EL GRAFO VIRTUAL PARA RUTEO GEOMÉTRICO EN UNA RED INALÁMBRICA AD-HOC

TESIS

Que para obtener el grado de

DOCTOR EN INGENIERÍA ELÉCTRICA

presenta

Héctor Tejeda Villela

Edgar Leonel Chávez González

Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Junio 2005

Deseo agradecer a mi asesor el Dr. Edgar Leonel Chávez González por su ayuda para realizar el presente trabajo. También deseo agradecer a mi familia por creer en mí, de igual forma a mis padres que me alentaron a seguirme superando, y a todos aquellos que se han mostrado interesados en este sueño. Sin lugar a dudas que los integrantes de la mesa de jurado, Dres. Félix Calderón Solorio, Juan José Flores Romero, Leonardo Romero Muñoz y Pedro Ruiz, ayudaron a que mi trabajo de tesis fuera mejor, deseo por igual manifestarles mi agradecimiento y amistad.

Esta investigación fue apoyada por el Consejo Nacional de Ciencia y Tecnología, CONACYT, como parte del proyecto "Algoritmos de Búsqueda en Espacios Métricos" No. 36911-A.

Resumen

Una red inalámbrica ad-hoc es un conjunto de dispositivos móviles equipados con un transmisor y un receptor, los cuales se autoorganizan para conectarse cuando no existe una infraestructura que los coordine. En esta tesis se aborda el problema de envío de datos en una red inalámbrica "ad-hoc" basado en la posición del transmisor. Los algoritmos empleados para este tipo de redes requieren que los nodos puedan determinar su ubicación geográfica, además algunos de los algoritmos requieren extraer un grafo plano, el cual represente la red física subyacente.

Las dos contribuciones importantes en este trabajo de tesis son: el *Dual del Grafo Plano* y el *Grafo Virtual*. El dual del grafo plano es una estructura para encaminar paquetes usando el algoritmo voraz, y donde siempre se garantiza la entrega de paquetes. Actualmente existen algunas topologías en las cuales el algoritmo voraz falla. Por su parte, el grafo virtual es una contribución importante para aquellos algoritmos que requieren obtener un grafo plano para encaminar, ya que comparado con el *grafo de Gabriel* y el *grafo de Morelia* se logran mejores resultados. Para comparar los resultados con otros grafos se hizo mediante simulación, para lo cual se creó un simulador con distintas opciones, que lo hacen sencillo y fácil de usar.

Abstract

A wireless ad-hoc network is a set of devices equipped with a transmitter and a receiver. Wireless hosts can communicate with each other in the absence of a fixed infrastructure. These networks typically consist of equal nodes that communicate over wireless links without central control. In this thesis the problem of routing data in a wireless ad-hoc network is approached by location aware. The algorithms used by these networks require that the nodes can determine their geographic location, in addition some algorithms require to extract a planar graph, which represents the underlying physical network.

The two important contributions in this thesis work are: the *Dual of a Plannar Graph* and the *Virtual Graph*. The dual of a planar graph is a structure for routing packages using the greedy algorithm, where always the delivery of packages is guaranteed. At this moment, with some topologies, greedy algortithm fails for sending packages. On the other hand, the virtual graph is an important contribution for those algorithms that require to obtain a planar graph for routing, when the virtuai graph is compared with respect to the *Gabriel Graph* and the *Morelia Graph* we obtain better results. A simulation study was developed using a proper simulator, it is simple and easy to use.

Índice general

Índice de figuras

Índice de cuadros

Lista de Publicaciones

En el capítulo 6 se discute con mas detalle la presentación del grafo virtual hecha en el taller.

> E. Chávez and H. Tejeda. "Geometric Routing Using a Virtual Graph". 3rd Workshop RAM *(Routing At Morelia)*,August 2004.

El capítulo 7 contiene a detalle las distintas opciones y ventajas del simulador que fue presentado en el taller.

> E. Chávez and H. Tejeda. "A Geographic Routing Workbench and a Tool for Planar Graphs, Spanning Trees and Geometric Routing". 3rd Workshop RAM *(Routing At Morelia)*, August 2004.

Capítulo 1

Introducción

Las redes inalámbricas ad-hoc consisten de varios dispositivos inalámbricos que se pueden comunicar entre ellos en ausencia de una infraestructura fija, como pudieran ser estaciones base en redes de telefonía, puntos de acceso en WLANs, etc. Este tipo de redes pueden surgir de forma espontánea y por sus características el medio inalámbrico es el que se usa en forma natural para comunicarse. Los dispositivos son usados en zonas de desastre, ambientes de batalla y conferencia, por lo cual han recibido bastante atención en los últimos años [IET, Macker98]. Se tiene la necesidad de diseñar redes que tengan un rendimiento eficiente y escalable, gracias a la disponibilidad reciente de receptores GPS de tamaño pequeño, de costo económico y con un consumo mínimo de energía. Por otra parte, no se requieren de un sistema de posicionamiento para determinar las coordenadas de un transmisor, ya que hay técnicas para encontrar las coordenadas relativas basadas en la potencia de la la señal. Algoritmos para encaminar en este tipo de redes fueron desarrollados en los últimos años, adicionalmente a unos cuantos métodos básicos propuestos hace aproximadamente unos quince años. Lo anterior, se prevé traerá consigo una proliferación de tecnologías inalámbricas en la próxima generación, las cuales tendrán la capacidad de proveer servicios *basados en la posición*, con lo que se conocerá la posición actual del usuario. Por otra parte, los vendedores de dispositivos inalámbricos han ofrecido incluir múltiples interfaces inalámbricas en los dispositivos en la próxima generación. Las interfaces que serán incluidas son la interfaz estándar celular y las interfaces estándar de redes inalámbricas de área local (*wireless local area network* WLAN). Los futuros teléfonos inalámbricos tendrán la capacidad de moverse libremente entre las redes celulares tradicionales y las redes WLAN que podrían estar presentes en centros de trabajos, centros comerciales o campus univer-

sitarios. Con lo anterior los proveedores inalámbricos empezarán a introducir los servicios *basados en la posición*, con lo cual se dispondrán un conjunto de aplicaciones para redes inalámbricas. Los servicios *basados en la posición* podrán ser tales como: mapas, servicios de directorios en edificios, anuncios dirigidos, entre otros. Este tipo de servicios estarán invadiéndonos todos los días de nuestra vida en un futuro muy cercano.

En esta tesis se estudia el problema de encaminamiento en redes inalámbricas ad-hoc con un enfoque en los algoritmos de encaminamiento *basados en la posición*. Los algoritmos basados en la posición aprovechan la ventaja del hecho de que todos los nodos en la red tienen conocimiento de su posición geográfica, por lo que esta información se puede usar para tomar decisiones de encaminamiento.

También en la tesis se explora la manera como puede lograrse que el encaminamiento voraz garantice la entrega de paquetes, ya que existen algunas configuraciones de redes que vencen al encaminamiento voraz. Para lograr lo anterior, se propone una nueva estructura el *Dual del Grafo Plano*.

1.1. Descripción del problema

El envío de información o encaminamiento en un sistema de comunicación inalámbrica es un problema importante que ha sido ampliamente estudiado. Usualmente, los dispositivos en una red tienen poco conocimiento acerca del ambiente que los rodea. Los algoritmos tradicionales, que serán discutidos en el capítulo 2, intentan dar a los nodos conocimiento global construyendo tablas de encaminamiento o de rutas usadas en el envío de paquetes de un nodo fuente a un nodo destino. Sin embargo, si los nodos conocen su posición, las posiciones de sus vecinos y la posición del destino, los algoritmos basados en la posición no necesitan construir las tablas anteriores o dar rutas para enviar datos, ya que pueden tomar decisiones de encaminamiento localmente.

En el presente trabajo se estudiará el encaminamiento en redes inalámbricas ad-hoc, conocidas también como redes móviles Ad-hoc o MANETs *(Mobile Ad-hoc Networks)*, las cuales consisten de transmisores inalámbricos que se comunican entre ellos en ausencia de una infraestructura fija. Se verán algunos algoritmos de encaminamiento tradicional desarrollados para redes ad-hoc; pero nos enfocaremos en los algoritmos de encaminamiento *basados en la posición* dando varios ejemplos.

El principal objetivo para algunos de los protocolos de encaminamiento basados en

la posición es obtener un *grafo plano*, ya que un algoritmo voraz no es capaz de garantizar la entrega de un paquete, debido a que el paquete puede llegar a un callejón, por lo tanto debe existir algún mecanismo para que se recupere el algoritmo; el algoritmo de *encaminamiento por caras* [Kranakis99] es el primer algoritmo basado en la posición que garantiza la entrega, el cual requiere un grafo plano. El *Grafo Virtual* que es propuesto en el presente trabajo construye un grafo plano a partir de la red inalámbrica subyacente, para que pueda ser empleado por el algoritmo de *encaminamiento por caras* como mecanismo de recuperación cuando así se requiera.

1.2. Modelo General y Notación

Un *grafo* $G = (V, A)$ está definido por un conjunto de n vértices V y un conjunto de m aristas A. Una arista $a \in A$ está formada por un par de vértices. Un grafo *plano* G es aquel donde ningún par de aristas se intersectan en ningún punto, excepto en algún vértice $v \in V$. Las aristas de G dividen el grafo en un conjunto de polígonos, llamadas *caras*. Los *vecinos* $N(v)$ de un vértice $v \in V$ es el conjunto de vértices adyacentes al vértice v. Una red es *conexa* si para cada par de vértices $u, v \in V$, existe una ruta de u a v recorriendo las aristas adyacentes.

1.3. Motivación

La motivación principal que se tiene para investigar el encaminamiento basado en la posición, es con la finalidad de lograr que el encaminamiento ad-hoc sea mas eficiente en cuanto a la sobrecarga de control. Las redes inalámbricas ad-hoc o redes móviles ad-hoc (MANET) están compuestas por un conjunto de nodos autónomos, los cuales actúan como encaminadores, o bien como anfitriones, colaborando en conjunto para formar una red productiva.

Los nodos forman una red "bajo demanda", sin ninguna infraestructura fija, usando enlaces inalámbricos y moviéndose de forma aleatoria. En la figura 1.1 se muestra una red inalámbrica ad-hoc y el movimiento del transmisor 4. Los nodos pueden estarse conectando y desconectando de la red en cualquier momento. Con lo anterior, resulta que se tiene siempre una topología que está cambiando, la cual es difícil de predecir.

Dos nodos pueden comunicarse en forma bidireccional si y solamente si la distan-

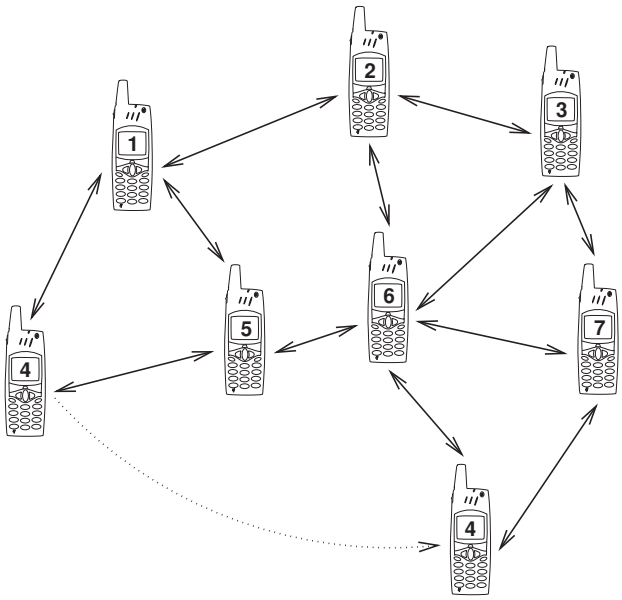


Figura 1.1: En la red inalámbrica Ad-hoc el nodo móvil 4 se mueve, por lo tanto cambia la topología de la red.

cia entre ellos es menor que el mínimo de sus rangos de transmisión. [1] Si un nodo desea comunicarse con otro nodo el cual no se encuentre dentro de su rango de transmisión, podría hacerlo enviando mensajes a través de otros nodos de la red realizando varios saltos, también conocido como modo "multisalto" . En la figura 1.2 se muestra como el nodo 1 se comunica con el nodo 3 enviando los paquetes de datos a través del nodo 2.

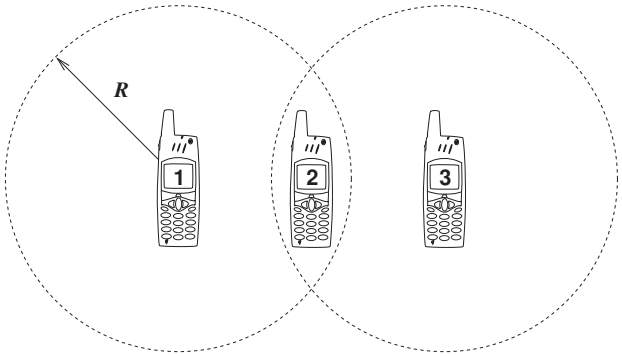


Figura 1.2: Como los nodos tienen rangos de transmisión limitados se requiere el encaminamiento multisalto para pasar paquetes entre algún par de nodos.

[1] En comunicaciones, se entiende que el rango de transmisión de un dispositivo, es la máxima distancia del dispositivo hacia otro donde la conectividad existe.

El *grafo del disco unitario* (GDU) es un grafo G donde una arista (u, v) existe si la distancia $d(u, v) \leq 1$. El GDU es típicamente usado para modelar una red ad-hoc donde todos los nodos en la red tienen el mismo rango de transmisión. Los nodos pueden ser representados como V, el conjunto de vértices, y hay una arista $a \in A$, entre dos nodos u y v si ellos están dentro del rango de transmisión del otro. La mayoría de las investigaciones de redes ad-hoc se basan en que los nodos tienen el mismo rango de transmisión. Sin embargo, hay un artículo interesante [Barriére01] donde los autores abordan el hecho de que los rangos de transmisión no son idénticos, donde dan un algoritmo que tolera hasta un 40 % de variación en los rangos de transmisión de los nodos móviles, las variaciones pueden deberse a obstáculos naturales o hechos por el hombre o condiciones ambientales.

Escalabilidad e Información de la posición.

Como se mencionó previamente, los algoritmos tradicionales de encaminamiento intentan dar a los nodos una vista global de la red. Esto requiere una gran capacidad de almacenamiento y genera una sobrecarga en las comunicaciones, ya que los nodos necesitan almacenar una cantidad de información proporcional al tamaño de la red. Lo anterior lleva a que en un momento dado en algún punto se consuma más ancho de banda para mantener la información, que para enviar los datos.

La ventaja de un algoritmo basado en la posición es que no se propaga información en la red para mantener una ruta. Los nodos en la red solamente necesitarán periódicamente informar su posición y su presencia a sus vecinos. Por lo que entonces, una mayor cantidad de ancho de banda está disponible para encaminar datos en vez de mantener la información global de la red.

1.4. Tesis y Perspectiva

La tesis propuesta en el presente trabajo considera que un algoritmo de encaminamiento voraz puede ser usado para encaminar paquetes y garantizar su entrega, cuando se hace uso de la estructura idónea, tal es el caso del *dual del grafo plano*, que es mostrado en el capítulo 5. También se considera la manera de como lograr que los algoritmos que realizan encaminamiento por caras mejoren su desempeño, cuando son usadas todas las aristas que representan a la red, lo cual no se logra siempre con los algoritmos que extraen un subgrafo plano de la red, por lo tanto se propone representar a la red mediante un *grafo*

virtual plano, el cual es discutido en el capítulo 6, que es usado como guía para realizar el encaminamiento por caras, y el encaminamiento en los nodos se hace considerando a todas las aristas del grafo.

En el capítulo 2 se hará una revisión de los algoritmos tradicionales de encaminamiento que han sido propuestos para redes inalámbricas ad-hoc. Se verá que pueden ser divididos en tres tipos, protocolos *proactivos*, *reactivos* e *híbridos* y se darán algunos ejemplos de estos protocolos.

En el capítulo 3 se definirá el modelo usado en los algoritmos basados en la posición. También se hará una revisión del desarrollo de los algoritmos basados en la posición comenzando con el algoritmo Voraz, el algoritmo de Encaminamiento por Caras y sus variantes, el algoritmo Voraz Progresivo, así como otros que hacen una combinación de los anteriores.

En el capítulo 3 se señala que existen algunos algoritmos que requieren obtener un grafo plano, por lo cual en el capítulo 4 se abordan algunos de los algoritmos que permiten obtenerlos, ya sea de forma global, o usando sólo información local.

1.5. Contribuciones de la tesis

En la tesis se hacen las siguientes contribuciones en el área de redes inalámbricas ad-hoc basadas en la posición.

1.5.1. El Dual del Grafo Plano

En el capítulo 5 se describirá una nueva estructura que permite encaminar en forma voraz y garantizar la entrega del paquete. Para los algoritmos voraces existe alguna topología de la red que impide garantizar la entrega del paquete, ya sea porque se quedan atrapados en un ciclo, o bien, porque el encaminamiento voraz falla para seguir enviando el paquete a través de los nodos intermedios.

1.5.2. El Grafo Virtual

En el capítulo 6 se presenta la propuesta de sobreponer un Grafo Plano Virtual a la red subyacente, de tal forma que modele a la red física subyacente. Las ventajas son:

- Un mejor desempeño, ya que se tiene la posibilidad de usar todas las aristas, lo cual no sucede con los algoritmos que extraen un subgrafo planar localmente porque excluyen un subconjunto de aristas, para obtener un grafo plano, por lo tanto, para algunas métricas logran un buen desempeño pero, con otras métricas es inferior.

- El grafo virtual se puede encontrar de forma local y de una manera sencilla, ya que la ubicación de algún nodo dentro del grafo se realiza con operaciones geométricas sencillas.

- El poder de cómputo requerido es mínimo porque la manera como se encuentra el grafo virtual, no requiere realizar operaciones costosas, por lo que la respuesta lograda usando el grafo es bastante buena.

Capítulo 2

Ruteo Tradicional en Redes Ad-hoc

2.1. Introducción

En este capítulo se revisarán los algoritmos tradicionales desarrollados para resolver el problema de encaminamiento en redes ad-hoc. Estos algoritmos se dividen en dos grupos de acuerdo a [Royer99], en: *proactivos* y reactivos. Se ha agregado otro grupo que es una combinación de los anteriores conocidos como *híbridos*.

Un algoritmo de encaminamiento *proactivo* es aquel que intenta guardar toda la información concerniente a la topología de la red y las rutas de todas las fuentes a todos los destinos. Estos protocolos son similares a los usados en redes alambradas pero, adaptados para el entorno ad-hoc. La ventaja de los protocolos de encaminamiento proactivos es que las rutas se tienen rápidamente disponibles cuando un nodo desea enviar un mensaje a otro. Una desventaja del encaminamiento proactivo es que el algoritmo requiere memoria en cada nodo para el almacenamiento de información la cual es proporcional al tamaño (número de nodos) de la red. Una segunda desventaja es la gran cantidad de mensajes de control que requieren ser enviados en la red, para mantener en orden la información. En la sección 2.2 se revisará el protocolo DSDV *(Destination Sequenced Distance Vector)* y el protocolo OLSR *(Optimized Link State Routing)* en la sección 2.3, que tiene un mejor desempeño.

Un protocolo de encaminamiento *reactivo* o bajo demanda es aquel que intenta minimizar el número de mensajes de control puestos en la red (menos ancho de banda

usada en los mensajes de control significa un mayor ancho de banda disponible para datos). Estos algoritmos solamente descubrirán una ruta entre un origen y un destino cuando el origen desea comunicarse con el destino. Una ventaja de estos protocolos, es que los nodos solamente necesitan almacenar información de las rutas que actualmente están usando. Una segunda ventaja es, como se dijo previamente, que estos protocolos intentan usar menos mensajes para mantener la información del encaminamiento. La principal desventaja de estos algoritmos, es que si los nodos no tienen una ruta al destino con el cual se quieren comunicar, deberá ser descubierta previamente. Con lo anterior se tiene un retraso antes de poder enviar el primer paquete. Sin embargo, una vez que la ruta ha sido descubierta, los mensajes pueden ser enviados normalmente. En la sección 2.4 será revisado el protocolo AODV *(Ad-hoc On Demand Distance Vector)* y en la sección 2.5 se discutirá el protocolo DSR *(Dynamic Source Routing)*.

El último tipo de algoritmos de encaminamiento que será discutido en el trabajo son los protocolos *híbridos* . Estos algoritmos intentan combinar ambos protocolos de encaminamiento, los proactivos y los reactivos, para obtener las ventajas que ofrecen. Estos protocolos típicamente dividen a la red ad-hoc en regiones en donde se emplea un protocolo de encaminamiento proactivo, y un protocolo de encaminamiento reactivo cuando se encamina entre regiones. En la sección 2.6 se verá el protocolo ZRP *(Zone Routing Protocol)*.

2.2. Protocolo Vector de Distancia por Secuencia de Destino

El protocolo DSDV *(Destination Sequenced Distance Vector)* es un protocolo vectorial de distancia salto a salto [Larson98, Perkins94], que permite el encaminamiento multisalto entre los nodos de una red ad-hoc. Cada nodo de la red mantiene una tabla de encaminamiento que contiene todos los posibles destinos y el número de saltos que daría un paquete que viaje hacia el destino especificado. La tabla de encaminamiento también almacena el siguiente vecino y un número de secuencia. DSDV requiere que todos los nodos periódicamente envíen toda la información disponible de encaminamiento que hayan aprendido, o que lo hagan inmediatamente cuando se presente un cambio en la información, por ejemplo alguna desconexión, este tipo de actualización es incremental. DSDV es un protocolo libre de ciclos, lo cual se logra usando el número de secuencia mencionado previamente para marcar la ruta. El número de secuencia indica que tan antigua es una ruta. Entre mayor sea el número de secuencia, más actualizada está la información de la ruta,

por lo que será la ruta usada. Una ruta x se considerará más favorable que otra ruta y; si el número de secuencia de la ruta x es mayor que el de y, en el caso de que sus números de secuencia sean iguales, entonces la ruta x deberá tener el menor número de saltos. Cuando un nodo descubre que una ruta en su tabla ha sido rota, el nodo incrementará el número de secuencia y pondrá el número de saltos en ∞. El nodo inmediatamente después avisará a sus vecinos del cambio de la información.

El protocolo DSDV es un protocolo vectorial de distancia con unas cuantas modificaciones para poder usarlo en las redes ad-hoc. Con la incorporación de la actualización inmediata, cuando un cambio ocurre en la topología de la red, se asegura que la información sea distribuida a todos los nodos de la red. DSDV requiere muchos mensajes de control para actualizar la información de las rutas a través de la red. En el protocolo DSDV se han definido dos tipos de mensajes de actualización:

- **Full Dump**. Este paquete transporta toda la información disponible sobre el encaminamiento y puede requerir que su envío se divida en varias unidades más pequeñas. Cuando los cambios en la red son pequeños, es raro que se use este tipo de paquete.

- **Incremental Dump**. El paquete sólo contiene la información que ha variado desde el último *full dump*.

Una de las principales desventajas del protocolo DSDV, al igual que muchos de los protocolos de encaminamiento proactivos para las redes ad-hoc, se debe al hecho de que la información deberá ser recogida para todas las rutas en la red. Esto significa que tomará algún tiempo para que la información converja y produzca un ambiente de encaminamiento estable. También, en cada nodo, los números de secuencia aseguran que las rutas se mantenga actualizadas.

2.3. Protocolo de Encaminamiento Basado en Estado de Enlace Optimizado

El protocolo de encaminamiento basado en estado de enlace optimizado o OLSR *(Optimized Link State Routing)* descrito en [Jacquet01] tiene como principales características el ser un protocolo de encaminamiento proactivo basado en estado de enlace. Al ser

proactivo los nodos que forman la MANET intercambian periódicamente mensajes de control que permiten aprender la topología de la red. Al estar basado en encaminamiento por estado de enlace los mensajes de control inundan a toda la red, y la información topológica que contienen consiste en el estado de los enlaces que posee el nodo que origino el mensaje con sus vecinos.

La operación de inundado de mensajes en la red es una operación costosa para los limitados recursos de las redes móviles inalámbricas, adicionalmente el problema se incrementa debido a que la inundación ocurre de forma periódica. La forma de solucionar lo anterior, por parte de OLSR, es usando los relevadores multipunto o MPR *(Multi Point Relays)*, que es un conjunto mínimo de vecinos que deben retransmitir los mensajes de control, para disminuir la sobrecarga de tráfico de control, gracias a que los MPR son los únicos encargados de retransmitir los mensajes de control.

2.4. Protocolo de Encaminamiento Ad-hoc Bajo Demanda por Vector de distancia

El protocolo de encaminamiento ad-hoc bajo demanda por vector de distancia o AODV *(Ad-hoc Demand Distance Vector)* [Perkins94, Perkins99] provee encaminamiento multisalto en redes ad-hoc. Se basa en el protocolo de encaminamiento vectorial de distancia. La ventaja principal de AODV es que es un protocolo reactivo, a diferencia del protocolo DSDV el cual es proactivo. Esto significa que AODV solamente descubre las rutas a los destinos para los cuales se está tratando de usar actualmente. Lo anterior hace que se tenga una reducción considerable de espacio de almacenamiento requerido para cada nodo.

Las optimizaciones que se consiguieron respecto al protocolo DSDV son: la reducción en el tiempo para obtener una ruta, la disminución del gasto de memoria y la reducción del tráfico de control en la red. Se conservó la idea de mantener números de secuencia y tablas de encaminamiento; pero se añadió el concepto de encaminamiento bajo demanda, es decir, sólo se guarda información de los nodos que intervengan en la transmisión de datos.

