Universidad Autónoma de San Luis Potosí

Facultad de Ingeniería

Algoritmo de Enrutamiento Para Redes de Sensores Inalámbricos Basado en el Modelo Oferta-Demanda que Optimiza el Uso de Energía

TESIS

Para obtener el Grado de

Maestría en Ingeniería de la Computación

PRESENTA:

Ing. Neftalí Reyes Izaguirre

ASESOR:

Dr. J. Jesús Acosta Elías

San Luis Potosí, S.L.P. Junio 2010

Agradecimientos

A mis padres y hermanos.

A mis compañeros y amigos.

A todas las personas que me apoyaron y acompañaron en la realización de este proyecto.

A Conacyt.

Un agradecimiento especial a mi asesor Dr. J. Jesús Acosta Elías un salvador de causas perdidas.

San Luis Potosí, 2010

UNIVERSIDAD AUTÓNOMA
DE SAN LUIS POTOSÍ

22 de abril de 2010

ING. NEFTALI REYES IZAGUIRRE
P R E S E N T E. –

En atención a su solicitud de Temario, presentada por el **Dr. J. Jesús Acosta Elías** Asesor de la Tesis que desarrollará Usted, con el objeto de obtener el Grado de ***Maestría*** **en** ***Ingeniería de la Computación.*** Me es grato comunicarle que en la Sesión de Consejo Técnico Consultivo celebrada el día 22 de abril del presente año, fue aprobado el Temario propuesto:

TEMARIO:

"ALGORITMO DE ENRUTAMIENTO PARA REDES DE SENSORES INALÁMBRICOS BASADO EN EL MODELO OFERTA-DEMANDA QUE OPTIMIZA EL USO DE LA ENERGÍA"

INTRODUCCIÓN
1. TRABAJO RELACIONADO.
2. ALGORITMO PROPUESTO.
3. IMPLEMENTACIÓN EN NS-2.
4. ANÁLISIS DE DESEMPEÑO.

CONCLUSIONES Y TRABAJO A FUTURO.
REFERENCIAS.

"MODOS ET CUNCTARUM RERUM MENSURAS AUDEBO"

A T E N T A M E N T E

ING. ARMANDO VIRAMONTES ALDANA
DIRECTOR

UNIVERSIDAD AUTÓNOMA
DE SAN LUIS POTOSÍ
FACULTAD DE INGENIERÍA
DIRECCIÓN

**bcb

FACULTAD
DE INGENIERÍA
Av. Manuel Nava 8
Zona Universitaria • CP 78290
San Luis Potosí, S.L.P.
tel. (444) 826 2330 al39
fax (444) 826 2336
www.uaslp.mx

Índice general

Introducción

Una red inalámbrica de sensores[1], consta de un conjunto de nodos autónomos distribuidos en una región determinada, funcionan de forma cooperativa para supervisar las condiciones físicas o ambientales. Cada nodo sensor tiene capacidad de procesamiento, y puede tener múltiples sensores integrados, operando en modos como el acústico, sísmico, infrarrojo (IR), magnético e incluso en modo de captura de imágenes y radar. También almacenan la información sobre los enlaces inalámbricos hacia los nodos vecinos, y el conocimiento de localización y posicionamiento a través del sistema de posicionamiento global (GPS) o bien a través de algoritmos de posicionamiento local, ya que para muchas de las aplicaciones potenciales de las redes de sensores es requisito imprescindible el conocimiento de la localización.

Por lo tanto, cada nodo está constituido al menos por los siguientes elementos:

- CPU
- Sistema de radio comunicación
- Conjunto de sensores

Los nodos se comunican entre sí por paso de mensajes a través de enlaces radio-eléctricos, ver figura 1. Debido a que disponen de potencia de transmisión limitada, ya sea por el uso de baterías ó por el uso de bandas libres (en las bandas libres tales como la de 900 mhz, 2.4 Ghz, etc. los gobiernos establecen límites en la potencia transmitida), por lo tanto la comunicación se realiza por medio de múltiples saltos radio-eléctricos, ver figura 2.

Los nodos no son colocados de forma determinista en posiciones exactas, sino que son esparcidos sobre la superficie de tal manera que quedan en posiciones aleatorias. Esto es

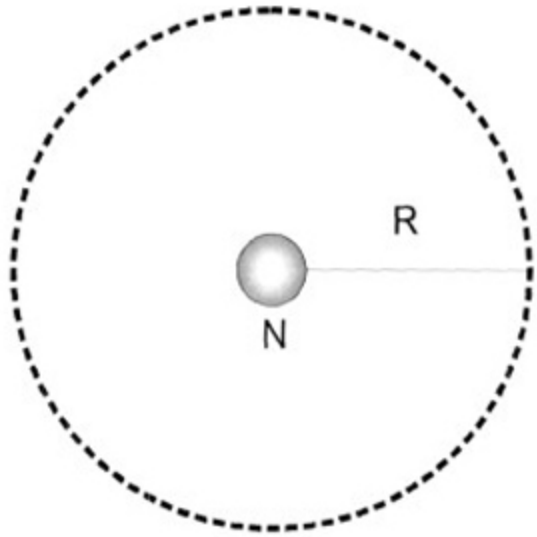

Figura 1: *Nodo sensor N en el que se puede observar el alcance de comunicación radio-eléctrico R.*

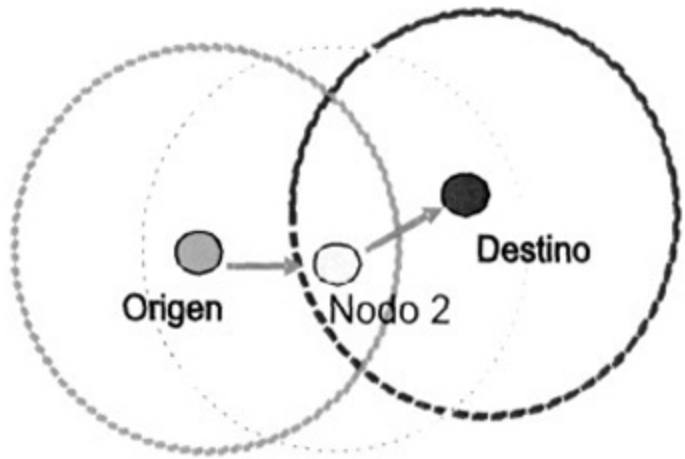

Figura 2: *Los círculo pequeños de color sólido representan al nodo y los círculos en línea punteada representan el alcance en la radio-comunicación de cada nodo. El nodo origen envía datos al nodo destino. Debido a que el nodo destino no está dentro de su radio de alcance, la comunicación se realiza a través del nodo 2. En otras palabras, la información enviada por el nodo origen llega hasta el nodo dos, de ahí salta al nodo destino*

debido a que las aplicaciones de estas redes están relacionadas con la medición de variables en lugares de difícil acceso, tales como campos de batalla, áreas bajo alerta sanitaria, lugares montañosos, etc. Una manera de esparcirlos es lanzarlos desde un avión (ver figura

3) que vuela sobre el área de interés.

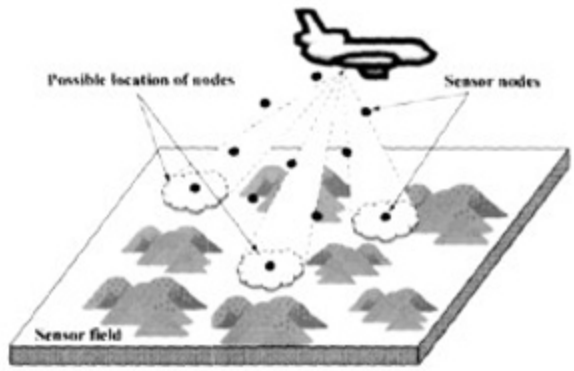

Figura 3: *Los nodos son lanzados desde un avión, al caer algunos pueden dañarse y el resto quedan en posiciones indeterminadas. La nube representa la región con mayor probabilidad de encontrarse un nodo. El circulo negro representa un nodo. La figura fué tomada de [2]*

El desarrollo de redes inalámbricas de sensores fue motivado por aplicaciones militares, como la vigilancia del campo de batalla. En la actualidad su uso se extiende a muchas áreas, como la industrial, civil, médica, etc. Una red inalámbrica de sensores normalmente constituye una red inalámbrica ad-hoc, con las siguientes características:

- Sin autoridad central. Es decir, no hay un nodo que sea el líder de todo el sistema, o un servidor que almacene la información actualizada y coordine de forma centralizada todas las acciones. Debido a que en un tiempo t es imposible saber que nodos están activos, cual es su posición, etc. Si se intentara desarrollar un sistema centralizado se gastaría ancho de banda y batería en transmitir información al nodo central. Pero como los retardos del sistema son altos e impredecibles, hasta después de un Δt se podría tener toda la información. Pero para entonces el sistema ya habrá cambiado.

