {
         global ns ns_ chan prop topo tracefd opt node_
```

```
        set ns_ [Simulator instance]
    set node_($id) [new Node/MobileNode]
        set node $node_($id)
        $node random-motion 0                ;# disable random motion
        $node topography $topo

        # XXX Activate energy model so that we can use sleep, etc. But put on
        # a very large initial energy so it'll never run out of it.
        if [info exists opt(energyModel)] {
                $node addenergymodel [new $opt(energyModel) $node 80 0.600 0.395]
        }

        #
        # This Trace Target is used to log changes in direction
        # and velocity for the mobile node.
        #
        set T [new Trace/Generic]
        $T target [$ns_ set nullAgent_]
        $T attach $tracefd
        $T set src_ $id
        $node log-target $T

        if ![info exist inerrProc_] {
                set inerrProc_ ""
        }
        if ![info exist outerrProc_] {
                set outerrProc_ ""
        }
        if ![info exist FECProc_] {
                set FECProc_ ""
        }

        $node add-interface $chan $prop $opt(ll) $opt(mac)          \
           $opt(ifq) $opt(ifqlen) $opt(netif) $opt(ant) \
           $topo $inerrProc_ $outerrProc_ $FECProc_

        #
        # Create a Routing Agent for the Node
        #
        create-ipow-routing-agent $node $id

    $ns_ at 0.0 "$node_($id) start"

    return $node
}
```

B.3 cbr100.tcl

```
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
# are met:
# 1. Redistributions of source code must retain the above copyright
#    notice, this list of conditions and the following disclaimer.
# 2. Redistributions in binary form must reproduce the above copyright
#    notice, this list of conditions and the following disclaimer in the
#    documentation and/or other materials provided with the distribution.
# 3. All advertising materials mentioning features or use of this software
#    must display the following acknowledgement:
#      This product includes software developed by the Computer Systems
#      Engineering Group at Lawrence Berkeley Laboratory.
# 4. Neither the name of the University nor of the Laboratory may be used
#    to endorse or promote products derived from this software without
#    specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS
IS'' AND
# ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE
# ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
BE LIABLE
# FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
# DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS
# OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY
# OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF
# SUCH DAMAGE.
#
#
# Ported from CMU/Monarch's code, nov'98 -Padma.
```

```
########
## Carlos Eduardo Silva Martínez
## Maeestría en Ingeniería Eléctrónica y de Computadores
#####
## Universidad de los Andes
## Bogotá Colombia 2007
########

## In this file is the traffic class to use, your rate and packet size.
## Furthermore here set the data source and sink
##


# IPOW routing agent settings
for {set i 0} {$i < $opt(nn)} {incr i} {
   $ns_ at 0.00002 "$ragent_($i) turnon"
   $ns_ at 20.0 "$ragent_($i) neighborlist"
}

$ns_ at 5.0 "$ragent_(0) startSink 10.0"

# IPOW routing agent dumps
$ns_ at 25.0 "$ragent_(24) sinklist"

# Upper layer agents/applications behavior
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(1)

set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_([expr $opt(nn)-1]) $udp_(1)

set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 32
$cbr_(1) set interval_ 2.0
$cbr_(1) set random_ 1
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 5.0 "$cbr_(1) start"
$ns_ at 65.0 "$cbr_(1) stop"
```

B.4 Grid-deploy10x10_rdm.tcl

```
########
## Carlos Eduardo Silva Martínez
## Maeestría en Ingeniería Eléctrónica y de Computadores
#####
## Universidad de los Andes
## Bogotá Colombia 2007
########

## In this file is implemented the method to place the nodes
## in randoms position on the geographic area
##



ns-random 0                     ;#Para que en cada simulación sea distinta
set posRNG [new RNG]
set pos_ [new RandomVariable/Uniform]    ;#Distribución uniforme
$pos_ set min_ 0
$pos_ set max_ 160   ;# 160 metros para 50 nodos
#$pos_ set max_ 230  ;# 230 metros para 100 nodos
#$pos_ set max_ 277  ;# 277 metros para 150 nodos
#$pos_ set max_ 320  ;# 320 metros para 200 nodos
#$pos_ set max_ 360  ;# 360 metros para 250 nodos
$pos_ use-rng $posRNG

# Para ubicar aleatoriamente el nodo fuente
set posFte_ [new RandomVariable/Uniform]        ;#Distribución uniforme
$posFte_ set min_ 0
$posFte_ set max_ 70
$posFte_ use-rng $posRNG