El protocolo usa varios mensajes de control para obtener información de las rutas y para mantener los enlaces. Cuando un nodo desea descubrir la ruta a otro nodo envía a sus nodos vecinos, también conocido como *broadcast*, un mensaje RREQ *(Route Request)* que será oído por sus vecinos. El RREQ se propagará a través de la red hasta que alcance

el nodo destino, o bien, que algún nodo tenga información de la ruta que está buscando el nodo origen. El nodo destino o el nodo que tenga información de la ruta contestará con un mensaje RREP *(Route Reply)* , el cual es regresado al nodo fuente en modo *unicast*, es decir, se envía de un nodo a otro nodo usando la lista de nodos recorridos que se usaron para llegar a tal nodo. Para mantener actualizada la información, los nodos periódicamente enviarán en modo broadcast una señal, por lo que sus vecinos inmediatos tendrán conocimiento de su participación continua en la red. Los vecinos que están usando el nodo (del cual ellos recibieron una señal) para encaminar pueden tener las rutas como válidas. Si un nodo vecino no responde después de cierto tiempo, entonces el nodo marcará el enlace al vecino como desconectado, y mandará un mensaje indicando que el enlace ha fallado a los nodos afectados, es decir, aquellos que usan ese enlace para encaminamiento.

2.4.1. Descubrimiento de rutas en AODV

Cuando un nodo necesita descubrir una ruta para un destino que no tiene, enviara un mensaje RREQ en modo broadcast. Esto sucederá cuando el nodo no tenga una ruta al destino, o la ruta haya caducado, o se le haya notificado al nodo que ha fallado. El nodo esperará hasta que se reciba el mensaje RREP. Si no se recibe RREP después de un tiempo fijo, el nodo intentará una cantidad fija de veces nuevamente, y si vuelve a fallar entonces el nodo asumirá que no existe ruta a ese destino.

Cuando un nodo recibe un mensaje RREQ para un destino x, una de las siguientes dos acciones podrá ocurrir.

- El nodo no tiene información de la ruta. En este caso el nodo reenviará el mensaje RREQ en modo broadcast y también guardará en forma temporal la ruta de regreso al nodo fuente para un posible uso futuro.

- El nodo es x o tiene información de la ruta hacia x. Para esta situación un mensaje RREP puede ser regresado directamente a la fuente en modo unicast. Como el mensaje RREP es regresado al nodo fuente, la información del encaminamiento es actualizado a lo largo de la ruta. Cuando la fuente recibe el mensaje RREP, una ruta al destino x habrá sido creada y podrá ser usada.

2.4.2. Mantenimiento de rutas en AODV

Cuando se establece una ruta entre dos nodos, la ruta se considera válida durante un periodo de tiempo. Esto es debido a que los nodos son móviles y una ruta que antes era óptima, después de un tiempo puede que ni siquiera sea válida. Para defenderse de estas situaciones, AODV utiliza el mantenimiento de rutas. Si el nodo origen de un envío se mueve (y altera la topología de la red), él debe reiniciar un nuevo descubrimiento de ruta hacia el destino. Sin embargo, si ha sido el nodo destino de los datos el que se ha movido o algún nodo intermedio, y hay algún mensaje dirigido hacia él, un mensaje especial de error en ruta (RERR) será enviado al nodo que originó el envío, para que el nodo origen sepa del cambio en la topología de la red. Es importante resaltar que no todos los cambios de los nodos ocasionan operaciones en el protocolo, recordemos que AODV encamina bajo demanda. Todos los nodos por los que atraviese este paquete (RERR), cancelarán las rutas que pasan por el nodo que se ha vuelto inaccesible. En el momento que el RERR llegue a su destino, éste puede decidir dar por terminado el envío o iniciar un nuevo RREQ si aún necesita establecer la comunicación. Es preciso mantener información actualizada de quienes son los vecinos de cada nodo cada cierto tiempo. Cada vez que un nodo recibe un paquete de algún vecino, la entrada para ese vecino en la tabla de rutas se refresca, pues se sabe con seguridad que sigue en su lugar. Si no hubiera entrada todavía para el vecino, se crearía una nueva entrada en la tabla de encaminamiento. Además, cada cierto intervalo de tiempo, se mandan paquetes HELLO a los vecinos para informarles que el propio nodo sigue activo. Esta información es usada por los vecinos para actualizar los temporizadores asociados a dicho nodo o en su defecto, para deshabilitar las entradas que se encaminen por el nodo que no responde.

2.4.3. Características

La mayor ventaja de AODV sobre los protocolos de encaminamiento diseñados para redes alambradas, tales como el vector de distancias, es que en AODV se tiene una reducción significativa del número de mensajes de control introducidos en la red, gracias a su naturaleza reactiva. AODV también es más parecido a los protocolos diseñados para redes alambradas, en oposición a DSR (en la siguiente sección), por lo que la integración de una red ad-hoc con una red alambrada es más fácil.

2.5. Encaminamiento Dinámico de la Fuente

El protocolo de encaminamiento dinámico de la fuente o DSR *(Dynamic Source Routing)* [Johnson96] es otro protocolo de encaminamiento reactivo para redes ad-hoc. Este difiere de los protocolos de vector de distancias en el sentido de que el nodo fuente conoce la ruta salto a salto que deberá ser tomada por el paquete de la fuente al destino. Esta ruta es guardada en la cabecera del paquete y los nodos intermedios a lo largo de la ruta simplemente envían el paquete a su siguiente vecino indicado en la ruta.

El protocolo DSR realiza dos fases. La primera es el *descubrimiento de rutas* y la segunda es el *mantenimiento de rutas*. El *descubrimiento de rutas* es usado cuando un nodo origen s desea mandar un paquete a un nodo destino t y no se tiene la ruta almacenada en memoria. El *mantenimiento de rutas* es un mecanismo donde un nodo s puede detectar los nodos de la red que se han movido, por lo tanto deberá prevenir el uso de esa ruta, cuando se está mandando un paquete a t. Si algunos enlaces a lo largo de la ruta han sido rotos, s puede usar una ruta alterna hacia t, o iniciar el mecanismo de descubrimiento de rutas otra vez. El mantenimiento de rutas es solamente usado cuando se envían paquetes de s a t.

2.5.1. Descubrimiento de rutas

Para descubrir una ruta a un destino, el emisor s iniciará un descubrimiento de ruta enviando un mensaje RREQ *(Route Request)* a sus vecinos. El mensaje RREQ identifica al emisor y al nodo destino, también contiene un identificador de solicitud único escogido por el emisor. El mensaje RREQ también contiene una lista de direcciones de todos los nodos intermedios a través de los cuales ha sido enviado el paquete. La figura 2.1 muestra un ejemplo del procedimiento de descubrimiento de rutas con el nodo A como el iniciador y el nodo D como el destino.

Cuando un nodo recibe un paquete RREQ, uno de los siguientes dos eventos ocurrirá:

- Si el nodo es el destino, éste mandará un mensaje RREP de regreso al emisor iniciador. En la respuesta, una copia del registro de rutas es incluida por lo que el emisor puede guardar esta ruta para su uso futuro.
- Si el nodo no es el destino, pero ha visto un RREQ de la misma fuente con el mismo identificador de solicitud, o ve su propia dirección en el registro de rutas, simplemente

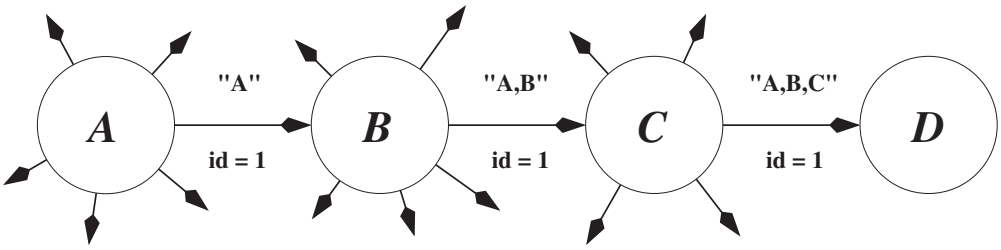


Figura 2.1: Un ejemplo de descubrimiento de rutas en DSR. A es un iniciador y D es el destino.

ignorará el mensaje y lo descartará. Si la petición es nueva, entonces el nodo agregará su dirección al final del registro de rutas y enviará la petición con el mismo identificador de solicitud a sus vecinos haciendo un broadcast local.

2.5.2. Mantenimiento de rutas

En DSR, el mantenimiento de rutas es hecho mientras los nodos están mandando paquetes a través de la ruta encontrada. Cada nodo que está enviando un paquete es responsable de asegurarse que el siguiente nodo haya recibido el paquete. Esto se hace usualmente con una pequeña sobrecarga al protocolo DSR a través del *IEEE 802.11 link-level acknowledgement frame* [IEEE97]. Un segundo método para que un nodo conozca que un paquete ha sido recibido por el siguiente nodo es un *reconocimiento pasivo* [Jubin87] donde el nodo que envía sobre escucha al siguiente nodo en el envío del paquete al próximo nodo. Si el nodo que envía no puede usar alguno de los dos métodos anteriores, existe una tercera alternativa. El nodo que envía puede poner un bit en la cabecera solicitando una confirmación de reconocimiento, que será mandada de regreso por el siguiente nodo.

Si un paquete no es confirmado, este será reenviado un determinado número de intentos. Si después de lo anterior no ha sido confirmado, el nodo que envía asumirá que hay un problema y generará un mensaje RERR *(Route Error)* que será regresado a la fuente emisora del paquete. El mensaje RERR permitirá al emisor saber que la ruta ha fallado, y específicamente cual transmisor a lo largo de la ruta causó el problema. En la figura 2.2 se puede ver que el nodo B no puede enviar un paquete a C. En este caso B enviará un mensaje RERR de regreso a A informando que el enlace BC está roto. El nodo A entonces eliminará esta ruta de su memoria. Si A quiere mandar paquetes adicionales al destino D deberá buscar una ruta alterna en su memoria o reiniciar el procedimiento de

descubrimiento de una ruta para el destino D.

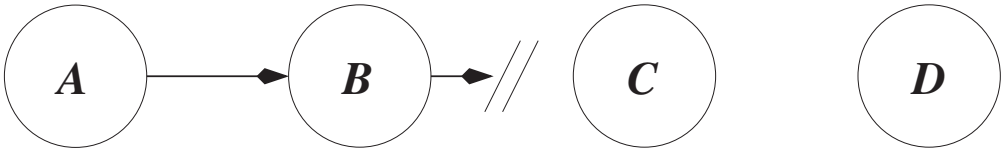


Figura 2.2: Un ejemplo del mantenimiento de rutas en DSR. A quiere mandar un paquete a D; pero el enlace del nodo B al nodo C esta roto.

2.5.3. Características

La principal ventaja del protocolo DSR es que los nodos intermedios en la red usados para mandar paquetes no requieren mantener actualizada la información del encaminamiento. En DSR, no hay necesidad de enviar periódicamente señales para actualizar la información de los vecinos. Esto reducirá el consumo de ancho de banda y de energía, en particular cuando hay poca o nada de movilidad del nodo.

Un nodo puede también descubrir información de la ruta escuchando los paquetes que recibe. Por ejemplo, si un nodo A que está enviando paquetes al nodo C a través del nodo B descubrirá las rutas hacia B y C.

El hecho de que los paquetes deban incluir la ruta fuente desde el origen hasta el destino se traduce en que cada paquete introduce una sobrecarga en el ancho de banda para guardar la información de la ruta. Esta sobrecarga es directamente proporcional al número de transmisores que el paquete deba recorrer de la fuente al destino.

Por último, en DSR los nodos operan en forma no segura al estar escuchando todos los paquetes que reciben para buscar información. Lo anterior es un riesgo en la seguridad y algún mecanismo se debe dar para hacer seguro el protocolo, por ejemplo alguna forma de criptografía.

2.6. Protocolo de Encaminamiento por Zonas

El protocolo de encaminamiento por zonas o ZRP *(Zone Routing Protocol)* [Larson98, Haas02] es una clase de protocolo de encaminamiento *híbrido*. Este combina un protocolo de encaminamiento reactivo y uno proactivo. La red es subdividida en regiones, llamadas

zonas y tienen un protocolo para encaminar dentro de la zona y otro para encaminar entre múltiples zonas.

Los dos protocolos son el IARP *(Intrazone Routing Protocol)* y el IERP *(Interzone Routing Protocol)*. El protocolo IARP es usado para encaminar paquetes dentro de una zona, y se podría usar un protocolo de encaminamiento *proactivo* como el DSDV. Distintas zonas podrían ejecutar protocolos diferentes. Cuando la topología de la red cambie como resultado de la movilidad, la información es solamente propagada dentro de la zona afectada y no en toda la red.

El protocolo IERP es un protocolo de encaminamiento *reactivo* y es usado para descubrir rutas a través de la red entre múltiples zonas de encaminamiento. Si el destino no se encuentra dentro de la misma zona de encaminamiento, el protocolo enviará un mensaje RREQ en modo broadcast a todas las zonas que se encuentren en la periferia de la zona actual (*bordercast*). Los nodos de la periferia enviarán el mensaje RREQ si el nodo destino no está dentro de su zona de encaminamiento. Lo anterior se repite hasta que el nodo destino es encontrado y se regresa un mensaje RREP al emisor. El protocolo IERP incluye el protocolo BRP *(Bordercast Resolution Protocol)* que se encarga de administrar los nodos en la periferia de la zona. Este protocolo permite enviar mensajes de un nodo a todos los nodos en la frontera de la zona actual.

2.6.1. La zona de encaminamiento

La zona de encaminamiento de un nodo A está definida por el conjunto de todos los nodos que puede alcanzar dentro de un número específico de estaciones a partir de A. En la figura 2.3 se puede observar que los nodos A, B, C, E, H, I, J están a una distancia de una o dos estaciones del nodo D. Supongamos el siguiente ejemplo donde el nodo A desea enviar un mensaje al nodo G, como el nodo G no se encuentra dentro de la zona de encaminamiento de A, un mensaje de petición de ruta es mandado a la periferia de la *Zona* A, en este caso al nodo C y D. Estos nodos revisarán si el nodo G se encuentra dentro de su zona de encaminamiento. Como el nodo G no es encontrado en C, entonces C envía una petición al siguiente conjunto de nodos periféricos A, D, F, H, y por otra parte el nodo D envía la petición a los nodos A, C, E, I, J. El nodo destino G es encontrado en las zonas de E y F, por lo que una respuesta es regresada al nodo A.

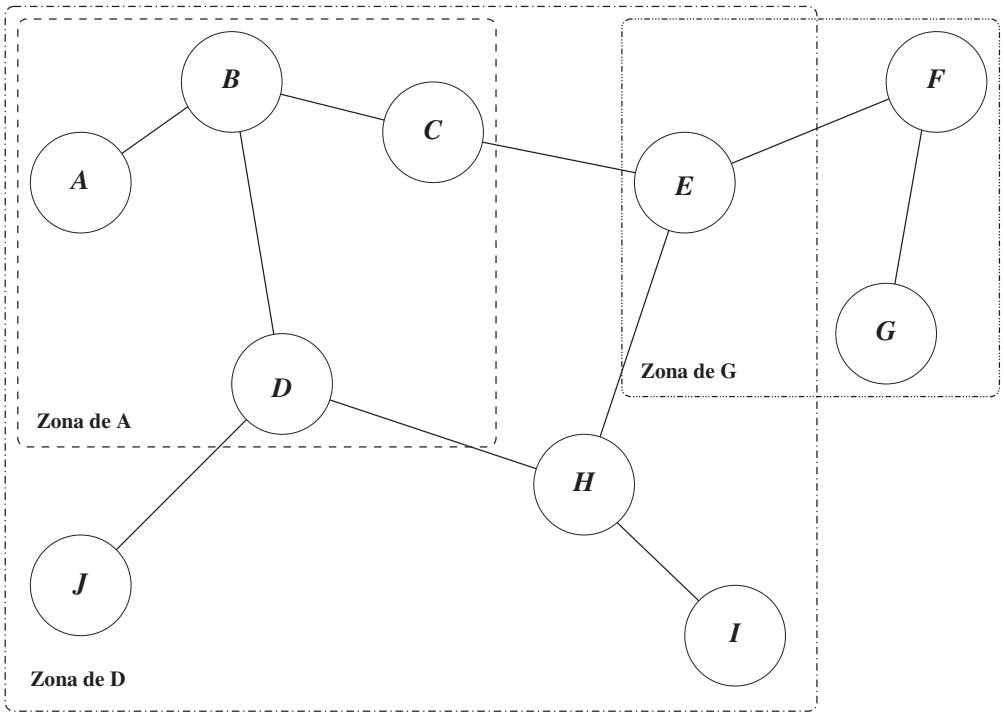


Figura 2.3: Ejemplo del protocolo ZRP en donde se muestran las zonas de los nodos A, D y G

2.6.2. Características

Este protocolo es un híbrido que aprovecha las ventajas de los protocolos proactivos y reactivos. Por una parte las rutas son descubiertas rápidamente dentro de una zona, y por otra parte los destinos fuera de la zona de encaminamiento son encontrados preguntando a un subconjunto de nodos dentro de la zona.

Otra ventaja del protocolo ZRP es que puede ser ajustado para trabajar con diferentes topologías. El diámetro de la zona (el número de estaciones) puede ser cambiado por algún administrador para optimizar la operación en distintos escenarios particulares.

2.7. Resumen

En este capítulo se han visto las características principales de los algoritmos tradicionales de encaminamiento en redes ad-hoc. La principal desventaja de estos algoritmos es que tienen una sobrecarga por el hecho de tener que dar mantenimiento a tablas de

encaminamiento, o para descubrir rutas. Estos protocolos "inundan" la red durante el descubrimiento de la ruta, o al actualizar las tablas. La principal ventaja de los algoritmos revisados en éste capítulo es que no requieren de información de posicionamiento, en el próximo capítulo se verán las ventajas de usar el posicionamiento para encaminar.

Capítulo 3

Encaminamiento basado en la posición

En este capítulo se describe una clase de algoritmos conocidos como *basados en la posición* o de *encaminamiento geométrico*. Esta clase de algoritmos tiene algunas ventajas adicionales respecto a los algoritmos tradicionales basados en la topología. Se revisarán estas ventajas en la descripción del modelo general para protocolos de encaminamiento geométrico. Estos algoritmos hacen uso de la información acerca de su posición física o geográfica de los nodos en la red. En las siguientes secciones se presentará un modelo general y una breve revisión de los algoritmos de encaminamiento basados en la posición. Con excepción de los dos últimos algoritmos presentados, estos algoritmos no "inundan" [1] la red durante su operación; pero seleccionan el siguiente "mejor" nodo para mandar los paquetes. Los algoritmos LAR y DREAM usan "inundado parcial" para descubrir la ruta, o para enviar paquetes, dentro de una región de la red definida por la posición del nodo que envía y el rango en el que se espera encontrar al nodo destino. La principal desventaja de los algoritmos basados en la posición respecto a los algoritmos tradicionales es el compromiso de memoria contra rutas óptimas. Los algoritmos tradicionales propagan información acerca de la red a los nodos, por lo que más rutas óptimas pueden ser encontradas.

[1] La operación de inundado es el envío por parte de cualquier nodo de un paquete a todos los nodos que se encuentren conectados con él; donde la retransmisión la hacen aquellos nodos que no habían recibido el paquete.

3.1. El modelo

Para describir el modelo de los algoritmos de encaminamiento basados en la posición, se pueden describir las siguientes características generales:

- Cada nodo en la red tiene forma de determinar su posición física dentro de su ambiente. La dirección puede ser obtenida a través de un receptor GPS o algún otro mecanismo.

- Todos los nodos deberán hacer uso de un *servicio de localización* para encontrar la posición del dispositivo con el cual ellos desean comunicarse. En el caso de no contar con el servicio, los algoritmos basados en la posición no pueden emplearse, en el trabajo de [Rao03] se da un algoritmo que puede emplearse en el caso de no contar con el servicio, el cual será discutido en la sección 3.9.

- Con la información de la posición del destino, las decisiones de encaminamiento en cada estación en la ruta pueden ser hechas basadas en la posición del nodo actual y la posición del destino.

- No hay necesidad de mandar mensajes adicionales en la red para mantener una tabla de encaminamiento. Los dispositivos en la red simplemente en forma periódica enviarán una *señal* conteniendo información tal como el identificador del nodo y sus coordenadas geográficas.

- Una ventaja adicional al emplear encaminamiento basado en la posición, es la posibilidad de mandar mensajes a los dispositivos de un área en particular. Un ejemplo de ésto podría ser una aplicación de anuncios dirigidos, donde la información es enviada a usuarios dentro de una cierta área, tal como una plaza comercial. Lo anterior es conocido como "geocasting" [Navas97] y no se tratará.

Hay algunas variaciones al modelo anterior. Por ejemplo, algunos algoritmos escogen guardar alguna información de las rutas que han descubierto, como lo hace LAR, al usar solamente información local para descubrir rutas y no para realizar el encaminamiento. Algunos algoritmos no usan explícitamente un servicio de localización, pero podrían propagar información acerca de los nodos dentro de los paquetes de datos.

3.2. Encaminamiento Tradicional Voraz

En esta sección se comentará un conjunto de algoritmos *voraces*, los cuales son revisados y clasificados en [Giordano01, Mauve01]. La definición de progreso es empleada en [Giordano01], la cual es usada para formar una clase de los esquemas de encaminamiento basados en la posición. Si se tiene un nodo s deseando enviar un paquete, el progreso de un nodo vecino n es la proyección sobre la recta que forma s al destino. Si el nodo n proyecta en la dirección de avance, entonces el nodo tiene *avance o incremento positivo*, caso contrario se dice que tiene *retroceso o incremento negativo*. En la figura 3.1 se muestran las proyecciones de los vecinos del nodo s.

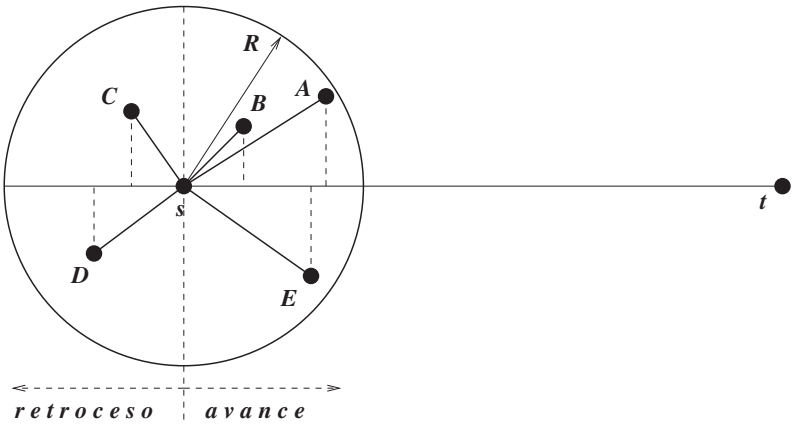


Figura 3.1: Progreso de los nodos vecinos de s

En el método de *progreso aleatorio* [Nelson87] , en cada estación a lo largo de la ruta, los paquetes para un nodo t son aleatoriamente enviados a cualquier nodo vecino que tenga un avance del nodo actual hacia el destino. En la figura 3.1 el nodo s podría seleccionar aleatoriamente el nodo A, B o E.

En el método MFR *(Most Forward within Radius)* [Takagi84] , el vecino seleccionado es aquel que está más cercano al destino, es decir, el que tenga el mayor avance.

El algoritmo MFR es modificado en [Hou86] donde los nodos pueden ajustar su potencia para sólo cubrir la distancia entre los nodos que están en comunicación. Con lo anterior, los paquetes son enviados al vecino más cercano con avance. Esto es conocido como NFP *(Nearest with Forward Progress)*.

En la figura 3.1, se pueden ver los nodos que presentan avance y retroceso del nodo s. Si se emplea el método MFR, el nodo A será seleccionado, mientras que con el método

NFP el nodo B será escogido.

Finalmente, se puede ver que los algoritmos de envío voraces por sí mismos no pueden garantizar la entrega de paquetes. Hay algunas topologías de redes, tal como se muestra en la figura 3.2, donde un paquete podría ser enviado a un nodo donde ya no se pueda seguir avanzando. En estos casos, deberá existir algún mecanismo de recuperación. Un algoritmo que tiene tal mejora es GPSR [Karp00], el cual se revisará más adelante.

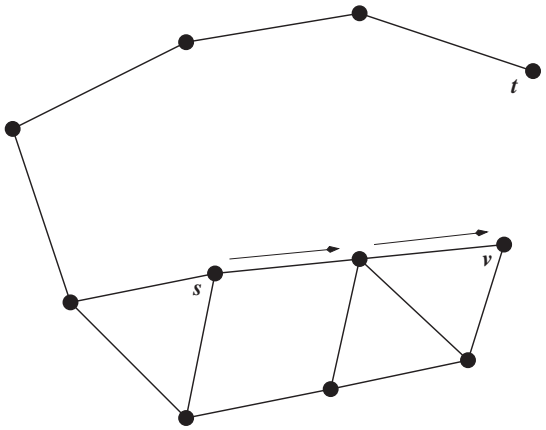


Figura 3.2: Falla del algoritmo voraz para encontrar una ruta de s a t al llegar al nodo v.

3.3. Encaminamiento por Brújula

En [Morin01, Kranakis99] se puede revisar la introducción al concepto de Encaminamiento por Brújula *(Compass Routing)*. La idea se deriva de la forma como un visitante en la vida real intenta orientarse usando alguna referencia local que pueda observar a la distancia. El viajero deberá ir por aquellas calles que lo acerquen más hacia el destino.