- Sin infraestructura. En las redes alambradas existen nodos especiales que se encargan de re-enviar datos(routers y gateways). En las redes de sensores no existen estos enrutadores, en cambio cada nodo tiene la capacidad de enrutar datos. Cada sensor soporta un protocolo de enrutamiento multi-saltos debido a que cada nodo debe de enviar la información recaudada a un punto de recaudación y la probabilidad es muy

alta de que el punto de recaudación esté a una distancia mucho mayor que su radio de alcance. Por lo tanto la información seguirá una ruta saltando de nodo en nodo hasta llegar al destino.

- Alta probabilidad de fallo. Tanto enlaces como nodos tienen una probabilidad muy alta de fallar. Por lo tanto el sistema debe soportar estos fallos.

- Los nodos son alimentados por baterías sin posibilidad de recargarse. Por lo tanto un sensor va a funcionar hasta que se le agote la batería.

En los algoritmos de enrutamiento es importante tomar en cuenta que cada nodo en la ruta gastará energía en retransmitir la información. La vida de los sensores está limitada por la batería, debido a esto se tienen que elaborar protocolos de enrutamiento especiales para optimizar el ahorro de energía, por tal motivo no es recomendable utilizar los protocolos clásicos de las redes ad-hoc.

Por ejemplo, el protocolo Ad hoc On Demand Distance Vector (AODV)[3], que es un protocolo basado en el principio de bajo demanda, es decir, construye la ruta entre los nodos que se van necesitando y mantiene la ruta mientras se necesite; este protocolo solo encuentra la ruta más corta, es decir la que genere menos saltos. Esto puede provocar que ciertos nodos se queden sin energía rápidamente y por consecuencia no se obtengan los datos de la zona que le corresponde al nodo en cuestión, ver figura 4.

Por este motivo, en años recientes, se ha empezado a desarrollar protocolos de enrutamiento que ayudan a optimizar el ahorro de energía del sistema. Estos algoritmos se tratarán en el capítulo 1 Trabajo relacionado,de esta tesis.

Protocolo de enrutamiento

Un protocolo de enrutamiento especifica el camino que debe seguir la información para llegar a su destino, buscando la ruta más óptima. Dependiendo del criterio para considerar qué es lo más óptimo, estos protocolos pueden variar. Algunos de los criterios más utilizados para encontrar el camino más óptimo son:

- La cantidad menor de saltos.

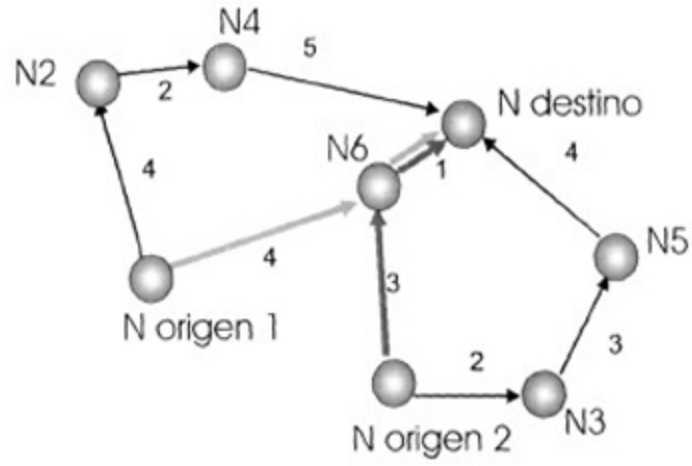

Figura 4: *Los círculos representan nodos, las flechas muestran las posibles rutas a seguir los datos desde los nodos origen al nodo destino. Se tienen dos nodos origen y un nodo destino. Si la ruta se elige en función de la menor cantidad de saltos, todo el tráfico pasaría a través del nodo N6, lo que originaría que al nodo N6 se le agote la batería mucho más rápido que al resto de nodos.*

- La menor latencia en la entrega de información.
- El ahorro de energía.

Actualmente existen varios protocolos de enrutamiento para redes de sensores inalámbricos, algunos de los considerados más importantes son:

- Flooding and Gossiping[13]. Son dos algoritmos de enrutamiento clásicos para el envío de datos en redes de sensores sin la necesidad de mantenimiento de rutas. En "Flooding", cada nodo que recibe un paquete de datos, lo re-envía a todos sus vecinos que están dentro de su radio de alcance. Este proceso continúa hasta que el paquete llega a su destino ó se alcanza el número máximo de saltos. En "Gossiping", cada nodo que recibe un paquete lo envía solamente a uno de sus vecinos, el cual ha sido seleccionado de manera aleatoria.
- Directed Diffusion[14]. Es un algoritmo de enrutamiento centrado en los datos. Cuando un nodo necesita datos, envía la petición especificando un conjunto de atributos que describen los datos deseados (interés). Los nodos que generan los datos,

únicamente envían información si han recibido alguna petición que coincide con sus datos. Un nodo intermedio almacena el interés, conjuntamente con los posibles vecinos más cercanos al nodo que solicitó los datos en un buffer. Cuando recibe datos que coinciden con el interés almacenado, selecciona del buffer el vecino al que le re-enviará el mensaje.

- LEACH[15]. Es un protocolo jerárquico conformado por cúmulos (clusters) de nodos. La formación de estos clusters es distribuida, basada en un subconjunto predeterminado de los nodos que se eligen aleatoriamente como líderes (cluster head). la función de los líderes es comprimir la información que reciben de todos los nodos integrantes del cluster y enviar un sólo mensaje con ésta información a la estación base, reduciendo de esta manera la cantidad de transmisiones.

Estos protocolos no han sido diseñados específicamente para el ahorro de energía para alargar la vida del sistema. Sin embargo desde hace algunos pocos años a la fecha, se está trabajando activamente en desarrollar protocolos de enrutamiento para redes ad hoc y redes de sensores con el objetivo de ahorro de energía.

Algunos de estos protocolos son complejos modelos matemáticos basados en la teoría de juegos, otros usan teoría de control. Pero son modelos que no toman en cuenta que las redes de sensores son sistemas distribuidos, en las que no es posible tener un control centralizado. Poseen alta e impredecible latencia, además de ser sistemas muy dinámicos. Es por eso que en esta tesis se plantea un nuevo algoritmo de routing basado en el modelo microeconómico de la oferta y la demanda. Este es un modelo totalmente distribuido, es escalable[16], tolera fallos y latencia.

La aplicación de estos modelos no son nuevos en las redes, por ejemplo Saraydar y colegas en [12] proponen un modelo basado en precios para el control de potencia en redes de datos inalámbricas.

Otro ejemplo en redes inalámbricas usando el protocolo "multiuser orthogonal frequency division multiplexing"(OFDM) es el realizado por Guo y colegas en[17] en el que describe un algoritmo de decisión para la asignación de sub-portadora de acuerdo con el valor máximo de la función de utilidad total, con respecto al tiempo medio de espera en la cola basado una función de utilidad, que es otra forma de ver el modelo oferta-demanda, en el cual cada participante de forma egoista busca maximizar su utilidad.