$node_([expr $opt(nn)-1]) set X_ [expr round([$posFte_ value])]
$node_([expr $opt(nn)-1]) set Y_ [expr round([$posFte_ value])]
$node_([expr $opt(nn)-1]) set Z_ 0.000000000000

for {set i 0} {$i < [expr $opt(nn)-1]} {incr i} {
        $node_($i) set X_ [expr round([$pos_ value])]
        $node_($i) set Y_ [expr round([$pos_ value])]
        $node_($i) set Z_ 0.000000000000
}
```

Referencias

[1]. J. al-Karaki y A. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey", en IEEE Wireless Communications Magazine, Vol 11, Issue 6, pp 6 – 28, Diciembre 2004.

[2]. S. Park, A. Savvides y M. Srivastaba, "Simulating Networks of Wireless Sensors", en Proceedings of the 2001 Winter Simulation Conference, pp 1330 – 1338, 2001.

[3]. 14. H. Tsen, S. Yang, P. Chuan, E Wu, y G. Chen, "An Energy Consumption Analytic Model for a Wireless Sensor MAC Protocol", en IEEE Vehicular Technology Conference VTC2004, Vol 6, pp 4533 - 4537, Septiembre 2004.

[4]. J. L. Hill, "System Architecture for Wireless Sensor Networks", PhD Thesis, Graduate Division, University of California, Berkeley, Primavera 2003.

[5]. I. Luna-Vazquez, "Implementation and Simulation of Routing Protocols for Wireless Sensor Networks", Master Thesis, Department of Electrical Engineering and Computer Sciences, Universitat Siegen, Siegen, Marzo 2006.

[6]. I. Teixeira, "Roteamento com Balanceamento de Consumo de Energia para Redes de Sensores sem Fio", Tesis de maestría, Coordenacao dos Programas de Pos-Graduacao, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Abril 2005.

[7]. Proyectos de investigación del Center for Embedded Network Sensing (CENS) de la Universidad de California en Los Angeles. Página web, http://research.cens.ucla.edu/areas/2007/Systems/projects.htm, visitada el 15 de Junio de 2007.

[8]. Proyecto Smart Dust de la Universidad de Berkeley. Página web, http://robotics.eecs.berkeley.edu/~pister/SmartDust/, visitada el 15 de Junio de 2007.

[9]. Proyectos de investigación en redes de sensores de la Universidad de Berkeley. Página web, http://www.truststc.org/sensornets/, visitada el 15 de Junio de 2007.

[10]. Proyecto de investigación en microsensores del Instituto Tecnológico de Massachussets. Página web, http://mtlweb.mit.edu/researchgroups/icsystems/projects.html, visitada el 15 de Junio de 2007.

[11]. Proyecto de investigación μAMPS. Página web, http://mtlweb.mit.edu/researchgroups/icsystems/uamps/, visitada el 15 de Junio de 2007

[12]. Proyecto PicoRadio. Página web, http://bwrc.eecs.berkeley.edu/Research/Pico_Radio/Default.htm, visitada el 15 de Junio de 2007.

[13]. Proyecto Gnomes. Página web, http://cmclab.rice.edu/projects/gnomes, visitada el 15 de Junio de 2007.

[14]. Proyectos del Autonomous Networks Research Group. Página web, http://ceng.usc.edu/~anrg/SensorNetBib.html, visitada el 15 de Junio de 2007.

[15]. S. Park, A. Savvides y M. Srivastaba, "Battery Capacity Measurement And Analysis Using Lithium Coin Cell Battery", en proceedings of the ISLPED 01, Agosto 2001.

[16]. K. Akkaya y M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks", en Elsevier Ad-Hoc Networks Journal, Vol. 3, Issue 3, pp 325 – 349. Mayo 2005.

[17]. W.R. Heinzelman, A. Chandrakasan y H. Balakrishnan, "Energy-efficient communication protocols for Wireless Microsensor Networks", en Proceedings of the 33rd Hawaii International Conference on System Sciences, pp 1 – 10, Enero 2000.

[18]. A. Menéndez, J. J. Pérez, J. P. Sebastiá, "Red de sensores inalámbricos para monitorización de terrenos mediante tecnología IEEE 802.15.4"

[19]. W. Ye y J. Heidemann, "Medium Access Control in Wireless Sensor Networks", USC/ISI Technical report ISI-TR-580, Octubre 2003.

[20]. K. Akkaya y M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks", en Elsevier Ad-Hoc Networks Journal, Vol. 3, Issue 3, pp 325 – 349. Mayo 2005.

[21]. W. Ye, J. Heidemann y D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks", en Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), volume 3, pp 1567 - 1576, Junio 2002.

[22]. S. Sing y C.S. Raghavendra, "PAMAS: Power Aware Multi-Access protocol with Singalling for Ad Hoc Networks", ACM-CCR, 28, 3, pp 5 – 26, Julio 1998.

[23]. R. Mini, M. do Val Machado, A. Loureiro y B. Nath, "Prediction-based energy map for wireless sensor networks", en Elsevier Ad-Hoc Networks Journal, Vol. 3, Issue 2, pp. 235 - 253, Marzo 2005.

[24]. 802.15.4-2003 IEEE Standar for Information Technology-Part 15.4: Wireless Medium Acces Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Área Networks (LR-WPANS), 2003.