Para mandar un paquete de un nodo origen a un nodo destino, las decisiones deberán hacerse en cada nodo a lo largo de ruta. El siguiente nodo que será cruzado en la red, es escogido basándose en la dirección del nodo destino, usando como referencia la dirección del nodo actual al nodo destino. El siguiente nodo escogido deberá ser aquel que tenga la dirección más cercana al destino. Si más de un nodo cumple con la propiedad anterior (hay un máximo de dos), entonces uno de ellos es escogido arbitrariamente. En la figura 3.3 se puede ver como trabaja el algoritmo.

No es suficiente con emplear solamente *compass routing* para garantizar la entrega de paquetes de la fuente al destino. Los autores [Kranakis99] han dado una topología de

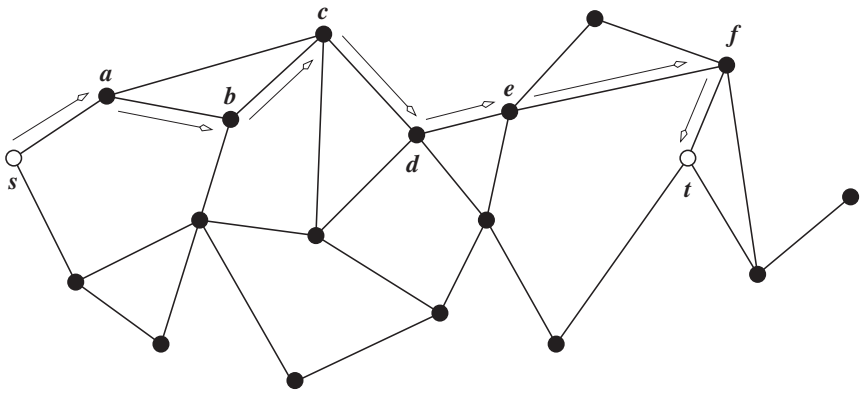


Figura 3.3: Uso del Encaminamiento por Brújula del nodo s al nodo t.

la red en la cual el encaminamiento por brújula entra en un ciclo, la cual se muestra en la figura 3.4, ya que al seguir la arista con el ángulo más pequeño desde el origen s al destino t se entrará en el ciclo $v_0, w_0, v_1, w_1, \ldots, v_3, w_3, v_0, \ldots$.

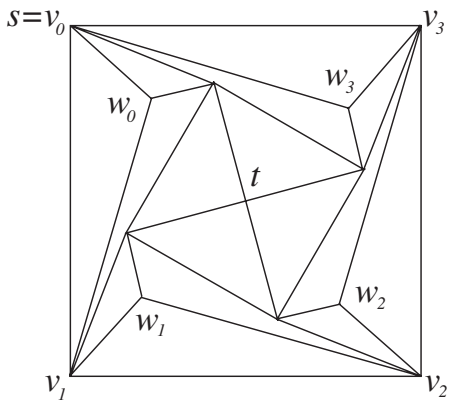


Figura 3.4: Ciclo al usar el Encaminamiento por Brújula

3.4. Encaminamiento por caras

En la presente sección se describen dos algoritmos de *encaminamiento por caras* [Kranakis99, Morin01, Bose01] , en donde los paquetes son progresivamente enviados a través de nodos intermedios de las *caras* adyacentes en un grafo plano hacia el nodo destino.

Cuando un grafo plano es dibujado sin ningún cruce, cualquier ciclo que rodee una región sin ninguna arista dentro de la región forma una *cara*. Un grafo plano conexo subdividirá el plano en un conjunto de *caras* adyacentes. Estas caras están limitadas por una

figura poligonal hecha por las aristas del grafo. Si un paquete llega a un vértice v, el cual se encuentra en una cara f, el paquete puede ser enviado alrededor del contorno de f usando la conocida *regla de la mano derecha* [Bondy76]. Esta regla indica que si con la mano derecha se va tocando la pared mientras se camina hacia adelante, todas las paredes de un laberinto pueden ser recorridas . Esta regla es la base para los algoritmos de *encaminamiento por caras*. Lo anterior se puede lograr también si el recorrido se hace con la mano izquierda.

El operación del primer algoritmo, *caras-1* , es mostrado en la figura 3.5. Todos los nodos conocen su posición además de la posición del nodo destino. La *línea(origen,destino)* es usada por el algoritmo para medir el progreso. El paquete es enviado siguiendo el contorno de la cara y los puntos de intersección de la cara con la *linea(origen,destino)* son guardados.

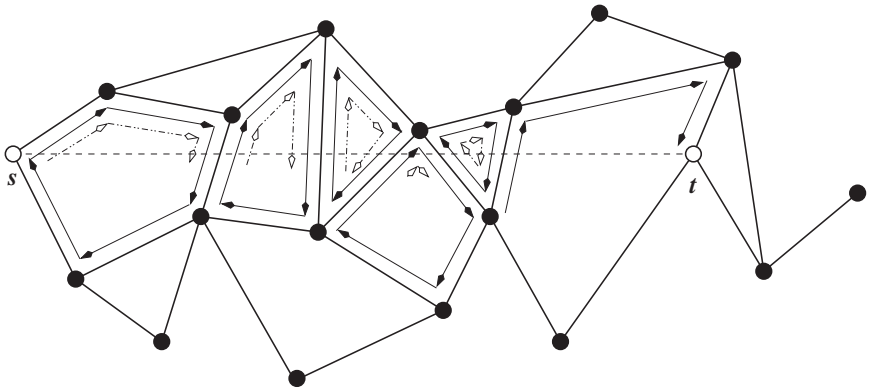


Figura 3.5: El algoritmo *caras-1* encaminando del nodo s al nodo t.

La arista que tenga el punto de intersección p más cercano al punto destino es escogida para "invertir" o "cambiar" el sentido del recorrido a la arista de la siguiente cara. Es importante señalar que una vez que se invierte el sentido del recorrido, se pasa a recorrer la cara adyacente. Una nueva *línea(p, destino)* es ahora usada como prueba de intersección al recorrer la siguiente cara. Este procedimiento se repite hasta que el destino haya sido alcanzado.

En [Bose01] se muestra que el algoritmo alcanzará el nodo destino a lo más en $4m$ saltos, donde m es el número de aristas en la red subyacente. En cada repetición, el paquete es enviado alrededor del perímetro completo de la cara. Una vez que se ha completado deberá ser nuevamente enviado de regreso a la arista de "inversión". Lo anterior se puede mejorar a $3m$ saltos enviando el paquete a la siguiente arista de "inversión" por el camino más corto de los dos que rodean el contorno de la cara.

Los autores en [Bose01] proponen una mejora al primer algoritmo, denominado como *caras-2*, que consiste en "invertir" la arista cuando se encuentre una intersección con la *línea(p,destino)*, tal que la nueva intersección este más cercana al destino que la anterior, en vez de recorrer toda la cara en cada repetición. De esta forma, se garantiza que el algoritmo avance hacia el nodo destino.

En la figura 3.6 se puede ver la forma como funciona el algoritmo *caras-2*. El algoritmo termina en una cantidad finita de pasos ya que la distancia al destino se va reduciendo con cada repetición. En la práctica, el algoritmo *caras-2* es mejor que el algoritmo *caras-1*, sin embargo se ha mostrado que en un caso extremo puede visitar $\Omega(n^2)$ aristas del grafo.

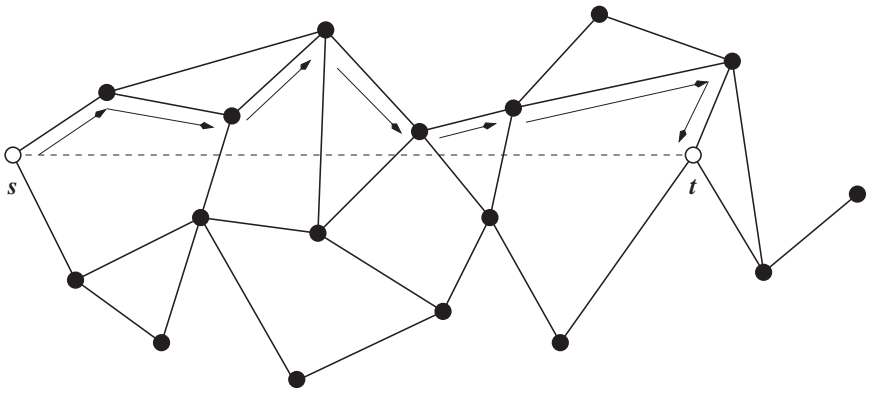


Figura 3.6: El algoritmo *caras-2* encaminando del nodo s al nodo t.

3.5. Encaminamiento Voraz-Perímetro

En [Karp00], los autores presentan el encaminamiento Voraz-Perímetro o GPSR *(Greedy Perimeter State Routing)*. El trabajo está basado en el algoritmo GFG *(Greedy-Face-Greedy)* el cual fue propuesto en [Bose01]. El algoritmo usa dos métodos para enviar paquetes en la red. El primero es el algoritmo *voraz progresivo* el cual es usado siempre que es posible, y el segundo algoritmo *perímetro progresivo* que es usado cuando el primer algoritmo falla, es decir, el segundo algoritmo es el método de recuperación del primer algoritmo.

3.5.1. Voraz Progresivo

Si los nodos conocen la posición de sus vecinos, entonces cada nodo puede tomar una decisión de encaminamiento local, escogiendo el vecino que se encuentre más cercano geográficamente al eventual nodo destino. En la figura 3.7 se puede ver un ejemplo de la selección del siguiente nodo. El nodo actual x ha recibido un mensaje para el nodo t. El nodo x escoge al nodo vecino y por encontrarse más cerca al nodo t. El círculo alrededor del nodo x muestra el rango de transmisión del nodo x y el arco punteado es el círculo alrededor del nodo t con un radio igual a la distancia que hay entre el nodo t y el nodo y. Este proceso se repite hasta que el mensaje llega a su destino el nodo t, o mientras sea posible.

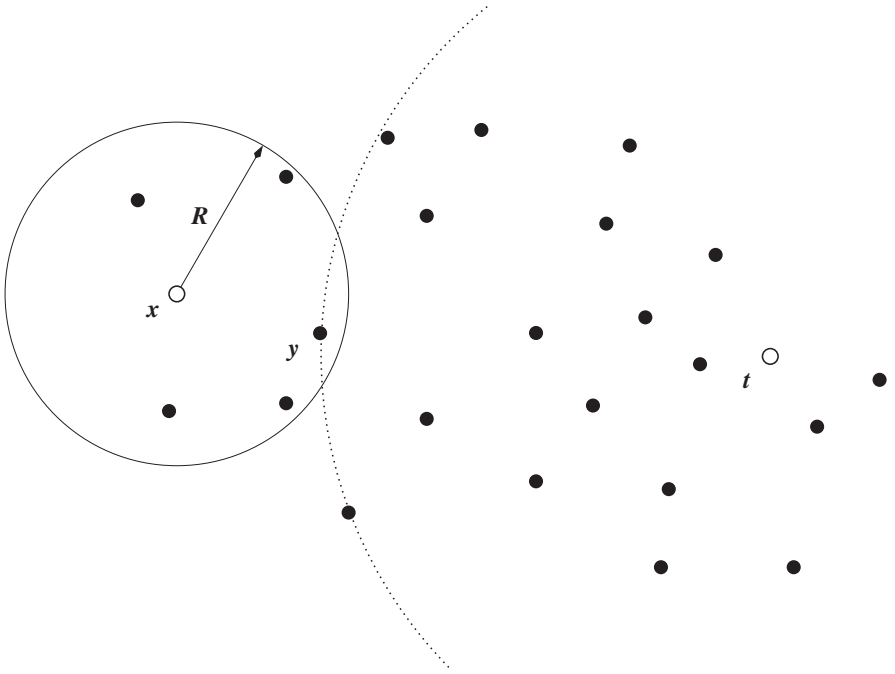


Figura 3.7: GPSR: avance *Voraz Progresivo*

El algoritmo voraz podría fallar para encontrar una ruta desde algún *origen* a un *destino*, ya que algunas topologías de redes harán que el siguiente nodo se encuentre más alejado del destino que el nodo actual, por lo que para solucionarlo, se necesita de algún mecanismo de recuperación, el cual se discutirá a continuación.

3.5.2. Perímetro Progresivo

Como se puede ver en la figura 3.8, la intersección del círculo de transmisión alrededor de x y el círculo alrededor del nodo t con radio $distancia(x, t)$ no tiene vecinos del nodo x. La región sombreada que no tiene vecinos es denominada *vacío*. El nodo x intentará encaminar alrededor del vacío.

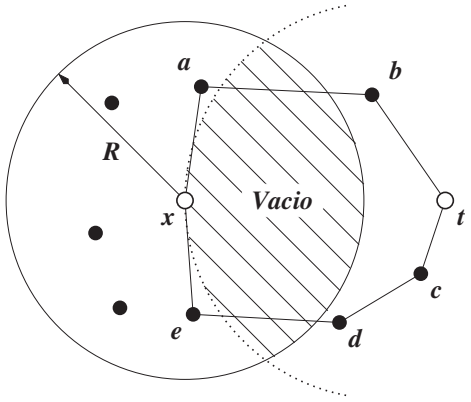


Figura 3.8: GPSR: encaminamiento alrededor del *vacío*.

El algoritmo ahora usa la bien conocida *regla de la mano derecha* para ayudar a encaminar alrededor del vacío. La aplicación de la regla es como sigue: cuando se llega a un nodo x desde un nodo y, el siguiente enlace será el próximo que aparece en sentido contrario a las manecillas del reloj de la arista (x, y). La regla de la mano derecha recorre la *cara* en el sentido de las manecillas.

Los autores usan este propiedad de recorrido en circuito para encaminar alrededor del *vacío*. En la figura 3.8, siguiendo la secuencia $(x \rightarrow a \rightarrow b \rightarrow t \rightarrow c \rightarrow d \rightarrow e \rightarrow x)$ al usar la regla de la mano derecha, se obtiene un encaminamiento alrededor del vacío. Este orden de recorrido es llamado el *perímetro*.

3.5.3. Combinación de algoritmos

El algoritmo GPSR combina el avance voraz con el perímetro progresivo dentro de un grafo *plano* cuando el avance voraz no puede emplearse. Los paquetes inician siendo enviados en *modo voraz*, y el nodo origen revisa su lista de vecinos para saber si hay alguno que se encuentre geográficamente más cercano al nodo destino. Si el vecino mencionado es encontrado, el paquete es enviado a ese nodo. Si no se encuentra ningún vecino, el paquete

es marcado y el algoritmo cambia al *modo perímetro*.

Cuando se encuentra en *modo perímetro*, los paquetes son enviados usando un grafo plano. Esto trabaja en forma similar al algoritmo de encaminamiento por caras, donde los paquetes son enviados a través de las caras adyacentes.

Cuando el paquete cambia al modo perímetro, éste se marca con la posición del punto donde dejó el modo voraz. El algoritmo usa la marca para determinar cual fue el punto de inicio del modo perímetro, de esta forma puede saber si se puede llegar al destino. El proceso se repite hasta que el paquete llega al destino.

3.6. Encaminamiento por Caras Adaptativo

El encaminamiento por caras adaptativo o AFR *(Adaptive Face Routing)* [Kuhn02b, Kuhn02a] es una extensión de *Encaminamiento por Caras*, el cual requiere una red subyacente plana. El algoritmo intenta acotar el encaminamiento por caras para minimizar la distancia recorrida de la *línea(fuente,destino)*. AFR realiza el encaminamiento en dos pasos los cuales se describen a continuación.

3.6.1. Paso BFR$[\hat{c}_d]$

En el paso encaminamiento acotado de cara o BFR *(Bounded Face Routing)*, los autores asumen que una cota superior $\hat{c}_d$ es conocida de antemano, la cual es la longitud $c_d(p^*)$ del camino más corto p^* de la fuente s al destino t. Ellos definen una elipse ϵ como el sitio de todos los puntos donde la suma de distancias a s y a t es $\hat{c}_d$. La elipse tiene focos en s y en t. Por definición de ϵ, el camino más corto de s a t a través de algunos puntos que estén afuera de ϵ será mayor a $\hat{c}_d$.

El algoritmo de encaminamiento por caras es ahora modificado de tal forma que cuando se recorren las caras, se mantiene dentro de los límites de la elipse ϵ. Con la definición de *BFR* se tiene ahora el bloque principal de construcción de *AFR*.

3.6.2. Paso AFR

El problema con *BFR* es que en una red funcionando, una cota superior de la longitud del camino más corto de s a t no es conocida. El algoritmo primero deberá determinar una $\tilde{c}_d$ estimada del valor desconocido $c_d(p^*)$. Para iniciar, sea $\tilde{c}_d := 2\overline{st}$, donde $\overline{st}$ es una

estimación entre el origen y el destino.

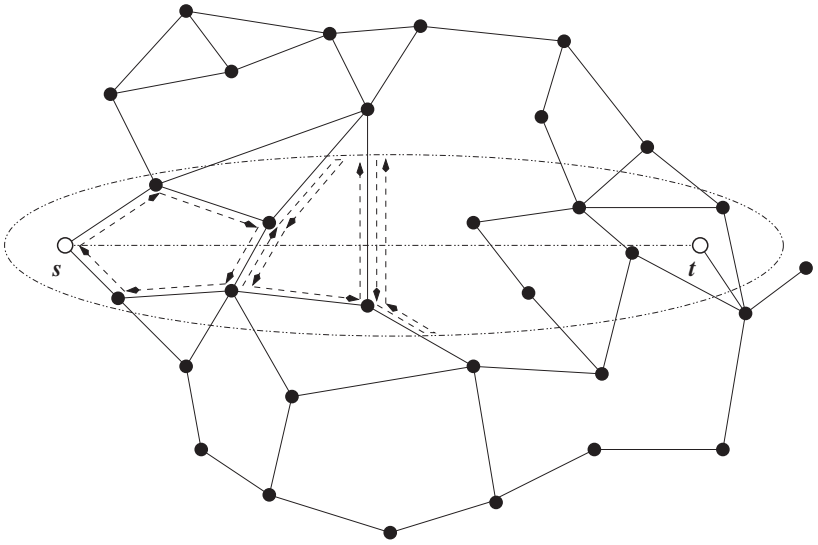


Figura 3.9: El paso BFR falla debido a que el tamaño de la elipse es muy pequeña.

El algoritmo tiene los siguientes pasos:

1. Ejecutar BFR[$\tilde{c}_d$].

2. Si en el paso anterior BFR[$\tilde{c}_d$] fue exitoso para encontrar una ruta al nodo t entonces el algoritmo ha terminado, en caso contrario, doblar la longitud de la estimación del camino más corto, $\tilde{c}_d := 2\tilde{c}_d$. Repetir los pasos iniciando en 1.

La figura 3.9 muestra una ejecución no exitosa de BFR, el límite fue puesto muy pequeño por lo que ninguna ruta al destino fue encontrada. En la figura 3.10 el límite fue incrementado, por lo que el algoritmo tuvo éxito para encontrar una ruta al destino.

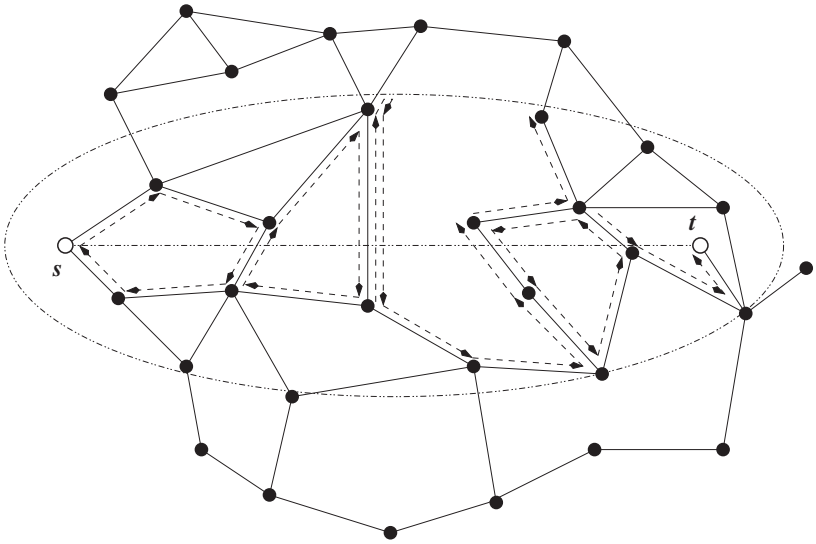


Figura 3.10: Paso AFR en el cual se ajustó el tamaño de la elipse.

3.7. Encaminamiento Auxiliado por Posición

El algoritmo de encaminamiento Auxiliado por Posición o LAR *(Location Aided Routing)* [Ko98] realiza encaminamiento bajo demanda al igual que el algoritmo DSR (el cual se revisó en el capítulo 2). LAR y DREAM (el cual se revisa en la siguiente sección) usan información adicional aparte de la posición de los nodos móviles, como por ejemplo la velocidad promedio de los nodos.

Comparando LAR con DSR, el primero manda su información posicional dentro de cada paquete, y haciendo lo anterior, se esperará reducir la sobrecarga de cualquier petición de ruta. En DSR, cuando un nodo tiene un paquete para mandar a un destino t, pero no tiene una ruta, inundará la red entera con una petición de ruta. Por otra parte, LAR hace uso de la información de la posición de nodos móviles para inundar dentro de una área en una determinada dirección. Esa área es conocida como la *zona de petición* . Usando una zona de petición el algoritmo espera limitar el inundado o encaminar los paquetes de petición dentro de una porción (afortunadamente pequeña) de la red entera.

El protocolo LAR define dos métodos para los nodos que serán usados para determinar la zona de petición, a los cuales se enviarán los paquetes de requerimiento de ruta.

3.7.1. La caja LAR

Con este método se define la zona de petición como un rectángulo con la fuente s en una esquina y la *zona esperada* en la esquina opuesta. La zona esperada es una área circular alrededor del destino t, el cual tiene un radio de acuerdo a la movilidad que podría tener el nodo t, respecto a la última vez que su posición fue actualizada. En la figura 3.11 se observan la zona de petición y la zona esperada dentro de la caja LAR. Los nodos vecinos que no se encuentran en la zona de petición no enviarán los paquetes de petición de ruta.

3.7.2. Paso LAR

Usando el método Paso LAR un nodo intermedio revisa para ver si está más cercano al destino t que el nodo del cual ha recibido el paquete. Si el método determina que está más cerca, lo considerará dentro de la zona de petición y enviará el paquete.

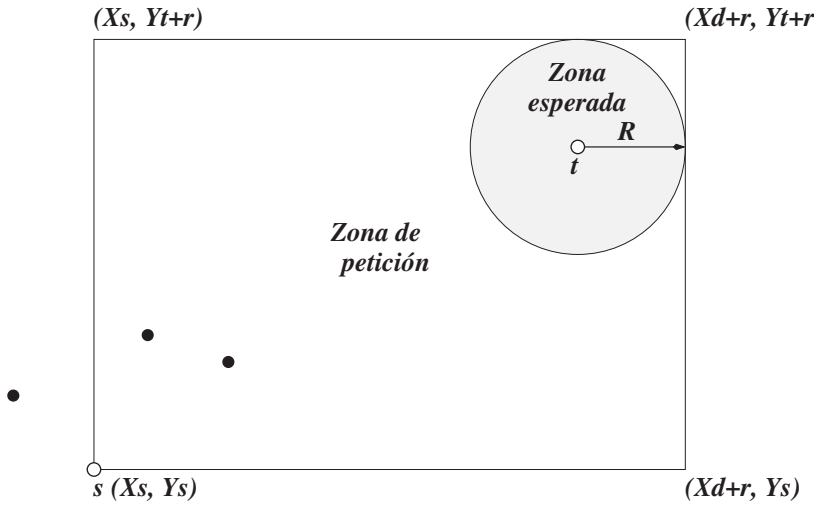


Figura 3.11: LAR: zona de petición y zona esperada. Los nodos que no se encuentran en la zona de petición no envían paquetes.

3.7.3. Encaminamiento

Sin importar cual de los procedimientos anteriores sea usado, el algoritmo determina una ruta mediante un mecanismo de dos estados. En el primer estado, un paquete es enviado usando uno de los dos métodos previos. Si una *respuesta de ruta* no es regresada dentro de un periodo definido (*request timeout*), el algoritmo pasa al segundo estado. En el segundo requerimiento de ruta, toda la red ad-hoc es inundada (como se hace en DSR). Si una respuesta de ruta todavía no es recibida, el nodo actual marcará el nodo destino t como inalcanzable. Si un nodo t permanece inalcanzable por más de 30 segundos, todos los paquetes para t serán desechados.

3.8. Algoritmo de Efecto Encaminamiento Distancia para Movilidad

A diferencia de LAR, el algoritmo de efecto encaminamiento distancia para movilidad o DREAM *(Distance Routing Effect Algorithm for Mobility)* [Camp02, Basagni98] es un protocolo de encaminamiento proactivo. Todos los nodos mantienen guardada la información de la posición de todos los nodos de la red ad-hoc. El mantenimiento de la tabla es hecha cuando los nodos actualizan la información de su posición a todos los nodos de la

red. Lo anterior es hecho de manera sencilla, ya que esta información es enviada a los nodos cercanos con mayor frecuencia que a aquellos nodos alejados. La premisa es que los nodos más alejados de algún observador parecen moverse más lentamente, y cambiando la relación de actualizaciones del algoritmo DREAM se espera limitar el uso de ancho de banda que se necesita para los mensajes de control.