El objetivo de este tesis es optimizar el uso de la energía de cada nodo, proveyendo mecanismos para descubrir la ruta óptima para entregar la información al punto de destino. Esta ruta óptima puede ser tanto en cantidad de saltos como en el costo de cada nodo para enviar la información. Para lograr este objetivo se utilizó la teoría económica de oferta y demanda, la cual indica lo siguiente: *El precio de un bien refleja su escasez. Cuando la demanda excede a la oferta, los precios suben, y viceversa*, siendo en este caso la energía el bien a negociar.

Un nodo con una energía inicial X va a tener un precio Y por un tiempo T cuando la energía X caiga por debajo de un umbral U (un bien reflejando su escasez) el precio Y se incrementará a $Y+I$ siendo I un incremento.

Uno de los puntos a determinar es el momento apropiado para decidir que el costo por enviar información por este nodo debe subir debido al decremento de la energía del nodo.

Capítulo 1

Trabajo relacionado

Desde hace algunos años a la fecha, se realiza intensa investigación en la búsqueda de algoritmos de enrutamiento que permitan el ahorro de energía en redes de sensores, estos esfuerzos han seguido diversas estrategias, por ejemplo Heinzelman y colegas en [5, 6] proponen una familia de protocolos llamada SPIN que diseminan la información de cada nodo al resto del sistema. Ellos asumen que cada nodo es potencialmente un recolector de datos, también suponen que nodos cercanos poseen información similar, por lo tanto solo se transmite la información a aquellos nodos que no la poseen. Además solo se transmiten meta-datos, que solamente describen la información. Esta familia de protocolos se basan en dos ideas básicas:

- Los nodos sensores operan más eficientemente y conservan energía enviando solamente datos que describen la información que posee el nodo en vez de enviar todos los datos.
- Se consume energía y ancho de banda al enviar datos de sensores que cubren las misma áreas

El algoritmo "Minimum Total Transmission Power Routing"(MTPR) de K. Scott y N. Bambos [7], elige de entre varias rutas la que tenga el menor consumo total de potencia. Este algoritmo no toma en cuenta la batería por lo tanto puede elegir rutas que consuman menor potencia, pero que pueden agotar rápidamente la batería de algunos nodos. Por ejemplo, ver figura 4 de la sección anterior, en esa imagen, los números a un lado de las

flechas representan el gasto en potencia de transmisión. Tanto el nodo origen 1, como el nodo origen 2 han elegido la ruta que consume menor potencia total, pero que pasa a través del nodo $N6$, por lo tanto este nodo pronto agotará su batería. Algo que no es deseable de cara a alargar la vida del sistema.

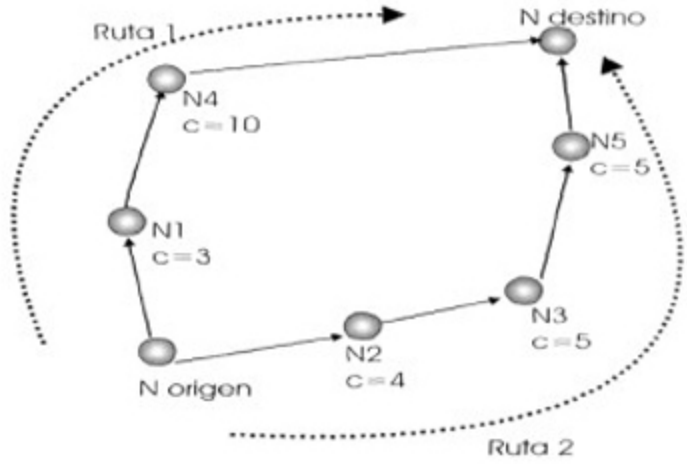

Figura 1.1: *Los círculos representan nodos, las flechas muestran las posibles rutas a seguir los datos desde los nodos origen al nodo destino. En la figura, existen dos posibles rutas entre el nodo fuente y destino. Aunque el nodo N4 tiene mucha menos batería restante que los demás nodos (su función de coste es sensiblemente superior), la ruta 1, la que incluye el nodo N4, presenta un menor coste total de batería. Por lo tanto la ruta 1 es la elegida, reduciendo el tiempo de vida del nodo N4, algo que como se ha venido comentando, no es deseable.*

Otra propuesta es la de Toh [8] en la que presenta el algoritmo "Minimum Battery Cost Routing"que toma en cuenta la batería residual de los nodos para evitar las rutas que contengan los nodos con menor batería. En este algoritmo se propone una función de coste que haga que un nodo sea más reacio, es decir, le cueste más encaminar, a medida que su capacidad residual de batería es más baja.

$$f_i(B_i^t) = \frac{1}{B_i^t} \tag{1.1}$$

Donde B el en energía en la bateria en un timpo t en un nodo i. Se calcula el coste de la batería C_l para la ruta l como la suma de los costes individuales de los nodos que la forma:

$$C_l = \sum_{i=0}^{D_l - 1} f_i(B_i^t) \tag{1.2}$$

Donde D es en numero de nodos que conforman la ruta l.

Sin embargo si en una ruta hay un nodo con poca batería y por lo tanto mayor costo, pero la suma total de costo es la menor. Esa ruta será la elegida(ver figura 1.1), y el sistema en vez de aumentar su vida, se verá reducida al fallar el nodo con poca batería residual. En un trabajo futuro se puede estudiar la función de costo. En los artículos presentados en esta tesis, la función es lineal. Por ejemplo en la ecuación 1.1 el costo es la inversa de la batería residual, pero se puede pensar en una función no-lineal, por ejemplo exponencial.

Tanto en [8] como en [9] se hace un análisis del desempeño para diferentes valores del umbral. El umbral es el porcentaje de batería residual en el que es necesario realizar un cambio de ruta. Sin embargo mi hipótesis es que el umbral deberá elegirse en función del diámetro de la red y del retardo asociado a él. Esto será parte del trabajo a futuro.

Jin Wang y colegas proponen en [10] un algoritmo que además de tomar en cuenta la energía también toma en cuenta el número adecuado de saltos para el ahorro de energía.

Capítulo 2

Algoritmo propuesto

En esta tesis se asume que existe un control de potencia[4], es decir que los nodos utilizan solamente la potencia mínima necesaria para transmitir. De esta manera nodos más alejados consumen más potencia que aquellos que están más cerca. En la figura 4 los números a un lado de las flechas indican la potencia necesaria para transmitir. De esta manera se puede observar que hay rutas que en un instante determinado consumen más potencia que otras.

La ruta se genera bajo demanda, cuando un nodo requiere conocer una ruta éste envía una solicitud de costo de ruta a sus nodos vecinos (se genera un broadcast de solicitud de costo) si el nodo que recibe la petición tiene el costo por enviar al nodo destino o es el nodo destino responde con un mensaje de respuesta de costo también de modo broadcast de esta manera otros nodos que no requieran el costo de la ruta en ese momento ya la conocerán y así ahorrarán energía al evitar una petición de ruta.

Con el paso del tiempo y el envió de mensajes la energía del nodo va decreciendo. Aquí es cuando aplicamos la teoría de la oferta y la demanda, si la energía cae por debajo del umbral se envía un mensaje broadcast de actualización de costo.