[25]. T. van Dam, K. Langendoem, "An Adaptative Energy-Efficient MAC Protocol for Wireless Sensor Networks", en Sensys 2003, Los Angeles, Noviembre, 2003.

[26]. B. Liu, N. Bulusu, H. Pham y S. Jha, "CSMAC: A Novel DS-CDMA Based MAC Protocol For Wireless Sensor Networks", en IEEE Global Telecommunications Conference Workshops, pp. 33 – 38, Diciembre, 2004.

[27]. G. Lu, B. Krishnamachari, C. Raghavendra, "An Adaptative Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks", en Proceedings of the 18th International Paralell and Distributed Procesing Symposium (IPDPS´04).

[28]. W. Ye, J. Heidemann, y D. Estrin, "Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks", IEEE/ACM Transactions on Networking, Vol. 12, Nº. 3, Junio, 2004.

[29]. S. Lindsey, C. Raghavendra, PEGASIS: Power-Efficient Gathering in Sensor Information Systems", IEEE Aerospace Conference Proceedings, 2002, Vol. 3, 9-16 pp. 1125-1130.

[30]. C. Intanagonwiwat, R. Govindan, y D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks", Proceedings of the ACM Mobi-Com'00, Boston, MA, pp. 56–67, Agosto 2000.

[31]. I. Raicu, "Efficient Even Distribution of Power Consumption in Wireless Sensor Networks", en ISCA 18th International Conference on Computers and Their Applications, CATA 2003, Honolulu, Hawai, USA, Marzo 2003. http://arxiv.org/abs/cs.NI/0411040

[32]. I. Raicu, L. Schwiebert, S. Fowler y S. Gupta, "e3D: An Energy-Efficient Routing Algorithm for Wireless Sensor Networks" en IEEE ISSNIP The International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Melbourne, Australia, Diciembre 2004. http://arxiv.org/abs/cs.NI/0411043

[33]. Q. Gao, K.J. Blow, D.J. Holding, I.W. Marshall y X. Peng, "Radio range adjustment for energy efficient wireless sensor networks" en Elsevier Ad-Hoc Networks Journal, Vol. 4, Issue 1, pp 75 – 82. Enero 2006.

[34]. B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in Proc. 6th annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '00). New York, NY, USA: ACM Press, 2000, pp. 243–254.

[35]. J. Kuruvila, A.Nayak, and I. Stojmenovic, "Progress and Location Based Localized Power Aware Routing for ad hoc and Sensor Wireless Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 11, pp. 1122–1133, November 2001.

[36]. J. A. Sánchez, "Algoritmos de Encaminamiento Multicast con reducido consumo energético para Redes de Sensores Inalámbricos", Tesis Doctoral, Departamento de Ingeniería de la Información y las Comunicaciones, Universidad de Murcia, Murcia, Septiembre de 2006.

[37]. A. Giridhar, y P. Kumar, "Maximizing the functional lifetime of sensor networks", en Fourth International Symposium on Information Processing in Sensor Networks, Abril, 2005.

[38]. The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns/

[39]. The VINT Project, "The ns Manual (formerly ns Notes and Documentation)", en http://www.isi.edu/nsnam/ns/doc-stable/ns_doc.pdf, Octubre 2006.

[40]. Cygwin. http://www.cygwin.com

[41]. Marc Greis. Tutorial for the Network Simulator "ns". http://www.isi.edu/nsnam/ns/tutorial/index.html

[42]. Ns by example. http://nile.wpi.edu/NS/menu.html

[43]. F. J. Ross y P. M. Ruiz, "Implementing a New Manet Unicast Routing Protocol in ns-2", http://masimum.dif.um.es/nsrt-howto/html/nsrt-howto.html

[44]. C. Intanagonwiwat, R. Govindan, R. E Estrin, D. Heiddeman, F. Silva, "Directed diffusion: for Wireless Sensor Networking," IEEE/ACM Transactions on Networking, Volume 11, no 1, pp. 2 – 16, Febrero 2003.

[45]. B.A. Galvao, "Um Protocolo Tolerante a Falhas para Disseminacao de Dados em Redes de Sensores Sem Fio", Tesis de maestría, Núcleo de computacao Eletronica / Instituto de Matemática da Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2005.

[46]. K. Liu, "GPSR: Greedy Perimeter Stateless Routing code for version 2.26 or later", State University of New York at Binghamton, http://www.cs.binghamton.edu/~kliu/research/ns2code/index.html

[47]. T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher. Speed: A staleless protocol for real-time communication in sensor networks. In ICDCS, Mayo, 2003.

[48]. S. Kim, V. Vasireddy, y K. Harfoush, "Scalable coordination for sensor networks in challenging environments", in Proceedings of the 2007 ACM symposium on Applied computing, Seoul, Korea, pp. 214 – 221, 2007.

[49]. S. Ratnaraj, S. Jagannathan, V. Rao, "OEDSR: Optimized Energy-Delay Subnetwork Routing in Wireless Sensor Networks", en Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, pp. 330 – 335, Abril, 2006.