Para mandar un paquete, el origen o algún nodo intermedio s determinan la dirección del destino t. Después usará la información de movilidad que tenga del nodo t para determinar un *rango* de una dirección angular. El paquete actual es mandado a todos los vecinos de s que se encuentren en el rango direccional. El área está determinada por dos tangentes del nodo actual a un rango circular alrededor del nodo destino t. El rango circular queda definido como el área que cubre todas las posibles posiciones a las que se esperaría que se mueva el nodo t, después de su última actualización y que es conocida como la *zona esperada*. El área incluida por las dos tangentes y la *zona esperada* es conocida como la *zona de petición* como se muestra en la figura 3.12.

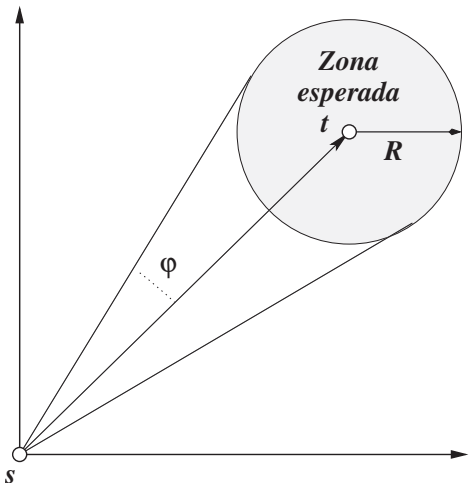


Figura 3.12: DREAM: zona de petición y zona esperada de t

A diferencia del algoritmo LAR, el cual define la zona de petición basada en el origen y la zona esperada del destino, la zona de petición es recalculada en cada nodo intermedio usando la descripción previa. El algoritmo LAR es usado para descubrir rutas (envía paquetes de petición de ruta), mientras DREAM es usado para enviar paquetes de datos. Cuando un paquete llega al destino un reconocimiento ACK (*acknowledgement*) es regresado al nodo fuente usando el mismo mecanismo que fue usado para mandar el dato

original a t.

Si por alguna razón un paquete de datos o el correspondiente ACK se pierden, el origen pasará a un mecanismo de *recuperación*. El nodo origen simplemente inundará la red entera con el paquete destinado a t.

3.9. Encaminamiento Geográfico sin Información de Localización

Como se ha visto en las secciones anteriores los algoritmos basados en la posición requieren que los nodos sepan su posición. Para algunos ambientes la suposición anterior es cierta, por ejemplo en redes de dispositivos de sensores con GPS. Existen otros donde no se dispone de un servicio de localización. Los autores [Rao03] se enfocan principalmente en el caso donde ningún nodo tiene información de su posición, la aproximación que emplean es dando coordenadas *virtuales* a cada nodo y entonces para usar los algoritmos basados en la posición se emplean las coordenadas virtuales. Las coordenadas virtuales no requieren ser exactas respecto a la red geográfica subyacente, sino reflejar la conectividad subyacente. Las coordenadas virtuales son construidas solamente usando información de la conectividad local, la cual está siempre disponible, porque los nodos conocen quienes son sus vecinos, por lo cual puede ser aplicada en distintos ambientes.

El método para construir las coordenadas virtuales sin tener información de la posición, se basa parcialmente en la analogía, prestada de la teoría de encaje de grafos, de que cada *enlace* – cada relación con el vecino – es representado como una fuerza que jala a los vecinos conjuntamente. Se supone que la fuerza en la dirección del eje x es proporcional a la diferencia de las coordenadas en las abscisas (similarmente para las ordenadas). Si se mantienen los vecinos fijos, entonces la posición de equilibrio de un nodo, aquel donde la suma de fuerzas es cero, es donde su abscisa es el promedio de las abscisas de sus vecinos, de forma similar para la ordenada. Con lo anterior se puede tener un procedimiento iterativo donde cada nodo que no se encuentre en el perímetro actualiza sus coordenadas virtuales de la siguiente forma:

$$x_i = \frac{\sum_{k \in conjunto\_vecino(i)} x_k}{|conjunto\_vecino(i)|}$$

$$y_i = \frac{\sum_{k \in conjunto\_vecino(i)} y_k}{|conjunto\_vecino(i)|}$$

Conforme las iteraciones van progresando, los nodos tienden a moverse hacia los nodos perimetral que se encuentran más cercanos a ellos en términos del número de saltos.

El escenario más general es cuando se asume que los nodos no conocen su posición y que tampoco saben que se encuentran en el perímetro. Por lo que el algoritmo para asignar las coordenadas virtuales consiste de los siguientes pasos:

1. Se designan dos nodos como arranque y señalización para que hagan *broadcast* en toda la red. Los nodos usan sus distancias a los nodos de arranque para determinar si son nodos del perímetro.

2. Cada nodo del perímetro manda un mensaje en modo *broadcast* a toda la red para permitir que cada nodo calcule su vector perímetro, es decir, las distancias de cada nodo a todos los nodos perimetrales.

3. Los nodos perimetrales y de arranque mandan en modo *broadcast* sus vectores perimetrales a toda la red.

4. Cada nodo usa las distancias interperimetrales para normalizar sus propias coordenadas y de los nodos perimetrales. Los nodos perimetrales proyectan sus coordenadas fuera del círculo.

5. Los nodos perimetrales se mantienen fijos mientras los otros nodos ejecutan un algoritmo de relajación, como el que fue descrito previamente.

6. Para manejar la movilidad, un nodo designado de arranque periódicamente en modo *broadcast* señaliza la red, con lo cual los nodos periódicamente revisan si se encuentran o no en el perímetro.

Del paso 1 al paso 4 se requieren solamente para iniciar la asignación de las coordenadas cuando la red es inicializada por vez primera y los pasos 5 y 6 son para la operación normal.

3.10. Resumen

En este capítulo se han revisado algoritmos de encaminamiento basados en la posición. Los algoritmos voraces tienen un buen desempeño pero, existen algunas topologías

con las cuales no pueden garantizar la entrega de paquetes, o bien, algunas topologías pueden generar ciclos como en el encaminamiento por brújula. Para resolver problemas como los anteriores, se hace necesario emplear otros algoritmos que permitan que se recuperen los algoritmos voraces. El encaminamiento por caras y sus variantes pueden recuperar a los algoritmos voraces, con lo cual se puede garantizar la entrega del paquete. El encaminamiento por caras requiere de un grafo plano para poder funcionar adecuadamente, estos grafos deberán encontrarse de forma local debido a las características de la red ad-hoc. En el siguiente capítulo se describen algunos algoritmos que permiten obtener un grafo plano.

Capítulo 4

Grafos Planos Calculados Localmente

El capítulo anterior se revisaron algoritmos de encaminamiento por caras, los cuales permiten a los algoritmos voraces que se recuperen cuando fallan o no pueden seguir encaminando. Los algoritmos de encaminamiento por caras requieren de un grafo plano para funcionar adecuadamente, lo cual será mostrado dando un ejemplo donde un cruce nos impide encaminar al destino. En este capítulo se verán varios algoritmos diseñados para derivar un *subgrafo plano* a partir de una red usando solamente información local, es decir, usando únicamente información acerca de la posición del nodo y sus vecinos.

En una red ad-hoc, los nodos individuales por lo general no tienen información *global* acerca de su ambiente. Los nodos conocen la información de su posición y pueden fácilmente obtener la posición de sus vecinos, por lo que los algoritmos locales deberán ser capaces de operar con las restricciones anteriores.

Como se vio previamente, un grafo plano es aquel donde dos aristas no se cruzan. El concepto de *subgrafo plano* se aplica a los algoritmos que obtienen de una red, un grafo que sea plano y que esté conectado.

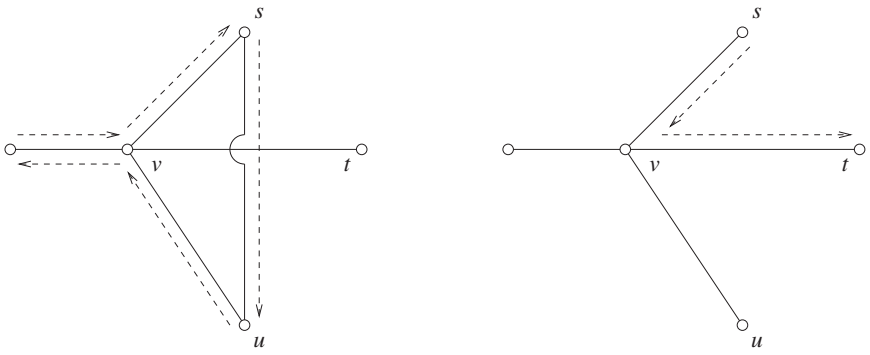
Se describirán primeramente dos algoritmos pioneros, el *Grafo de Vecindad Relativa* y el *Grafo de Gabriel*, los cuales simplemente revisan una región fácilmente definida para la presencia de algún vecino. Posteriormente se describirán dos algoritmos más recientes, el primero de ellos calcula una triangulación de Delaunay local y que remueve algunas aristas para obtener un grafo plano, y en el segundo, el *Grafo de Morelia*, se revisa si para algún

enlace, existen enlaces más grandes que los crucen para obtener un subgrafo plano.

Los algoritmos que obtienen un grafo plano localmente hacen referencia al grafo del Disco Unitario o UDG *(Unit Disk Graph)* , ya que es una forma de modelar la red física subyacente. El UDG es el grafo de intersección de un conjunto de discos cerrados de diámetro unitario en el plano. Es decir, cada vértice corresponde a un disco en el plano, y dos vértices son adyacentes en el grafo si el disco correspondiente esta en la intersección. Se dice que el conjunto de discos realizan el grafo. La distancia unitaria no es crítica, ya que el disco realiza el mismo grafo si el sistema coordenado es escalado por una cantidad conveniente. Observar que dos discos se interceptan si y solo si la distancia entre sus centros es a lo más el diámetro del disco. Por lo tanto, el UDG puede ser realizable también con el conjunto de puntos en el plano, considerando que dos vértices son adyacentes en el grafo cuando la distancia euclidiana entre ellos es a lo más 1.

4.1. Justificación del uso de un grafo plano

Una inquietud natural al realizar el encaminamiento por caras es saber si se requiere obtener un subgrafo plano para poder garantizar la entrega de paquetes. Mediante un ejemplo se mostrará la necesidad de tener un grafo plano para que el encaminamiento por caras funcione. Como se describió en el capítulo anterior, el encaminamiento por caras recorre la cara usando la regla de la mano derecha, en la figura 4.1 del lado izquierdo se muestra el recorrido de la *pseudocara* usando la regla mencionada, en la cual se puede apreciar que no hay avance del nodo s hacia el nodo t; del lado derecho de la misma figura se muestra que se puede llegar al nodo destino una vez que se elimina el cruce quitando la arista que se encuentra entre s y u, si se hubiera quitado la arista de v a t el grafo resultante no sería conexo, y por lo tanto no se podría llegar a t.



a) Grafo con cruces b) Grafo sin cruces

Figura 4.1: Justificación del grafo plano para encaminamiento por caras

4.2. Grafo de Vecindad Relativa

En esta sección se describirá la manera como se obtiene el *Grafo de Vecindad Relativa* o RNG *(Relative Neighbourhood Graph)* de acuerdo a [Toussaint98]. Un subgrafo plano de una red puede ser obtenido aplicando una prueba a cada par de nodos que estén dentro del rango de transmisión del otro. Sean u y v dos nodos tal que su distancia sea menor que el rango de transmisión R de la red. Considerar la región delimitada por la intersección de los círculos centrados en A y B, donde el radio de los círculos está determinado por la transmisión de las estaciones u y v, como se ve en la figura 4.2. Si no existe algún nodo en la intersección entonces la liga entre u y v es conservada. Sin embargo, si existiera un nodo en la región achurada o luna, la arista deberá ser removida. En la figura 4.2 la arista se preserva, ya que el nodo w se encuentra fuera de la luna. Por lo tanto, la arista $a = (u, v)$ se encuentra en $RNG(V)$ sí y solo sí *no* existe un nodo w tal que $d(u, w) < d(u, v)$ y $d(v, w) < d(u, v)$.

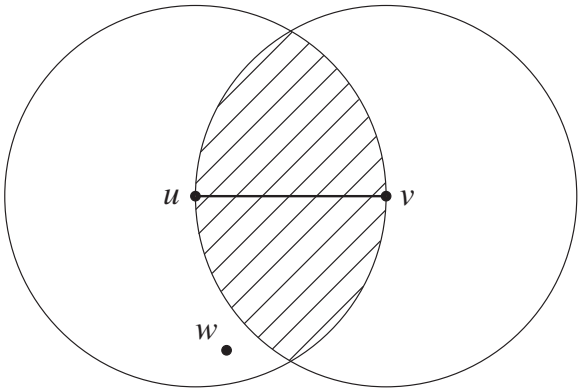


Figura 4.2: La prueba RNG para generar un subgrafo plano.

4.3. Grafo de Gabriel

Una de las pruebas más importantes para eliminar cruces en una red inalámbrica es la prueba de Gabriel [Gabriel69] la cual, de forma parecida a la prueba RNG, es aplicada a cada enlace de la red. La diferencia principal entre la prueba de Gabriel y la prueba anterior es que se considera una región de tamaño menor para la eliminación del enlace.

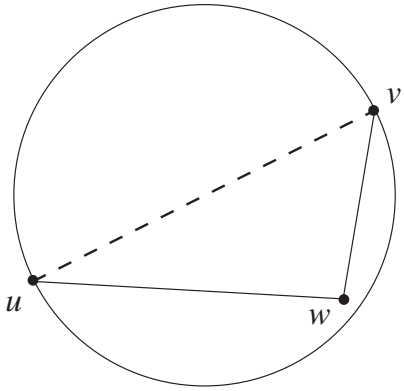


Figura 4.3: Eliminación del enlace uv por existir un nodo dentro del disco con diámetro uv.

Sean u y v dos nodos tal que la distancia uv es menor que el rango de transmisión R de la red. En la prueba de Gabriel, si no hay un nodo en el círculo con diámetro uv, entonces el enlace entre u y v es conservado. Sin embargo, si hay un nodo w en el círculo de diámetro uv, como se muestra en la figura 4.3, entonces los nodos u y v quitan su enlace directo. En particular, cuando a u se le solicita encaminar datos a v la tabla de encaminamiento en u lo mandará a través del nodo w, o alguno otro nodo similar, si existe más de un nodo dentro

del círculo de diámetro uv, del mismo modo se haría en el caso de encaminar datos de v hacía u.

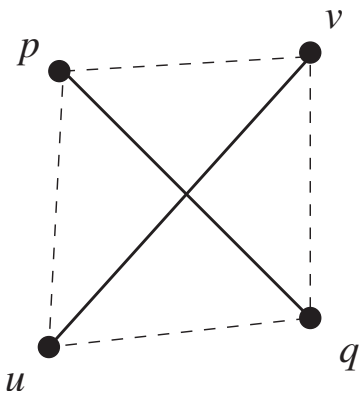


Figura 4.4: Prueba por Contradicción de la planaridad del Grafo de Gabriel

La ventaja de hacer el reencaminamiento de datos descrito previamente, es que el grafo resultante es un grafo plano. Para probar que el grafo de Gabriel es plano se puede hacer por contradicción, usando la prueba alterna que dice que la arista uv pertenece al grafo de Gabriel si $\angle uwv < \pi/2$ para cualquier punto w. Suponiendo que $uv, pq \in GG(S)$ y $uv \cap pq$, como se muestra en la figura 4.4, entonces $\angle puq < \pi/2$, $\angle pvq < \pi/2$, $\angle upv < \pi/2$, $\angle uqv < \pi/2$, por lo tanto, la suma de ángulos de $upvq < 2\pi$ cuando la suma de los ángulos internos de cualquier cuadrilátero es 2π.

Si se compara la prueba de Gabriel con la prueba RNG, en el primero de ellos se reduce la región de prueba y se crea un subgrafo plano que conserva algunos de los enlaces que podrían ser eliminados con la prueba RNG. Sin embargo, el grafo de Gabriel resultante podría todavía sufrir el efecto de salto múltiple.

Por ejemplo, si se considera un conjunto de nodos como el mostrado del lado izquierdo de la figura 4.5, todos los nodos son mutuamente alcanzables. Sin embargo, cuando se aplica la prueba de Gabriel se obtiene la configuración del lado derecho de la figura 4.5. Sin embargo se puede observar que los nodos u y v podrían alcanzarse el uno al otro directamente en un sólo salto, en vez de que ellos manden sus datos a través de una secuencia de varios nodos.

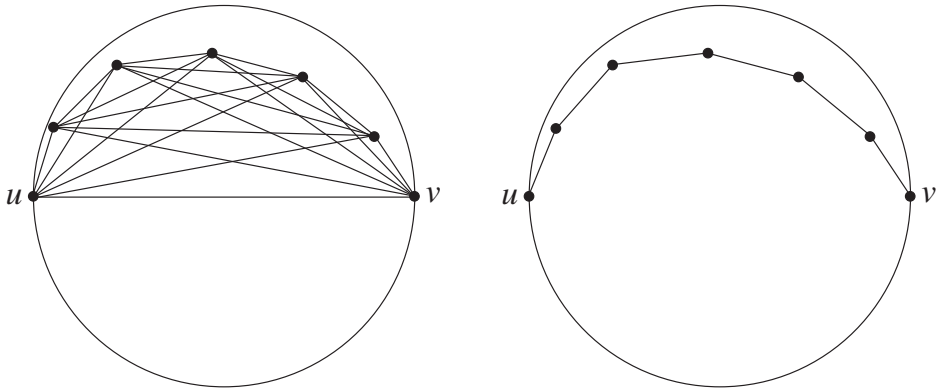


Figura 4.5: El efecto del salto múltiple se presenta cuando se elimina un enlace (el segmento de línea uv) aplicando la prueba de Gabriel.

4.4. Triangulación Local Unitaria de Delaunay

En un trabajo reciente, los autores en [Li03a] describen un algoritmo para construir localmente un subgrafo plano a partir del grafo del disco unitario. Para realizarlo usan la construcción de $LDel^{(1)}(V)$ como la base del algoritmo. Los autores muestran que al aplicar $LDel^{(1)}(V)$ se podría obtener un grafo no plano, por lo que se deberán quitar algunas aristas para hacerlo plano.

Se observa que el triángulo $\triangle uvw$ pertenece a una triangulación de Delaunay $Del(V)$ si el círculo que pasa por todos los vértices del triángulo $disco(u, v, w)$ no contiene ningún otro nodo de V en su interior, como se muestra en la figura 4.6.

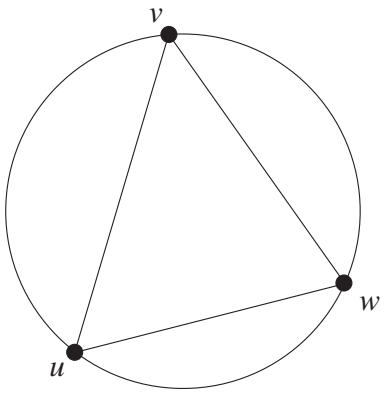


Figura 4.6: Aristas de una Triangulación de Delaunay ya que el $disco(u, v, w)$ no contiene puntos.

Algoritmo Localized Unit Delaunay Triangulation

1. Cada nodo inalámbrico u envía en modo broadcast un mensaje de anuncio con su identidad y localización a sus vecinos $N_1(u)$ y escucha de otros nodos en $N_1(u)$.

2. Asumiendo que el nodo u ha recibido toda la información de $N_1(u)$, calcula la triangulación de Delaunay $Del(N_1(u))$ con sus vecinos $N_1(u)$, incluyendo a u misma.

3. Para cada $arista(u, v)$ de $Del(N_1(u))$, sean $\triangle uvw$ y $\triangle uvz$ dos triángulos incidentes en la $arista(u, v)$, la $arista(u, v)$ es una arista de Gabriel si ambos ángulos $\angle uwv$ y $\angle uzv$ son menores que $\pi/2$. El nodo u marca todas las aristas de Gabriel y estas nunca serán removidas.

4. Para todos los nodos u, estos encuentran todos los triángulos $\triangle uvw$ de $Del(N_1(u))$ tal que todas las aristas de $\triangle uvw$ tienen longitud ≤ 1 unidad y $\angle wuv \geq \pi/3$. El nodo ordena las aristas de los triángulos en el sentido de las manecillas del reloj y manda un mensaje de propuesta (*proposal*) en modo broadcast. El mensaje de propuesta contiene el identificador *(ID)* de u, seguido por los IDs de los nodos v de acuerdo al $\triangle uvw$ anterior, y se dice que u esta proponiendo que el $\triangle uvw$ sea agregado a $LDel^{(1)}(V)$. Entonces el nodo u escucha los mensajes de otros nodos en $N_1(u)$.

5. Cada nodo u después de recibir el mensaje de propuesta de v, revisará si v está proponiendo agregar $\triangle uvw$ y $\triangle uvz$ al $LDel^{(1)}(V)$. Entonces el nodo u revisará si algún otro triángulo propuesto por otros nodos pertenece a $Del(N_1(u))$. Después de calcular todos los triángulos, u manda en modo broadcast un mensaje de aceptación (*accept*) el cual contiene los IDs de u, la lista de vértices v en el sentido de las manecillas del reloj tal que u calculada para algún w del $\triangle uvw$ pertenece a $Del(N_1(u))$, y un bit indicando si dos consecutivos v,v' forman un triángulo $\triangle uvv'$ de $Del(N_1(u))$.

6. El nodo u agrega las aristas uv y uw a su conjunto de aristas incidentes si el triángulo $\triangle uvw$ está en $Del(N(u))$ y ambos v y w han mandado un mensaje de propuesta o de aceptado para el triángulo $\triangle uvw$.

   Se tiene que las aristas agregadas por todos los vértices en este paso forman $LDel^{(1)}(V)$. En este momento, la fase de planarización inicia.

7. Cada nodo u enviará en modo *broadcast* un mensaje de revisión (*check*), incluyendo su ID y la posición de todos los vértices v, tal que $uv \in LDel^{(1)}(V)$.

8. El nodo u, usando los mensajes de revisión de sus vecinos, revisa todos los triángulos locales de Delaunay $\triangle uvw$ para saber si hay algún nodo dentro del círculo circunscrito al triángulo $\triangle uvw$. Si tal nodo es encontrado, entonces u removerá el triángulo $\triangle uvw$. Entonces el nodo u entonces ordena en el sentido de las manecillas del reloj las aristas uv las cuales ni son del grafo de Gabriel o pertenecen al triángulo de $LDel^{(1)}(V)$ que no fue descartado.

9. Posteriormente todos los nodos u enviarán un mensaje de presente (*alive*) incluyendo su ID, seguido de los IDs de los nodos incidentes a u y un bit entre dos nodos v y w diciendo si $\triangle uvw \in LDel^{(1)(V)}$ y el $\triangle uvw$ no fue descartado.

10. El nodo u conserva la arista uv en su conjunto de aristas incidentes si es una arista de Gabriel, o hay triángulo $\triangle uvw \in LDel^{(1)}(V)$ que aparezca en los mensajes alive de u, v y w. Esto implícitamente crea el grafo $PLDel(V)$ y termina el algoritmo.

Los autores amplían su trabajo en [Wang03], donde ellos encuentran un grafo de expansión plano con valencia acotada usando la estructura de Yao [Yao82] con el algoritmo descrito en la presente sección. Ellos también describen en [Li03b] un algoritmo para generar triangulaciones de Delaunay bajo el grafo subyacente en una red scatternet [1] con tecnología Bluetooth.

4.5. El Grafo de Morelia

El algoritmo del Grafo de Morelia [Boone04] crea un subgrafo plano. La prueba empleada crea un subgrafo que conserva los enlaces largos respecto a los enlaces cortos en promedio respecto al grafo de Gabriel, adicionalmente tiene la ventaja de que ayuda a reducir el impacto del *efecto multisalto*, en particular cuando la red tiene áreas dispersas.

Los características del algoritmo son:

- Conservar los enlaces largos con la finalidad de tener un cantidad reducida de saltos. Cuando se tienen muchos saltos se podría requerir procesamiento adicional en los nodos por que se incrementa la probabilidad de falla.

[1] Scatternet es un grupo de *piconets* que comparten al menos un dispositivo bluetooth común. Una *piconet* es una red de dispositivos conectados en modo ad-hoc usando la tecnología Bluetooth.

- Eliminar los cruces para poder obtener un grafo plano usando pruebas locales y donde el grafo resultante sea un grafo que abarque la red original.

Se debe mencionar que el primer objetivo no ayuda a minimizar el consumo de energía, ya que se requiere gastar mayor cantidad de energía para alcanzar nodos lejanos respecto a los nodos cercanos.

La prueba que se aplica para obtener el grafo de Morelia con el objetivo de conservar los enlaces largos y los eliminar los cruces entre los enlaces, es al igual que la prueba *RNG* y la prueba de *Gabriel* aplicada con cada enlace de la red. La diferencia con estas dos últimas pruebas es que el criterio usa la información de los enlaces vecinos para decidir si se conserva el enlace o si se remueve. Los autores definen un círculo interno con diámetro uv para un par de nodos u, v, tal que la distancia es menor que el rango de transmisión R de la red, y dos círculos correspondientes a los radios de transmisión de u y v como se aprecia en la figura 4.7.

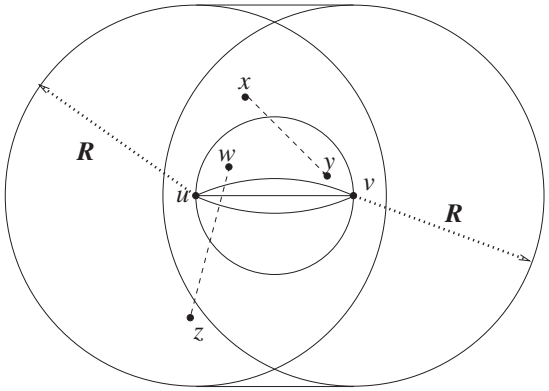


Figura 4.7: Prueba de Morelia: eliminación de la arista uv debido a la arista wz.