Cuando un nodo recibe un mensaje de actualización de costo éste actualiza el precio de la ruta en sus tablas de enrutamiento y verifica si esta sufrió cambio de costo, si sufre un aumento se envía un broadcast notificando el cambio del valor de la ruta y este proceso se repite. Para validar este protocolo de enrutamiento se implementó sobre el simulador ns-2[11]. Para el desarrollo de un protocolo sobre ns-2 se programa en dos lenguajes, en C++ en el cual se construyen los agentes y en Tool Command Language (TCL) se realiza

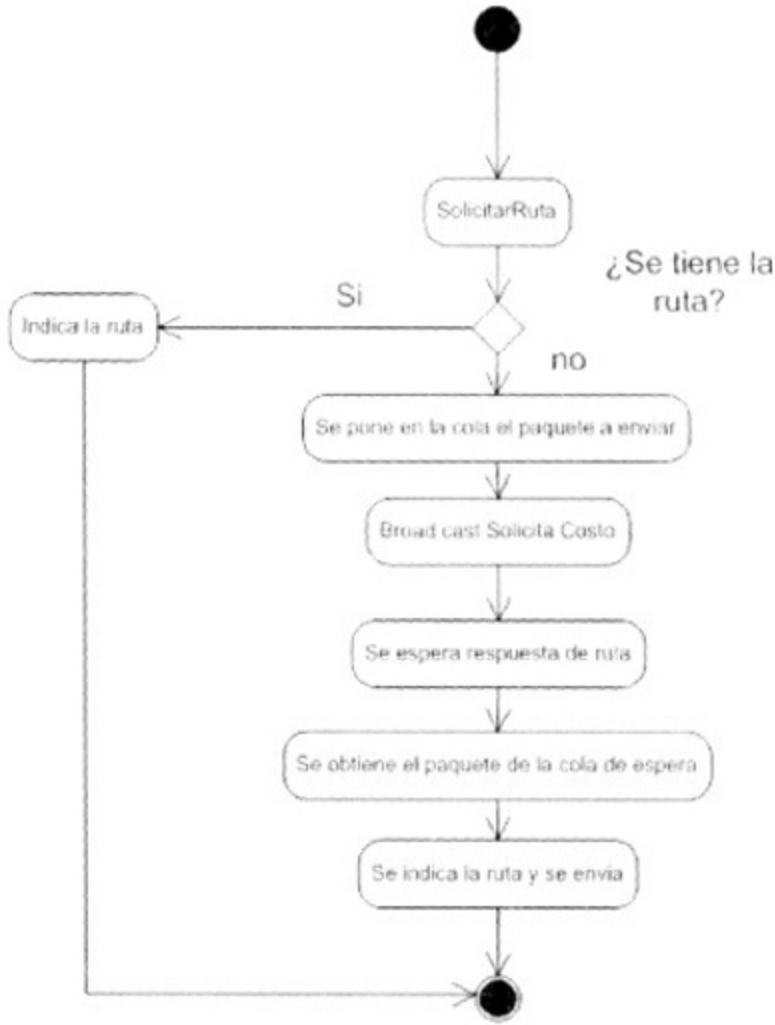

Figura 2.1: *Este diagrama ilustra el procedimiento de localización de una ruta*

la configuración de los escenarios y se lleva a cabo la planificación de la simulación.

Figura 2.2: *Este diagrama ilustra la solicitud de costo*

Figura 2.3: *Este diagrama ilustra el cambio de costo del nodo*

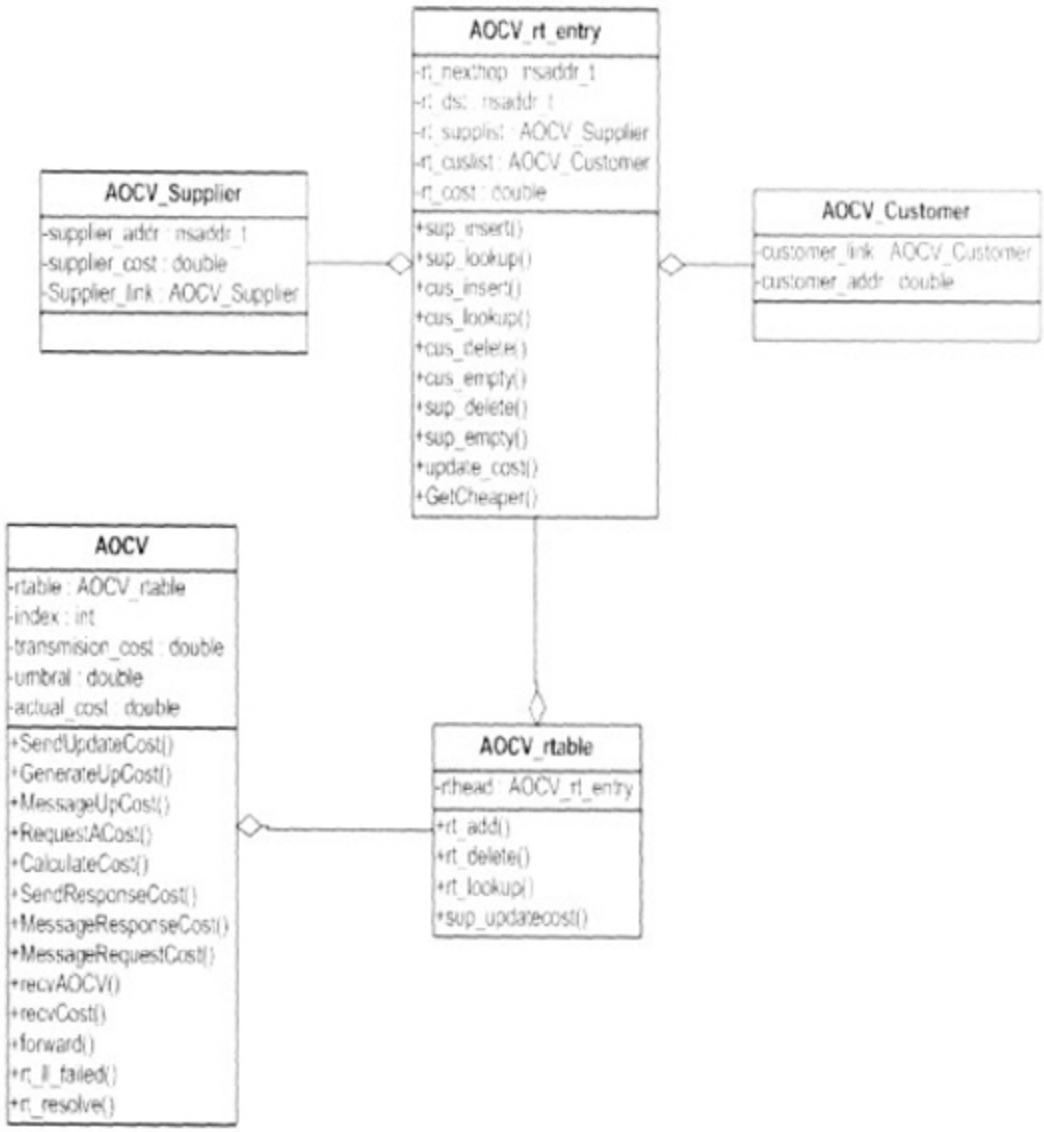

Figura 2.4: *Este diagrama ilustra las clases principales implementadas sobre C++ para ns-2*

Capítulo 3

Implementación en ns-2

A continuación se describe la configuración básica para realizar un escenario de simulación sobre TCL y las posibles configuraciones.

Primero procedemos a definir las opciones de configuración para la simulación:

1. *set val(chan) Channel/WirelessChannel*

   Aquí se establece que tipo de canal se va a utilizar, en este caso como trabajaremos con sensores inalámbricos usaremos la opción Channel/wirelessChannel, esta opción nos representa el medio físico de transmisión.

2. *set val(prop) Propagation/TwoRayGround*

   Esta opción sirve para definir el modelo de propagación. Ns2 cuenta con varios modelos de propagación entre los cuales se distinguen los siguientes:

   Free space model: El modelo de propagación en el espacio libre asume condiciones ideales de propagación. Solo considera el emisor y el receptor en línea de vista. La potencia de recepción es calculada por la ecuación de Friis y que está dada por: $P_r(d) = \frac{P_t G_t G_r \gamma^2}{(4\pi)^2 d^2 L}$ donde Pt es la potencia de la señal transmitida, Gt y Gr es la ganancia de la antena del transmisor y del receptor respectivamente. L(L=1) es la pérdida del sistema , γ es la longitud de onda y d es la distancia del transmisor al receptor.

Two-ray ground reflection model: En este modelo se considera además de la señal directa, la reflejada por el terreno. Este modelo da resultados más precisos a larga distancia que el de propagación en el espacio libre. La ecuación para calcular la potencia recibida es: $P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L}$ donde ht y hr son la altura de la antena del transmisor y receptor respectivamente.

Shadowing model: En los modelos anteriores predicen la potencia recibida con una función determinística en base a la distancia. En realidad la potencia recibida a cierta distancia es una variable aleatoria debido a los efectos de propagación. La ecuación utilizada es: $\frac{P_r(d_0)}{P_r(d)} = (\frac{d}{d_0})^{\beta}$ donde d es la distancia en linea lineal, d_0 es la distancia no lineal y β es exponente de pérdida y usualmente es determinado empíricamente.