Para aplicar el conjunto de reglas que se definen para el Grafo de Morelia, el círculo con diámetro uv es dividido en 4 regiones usando el diámetro uv y los arcos trazados entre uv con centro en la intersección de los círculos externos, como se muestran en la figura 4.7. También se consideran dos regiones que no son alcanzables ni por u, ni por v, que se encuentran fuera de los círculos externos y limitados por las tangentes de los círculos externos. Un conjunto de 5 reglas decide si un enlace debe ser conservado o borrado, las cuales se pueden resumir en que la arista uv será removida cuando exista un enlace wz, tal que uno de los extremos o ambos se encuentren dentro del círculo de diámetro uv y que intercepte al enlace uv. En la figura 4.7 la arista uv es removida ya que la arista wz la cruza

y el punto w está dentro de la circunferencia de diámetro uv; por otra parte, la arista xy no puede remover a las arista uv ya que no cruza la arista uv, a pesar de que y se encuentra dentro del círculo uv.

En la figura 4.8 se muestra como el Grafo de Morelia elimina el efecto multisalto al conservar la arista uv. El lado izquierdo muestra el enlace uv y los nodos dentro del círculo con diámetro uv y en el lado derecho se observan las aristas que fueron conservadas al aplicar el Grafo de Morelia, la arista uv no es eliminada ya que no hay cruce con ninguna de las otras aristas presentes.

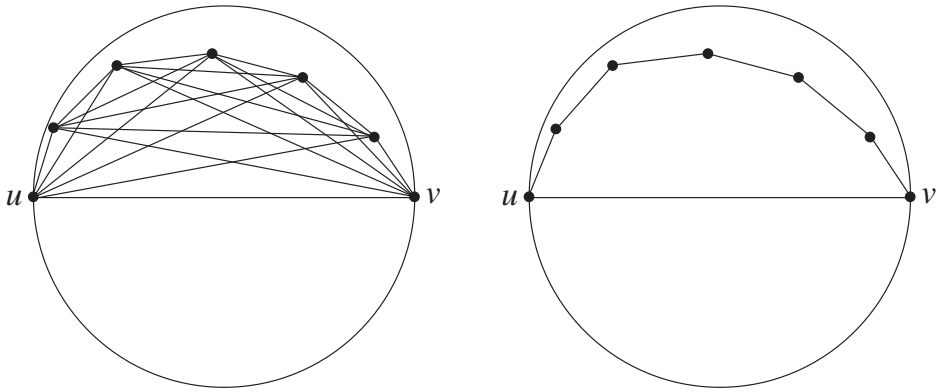


Figura 4.8: Eliminación del efecto multisalto al ser conservada la arista uv al aplicar el Grafo de Morelia.

4.6. Resumen

En este capítulo se han revisado algoritmos que obtienen un grafo plano localmente, realizando algunas pruebas en donde se involucran nodos, como en el grafo de Gabriel, o bien, se usan aristas, como en el grafo de Morelia. Estas pruebas eliminan algunas aristas del modelo del grafo del Disco Unitario, con lo cual aparecen algunas situaciones no deseadas, como el efecto multisalto. Dependiendo de la métrica empleada puede ser que con algunas aristas eliminadas se hubiera tenido un mejor desempeño para realizar el encaminamiento, por lo tanto es deseable que la red subyacente sea representada por algún grafo que sea plano, como se muestra en el capítulo 6, y que conserve todo el conjunto de aristas del grafo unitario.

La importancia que tienen los grafos planos es que permiten garantizar la entrega

de paquetes, como se vio en el capítulo anterior pero, se desea conocer si existe alguna estructura en donde se garantice la entrega de paquetes y sea libre de ciclos. La estructura que permite realizar lo anterior es mostrada en el capítulo 6, en donde además se indica la manera como se debe realizar el encaminamiento voraz.

Capítulo 5

El Dual del Grafo Plano

Este capítulo presenta el dual del grafo plano y la manera como se puede hacer el encaminamiento, el cual resulta sorprendentemente sencillo. El dual del grafo plano es un grafo de proximidad, ya que es posible ir de un origen a un destino siempre y cuando el grafo sea conexo. Para la construcción del grafo plano se requiere conocer información de las caras vecinas, la cual puede ser encontrada en forma local. En el capítulo 4 se vio la manera de construir un grafo plano localmente.

5.1. Dual del Grafo Plano

Si se recorre la cara de algún grafo plano sin marcar o recordar el nodo inicio se generará un ciclo. Por lo tanto, es importante que se sepa en donde inicio el recorrido para no estar en la situación anterior y poder detectar si se puede llegar al destino. Una manera de avanzar en la cara hacia el destino, consiste en hacer un cambio hacia aquella cara que nos aproxime al destino, respecto al nodo donde se inició el recorrido de la cara. Se pueden considerar a las aristas o los vértices para realizar el cambio. Se observa en la figura 5.1 que las aristas siempre pertenecen a dos caras, en cambio los vértices pueden estar presentes en más de dos caras, como sucede con algunos de los nodos interiores. Por lo tanto, es preferible usar como referencia de cambio a las aristas en vez de los nodos, debido a que no involucran más que dos caras.

El cambio entre caras usando las aristas puede hacerse de manera voraz de tal forma que partiendo de algún origen se pueda llegar hacia el destino, si el grafo dado es conexo. En el caso de que llegue a fallar el recorrido en una dirección, bastará hacer el

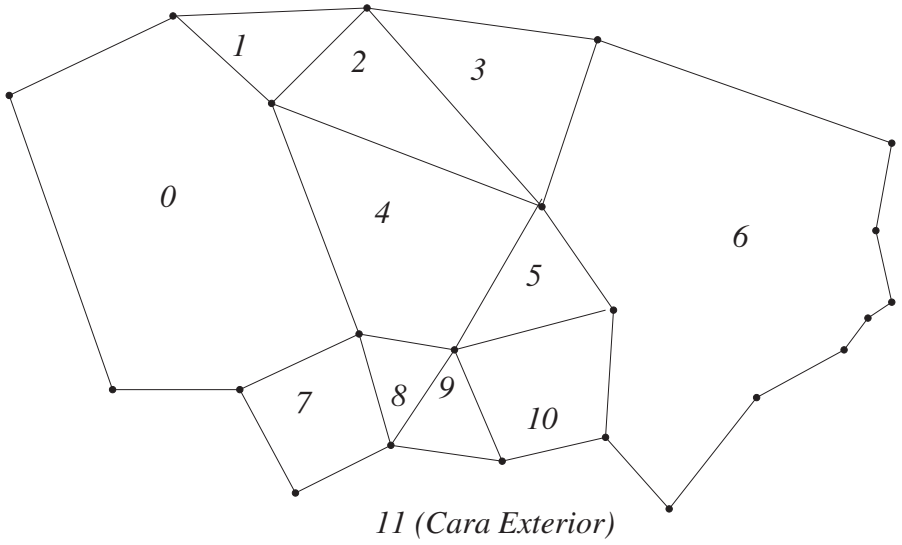


Figura 5.1: Grafo plano con doce caras.

recorrido en el sentido contrario. Observar que al realizar el cambio en la dirección del recorrido nos coloca en otra cara. En la figura 5.2 se muestra que el recorrido 1 falla para llegar del nodo s al nodo t pero, al hacer un cambio de dirección, como se muestra en el recorrido 2, se puede llegar al nodo t.

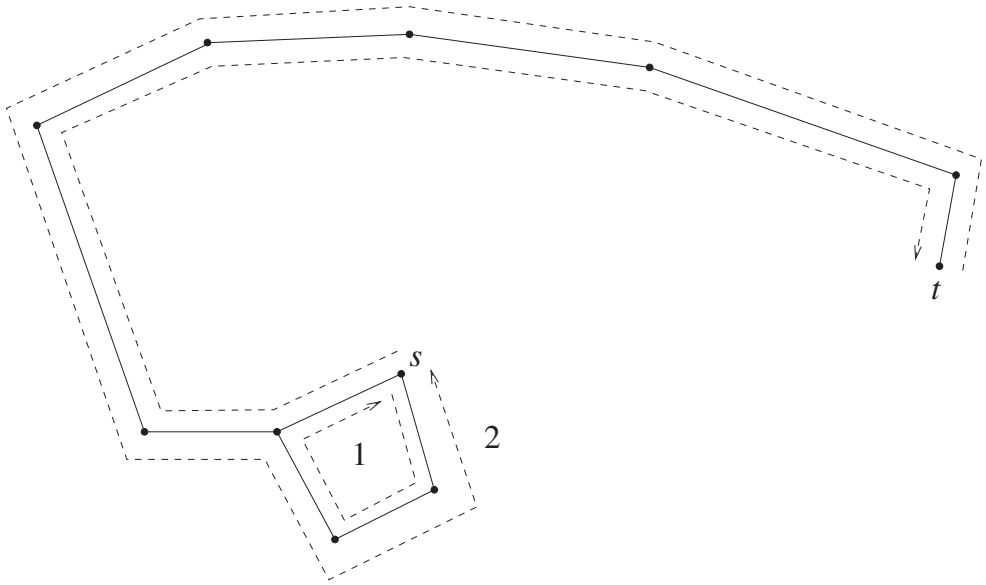


Figura 5.2: Recorridos en un grafo plano para llegar de s a t.

5.2. Construcción del Dual del Grafo Plano

Para la construcción del dual del grafo plano se realiza de la siguiente forma:

1. Cada cara es representada por un punto virtual, a excepción de la cara exterior.
2. Se colocan aristas virtuales entre los puntos virtuales cuando tienen una o mas aristas que son vecinas entre dos caras.
3. Las caras que son vecinas a la cara exterior quedan interconectadas entre ellas.

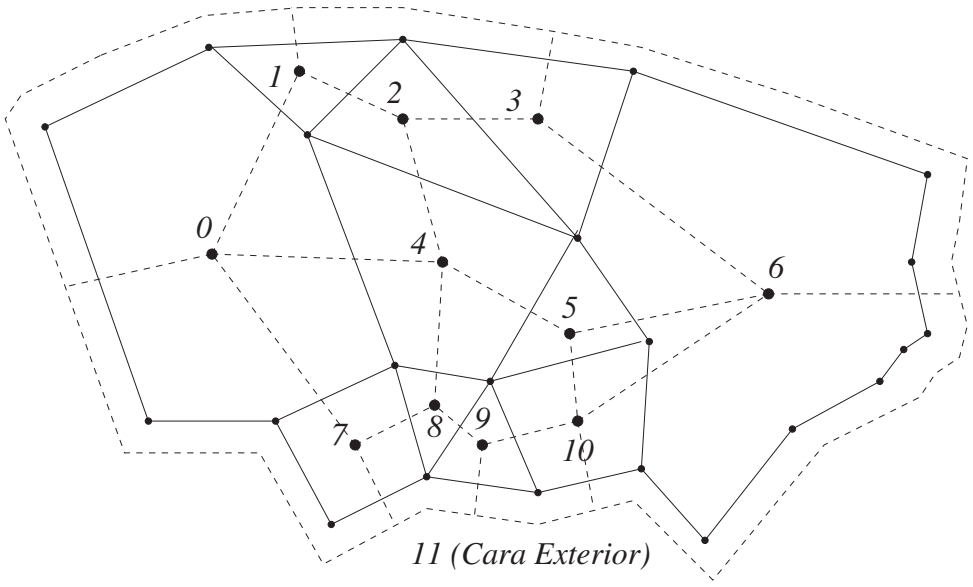


Figura 5.3: Dual del Grafo Plano.

Con la construcción anterior el grafo resultante que se obtiene se muestra en la figura 5.3, en el cual se puede ir de la cara 0 a la cara 6 de dos formas principalmente, usando las intercomunicación que existe entre ellas gracias a que son vecinas de la cara exterior, o bien, empleando las caras internas 4 y 5 que se encuentran entre ellas.

5.3. Encaminamiento en el Dual del Grafo Plano

Empleando el dual del grafo plano es posible encaminar desde cualquier nodo a otro, haciéndolo desde la cara donde se encuentra el nodo origen a la cara del nodo destino. La manera de hacerlo es simplemente cambiándose a alguna cara que se encuentre

más próxima al nodo destino, por lo tanto, el tipo de encaminamiento será voraz. Para seleccionar una arista para realizar el cambio entre caras se tienen los siguientes casos, que dependen de que sea vecina a la cara exterior:

a) Si la cara es interna, o no es vecina de la cara exterior, entonces la arista seleccionada por la cara se encuentra solamente en la cara.

b) Si la cara es vecina a la cara exterior, entonces las arista para realizar el cambio se encuentra en la misma cara, o bien, se encuentra en la cara exterior. Para este caso se deben realizar dos recorridos, uno de ellos es en la cara interna, y el otro en la cara exterior.

En el primer caso el encaminamiento no falla, ya que al ser una cara interna, habrá por lo menos una cara vecina que se encuentre más próxima del nodo destino. En el segundo caso el encaminamiento dentro de la cara puede fallar pero, al considerar el segundo recorrido se puede recuperar. La única forma como podría fallar el segundo caso, es que el grafo sea no conexo.

Se muestra a continuación la forma como se realiza el encaminamiento por caras y el encaminamiento en dual del grafo plano entre el nodo s al nodo t, para mostrar que el encaminamiento por caras es un caso particular del encaminamiento en el dual del grafo plano. En el encaminamiento del dual del grafo plano se consideran varias caras para hacer el cambio entre caras, y en el encaminamiento por caras sólo una, que es aquella donde el punto de intersección se encuentra más cerca del destino.

El recorrido hecho por el algoritmo de encaminamiento por caras se muestra en la figura 5.4 en donde se emplea la línea st para medir el progreso hecho.

En la figura 5.5 se muestran las aristas virtuales del dual del grafo plano que sirvieron como referencia para recorrer el grafo.

Se debe considerar que para la construcción del dual del grafo plano, se requiere tener en los nodos información de la caras vecinas para evitar recorrer toda la cara, cada vez que se use el dual del grafo plano.

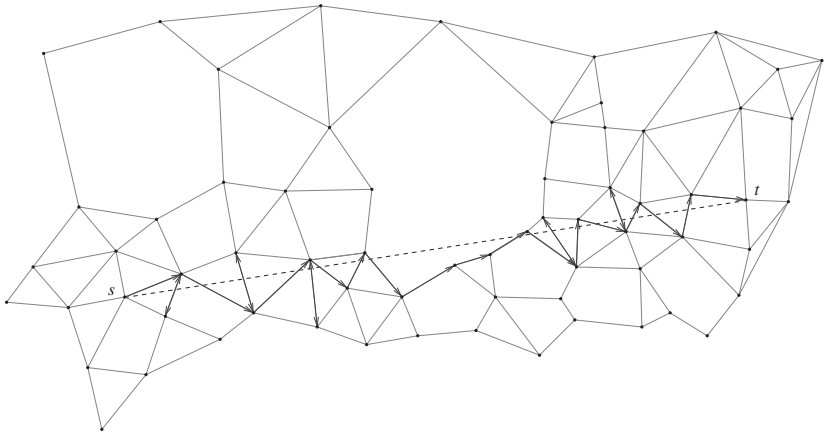


Figura 5.4: Recorrido de aristas al usar el encaminamiento por caras.

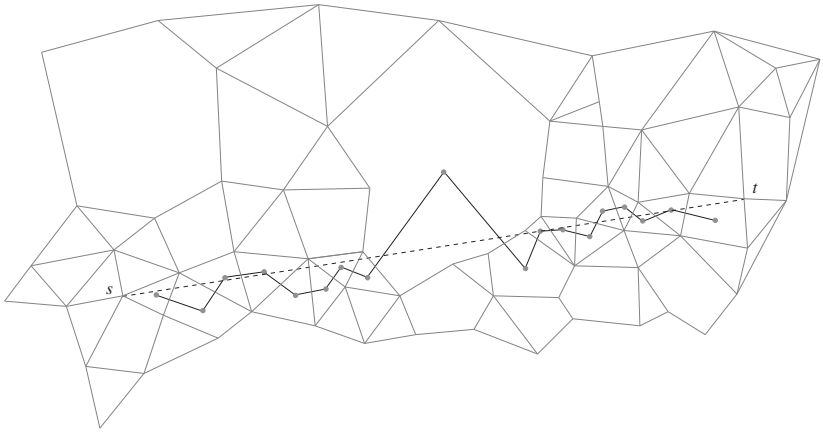


Figura 5.5: Encaminamiento en el dual del grafo plano del nodo s al nodo t.

5.4. Extensión del Dual del Grafo Plano al Grafo Virtual

Como se había comentado previamente para construir el dual del grafo plano, se requiere información de las aristas de la cara, lo cual puede considerarse global si por ejemplo una red esta formada por caras muy grandes. Lo anterior tiene algunos inconvenientes como

el incremento en los mensajes de control que permitan conocer las caras vecinas. Por lo tanto, se desea una construcción similar al dual del grafo plano que pueda ser construido localmente. Una manera de lograr lo anterior, es teniendo caras virtuales tal que cualquier par de nodos que se encuentren en la cara se puedan ver o alcanzar. Las caras virtuales existirán sólo si contienen algún nodo de la red.

La forma que puede tener la cara o celda puede ser muy variada, en particular, nos inclinamos por un sólo tipo de cara para todo el plano y que la cara sea un polígono regular, por la facilidad para ubicar puntos dentro de las celdas. Los únicos polígonos que cumplen con lo anterior son los triángulos, los cuadrados y los hexágonos.

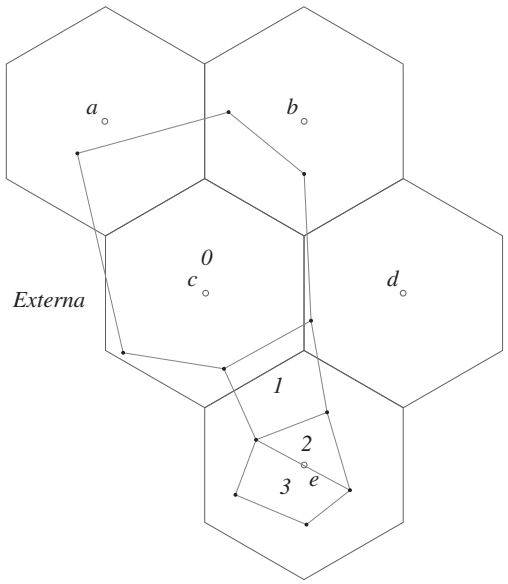


Figura 5.6: Superposición de caras hexagonales a un grafo plano.

En la figura 5.6 se muestra una superposición con caras hexagonales virtuales a un grafo plano, en el cual se observa que la cara 0 queda repartida en las caras hexagonales virtuales a, b, c y d, al igual que la cara 1 la cual se encuentra entre las caras c, d y e, por otra parte las caras 2 y 3 se encuentran en la cara hexagonal virtual e. Los casos que se presentan al superponer las caras virtuales son:

a) Una cara o varias del grafo plano queda contenida en una cara virtual.

b) Una cara del grafo plano queda contenida en varias caras virtuales.

Para el caso a) se tiene la ventaja de que una cara o varias del grafo plano se

encuentra en una celda, con lo cual no se necesita recorrer la cara, basta moverse vorazmente hacia un mejor punto, además de que se reduce el número de caras. En el caso b) se agregan más caras que las que se tienen en el grafo plano; pero con la ventaja de que el contenido es local debido al tamaño escogido de la cara virtual.

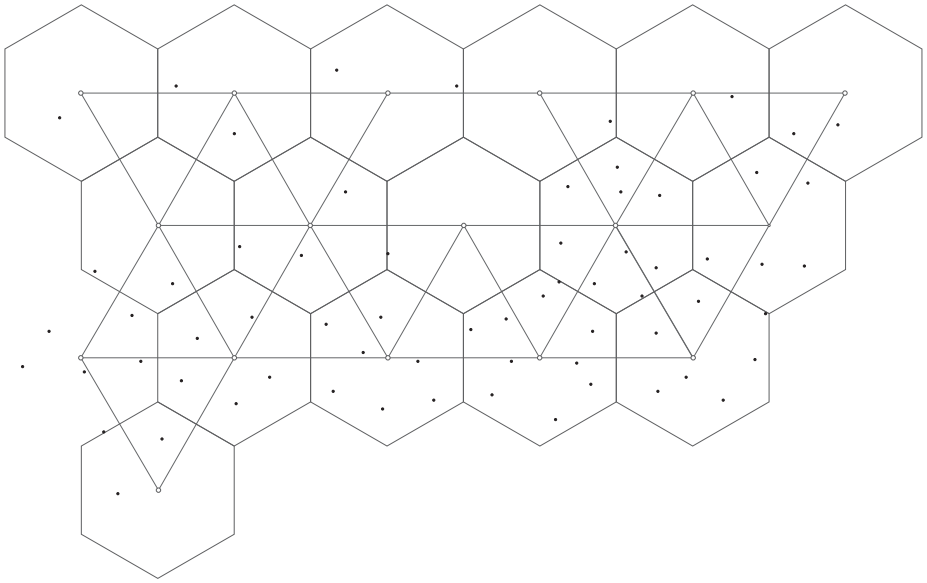


Figura 5.7: Grafo Virtual Dual con celdas hexagonales.

Aplicando un criterio similar para obtener el dual del grafo plano se puede derivar un nuevo grafo, el cual denominaremos el "Grafo Virtual". Las caras virtuales son reemplazadas por nodos virtuales, y se coloca una arista virtual entre los nodos virtuales si existe un par de nodos que sean alcanzables y se encuentren en celdas distintas virtuales, por lo tanto, para obtener el grafo virtual es suficiente con saber quienes son los nodos vecinos que son alcanzables fuera de la celda, es decir, quienes son los vecinos de los vecinos, en la figura 5.7 se muestra el grafo virtual.

Una de las desventajas derivadas de la localización hecha, es que en el grafo virtual no basta con encaminar vorazmente para poder mandar un paquete de un nodo a otro, ya que algunas caras del grafo plano no quedan contenidas en una cara virtual, sino segmentadas en varias, por lo que no es posible pasar a la siguiente cara, sin haber recorrido previamente las caras virtuales, por ejemplo en la figura 5.6 se muestra que no es posible ir de la cara 0 a la cara 1. Usando la información sólo de la cara virtual a, se puede hacer si usamos la cara virtual c o d, lo cual si puede hacerse en el dual del grafo plano, al existir una arista

entre la cara 0 y la cara 1. Para solucionar lo anterior, se debe usar en algunos casos en el grafo virtual el encaminamiento por caras.

5.5. Resumen

Se concluye que teniendo información global con el dual del grafo plano se puede encaminar vorazmente; pero si se tiene sólo información local, por ejemplo como en el grafo virtual, se tendrán situaciones en donde el encaminamiento voraz fallará, por lo que deberá proveerse una manera de que se recupere, siendo una de éstas técnicas el encaminamiento por caras. En el siguiente capítulo se verá la forma de construir el grafo virtual.

Capítulo 6

El Grafo Virtual

6.1. Introducción

En este capítulo se presentan los algoritmos que crean los *grafos virtuales planos*. Uno de los objetivos es mejorar el desempeño del encaminamiento de paquetes que se hace con el Grafo de Morelia [Boone04] y el Grafo de Gabriel. Una de las características importantes que debe ser mencionada del Grafo Virtual, es el hecho de que el Grafo Virtual no contiene aristas, ni nodos de la red subyacente. Sin embargo, el Grafo Virtual que se obtiene se deriva de los enlaces presentes. El Grafo Virtual se encuentra formado por un conjunto de nodos virtuales y enlaces virtuales, donde un nodo virtual representa a uno o más nodos reales, y una arista virtual representa una o varias aristas reales. El Grafo Virtual puede ser visto como una representación ligera de la red, en donde el encaminamiento se hace usando el conjunto de nodos y enlaces reales teniendo como referencia al grafo virtual. Con lo anterior, se tiene la posibilidad de escoger el enlace real que más convenga dependiendo de lo que se quiere optimizar.

En este capítulo se describen cuales son las metas que se quieren alcanzar al usar el Grafo Virtual y la manera como deberá ser construido el Grafo Virtual con cada variante. Se concluye presentando un conjunto de simulaciones diseñadas e implementadas con la finalidad de mostrar lo siguiente:

- El Grafo Virtual con cualquier tipo de métrica puede ser usado en una red ad-hoc, ya sea la distancia euclidiana, el número de transmisiones, el consumo de energía o alguna otra. Los otros grafos presentan un buen desempeño con algún tipo de métrica,

aunque tienen casos con los que presentan un resultado poco satisfactorio.

- El algoritmo de encaminamiento por caras con las distintas métricas usadas presenta una mejoría importante.

6.2. Metas

Las dos metas importantes que deberán ser tenidas en cuenta al momento de obtener el grafo virtual plano a partir de la red inalámbrica son:

- Usar todos los enlaces.
- Ausencia de cruces.

Como se describió en el capítulo 4 los algoritmos que obtienen un grafo plano local omiten algunas aristas del grafo del disco unitario, pudiendo ser que algunas de las aristas removidas permitan tener un mejor desempeño con alguna métrica respecto a las aristas del subconjunto obtenido por la planarización. Por lo anterior es importante que se puedan emplear todo el conjunto de aristas del grafo del disco unitario. Para lograrlo el grafo virtual se sobrepone a la red física subyacente, de tal forma que la represente de una forma sencilla. El grafo virtual será una referencia para el encaminamiento, en donde la manera de seleccionar el próximo vecino dependerá de la métrica, y no de algún subconjunto de aristas.

Respecto a la segunda meta anterior, las pruebas deberán ser aplicadas localmente por los nodos individuales, ya que en muchos casos no existe información global disponible acerca de la red, a la vez que los nodos están modificando la topología de la red. La forma como se considera a una arista virtual está basada en el requerimiento de que no existan cruces entre aristas.

Se debe recordar que se requiere de un grafo plano si se quiere encaminar usando el encaminamiento por caras pero, en el caso de que se use el encaminamiento GFG, el algoritmo voraz no requiere de algún grafo plano para encaminar, el método de encaminamiento si requiere de un grafo plano y es usado para recuperar al algoritmo voraz cuando llega a fallar.

6.3. Particiones Regulares del Plano

Para poder sobreponer el grafo virtual al grafo del disco unitario, se divide el plano en regiones. Las maneras como se puede realizar la división son:

- Usando un conjunto uniforme de figuras geométricas que cubran el plano, por ejemplo con rombos.
- Con alguna combinación finita de figuras planas, por ejemplo, usando cuadrados y octágonos.
- Realizando una partición regular, es decir, empleando algún polígono regular. Los únicos polígonos regulares que se pueden emplearse son los triángulos, los cuadrados y los hexágonos.
- Con una partición no regular, por ejemplo usando romboides.

En este trabajo se explora lo que se puede realizar usando particiones regulares o teselaciones (*regular tessellations*), las cuales se muestran en la figura 6.1.

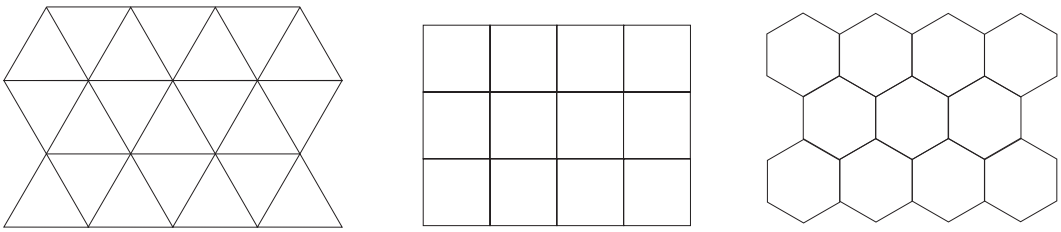


Figura 6.1: Particiones regulares del plano (*regular tessellations*).

Al emplear la partición regular dependiendo de la topología de la red, algunas regiones o celdas estarán activas debido a que contienen algún nodo de la red, y otras estarán apagadas por no contener ningún nodo. Cada celda puede ser representada por el centro del polígono equilátero, el cual a su vez será el nodo virtual. Para poder obtener el grafo virtual se hará necesario conectar algunos de los nodos virtuales, lo cual estará en función de la conexión que tengan los nodos en el grafo del disco unitario.

Otro aspecto que debe ser tomado al momento de colocar la malla de polígonos es el tamaño de los polígonos. Para el tamaño del polígono se pueden tomar los siguientes criterios:

a) El tamaño del polígono es mayor que el rango de transmisión R, por lo que no se garantiza que un nodo de la red alcance cualquier punto que se encuentra dentro del polígono, teniendo entonces que realizar tareas adicionales para decidir la conexión de la celda.

b) El tamaño del polígono contiene al rango de transmisión, es decir, cualquier par de puntos que se encuentren en el perímetro del polígono se encuentran a lo más el rango de transmisión R. A diferencia del caso anterior, cualquier nodo dentro del polígono alcanza a cualquier otro nodo que se encuentre en el polígono.

c) El rango de transmisión es mayor respecto al par de puntos mas alejados que se encuentran en el perímetro del polígono, con lo anterior se logra que un nodo dentro de alguna celda alcance un número mayor de nodos que se encuentren en las celdas vecinas. Un caso interesante es cuando el rango de transmisión es igual al par de puntos mas alejados que se encuentran en el perímetro de dos polígonos vecinos por su lado, ya que algún nodo que se encuentre en alguna celda podrá ver a todos los nodos de sus vecinos.

En la figura 6.2 se muestran los distintos casos comentados previamente considerando una partición regular del plano usando hexágonos.

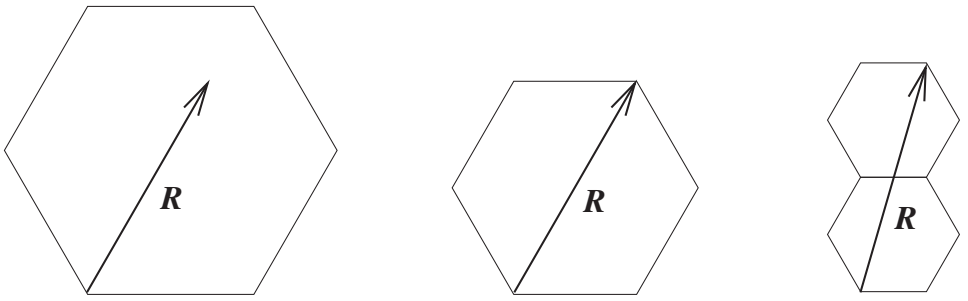


Figura 6.2: Variación del tamaño del polígono dependiendo de la cobertura deseada de los nodos dentro de una celda.

6.4. Construcción con Triángulos

Se detalla a continuación la forma como se construye el grafo virtual con una partición regular de triángulos equiláteros. El tamaño del triángulo se escogió tal que un

lado del triángulo sea igual al radio de transmisión, por ser la mayor distancia que existe en un triángulo.

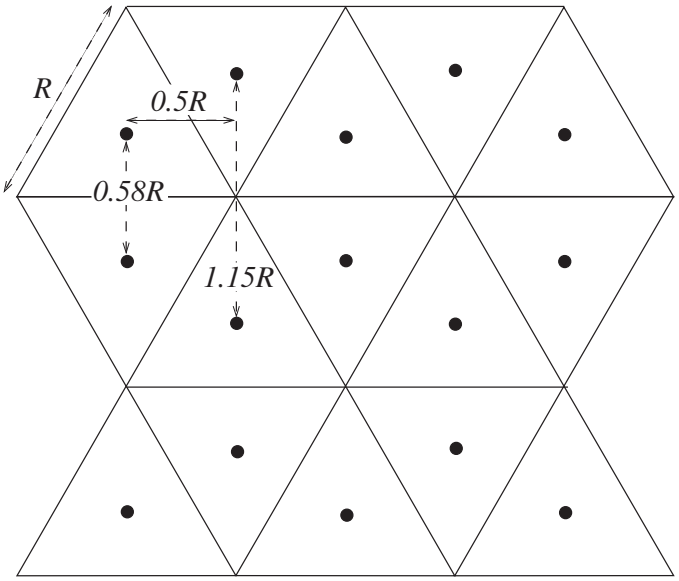


Figura 6.3: Distancias entre nodos virtuales con teselación triangular.

Como se había comentado en la sección previa, cada nodo de la red será asociado con alguna celda, y cada celda estará representada por un nodo virtual. El nodo virtual es el centro de cada triángulo, por lo que entonces cada centro tendrá un subconjunto de i vértices V. En la figura 6.3 se muestran las distancias horizontales y verticales aproximadas al considerar la longitud de los lados del triángulo equilátero igual al radio de transmisión. La separación horizontal entre un par de nodos virtuales adyacentes es $R/2$; la separación vertical varía ya que para un par de triángulos con lados comunes la distancia es $R/\sqrt{3}$, y la separación vertical si el par de triángulos se oponen por su vértice común es $2R/\sqrt{3}$.

La ubicación de la malla de triángulos se realiza de forma arbitraria en el plano cartesiano como se muestra en la figura 6.4. En la misma figura también se muestra la forma como se identifican a los centros de cada celda usando un par ordenado (x, y). Observar que el centro del primer triángulo $(0, 0)$ no está en el origen del sistema de coordenadas pero, si el vértice superior del triángulo con la etiqueta $(0, 0)$.

Las coordenadas del centro $c(x_c, y_c)$ de algún triángulo se determinan empleando la posición de cada triángulo (x, y) dentro de la malla y el radio de transmisión R como sigue:

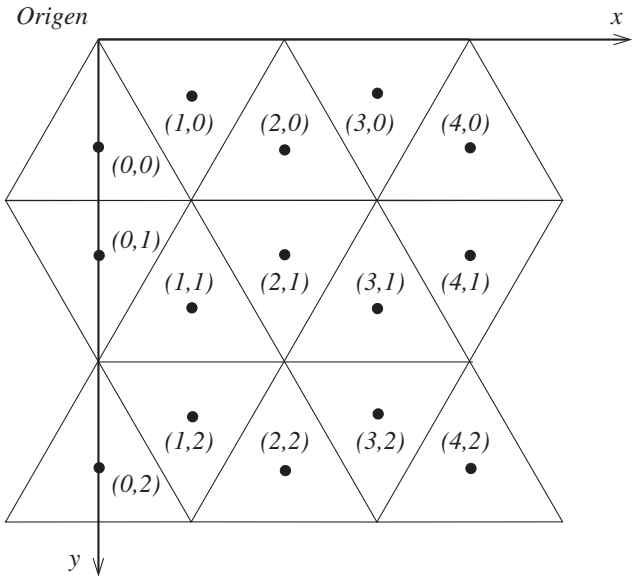


Figura 6.4: Ubicación de Triángulos en el Plano

$$x_c \leftarrow \frac{Rx}{2} \tag{6.1}$$

$$\begin{aligned} \text{si} \quad & (x+y) \text{ mód } 2) = 0 \\ \text{entonces} \quad & y_c \leftarrow \sqrt{3}R(y+2/3)/2 \\ \text{sino} \quad & y_c \leftarrow \sqrt{3}R(y+1/3)/2 \end{aligned} \tag{6.2}$$

Es decir, a la ordenada del centro del triángulo se le agregan $2/3$ si el triángulo apunta hacia arriba, como por ejemplo al triángulo $(0,0)$ o al $(3,1)$. Si el triángulo apunta hacia abajo entonces se agrega $1/3$, como por ejemplo a $(1,0)$ o $(3,2)$. El triángulo apunta hacia arriba si la suma de las coordenadas de su posición es par, y apunta hacia abajo si la suma de las coordenadas de su posición es impar.

Como cada nodo será identificado con algún triángulo, se indicarán las coordenadas (x,y) de la celda en la cual se encuentra, por ejemplo $(2,1)$ o $(3,4)$. La manera de obtener el par ordenado para un nodo con coordenadas $p(x_n, y_n)$ es:

$$x \quad \leftarrow \quad \text{truncar}\left(\frac{x_n}{2R}\right)$$

$$y \leftarrow \text{truncar}\left(\frac{2y_n}{\sqrt{3}R}\right)$$

El valor de x debe ser ajustado ya que los triángulos vecinos por su lado izquierdo y derecho, no comparten una arista "vertical". Una manera sencilla de saberlo, es usando la distancia euclidiana del punto $p(x_n, y_n)$ a la celda (x, y) y a la celda vecina de la derecha, es decir, a $(x+1, y)$ como se muestra a continuación:

$$\begin{aligned} d(p,c) &\leftarrow \sqrt{(x_n - X_c)^2 + (y_n - Y_c)^2} \\ d(p,c_{vec}) &\leftarrow \sqrt{(x_n - X_{c_{vec}})^2 + (y_n - Y_{c_{vec}})^2} \\ \text{si} &\quad (d(p,c_{vec}) < d(p,c) \\ \text{entonces} &\quad x \leftarrow x+1 \end{aligned}$$

Las coordenadas de c y c_{vec} se determinan con las expresiones dadas en 6.1 y 6.2.

6.4.1. Colocación de Aristas en el Grafo Virtual

Se detalla a continuación los criterios usados para colocar las aristas virtuales entre los nodos virtuales con el fin de crear el Grafo Virtual.

Debido al tamaño del triángulo que se escogió (los lados son iguales al radio de transmisión), un nodo de la red tiene posibilidades de alcanzar otras celdas adicionales a las celdas que le son vecinas por su lado. En particular si el nodo se encuentra próximo a un vértice puede alcanzar celdas que se encuentran en los triángulos vecinos por su vértice, caso a) en la figura 6.5; si un nodo se encuentra cercano al punto medio de algún lado del triángulos, se alcanza otro conjunto de celdas como se muestra en la figura 6.5. Para la partición regular que se discute en esta sección el total de triángulos alcanzables para algún triángulo es de 24 triángulos, por lo que potencialmente se pueden tener hasta 24 aristas para algún nodo virtual, lo anterior se limita por el hecho de que el grafo virtual debe ser plano, es decir las aristas virtuales no deben presentar cruces.

Para realizar las pruebas con las celdas que son alcanzables desde t, las celdas a revisar fueron divididas en dos grupos, el grupo interno I formado por 12 celdas que son las celdas vecinas a t ya sea por su lado, o por su vértice. El otro grupo esta formado por las externas E y son el resto de celdas. En la figura 6.6 se muestran los dos grupos y la numeración arbitraria que se hizo para cada grupo.

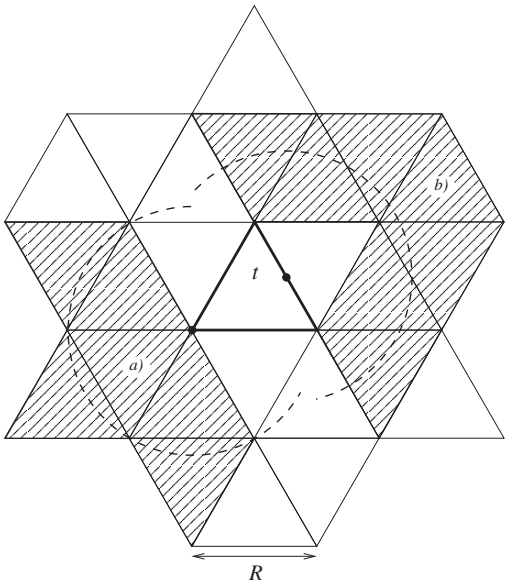


Figura 6.5: Celdas alcanzables desde t para una triangulación.

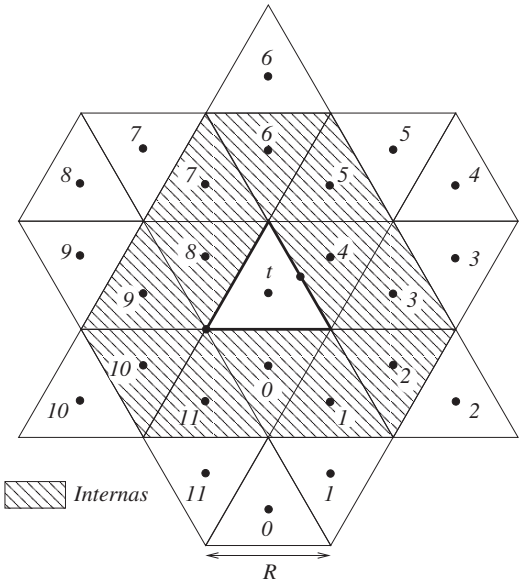


Figura 6.6: Etiquetado de celdas internas y externas alcanzables desde t.

Se muestran a continuación los algoritmos empleados para agregar aristas virtuales entre un triángulo t y las celdas que le son alcanzables. El algoritmo 1 recibe las posiciones de las celdas internas y las celdas externas. La función $existeArista(i, j)$ indica si existe un par de puntos que son mutuamente alcanzables y que se encuentran en celdas distintas y

la función $agregaArista(i, j)$ pone una arista entre i y j. El algoritmo 1 llama a su vez al algoritmo 2 que le indica si existen enlaces entre la celdas comprendidas por la celda t y las que se le oponen por su vértice, por ejemplo, cuando se revisa si hay enlaces en las celdas que se encuentran entre las celdas t e I_2, se revisan dos bloques, el primer bloque de enlaces es (I_{11}, I_3), (I_0, I_3) e (I_0, E_3) y el segundo bloque es (I_1, I_4), (I_1, I_5) y (E_1, E_4). Si alguno de los enlaces existiera, entonces no se pone el enlace entre t e I_2 porque se genera un cruce.

El procedimiento para revisar las celdas externas E a t se muestra en los algoritmos 3 y 4

En la figura 6.7 se observan las aristas que aparecen cuando solamente se revisan las celdas vecinas por su lado para alguna celda, es decir, cuando solo se toman en cuenta las $0, 4$ y 8 de acuerdo a la designación hecha en la figura 6.6. El resultado obtenido es un grafo no conexo a pesar de que la red subyacente si es conexa.

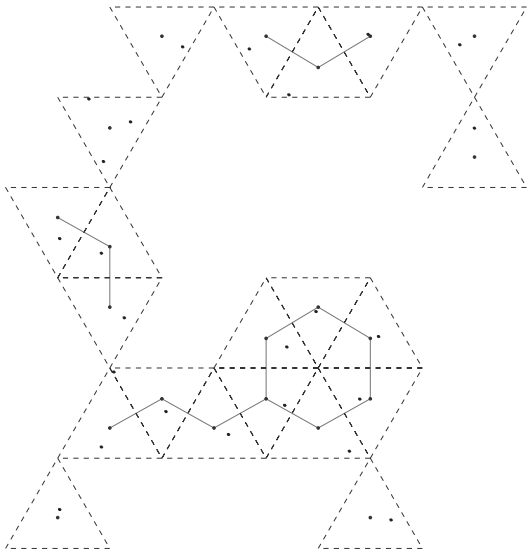


Figura 6.7: Revisión sólo de las celdas vecinas por su lado de t

Cuando se aplican el resto de las pruebas el grafo virtual triangular obtenido es conexo, como se aprecia en la figura 6.8.

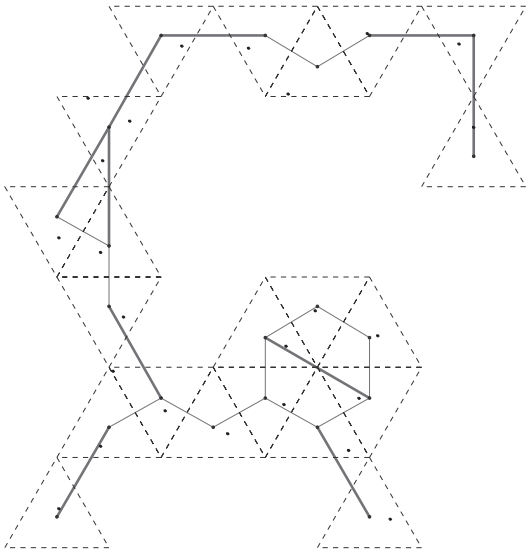


Figura 6.8: Revisión completa de todas las celdas alcanzables

Algoritmo 1 Revisión de las celdas las internas a t

```
1: procedimiento RevisaTriangularesInternas(I, E, t)
2:     k ← 0                                   ▷ Revisión de las celdas internas 0, 4 y 8
3:     mientras k < 12 haz
4:         si existeArista(t, I_k) entonces
5:             agregaArista(t, I_k)
6:         fin si
7:         k ← k + 4
8:     fin mientras
9:
10:    k ← 0                                   ▷ Revisión de las celdas internas 1, 5 y 9
11:    mientras k < 12 haz
12:        l ← k − 1
13:        b1 ← !existeArista (t, I_l)
14:        b2 ← !existeArista (I_l, k)
15:        b3 ← existeArista (t, I_k)
16:        si (b1 O b2) Y b3 entonces
17:            agregaArista(t, I_k)
18:        fin si
19:        k ← k + 4
20:    fin mientras
21:
22:    k ← 3                                   ▷ Revisión de las celdas internas 3, 7 y 11
23:    mientras k < 12 haz
24:        l ← k + 1
25:        b1 ← !existeArista (t, I_l)
26:        b2 ← !existeArista (I_l, k)
27:        b3 ← existeArista (t, I_k)
28:        si (b1 O b2) Y b3 entonces
29:            agregaArista(t, I_k)
30:        fin si
31:        k ← k + 4
32:    fin mientras
33:
34:    RevisaEnlaces(bEnl, I, E)
35:
36:    k ← 2                                   ▷ Revisión de las celdas 2, 6 y 10
37:    l ← 0
38:    mientras k < 12 haz
39:        si (!bEnl(l) Y !bEnl(l + 1) Y existeArista(t, I_k) entonces
40:            agregaArista(t, I_k)
41:        fin si
42:        k ← k + 4
43:        l ← l + 2
44:    fin mientras
45: fin procedimiento
```

Algoritmo 2 Revisión de existencia de aristas entre t y $2, 6, 10$

```
1: procedimiento RevisaEnlaces(bEnl, I, E)
2:     k ← 2                          ▷ Revisar que no existan aristas entre las celdas que
3:     l ← 0                          ▷ están entre t y las que se le oponen por su vértice
4:     mientras k < 12 haz
5:         m ← k − 2                                              ▷ m ← 0, 4, 8
6:         n ← k + 1                                              ▷ n ← 3, 7, 11
7:         o ← (k + 9) mód 12                                     ▷ o ← 11, 3, 7
8:         si existeArista(I_m,I_n) entonces
9:             bEnl_l ←Verdadero
10:        sino
11:            si existeArista(I_n,I_o) entonces
12:                bEnl_l ←Verdadero
13:            sino
14:                si existeArista(I_m,E_n) entonces
15:                    bEnl_l ←Verdadero
16:                sino
17:                    bEnl_l ←Falso
18:                fin si
19:            fin si
20:        fin si
21:
22:        m ← k − 1                                              ▷ m ← 1, 5, 9
23:        n ← (k + 2) mód 12                                     ▷ n ← 4, 8, 0
24:        o ← (k + 3) mód 12                                     ▷ o ← 5, 9, 1
25:        si existeArista(I_m,I_n) entonces
26:            bEnl_l ←Verdadero
27:        sino
28:            si existeArista(I_m,I_o) entonces
29:                bEnl_l ←Verdadero
30:            sino
31:                si existeArista(E_m,I_n) entonces
32:                    bEnl_l ←Verdadero
33:                sino
34:                    bEnl_l ←Falso
35:                fin si
36:            fin si
37:        fin si
38:        l ← l + 2
39:        k ← k + 4
40:    fin mientras
41: fin procedimiento
```

Algoritmo 3 Revisión de las celdas triangulares externas (1ª parte)

```
1: procedimiento RevisaTriangularesExternas(I, E, t)
2:     k ← 0                                   ▷ Revisión de las celdas externas 1, 5 y 9
3:     mientras k < 12 haz
4:         l ← k − 1
5:         b0 ←!existeArista(t, I_l)
6:         b1 ←!existeArista(t, I_k)
7:         b2 ←!existeArista(I_l, I_k)
8:         b3 ←!existeArista(I_k, E_k)
9:         b4 ←!existeArista(I_l, E_k)
10:        b5 ←existeArista(t, E_k)
11:        si (((b0 Y b2) O b1) Y b3) O (b0 y b4) entonces
12:            b ←Falso
13:        sino
14:            b ←Verdadero
15:        fin si
16:        si b Y b5 entonces
17:            agregaArista(t, E_k)
18:        fin si
19:        k ← k + 4
20:    fin mientras
21:
22:    k ← 3                                   ▷ Revisión de las celdas externas 3, 7 y 11
23:    mientras k < 12 haz
24:        l ← (k + 1) mód 12
25:        b0 ←!existeArista(t, I_l)
26:        b1 ←!existeArista(t, I_k)
27:        b2 ←!existeArista(I_l, I_k)
28:        b3 ←!existeArista(I_k, E_k)
29:        b4 ←!existeArista(I_l, E_k)
30:        b5 ←existeArista(t, E_k)
31:        si (((b0 Y b2) O b1) Y b3) O (b0 y b4) entonces
32:            b ←Falso
33:        sino
34:            b ←Verdadero
35:        fin si
36:        si b Y b5 entonces
37:            agregaArista(t, E_k)
38:        fin si
39:        k ← k + 4
40:        k ← l + 2
41:    fin mientras
```

Algoritmo 4 Revisión de las celdas triangulares externas (2ª parte)

```
42:     k ← 0                                   ▷ Revisión de las celdas externas 0, 4 y 8
43:     mientras k < 12 haz
44:         b ←existeArista(t, E_k)
45:         b0 ←!existeArista(t, I_k)
46:         b1 ←!existeArista(I_k, E_k)
47:         si b Y b0 Y b1 entonces
48:             agregaArista(t, E_k)
49:         fin si
50:         k ← k + 4
51:     fin mientras
52:
53:     k ← 2                                   ▷ Revisión de las celdas externas 2, 6 y 10
54:     mientras k < 12 haz
55:         b ←existeArista(t, E_k)
56:         b0 ←!existeArista(t, I_k)
57:         b1 ←!existeArista(I_k, E_k)
58:         si b Y b0 Y b1 entonces
59:             agregaArista(t, E_k)
60:         fin si
61:         k ← k + 4
62:     fin mientras
63: fin procedimiento
```

6.5. Construcción con Cuadrados

En esta sección se describe la forma como se construye el grafo virtual usando cuadrados. Se detallan las pruebas que deben realizarse para garantizar que se obtiene un grafo virtual conexo.

El tamaño elegido para los cuadrados fue tal que la diagonal fuera igual al radio de transmisión R, por lo tanto, la longitud de un lado del cuadrado es igual a $\sqrt{2}R$, como se muestra en la figura 6.9. El tamaño se escogió de esa forma, ya que se pretende que cualquier nodo dentro del cuadrado sea capaz de alcanzar a cualquier otro nodo que también se encuentre en el cuadrado, por lo tanto, la mayor distancia que se tiene entre los puntos dentro del cuadrado es en la diagonal. Los nodos virtuales para la partición con cuadrados fueron seleccionados de los centros de los cuadrados.