3. *set val(netif) Phy/WirelessPhy*

   Esta opción sirve para definir el tipo de interfaz de red que será usada como una interfaz de hadware para acceder al canal de transmisión. Para simulaciones inalámbricas se utiliza WirelessPhy.

4. *set val(mac) Mac/802_11*

   Aquí se establece el protocolo de la capa MAC a utilizar, ns2 tiene 2 protocolos implementados para nodos inalámbricos.

   802_11 que es la implementación de IEEE 802.11

   Time division multiple access (TDMA), este protocolo separa una porción del tiempo de transmisión para cada nodo, durante el cual pueden enviar y recibir paquetes.

5. *set val(ifq) Queue/DropTail/PriQueue*

   Aquí se define una cola de prioridades para los paquetes.

   PriQueue, le da prioridad a los paquetes de enrutamiento insertándolos al inicio de la cola.

6. *set val(ll) LL*

   Aquí se estable la capa de enlace de datos, para el caso de nodos inalámbricos es link layer (LL), que tiene un modulo Protocolo de resolucion de direcciones (ARP).

7. *set val(ant) Antenna/OmniAntenna*

   La opción OmniAntenna (antena omni-direccional) es utilizada por los nodos móbiles

8. *set val(ifqlen) 50*

   Cantidad de paquetes a almacenar en la cola de prioridades

9. *set val(nn) 6*

   Se estableed la cantidad de nodos a simular, esta variable es para generar ambientes más flexibles, su uso no es parte de la configuración.

10. *set val(rp) AOCV*

    Protocolo de enrutamiento a utilizar. Aquí establecemos como protocolo a probar el Ad hoc On-Demand Cost Vector Routing (AOCV que es el protocolo que propone esta tesis), algunos otros Ad hoc On-Demand Distance Vector Routing (AODV) o Ad-hoc On-demand Multipath Distance Vector (AOMDV)

11. *set opt(x) 670*

12. *set opt(y) 670*

    Se establece el tamaño del ambiente a simular en coordenadas x, y, z, su valor por omisión es 0.

    Después se procede a la declaración de variables globales.

13. *set ns_ [new Simulator]*

    Se crea una instancia del simulador.

14. *set tracefd [open trace.tr w]*

    Variable para generar un archivo de rastreo, donde trace.tr es el nombre del archivo a generar.

15. *set namtrace [open archivo.nam w]*

    Variable para crear un archivo nam que servirá para visualizar la simulación de manera gráfica, donde archivo.nam es el nombre del archivo.

16. *set topo [new Topography]*

    Se crea la topología del ambiente a simular.

    Se establece la configuración.

17. *$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)*

    Este comando es usado para incializar el archivo nam para almacenar todos los movimientos de los nodos.

18. *$ns_ trace-all $tracefd*

    Comando para habilitar el rastreo de cada paquete que viaja a través de la red.

19. *$topo load_flatgrid $opt(x) $opt(y)*

    Se establece una topología plana con unas dimensiones x, y, un parámetro opcional es la resolución, por omisión es 1.

    Otra opción para cargar una topografía es a través de un archivo tipo DEM (formato de mapas de Garmin's), el cual se carga utilizando $topo load_demfile <archivo dem>

20. *create-god $val(nn)*

    Se crea un "dios" pasándole como parámetro el número de nodos, este creará una matriz donde se almacenara la información de la conectividad de los nodos.

21. *set chan_1_ [new $val(chan)]*

    Se crea el canal a utilizar.

22. *$ns_ node-config -adhocRouting $val(rp) \-llType $val(ll) \-macType $val(mac) \-ifqType $val(ifq) \-ifqLen $val(ifqlen) \-antType $val(ant) \-propType $val(prop) \-phyType $val(netif) \-channel $chan_1_ \-topoInstance $topo \-energyModel "EnergyModel"\-rxPower .5 \-txPower 1 \-agentTrace ON \-routerTrace ON \-macTrace OFF \-movementTrace OFF \-initialEnergy 3*

    Se establece la configuración de cada nodo.

    Se incluye la opción energyModel en el cual se indica que se simulara el consumo de energía.

    –rxPower indica la energía consumida por la recepción de un paquete.

    –txPower indica la energía consumida por el envío de un paquete.

    –initialEnergy indica la energía inicial del nodo.

    –routerTrace indica si se guardara la información de los paquetes de enrutamiento

    –macTrace indica si se guardará la informacion de los paquetes en la capa mac.

    movementTrace indica si se guardará la información de desplazamiento de los nodos móbiles.

23. *for {set i 0} {$i < $val(nn) } {incr i} { set node_($i) [$ns_ node] $node_($i) random-motion 0}*

24. *set node_($i) [$ns_ node]* Se procede a crear los nodos.

25. *$node_($i) random-motion 0* Se indica si serán nodos móbiles (1) o fijos(0), aqui utilizaremos nodos fijos.

26. *$ns_ at 0.0 "[$node_(0) agent 255] umbral 40"*

    Se procede a establecer el valor del umbral el cual indicará un cambio en el costo del nodo donde 0.0 es el momento en el tiempo en el cual se quiere cambiar el valor.

27. *$node_(0) set X_ 5.0 $node_(0) set Y_ 50.0 $node_(0) set Z_ 0*

    Se establece la posición del nodo dándole valores a x, y, z aquí hay que tener en cuenta que no podemos poner un valor más grande del que indicamos en la topografía.

28. *set tcp [new Agent/TCP]*

29. *$tcp set class_ 2*

30. *set sink [new Agent/TCPSink]*

31. *$ns_ attach-agent $node_(0) $tcp*

32. *$ns_ attach-agent $node_(5) $sink*

33. *$ns_ connect $tcp $sink*

34. *set ftp [new Application/FTP]*

35. *$ftp attach-agent $tcp*

36. *$ns_ at 0.2 "$ftp start"*

    Este código genera una conexión del nodo 0 al nodo 5 utilizando Transmission Control Protocol (TCP) como protocolo de transmisión y Transmission Control Protocol (FTP) para generar tráfico.

37. *$ns_ at 19.0 "stop"*

    Esta instrucción indica la duración de la simulación.

38. *$ns_ at 19.01 "puts \"NS EXITING...\"; $ns_ halt"*

    Esta instrucción indica que en un tiempo t ponga un mensaje y se ejecute la rutina de salida (proc stop).

39. *proc stop {} { global ns_ tracefd $ns_ flush-trace close $tracefd}*

    Este procedimiento indica al simulador que escriba en los archivos la información obtenida.

40. *$ns_ run* Inicio de la simulación.

Protocolo de enrutamiento

A continuación se describe el procedimiento para la integración de un protocolo dentro del código de ns2 usando C++.

Para realizar la integración de un nuevo protocolo sobre este simulador es necesario definir una constante (PT_AOCV) para identificarlo, esto en el archivo Packet.h localizado en el directorio common, debe asignársele el valor de la constante PT_NTYPE y a esta constante sumarle 1 a valor. En este caso la constante PT_AOCV quedó con el valor de 62 y PT_NTYPE con el valor de 63.

El siguiente paso es indicar que el tipo de paquete PT_AOCV pertenecerá a paquetes de enrutamiento, para esto es necesario modificar el método classify de la clase p_info ubicada en el archivo Packet.h, la modificación necesaria es agregar la siguiente comparación *type ==PT_AOCV*; en la misma clase hay que inicializar el nombre del protocolo, esto es necesario para identificar el tipo de paquete en el archivo de trace, para lograr esto hay que agregar la siguiente línea *name_[PT_AOCV]= "AOCV";* en el método *initName*.

Después se procede a agregar el tipo de paquete a las colas de prioridad del ns2; se modifica el método *prq_assign_queue* perteneciente a la clase *CMUPriQueue* hubicada en el archivo dsr-priqueue.cc que se encentra en el directorio queue; agregando la línea *case PT_AOCV: return IFQ_RTPROTO* . Dentro del mismo directorio se encuentra el archivo priqueue.cc el cual contiene la clase *PriQueue* en donde se modificó el método recv agregando *case PT_AOCV: recvHighPriority(p, h);break*.

La clase principal del protocolo AOCV es AOCV ubicada en el archivo aocv.cc dentro de la carpeta aocv.

Dentro de esta clase se encuentran varios métodos para la creación de las tablas de enrutamiento y manejo de paquetes (tanto reenvío como creación de rutas). A continuación se explican los principales métodos involucrados en el funcionamiento de este protocolo:

1. *recv(Packet *p, Handler*)*

   a) Este método se encarga de recibir todos lo paquetes que llegan al nodo.

   b) Clasificar el tipo de paquete, es decir identificar si es un paquete de enrutamiento recibido, un paquete para reenviar, o un paquete que se quiere enviar y hay que buscar la ruta.

2. *forward(aocv_rt_entry *rt, Packet *p, double delay)*

   a) Este método está a cargo del reenvío de paquetes, así como asignar la prioridad para el envío.

3. *rt_resolve(Packet *p)*

   a) Este método está a cargo de indicar la ruta a seguir, de a cuerdo a la tabla de enrutamiento.

   b) En caso de no contar con la información de enrutamiento en la tabla, aquí se origina la solicitud de costo de la ruta.

4. *RequestACost(nsaddr_t src,nsaddr_t dst,nsaddr_t req)*

   a) Este método se encarga de armar y enviar el paquete de solicitud de costo.

5. *MessageRequestCost(Packet *p)*

   a) Aquí se manejan las peticiones de costo.

   b) En caso de contar con el costo de la ruta, aquí se origina la respuesta de la petición de costo.

   c) De lo contrario se origina una petición de costo.

6. *SendResponseCost(double cost,nsaddr_t dst,nsaddr_t req, nsaddr_t next)*

   a) Se encarga de armar el paquete de respuesta de costo y enviarlo.

7. MessageResponseCost(Packet *p)

   a) Se encarga de recibir los mensajes de respuesta de costo.

   b) Agrega la ruta en la tabla.

   c) En caso de tener peticiones solicitando el costo de esa ruta, envía el costo de la ruta.

8. SendUpdateCost(double newcost,nsaddr_t dst)

a) Aquí se arman los paquetes de actualización de costo.

9. MessageUpCost(Packet *p)

   a) Aquí se reciven los mensajes de actualización de costo.

   b) Si el costo de la ruta se incrementa se envía un paquete de actualización de costo.

10. CalculateCost()

    a) Aquí se calcula el costo de envío.

11. command (int argc, const char*const* argv)

    a) En este método traducen los comando enviados por TCL a acciones dentro del protocolo.

    b) Siendo el parámetro argc el número de argumentos que están en argv.

Se manejan 3 timers los cuales son:

1. RouteTimer

   a) Debido a que podemos recibir múltiples rutas para llegar a un destino, cuando se recibe una respuesta del costo para el destino se activa un timer para la espera de posibles respuestas.

   b) Una vez pasado 200 ms se procede a notificar el costo para la ruta.

   c) Para seleccionar el tiempo de espera se consideró lo siguiente:

      1) Cuando se crea por primera vez la ruta, teniendo en cuenta que las tablas están vacías, entre más tarde el nodo en responder hay mas de un 90 % de probabilidad que el costo de la ruta por ese nodo sea alto, esto es debido a que en un tiempo 0 todos los nodos tienen el mismo costo, entre más nodos recorra o más lejano este el siguiente nodo se generará una latencia mayor para conocer la ruta y por ende será mas costosa la ruta. Para peticiones posteriores una respuesta rápida significaría que la ruta ya estaba en las

tablas del nodo vecino con lo cual no se nos garantiza el costo mas óptimo al elegir la primera respuesta.

2) Si ya hay datos en la tabla la respuesta es inmediata por los nodos vecinos.

2. UmbralTimer

   a) Este se encarga de monitorear el nivel de energía.

   b) Cuando la energía caiga por debajo de el umbral se enviará una notificación de aumento de costo.

3. PacketTimer

   a) Se encarga de enviar los paquetes de datos que están pendientes por una ruta a seguir.

La tabla de enrutamiento(*aocv_rtable*) está formada de la siguiente manera:

1. Se compone por una lista de entradas.

2. Cada entrada a la vez esta compuesta por:

   a) Un destino.

   b) Un siguiente salto.

   c) Una lista de Proveedores.

      1) Cada proveedor esta formado por una dirección y un costo.

   d) Una lista de clientes.

   e) Un costo (siempre es el costo más bajo de la lista de proveedores).

Una vez terminado el desarrollo del protocolo se crea la referencia a este código por medio del lenguaje TCL. Para esto hay que modificar el ns_tcl.cc ubicado en el folder gen agregando las siguientes líneas

set aocvonly [string first \"AOCV\"[$agent info class]] \n\if {$aocvonly != -1 } {\n\$agent if-queue [$self set ifq_(0)] ;# ifq between LL and MAC\n\}\n\\n\AOCV

{\n\set ragent [$self create-aocv-agent $node]\n\}\n\\n\Simulator instproc create-aocv-agent { node } {\n\set ragent [new Agent/AOCV [$node node-addr]]\n
$self at 0.0 \"$ragent start\";# start BEACON/HELLO Messages\n\$node set ragent_ $ragent\n
return $ragent\n\}\n, al agregarlas y compilar creará un archivo con código TCL el cual es el corazón del ns-2.

Por último se procede a agregar las referencias a los nuevos archivos en el archivo FILES para incorporarlos en la compilación del simulador, hay que agregar las siguientes líneas:

1. *aocv/aocv.cc*
2. *aocv/aocv.h*
3. *aocv/aocv_logs.cc*
4. *aocv/aocv_packet.h*
5. *aocv/aocv_rqueue.cc*
6. *aocv/aocv_rqueue.h*
7. *aocv/aocv_rtable.cc*
8. *aocv/aocv_rtable.h*

Después se procede a modificar los archivos make, agregando a la variable *OBJ_CC* las siguientes líneas

1. *aocv/aocv_logs.o aocv/aocv.o* \
2. *aocv/aocv_rtable.o aocv/aocv_rqueue.o* \

Una vez terminado esto se procede a ejecutar la compilación a través del comando make, con lo cual se generará la aplicación ns2 .

Capítulo 4

Análisis de desempeño

Para validar el resultado del protocolo se hizo comparación contra AODV, se tomaron en cuenta 2 criterios:

- Cantidad de paquetes enviados.
- Tiempo de duración de la simulación.

A continuación se presenta el código TCL utilizado para esta comparación, se utilizó la misma topología, consumo de energía y energía inicial de los nodos para cada prueba.

Todas las simulaciones fueron realizadas en una máquina virtual con las siguientes caractersticas:

- Sun VirtualBox 3.1.6 como software de virtualización.
- S.O.: Ubuntu 9.10.
- Procesador: Core 2 duo usando un solo núcleo.
- RAM: 1 GB.
- HD: 16 GB dinámico.