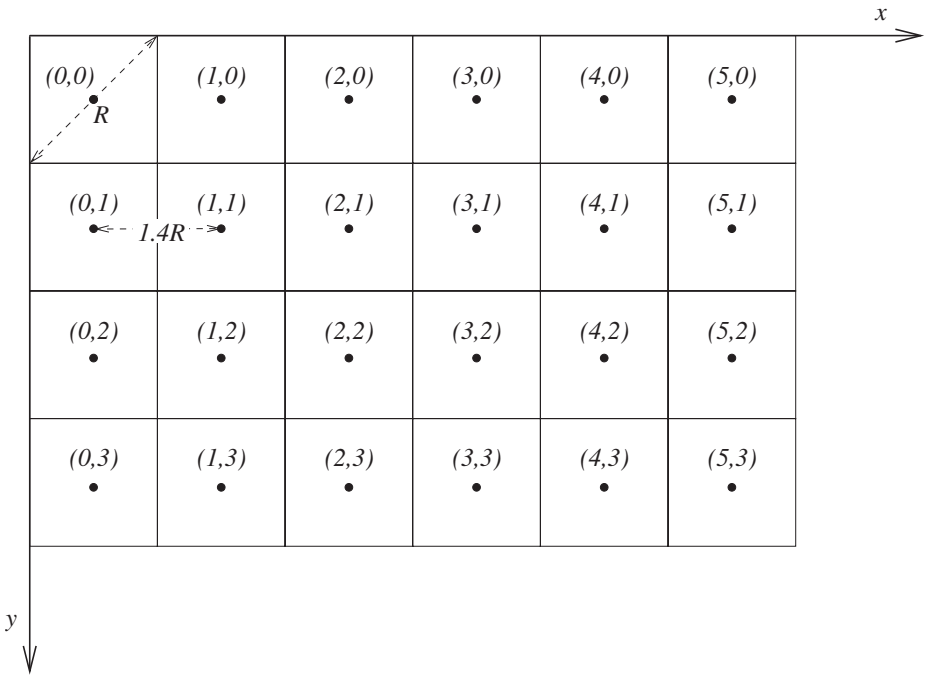


Figura 6.9: Partición del plano usando cuadrados

La manera de realizar la partición con cuadrados se hizo colocando el vértice del cuadrado con la etiqueta $(0, 0)$ en el origen de coordenadas y sus lados coincidiendo con los ejes de coordenadas positivos, como se ve en la figura 6.9. Los otros cuadrados se colocan a los lados del cuadrado $(0, 0)$ para formar la malla de cuadrados.

Conociendo la posición (x, y) de algún cuadrado en la teselación y el valor del

radio de transmisión R se puede determinar las coordenadas del centro $c(x_c, y_c)$ usando las siguientes expresiones:

$$x_c \leftarrow \frac{(x + 0.5)R}{\sqrt{2}} \tag{6.3}$$

$$y_c \leftarrow \frac{(y + 0.5)R}{\sqrt{2}} \tag{6.4}$$

Como cada nodo será identificado con algún celda cuadrada, se indicará la posición (x, y) en la malla de cuadrados, por ejemplo si un nodo tuviera como coordenadas $p(123, 32)$, su posición en la teselación podría ser $(3, 0)$. La forma de obtener la posición (x, y) del nodo $p(x_n, y_n)$ es:

$$\begin{aligned} x &\leftarrow \text{truncar}\left(\frac{\sqrt{2}x_n}{R}\right) \\ y &\leftarrow \text{truncar}\left(\frac{\sqrt{2}y_n}{R}\right) \end{aligned}$$

En la figura 6.10 se muestra que para una celda t se pueden alcanzar hasta 20 celdas, por ejemplo si un nodo de la red estuviera próximo a un vértice, entonces las celdas alcanzables son nueve. Procediendo de una forma similar con los otros vértices se puede obtener que el número de celdas alcanzables es 20.

Para decidir la existencia de aristas virtuales entre las celdas, las celdas alcanzables fueron divididas en dos grupos, como se muestra en la figura 6.10, el primer grupo está formado por las celdas que son vecinas por su lado a la celda t, las cuales tienen las etiquetas del 0 al 7, el segundo grupo esta formado por el resto de las celdas, las cuales se ubican en la periferia, etiquetadas del $0'$ al $11'$.

A continuación se describen los algoritmos que permiten decidir la existencia de aristas virtuales, y que a su vez nos garantizan obtener un grafo plano.

6.5.1. Primera Prueba Local

.

El algoritmo 5 recibe la posición de la celda t que será revisada, así como las posiciones de las celdas del primer grupo I respecto a la celda t. La función existeArista(i, j)

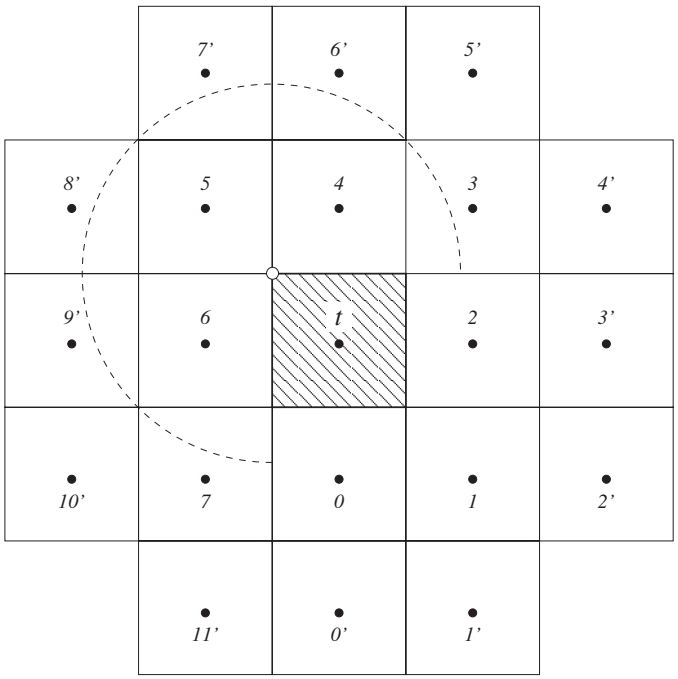


Figura 6.10: Alcance de la celda t con sus celdas vecinas.

devuelve verdadero si hay un par de nodos alcanzables, donde uno de ellos se encuentre en la celda i y el otro en la celda j. La función agregaArista(i, j) coloca una arista de i a j.

En la figura 6.11 se muestra el resultado de aplicar la primera prueba, cabe mencionar que con el primer grupo de celdas revisadas puede obtenerse un grafo no conexo.

6.5.2. Segunda Prueba Local

La primera prueba local no garantiza que el grafo virtual sea conexo, por lo tanto es necesario revisar el segundo grupo de celdas E, es decir, las que se encuentran en la periferia.

El algoritmo 6 recibe la posición de la celda t que será revisada, así como las posiciones de las celdas del primer grupo I y del segundo grupo E respecto a t. Las funciones existeArista(i, j) y agregaArista(i, j) tienen el mismo comportamiento descrito en la subsección 6.5.1.

Entre la línea 2 y la 13 del algoritmo 6 se revisan las celdas $0', 3', 6'$ y $9'$. Para poder colocar una arista entre t y alguna de las celdas, debe verificarse que no se presente un cruce localmente, para estos casos se considera a la celda que se encuentra entre ellas,

Algoritmo 5 Revisión del 1er grupo de celdas

```
1:  procedimiento RevisaCuad1(I, t)
2:      k ← 0                                   ▷ Revisión de t a las celdas 0, 2, 4 y 6
3:      mientras k < 8 haz
4:          si existeArista(t, I_k) entonces
5:              agregaArista(t, I_k)
6:          fin si
7:          k ← k + 2
8:      fin mientras
9:
10:     k ← 0                          ▷ Revisar existencia de enlaces de 0-2, 2-4, 4-6, 6-0
11:     mientras k < 4 haz
12:         a ← 2k                                                    ▷ a ← 0, 2, 4, 6
13:         b ← (2k + 2) mód 8                                        ▷ b ← 2, 4, 6, 0
14:         si existeArista(I_a, I_b) entonces
15:             bVer_k ← Verdadero
16:         sino
17:             bVer_k ← Falso
18:         fin si
19:         k ← k + 1
20:     fin mientras
21:
22:     k ← 0                                   ▷ Revisión de t a las celdas 1, 3, 5 y 7
23:     mientras k < 4 haz
24:         a ← 2k + 1
25:         si !bVer_k Y existeArista(t, I_a) entonces
26:             agregaArista(t, I_a)
27:         fin si
28:         k ← k + 2
29:     fin mientras
30: fin procedimiento
```

además del posible enlace que pasa por la celda intermedia y que se forma con las celdas que se encuentran a sus lados. Por ejemplo, para poner una arista entre t y la celda $0'$, no debe existir el enlace entre la celda t y 0, ni entre 7 y 1, tal como se muestra en la figura 6.12

Las celdas $1', 4', 7'$ y $10'$ se revisan en el algoritmo 6 con el pseudocódigo de la línea 15 a la línea 28. El criterio usado es que no existan las aristas virtuales de corto alcance que podrían ser cruzadas por la arista que quiere ponerse, las aristas de corto alcance son aquellas que se colocan entre celdas que son vecinas por su lado, o por su vértice. Por ejemplo, se pone una arista entre t y $4'$ si las aristas de corto alcance que podrían ser

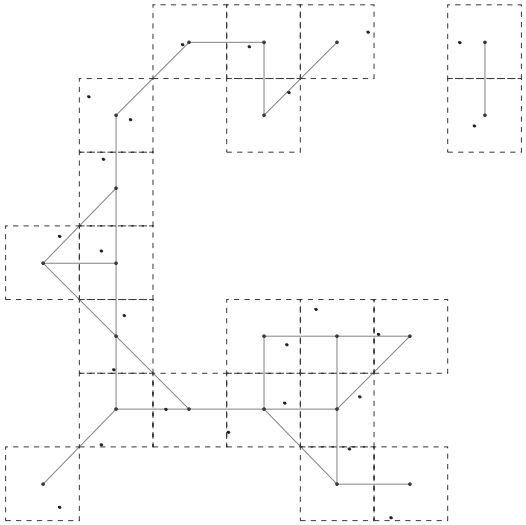


Figura 6.11: Construcción del grafo virtual cuadrado usando la primera prueba.

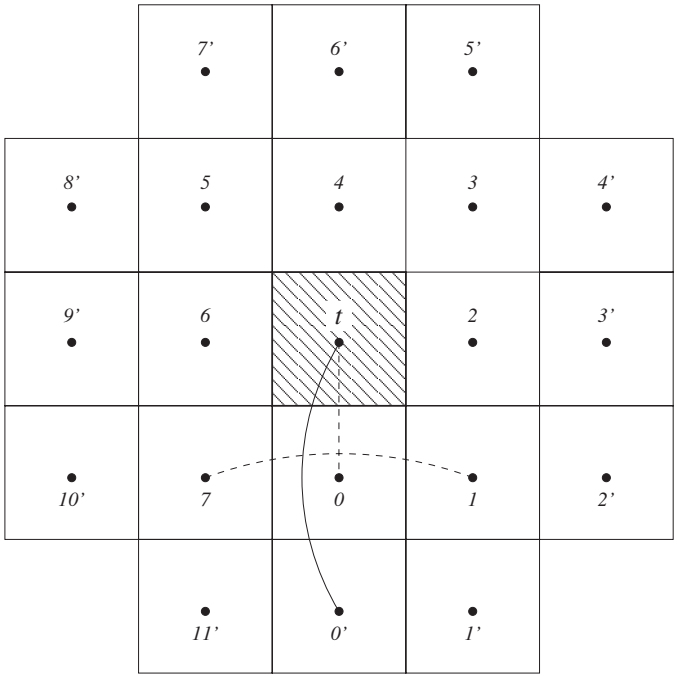


Figura 6.12: Arista virtual entre t y $0'$ si no hay arista entre t y 0, ni entre 7 y 1.

cruzadas no existen, en este caso las aristas de corto alcance son las que se forman entre las celdas $2-4$, $2-3$ y $3-3'$, como se muestra en la figura 6.13

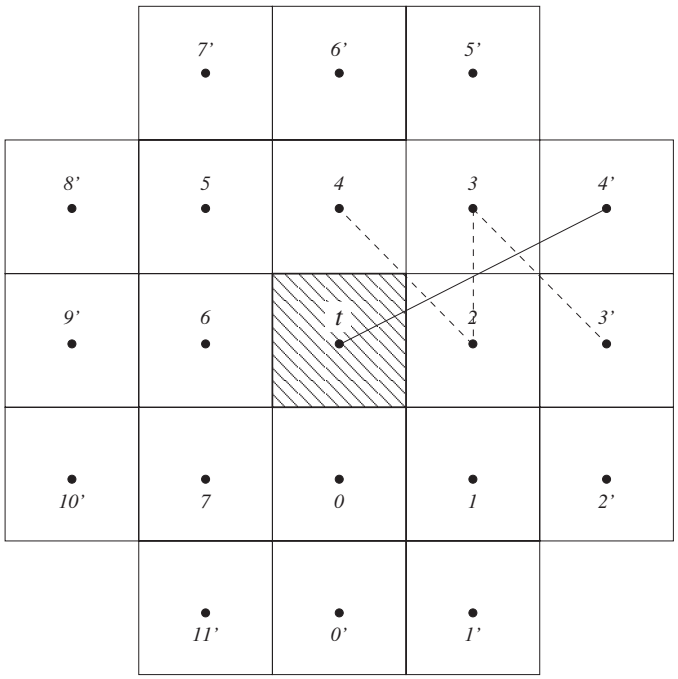


Figura 6.13: Arista virtual entre t y $4'$ si no existen cruces con aristas de corto alcance.

Las celdas restantes, $2', 5', 8'$ y $11'$, son revisadas por el algoritmo 6 usando desde la línea 30 a la 43. El criterio es igual que el anterior, es decir, que no existan aristas virtuales de corto alcance entre las celdas que son vecinas por su lado, o por su vértice. En la figura 6.14 se muestran las aristas de corto alcance que no deben existir, para poner una arista entre t y $8'$.

En la figura 6.15 se muestra el resultado al haber aplicado la segunda prueba local. El ejemplo que se revisa es con el mismo conjunto de nodos que fueron empleados para el grafo virtual triangular. Al comparar el grafo de la figura 6.15 con 6.11, se observa que se agregó una arista en la parte superior derecha con los que el grafo virtual se hace conexo.

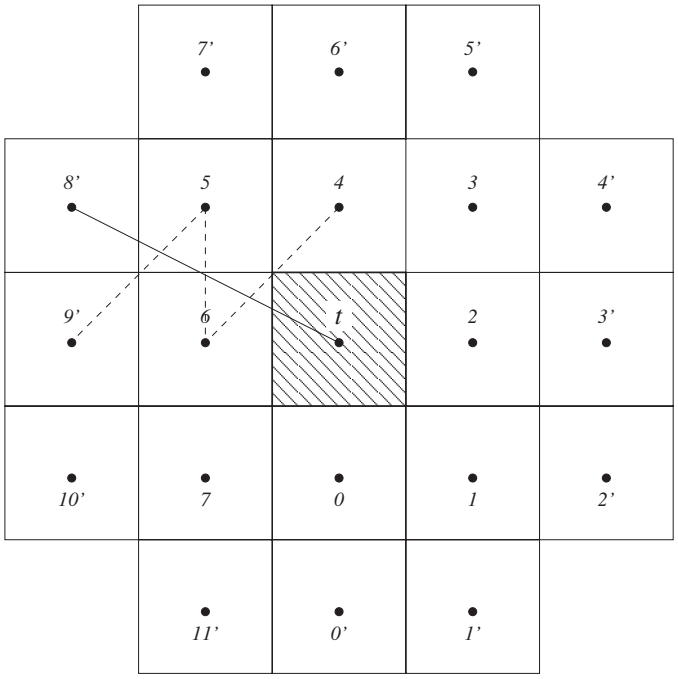


Figura 6.14: Arista virtual entre t y $8'$ si no existen cruces con aristas de corto alcance.

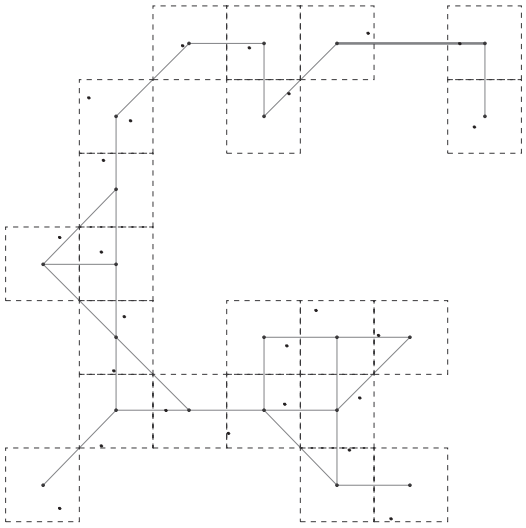


Figura 6.15: Construcción del grafo virtual cuadrado usando las 2 pruebas.

Algoritmo 6 Revisión del 2º grupo de celdas

```
1:  procedimiento RevisaCuad2(I, E, t)
2:      k ← 0                                   ▷ Revisión de las celdas externas 0, 3, 6 y 9
3:      mientras k < 4 haz
4:          a ← (2k + 7) mód 8, b ← 2k          ▷ a ← 7, 1, 3, 5, b ← 0, 2, 4, 6
5:          c ← 2k + 1, d ← 3k                  ▷ c ← 1, 3, 5, 7, d ← 0, 3, 6, 9
6:          b0 ← !existeArista(t, I_b)
7:          b1 ← !existeArista(I_a, I_c)
8:          b2 ← existeArista(t, E_d)
9:          si b0 Y b1 Y b2 entonces
10:             agregaArista(t, E_k)
11:         fin si
12:         k ← k + 1
13:     fin mientras
14:
15:     k ← 0                                   ▷ Revisión de las celdas externas 1, 4, 7 y 10
16:     mientras k < 4 haz
17:         a ← 3k + 1, b ← 2k                  ▷ a ← 1, 4, 7, 11,b ← 0, 2, 4, 6
18:         c ← (2k + 2) mód 8, d ← 3k          ▷ c ← 2, 4, 6, 0, d ← 0, 3, 6, 9
19:         e ← 2k + 1                          ▷ e ← 1, 3, 5, 7
20:         b0 ← !existeArista(I_b, I_c)
21:         b1 ← !existeArista(E_d, I_e)
22:         b2 ← !existeArista(I_b, I_e)
23:         b3 ← existeArista(t, E_a)
24:         si b0 Y b1 Y b2 Y b3 entonces
25:             agregaArista(t, E_k)
26:         fin si
27:         k ← k + 1
28:     fin mientras
29:
30:     k ← 0                                   ▷ Revisión de las celdas externas 2, 5, 8 y 11
31:     mientras k < 4 haz
32:         a ← 3k + 2, b ← 2k                  ▷ a ← 2, 5, 8, 11, b ← 0, 2, 4, 6
33:         c ← (2k + 2) mód 8, d ← 2k + 1      ▷ c ← 2, 4, 6, 8, d ← 1, 3, 5, 7
34:         e ← (3k + 3) mód 12                 ▷ e ← 3, 6, 9, 11
35:         b0 ← !existeArista(I_b, I_c)
36:         b1 ← !existeArista(I_d, E_e)
37:         b2 ← !existeArista(I_d, I_c)
38:         b3 ← existeArista(t, E_a)
39:         si b0 Y b1 Y b2 Y b3 entonces
40:             agregaArista(t, E_k)
41:         fin si
42:         k ← k + 1
43:     fin mientras
44: fin procedimiento
```

6.6. Construcción con Hexágonos

En esta sección, se describe la forma como se construye el grafo virtual empleando hexágonos, que como se había mencionado previamente, realizan una partición regular del plano.

Para esta partición con hexágonos se consideró que el tamaño del hexágono es tal que el par de puntos más alejados en el perímetro se encuentran dentro del rango de transmisión. Estos puntos se encuentran en algún vértice del hexágono y el vértice que se le opone. En la figura 6.16 se muestra la distancia que existe verticalmente y horizontalmente entre los nodos virtuales.

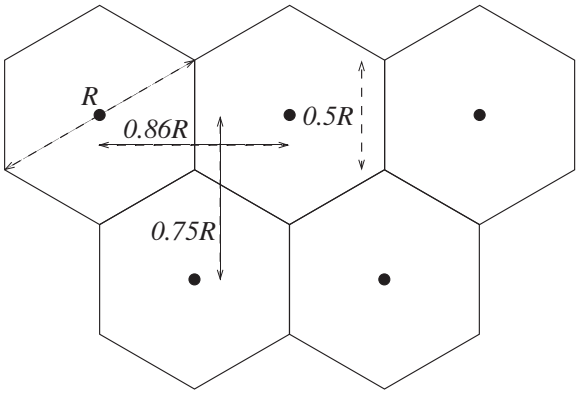


Figura 6.16: Distancias de los nodos virtuales con una partición regular de hexágonos.

La colocación de los hexágonos en el plano cartesiano se hizo colocando un hexágono con su vértice superior izquierdo coincidiendo con el origen del plano cartesiano, y haciendo que su lado esté en el eje positivo de las ordenadas, como se muestra en la figura 6.17. La posición de cada celda hexagonal se determina mediante un par ordenado (x, y), comenzando con la que tiene su vértice en el origen de coordenadas, la cual recibió el par $(0, 0)$. El resto de las posiciones de las celdas se muestra en la figura 6.17.

Usando la posición que tenga una celda hexagonal (x, y) y el valor del radio de transmisión R se puede determinar las coordenadas reales del centro $c(x_c, y_c)$ de cada celda, mediante las siguientes expresiones:

$$\begin{aligned} \text{si} \quad & y \text{ mód } 2 = 0 \\ \text{entonces} \quad & x_c \leftarrow \sqrt{3}R(0.5 + x)/2 \end{aligned} \tag{6.5}$$

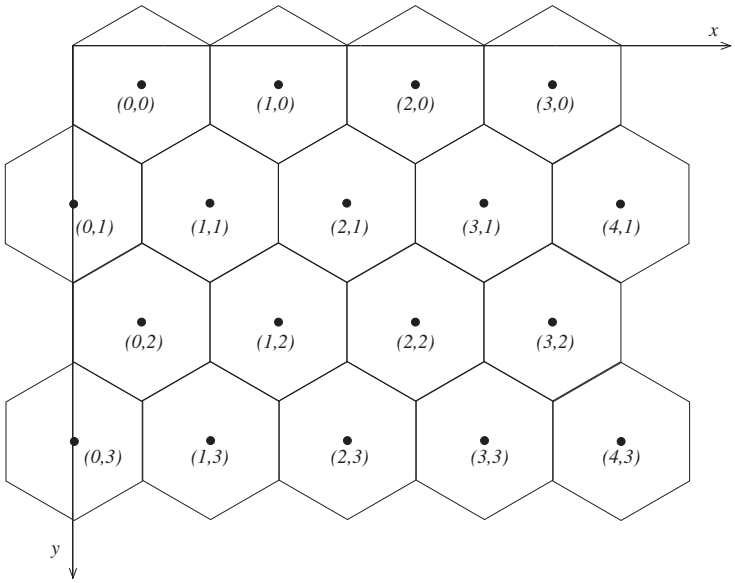


Figura 6.17: Colocación de la malla hexagonal en el plano

$$\text{sino} \qquad x_c \leftarrow \sqrt{3}Rx/2$$

$$y_c \leftarrow 0.75R(y + 1/3) \tag{6.6}$$

Observar que en la expresión 6.5 se debe ajustar el valor de la abscisa debido a que la primera fila, la tercera, etc., presentan un corrimiento hacia la derecha los hexágonos respecto de la segunda fila, la cuarta, etc.

Para identificar cada nodo con su correspondiente celda hexagonal, se dará el par ordenado (x, y) que indica la posición que tiene en la malla de hexágonos. Para obtener una aproximación a la posición de la celda en la cual se encuentra un nodo $p(x_n, y_n)$ se hace lo siguiente:

$$\begin{aligned}
& y \leftarrow \text{truncar } \left(\frac{y_n}{0.75R}\right) \\
\text{si} \quad & (y \text{ mód } 2) = 0 \\
\text{entonces} \quad & x \leftarrow \text{truncar } \left(\frac{2x_n}{\sqrt{3}R}\right) \\
\text{sino} \quad & x \leftarrow \text{truncar } \left(\frac{2x_n + \sqrt{3}R/2}{\sqrt{3}R}\right)
\end{aligned}$$

Para ajustar la posición del hexágono al cual pertenece el nodo p, se debe revisar contra la posición del hexágono vecino más próximo. Se calcula la distancia euclidiana de p al centro del hexágono dado por la expresión anterior, y de p al centro del hexágono vecino más cercano, si la primera distancia es menor respecto a la segunda entonces la posición es la correcta, sino la posición se ajusta a la del hexágono vecino.

En la figura 6.18 se muestra que para una celda t la cantidad de celdas alcanzables son 18. Se tienen 6 celdas que son vecinas por su lado, las cuales serán identificadas por I, y 12 que se encuentran en la periferia representadas por E. En la figura 6.18 se muestran las etiquetas que les fueron asignadas a las cuales se hace referencia cuando se utilizan los algoritmos para encontrar las aristas virtuales. En la misma figura se observa que cuando un nodo se encuentra cercano a un vértice de la celda hexagonal se alcanzan 11 celdas, lo cual si se repite en los otros vértices se encontraría el conjunto mostrado en la figura.