 # escenario3_6n.tcl
 # A simple example for wireless simulation
 # Define options
set val(chan) Channel/WirelessChannel ;# channel type

```
set val(prop) Propagation/TwoRayGround ;# radio-propagation model
set val(netif) Phy/WirelessPhy ;# network interface type
set val(mac) Mac/802_11 ;# MAC type
set val(ifq) Queue/DropTail/PriQueue ;# interface queue type
set val(ll) LL ;# link layer type
set val(ant) Antenna/OmniAntenna ;# antenna model
set val(ifqlen) 80 ;# max packet in ifq
set val(nn) 6 ;# number of mobilenodes
#set val(rp) AODV ;# routing protocol
set val(rp) AOCV ;# routing protocol
set opt(x) 670 ;# x coordinate of topology
    set opt(y) 670 ;# y coordinate of topology
    set val(umb) 80
# Main Program
#
================================================================:
#
# Initialize Global Variables
#
set ns_ [new Simulator]
set tracefd [open escenario3_6n-$val(rp)-$val(umb).tr w]
set namtrace [open escenario3_6n-$val(rp)-$val(umb).nam w]
#$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

$ns_ trace-all $tracefd

# set up topography object
set topo [new Topography]

$topo load_flatgrid 500 500
```

```
#
# Create God
#
create-god $val(nn)
set chan_1_ [new $val(chan)]

# configure node

 $ns_ node-config -adhocRouting $val(rp) \
      -llType $val(ll) \
      -macType $val(mac) \
      -ifqType $val(ifq) \
      -ifqLen $val(ifqlen) \
      -antType $val(ant) \
      -propType $val(prop) \
      -phyType $val(netif) \
      -channel $chan_1_ \
      -topoInstance $topo \
      -energyModel ."ergyModel"\
      -rxPower 5 \
      -txPower 10 \
      -agentTrace ON \
      -routerTrace ON \
      -macTrace OFF \
      -movementTrace OFF \
      -initialEnergy 30 \
      -sleepPower 0.001 \
      -transitionPower .02 \
      -transitionTime 0.005 \

      for {set i 0} {$i < $val(nn) } {incr i} {
      set node_($i) [$ns_ node]
```

```
    $node_($i) random-motion 0 ;# disable random motion

    }
$ns_ at 0.0 "[$node_(0) agent 255] umbral $val(umb)"
$ns_ at 0.0 "[$node_(1) agent 255] umbral $val(umb)"
$ns_ at 0.0 "[$node_(2) agent 255] umbral $val(umb)"
$ns_ at 0.0 "[$node_(3) agent 255] umbral $val(umb)"
$ns_ at 0.0 "[$node_(4) agent 255] umbral $val(umb)"
$ns_ at 0.0 "[$node_(5) agent 255] umbral $val(umb)"



#
# Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes
#
$node_(0) set X_ 5.0
$node_(0) set Y_ 50.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 180.0
$node_(1) set Y_ 2.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 340.0
$node_(2) set Y_ 70.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 100.0
$node_(3) set Y_ 200.0
$node_(3) set Z_ 0.0
```

```
#
for {set i 0} {$i < $val(nn) } {incr i} {
 $ns_ at 30.0 "$node_($i) reset";
}
$ns_ at 29.0 "stop"
$ns_ at 29.01 "puts \"NS EXITING...\"; $ns_ halt"

proc stop {} {
 global ns_ tracefd
 $ns_ flush-trace
 close $tracefd
}

puts "Starting Simulation..."
$ns_ run
```

Este ejemplo utiliza una topología formada por 6 nodos ver figura 4.1

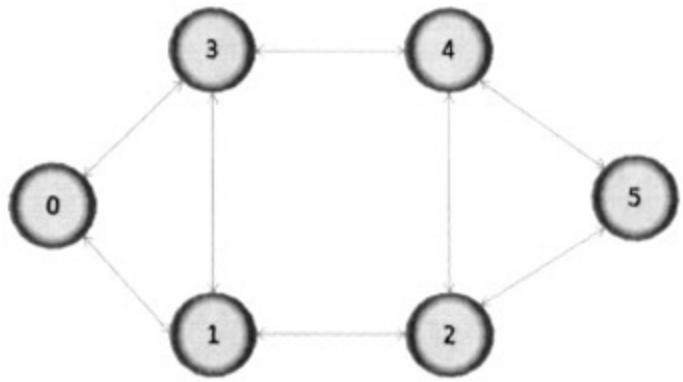

Figura 4.1: *El nodo 0 tiene comunicación con los nodos 1 y 3,el nodo 1 tiene comunicación con los nodos 0,2,3,el nodo 2 tiene comunicación con los nodos 1,4,5,el nodo 3 tiene comunicación con los nodos 0,1,4,el nodo 4 tiene comunicación con los nodos 3,2,5,el nodo 5 tiene comunicación con los nodos 4,2*

Cabe aclara que ns-2 termina las simulaciones cuando ya no hay mas envió de paquetes, en este caso se estableció un tiempo de simulación mayor al que el sistema podía soportar (30 segundos).

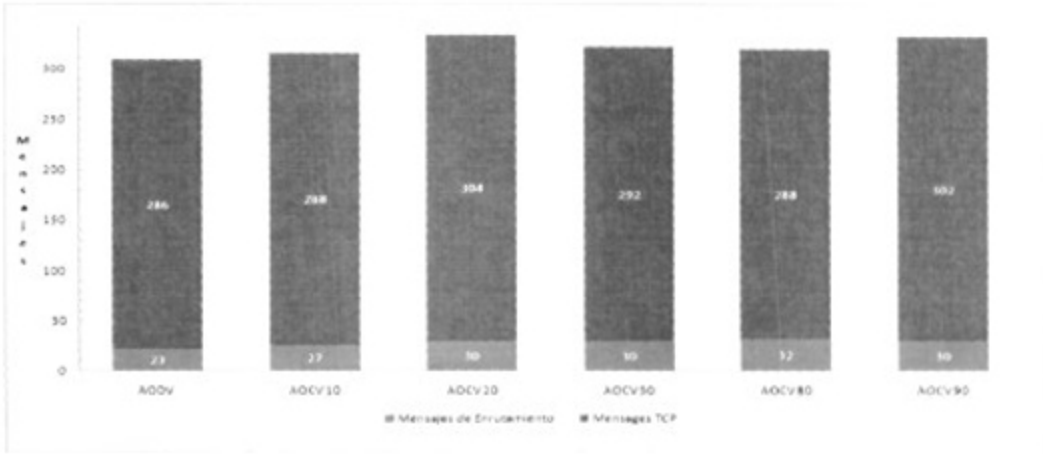

Figura 4.2: *Comparación del total de mensajes enviados durante la simulación, entre AODV y AOCV con distintas configuraciones*

En la figura 4.2 se muestra la cantidad de paquetes sobre TCP enviados y cuantos mensajes de enrutamiento, debajo de la leyenda AOCV se encuentra el % del umbral.

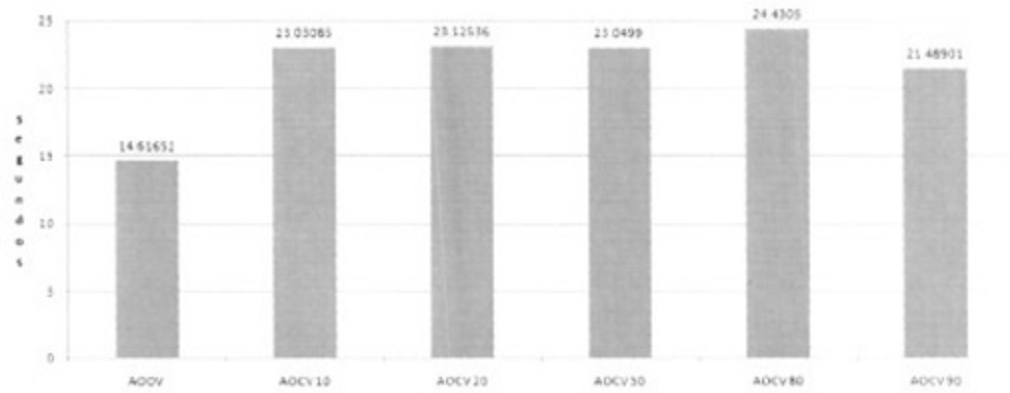

Figura 4.3: *Comparación del tiempo transcurrido antes de agotar la energia de los nodos, entre AODV y AOCV con distintas configuraciones*

En la gráfica anterior figura 4.3 comprobamos que el tiempo de simulación se ha incrementado.

En conclusión podemos comprobar que AOCV utiliza de manera eficaz los nodos vecinos (ver figura 4.4 y 4.4) para el envío de información, alargando la vida de la red y por ende enviando más paquetes de información.

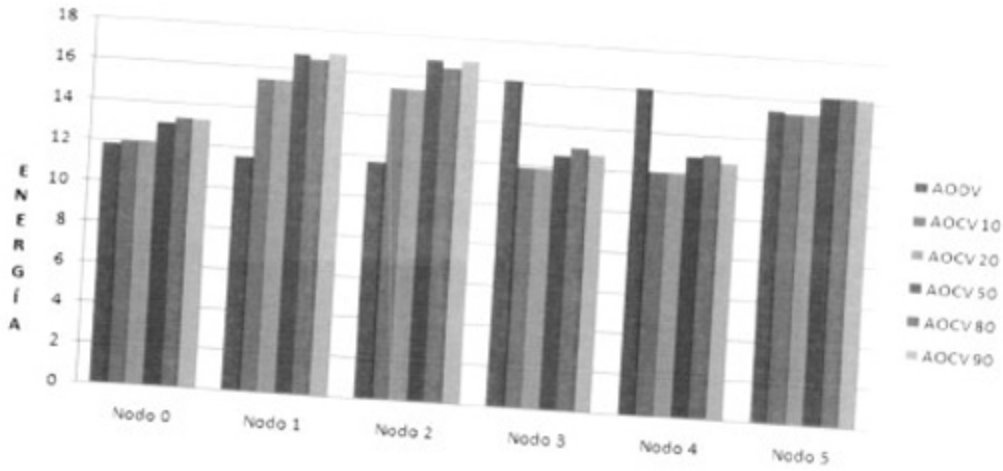

Figura 4.4: *Comparación de la energía restante en cada nodo en un tiempo aproximado de 3 segundos, entre AODV y AOCV con distintas configuraciones*

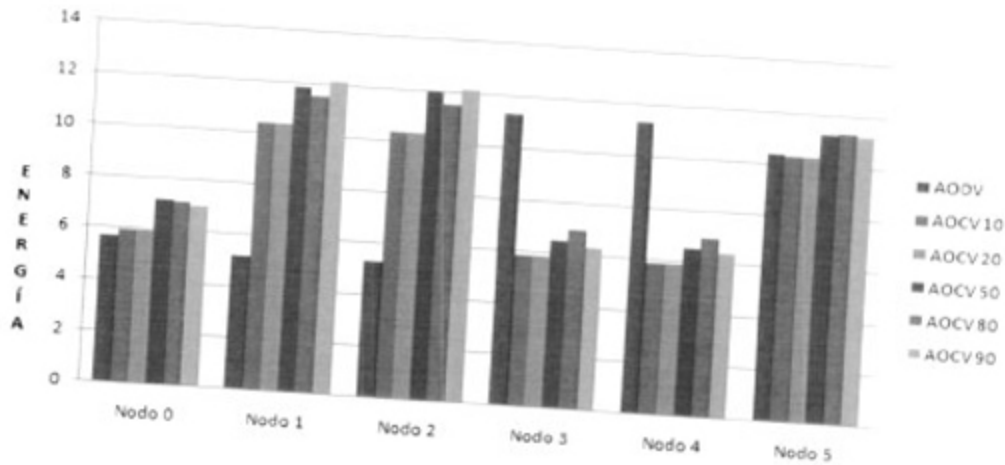

Figura 4.5: *Comparación de la energía restante en cada nodo en un tiempo aproximado de 4 segundos, entre AODV y AOCV con distintas configuraciones*

Capítulo 5

Conclusiones y trabajo a futuro

En conclusión vemos que se ha logrado el objetivo de esta tesis, presentar un nuevo protocolo de enrutamiento para redes de sensores, distribuido, que no necesita conocimiento global del sistema y que incrementa el tiempo de duración de la batería de los nodos y por ende se incrementa el número de mensajes enviados en la red. La comparación se realizó contra AODV debido a que es un protocolo muy validado y que está codificado en ns-2.

Como se ha mencionado en algunas partes de éste documento, para trabajo a futuro de este algoritmo, se tiene planeado buscar una función en la asignación del costo que permita que el sistema reaccione de forma más eficiente en el momento en que un porcentaje determinado de la batería llegue a un umbral.

La función de costo es una función escalón, no es una función contínua. Entonces el problema es determinar el tamaño de ese escalón. Por ejemplo que el costo cambie cada vez que la batería ha bajado un décimo ó un céntimo.

Otro problema relacionado con lo mismo es que esta función escalón se ha definido como lineal, pero también podría ser no-lineal, de esta manera podría el costo aumentar de forma exponencial por ejemplo para evitar que nodos con batería baja reciban mucho tráfico.

Otra cuestión que se tiene planeada investigar es la relación entre el retardo del sistema y el umbral. Algunos investigadores han estudiado el umbral, por ejemplo Dongkyun Kim y colegas en[9] estudiaron la respuesta del sistema a umbrales del 25 %, 50 % y 75 % de batería residual, y encontraron que el mejor desempeño se obtiene al 50 %. Sin

embargo, esto es válido para esa topología específica, con ésa cantidad de nodos y con un determinado retardo, etc. La pregunta que surge es: ¿Ese resultado es válido para otra topología muy distinta? Mi hipótesis es que *no*, dado el resultado obtenido en esta tesis el valor optimo del umbral es un 20 % y no un 50 % como en los estudios de Dongkyun. En el trabajo futuro se tiene planeado investigar la relación entre algunas propiedades topológicas de las redes y el umbral

Referencia

[1] Toh,C-K. Ad hoc Mobile Wireless Networks, Procols and systems. Prentice-Hall (2002).

[2] Zou Y.,Chakrabarty K., Uncertainty-Aware and Coverage-Oriented Deployment for Sensor Networks, Journal Parallel and Distributed Computing, vol. 64, pp. 788-798, Jan. 2004.

[3] Charles E. Perkins and Elizabeth M. Royer. Ad hoc On-Demand Distance Vector Routing. Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90-100.

[4] Vikas Kawadia and P.R. Kumar.Principles and Protocols for Power Control in Wireless Ad Hoc Networks. IEEE Journal On Selected Areas In Communications: Special Issues On Wireless Ad Hoc Networks vol. 1, 2005

[5] W. Heinzelman, J. Kulik, and H. Balakrishnan. AdaptiveProtocols for Information Dissemination in WirelessSensor Networks. Proc. 5th ACM IEEE Mobicom, Seattle, WA, Aug. 1999

[6] J. Kulik, W. R. Heinzelman, and H. Balakrishnan. Negotiation-Based Protocols for Disseminating Infor-mation in Wireless Sensor Networks.Wireless Networks, vol. 8, 2002

[7] Scott, K., Bambos, N. Routing and channel assignment for low power transmission in PCS. Proceedings of IEEE International Conference on Universal Personal Communications (ICUPC 96), Cambridge.(1996).

[8] Toh, C-K. Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks. IEEE Communications magazine(2001)

[9] Dongkyun Kim, J.J Garcia-Luna-Aceves,Katia Obraczka,Juan-Carlos Cano,and Pietro Manzoni. CMDR:Conditional Minimum Drain Rate Protocol for Route Selection in Mobile Ad-Hoc Networks. ICOIN 2003

[10] Jin Wang, Jinsung Cho, Sungyoung Lee, Kwang-Cheng Chen and Young-Koo Lee. Hop-Based Energy Aware Routing Algoritm for Wireless Sensor Networks. IEICE Trans. Comm. Vol. E93. Feb 2010

[11] http://www.isi.edu/nsnam/ns/

[12] Cem U. Saraydar, Narayan B. Mandayam, David J. Goodman. Efficient power control via pricing in wireless data networks. IEEE Transactions on Communication, Vol 50, no 2, 2002.

[13] S. Hedetniemi and A. Liestman, A survey of gossiping and broadcasting in communication networks, Networks, Vol. 18, No. 4, 1988.

[14] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in the Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00), Boston, MA, August 2000.

[15] Wendi Rabiner Heinzelman, Anantha Chandrakasan, Hari Balakrishnan.Energy-Efficient Communication Protocols for Wireless Microsensor Networks .Proc. Hawaaian Int'l Conf. on Systems Science, January 2000.

[16] Neuman C, "Scale in Distributed Systems. In Readings in Dist.Comp. Syst.", IEEE Computer Society Press, 1994

[17] Guo Kunqia, Sun Lixin and Ji Shilou. Utility function based fair data scheduling algorithm for OFDM wireless network. Journal of Systems Engineering and Electronics Volume 18, Issue 4, December 2007.