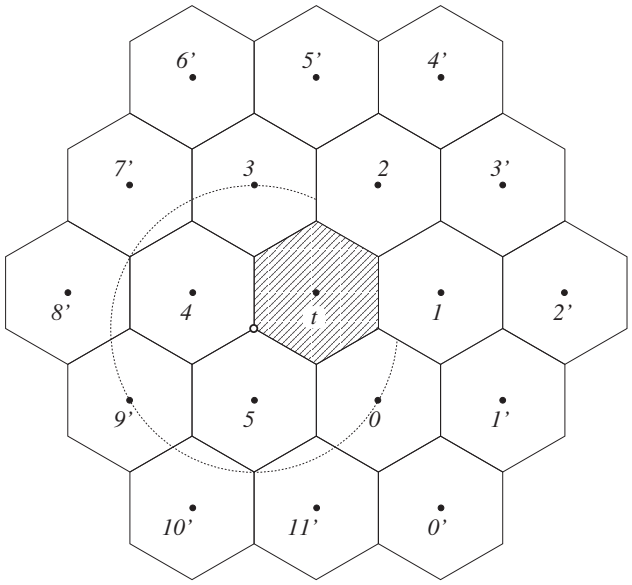


Figura 6.18: Alcance de la celda t con sus celdas vecinas.

Para colocar las aristas entre los centros y obtener el grafo virtual se aplican dos pruebas locales. La primera prueba permite poner aristas entre los hexágonos vecinos por su lado, etiquetados en la figura 6.18 como $1, 2, \ldots, 5$, mientras la segunda prueba nos permite garantizar que el grafo sea conexo, las celdas revisadas en la segunda prueba tienen las etiquetas $1', 2', \ldots, 11'$.

6.6.1. Primera Prueba Local

En la primera prueba local se determina si es posible colocar una arista entre el centro de algún hexágono y alguno de los centros de los hexágonos que son vecinos por su lado; la arista virtual se pone si existe al menos un par de nodos reales $u, v \in V$ tal que $d(u, v) \leq R$, y que u y v se encuentren en hexágonos distintos. El algoritmo 7 muestra la forma como se realiza la prueba descrita, la función existeArista(i, j) revisa si existe un par de puntos que sean mutuamente alcanzables, donde uno de los puntos se encuentre en la celda hexagonal i, y el otro en la celda j. La función agregaArista(i, j) pone una arista virtual entre i y j. El algoritmo requiere recibir la posición de las celdas vecinas por su lado I respecto a t y la posición de la celda t.

Algoritmo 7 Revisión del 1er grupo de celdas

```
1: procedimiento RevisaHexagonos1(I, t)
2:     k ← 0                          ▷ Revisión de t a las celdas internas 0, 1, ..., 5
3:     mientras k < 6 haz
4:         si existeArista(t, I_k) entonces
5:             agregaArista(t, I_k)
6:         fin si
7:         k ← k + 1
8:     fin mientras
9: fin procedimiento
```

Empleando el mismo conjunto de puntos que fue usado para mostrar la construcción del grafo virtual triangular y cuadrado, se muestra en la figura 6.19 el resultado de aplicar la primera prueba.

6.6.2. Segunda Prueba Local

Con el fin de garantizar que el grafo virtual hexagonal sea conexo se define otro conjunto de pruebas para las 12 celdas de la periferia E. Las celdas de la periferia se pueden ver en la figura 6.18.

Las celdas externas de la periferia son dividas en dos grupos para colocar sus aristas virtuales. El primer grupo de ellas esta formada por las celdas $0', 2', 4', 6', 8'$ y $10'$ y el segundo grupo por las celdas $1', 3,', 5', 7', 9'$ y $11'$. Entre la celda t y algunas de las celdas del primer grupo se observa en la figura 6.18 que hay una celda entre ellas, por ejemplo, entre t y $2'$, la celda intermedia es la celda 1. En el caso del segundo grupo observamos que

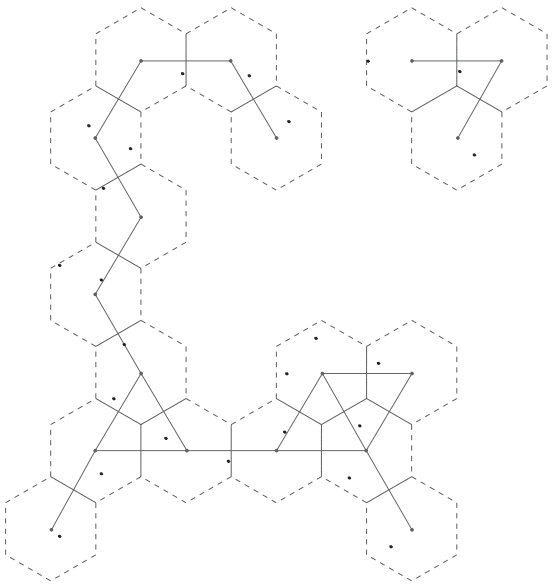


Figura 6.19: Construcción del grafo virtual hexagonal usando la primera prueba.

existen dos celdas intermedias entre t y alguna celda externa, por ejemplo, entre t y $5'$ se tienen las celdas intermedias 2 y 3.

Para colocar una arista virtual entre t y alguna de las celdas virtuales del primer grupo, se requiere que no exista un camino entre la celda t y la celda intermedia correspondiente hacia la celda externa, por ejemplo, se pondrá una arista virtual entre t y el nodo $2'$, como se muestra en la figura 6.20 si se cumple lo siguiente:

- No existe camino entre t y $2'$ a través de la celda intermedia 1, ya sea porque la arista entre t y 1 no existe o porque la arista entre 1 y $2'$ no existe.
- Existe un par de nodos que se alcanzan donde uno de ellos se encuentra en la celda del nodo t y el otro en la celda del nodo $2'$.

La colocación de aristas virtuales entre la celda t y el segundo grupo de celdas externas, debe considerar que no exista un camino entre t y alguna de las celdas intermedias correspondientes hacia la celda externa, por ejemplo, se pondrá una arista virtual entre t y el nodo $5'$, como se muestra en la figura 6.20, si se cumple lo siguiente:

- No existe camino entre t y $5'$ a través de la celda intermedia 2, ya sea porque la arista entre t y 2 no existe o porque la arista entre 2 y $5'$ no existe.

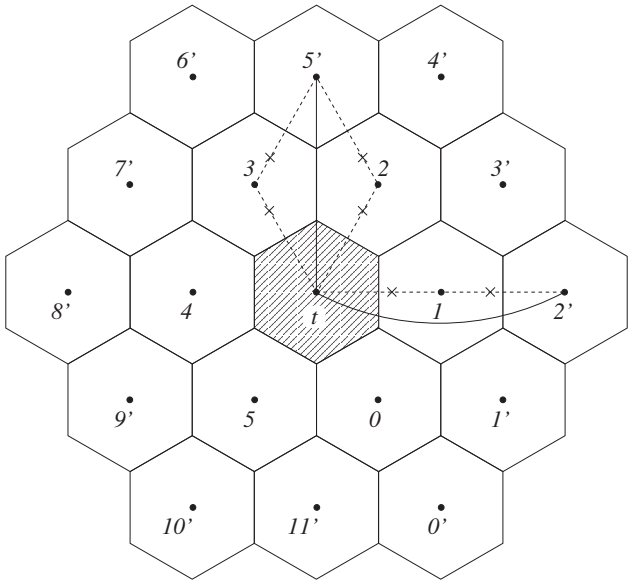


Figura 6.20: Ejemplos de revisiones en la 2ª prueba de la malla de hexágonos.

- No existe camino entre t y $5'$ a través de la celda intermedia 3, ya sea porque la arista entre t y 3 no existe o porque la arista entre 3 y $5'$ no existe.
- Existe un par de nodos que se alcanzan donde uno de ellos se encuentra en la celda del nodo t y el otro en la celda del nodo $5'$.

El resultado de aplicar la segunda prueba se puede apreciar en la figura 6.21. La arista agregada aparece mas gruesa.

El algoritmo 8 describe las pruebas que conforman el segundo grupo, el cual recibe las posiciones de las celdas vecinas por su lado I, las posiciones de las celdas de la periferia, así como la posición que tiene la celda t.

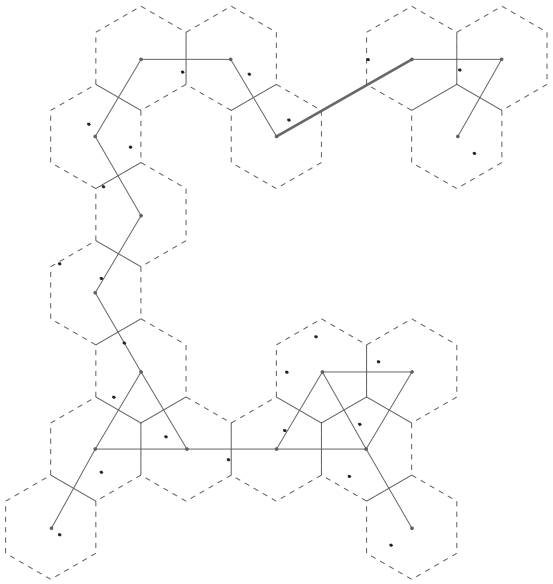


Figura 6.21: Construcción del grafo virtual hexagonal usando las 2 pruebas.

Algoritmo 8 Revisión del 2º grupo de celdas

```
1:  procedimiento RevisaHexagonos2(I, E, t)
2:      k ← 0                          ▷ Revisión de t a las celdas externas 0, 2, 4, ..., 10
3:      mientras k < 6 haz
4:          a ← 2k                                              ▷ a ← 0, 2, 4, 6, 8, 10
5:          b0 ←!existeArista(t, I_k)
6:          b1 ←!existeArista(I_k, E_a)
7:          b2 ←existeArista(t, E_k)
8:          si b0 Y b1 Y b2  entonces
9:              agregaArista(t, E_k)
10:         fin si
11:         k ← k + 1
12:     fin mientras
13:
14:     k ← 0                          ▷ Revisión de t a las celdas externas 1, 3, 5, ..., 11
15:     mientras k < 6 haz
16:         a ← (k + 1) mód 6                                   ▷ a ← 1, 2, 3, 4, 5
17:         b ← 2k + 1) mód 6                                   ▷ b ← 1, 3, 5, 7, 11
18:         b0 ←existeArista(t, I_k)
19:         b1 ←existeArista(I_k, E_b)
20:         b2 ←existeArista(t, I_a)
21:         b3 ←existeArista(I_a, E_b)
22:         b4 ←existeArista(t, E_b)
23:         si !(b0 Y b1) Y !(b2 Y b3) Y b4 entonces
24:             agregaArista(t, E_b)
25:         fin si
26:         k ← k + 1
27:     fin mientras
28: fin procedimiento
```

6.7. Encaminamiento en el Grafo Virtual

La manera como se hará el encaminamiento es usando el algoritmo GFG *(Greedy-Face-Greedy)* propuesto en el trabajo de [Bose01]. Este algoritmo emplea dos métodos para enviar paquetes en la red. El primero es el método *voraz progresivo*, el cual se usa siempre que sea posible, y el segundo *encaminamiento por caras*, es usado cuando el primero no se puede emplear. El primer método será usado nuevamente tan pronto como sea posible.

Cuando se emplea el algoritmo *voraz* no se requiere extraer de la red de nodos un grafo plano, ya que los nodos localmente conocen la dirección de sus vecinos y del nodo destino, por lo que basta con escoger al vecino que se encuentre más cercano geográficamente al eventual nodo destino. Los nodos locales pueden aplicar otros criterios con la finalidad de mejorar la métrica usada, por ejemplo, si se desea minimizar el consumo de energía, entonces el nodo local deberá escoger al nodo vecino más cercano y que presente progreso hacia el nodo destino.

Como se ha revisado previamente, el algoritmo voraz podría fallar al intentar descubrir una ruta desde algún *origen* a un *destino*, ya que estas topologías harán que el siguiente nodo se encuentre más alejado del destino que el nodo actual, por lo tanto, el mecanismo de recuperación empleado es el *encaminamiento por caras*.

Al usar el algoritmo de *encaminamiento por caras* [Kranakis99, Morin01, Bose01] se garantiza que no existe ninguna topología que pueda vencer al algoritmo. El grafo plano que es usado por este algoritmo es el propuesto en este trabajo, es decir, el Grafo Virtual, donde cada nodo localmente construye con la información de sus vecinos las aristas virtuales del Grafo Virtual. El encaminamiento usando el Grafo Virtual permite orientar el encaminamiento real en la red, ya que los nodos virtuales son solamente una referencia para los nodos reales.

En la figura 6.22 se muestra lo que sucede al emplear el encaminamiento por caras, donde los nodos virtuales forman el grafo plano sobre el cual el algoritmo de encaminamiento por caras, y la selección de los nodos reales se basa siguiendo el camino virtual encontrado por el *encaminamiento por caras*. Se debe observar que cada nodo puede escoger a cualquier vecino, ya que el Grafo Virtual que es extraído de la red, no depende de los nodos, sino de la forma como se encuentran distribuidos los nodos, por lo tanto dependiendo de la métrica empleada, se pueden usar distintas heurísticas para minimizar el costo. Por ejemplo, cuando un nodo en un hexágono x decide encaminar hacia el hexágono y, la decisión de que nodo

del hexágono y será el elegido, depende de la métrica, si se quiere minimizar el número de saltos se escogerá aquel nodo de la celda y que se encuentre más cercano al destino pero, si se quiere minimizar el gasto de energía y los nodos pueden variar la potencia de su señal, entonces el nodo seleccionado será aquel que se encuentre más próximo al nodo actual.

En el ejemplo de la figura 6.22 se ilustra la forma como funciona el encaminamiento por caras para ir del nodo 22 al nodo 24. El primer paso consiste en determinar la posición de la celda en la cual se encuentra el nodo 22 y 24, determinando que se encuentran en la celdas $(4, 5)$ y $(5, 1)$ respectivamente. Como el encaminamiento se realizará empleando caras exclusivamente, se visita la cara formada por las celdas $4, 5$, $3, 4$ y $4, 4$ y se decide que la cara más próxima es $4, 4$ por lo que enruta hacia esa posición, y se hace el cambio de cara. Cuando se explora la siguiente cara se descubre la celda destino después de recorrer toda la cara externa, como la trayectoria sea más corta en el sentido de las manecillas del reloj el camino virtual será $(4, 4)$, $(3, 4)$, $(3, 5)$, $(2, 5)$, $(1, 4)$, $(1, 3)$, $(1, 2)$, $(1, 1)$, $(1, 0)$, $(2, 0)$, $(3, 1)$, $(4, 0)$, $(5, 0)$ y $(5, 1)$.

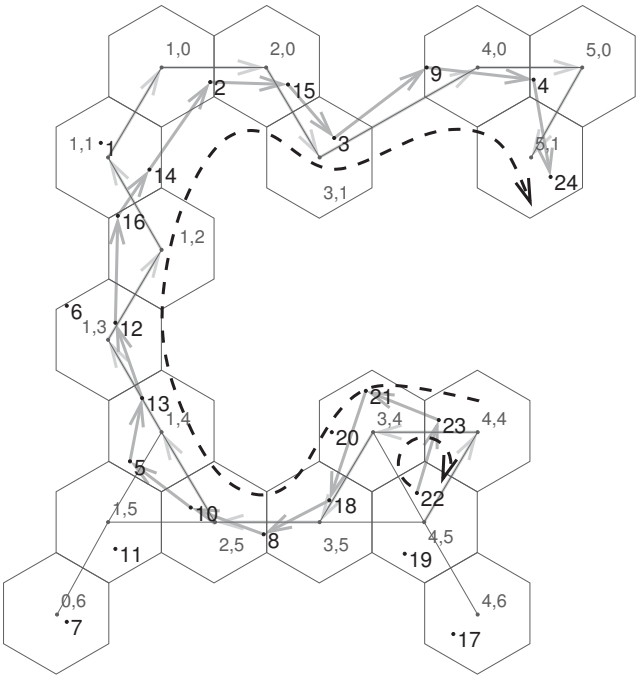


Figura 6.22: Encaminamiento por caras en el Grafo Virtual del nodo 22 al 24

Como se había citado previamente el grafo virtual es tan solo una referencia, ya que el encaminamiento real se hace a través de los nodos, por lo que el camino con los

nodos es 22, 23, 21, 18, 8, 10, 5, 13, 12, 16, 14, 2, 15, 3, 9, 4 y 24. Se observa también que en algunos casos se llega hacer doble salto dentro de alguna celda, por ejemplo en la celda $(1, 4)$, ya que no se puede ir a la celda indicada si no se avanza dentro de la celda hacía esa posición.

En la figura 6.23 se muestra un ejemplo donde se presenta la forma como trabaja el algoritmo GFG. El nodo origen s es el nodo 17, a partir del cual se empieza a emplear el algoritmo voraz, el cual puede avanzar hasta el nodo 23 pero, después no existe ningún vecino que se encuentre más próximo al nodo 24, por lo que cambiará al modo de encaminamiento por caras, con lo que explora toda la cara externa y descubre la celda destino. Como el trayecto sea más corto en el sentido de las manecillas del reloj se emplea esa dirección.

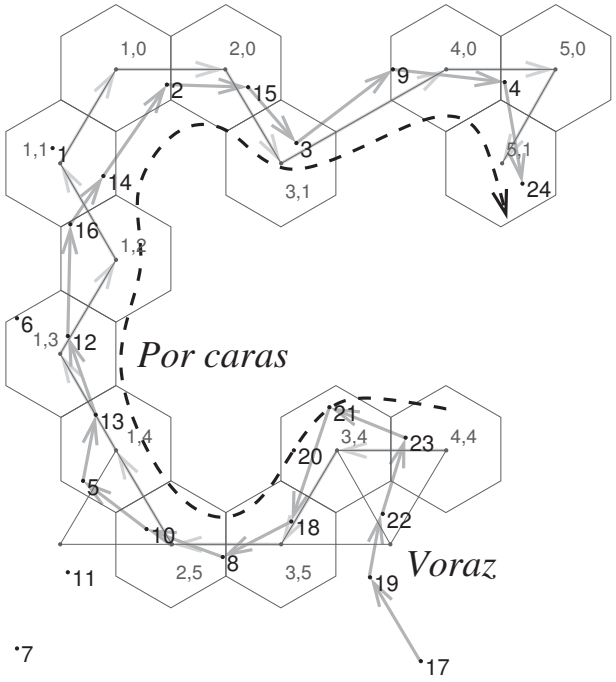


Figura 6.23: Encaminamiento GFG en el Grafo Virtual del nodo 17 al 24

6.8. Resumen

En este capítulo se ha revisado la forma de construir el grafo virtual empleando teselaciones regulares. Para un mismo conjunto de puntos si se comparan las distintas construcciones se puede comentar que la menor cantidad de celdas es hecha con los hexágonos,

luego los cuadrados, y la mayor cantidad con los triángulos, lo que deriva como resultado que las pruebas para colocar aristas virtuales entre las celdas sean mas sencillas en los hexágonos, después en los cuadrados, y más complejas en los triángulos. La razón de lo anterior es que para una celda t, en el caso de triángulos se revisan 24 celdas, con cuadrados son 20, y en un hexágono son 18. También con los hexágonos se colocan el mayor número de aristas virtuales a celdas vecinas por su lado, respecto a las particiones con cuadrados y triángulos, con lo cual las aristas colocadas por la segunda prueba serán pocas. Por otra parte, con el grafo virtual hexagonal se logra una mejor descripción de la red física subyacente, por tener cada vértice virtual el valor de su valencia o grado mayor que los vértices de las celdas cuadradas o triangulares.

Capítulo 7

Un Simulador de Arquitectura Abierta

En este capítulo se describe el simulador que fue usado para comparar el desempeño del grafo virtual contra otros grafos que obtienen un grafo plano localmente. También se revisa su arquitectura y las opciones implementadas.

En la parte de la arquitectura del sistema, se discute cual fue el diseño propuesto para el funcionamiento del simulador, pretendiendo que los distintos módulos que lo componen se parezcan, en la medida de lo posible, a una implementación real.

El simulador permite generar dos tipos de experimentos: variando la cantidad de puntos para una malla fija, o bien, variando el tamaño de la malla para un conjunto fijo de nodos. Con estos experimentos se logra variar la densidad par poder comparar el algoritmo del Grafo Virtual con otros algoritmos que también crean un grafo plano. Al realizar los experimentos con el simulador es posible seleccionar distintos algoritmos de encaminamiento, lo que permite conocer mejor su desempeño dados distintos escenarios.

7.1. Arquitectura del simulador

En la figura 7.1 se muestra en forma esquemática los distintos módulos que constituyen al simulador. Estos módulos son usados para encontrar el siguiente vecino al cual será enviado el paquete de datos. Dependiendo del tipo de encaminamiento escogido puede variar la cantidad de módulos usados. Se emplean todos los módulos cuando se emplea el algoritmo *encaminamiento por caras*. Al emplear el algoritmo *voraz* no se requiere la capa

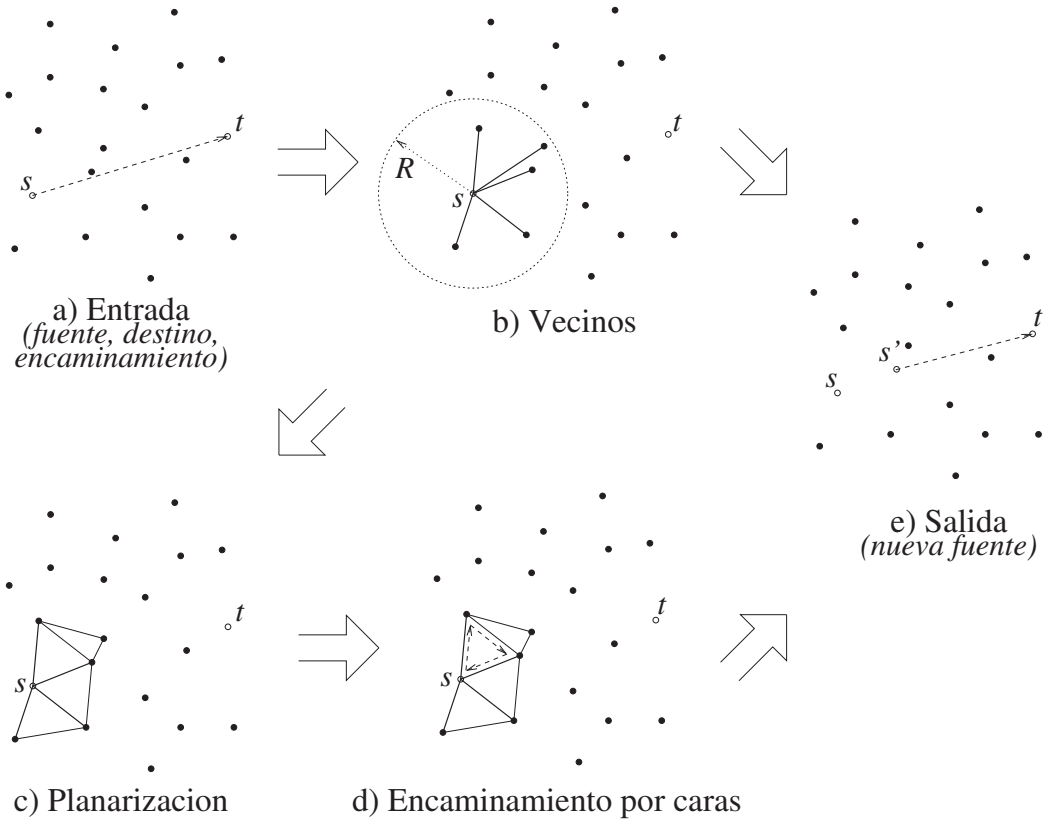


Figura 7.1: Módulos usados para obtener el siguiente vecino al encaminar del nodo s al nodo t.

de planarización.

Se describen a continuación los distintos módulos del simulador.

a) **Entrada.** Se encarga de recibir un paquete de datos acompañado por las coordenadas del nodo destino para algún nodo. El nodo que recibe el paquete de datos se le denomina el nodo fuente.

b) **Vecinos.** Esta capa se encarga de descubrir los nodos vecinos del nodo fuente s, es decir, aquellos nodos que se encuentren dentro de su rango de transmisión. Se debe señalar que los nodos sólo son capaces de obtener información local. Si se usa encaminamiento voraz entonces se escoge el siguiente vecino y se pasa al módulo *salida*, en caso de que se esté empleando el encaminamiento por caras entonces la información de los vecinos es dada al módulo de *planarización*.

c) **Planarización.** Este módulo obtiene un grafo plano local con la información dada por el módulo *vecinos*. El grafo plano es empleado por algún algoritmo de encaminamiento por caras.

d) **Encaminamiento por caras.** Recorriendo la cara descubre al nodo más próximo a la línea proyectada del origen al destino, y encamina al vértice encontrado.

e) **Salida.** Devuelve el siguiente nodo que recibirá el paquete de datos, así como la información necesaria para seguir encaminando, por ejemplo: las coordenadas del nodo fuente y el tipo de encaminamiento. Si no se ha llegado al destino la información anterior será transferida al módulo de *entrada*.

La razón de emplear módulos dentro del simulador, es con el fin de que al usar la herramienta se puedan probar nuevos algoritmos de encaminamiento o de planarización que sean propuestos por algún investigador, sin que tenga que preocuparse por la interacción con otros módulos.

Cuando se desea agregar un nuevo algoritmo de encaminamiento, éste pertenecerá al módulo de *encaminamiento por caras* y aparecerá dentro del mismo como una función o conjunto de funciones. En el caso de que el algoritmo de encaminamiento proponga una nueva función para obtener un grafo plano, entonces deberá agregarse la función o las funciones respectivas en el módulo de *planarización*.

7.2. Descripción del simulador

Con esta herramienta, se pretende tener un ambiente de trabajo donde sea posible por parte del usuario explorar algunos algoritmos de encaminamiento en nubes de puntos o grafos propuestos por el usuario. La implementación del simulador fue tal que tuviera un aspecto claro y visualmente atractivo, de tal forma que cualquier caso estudiado pueda ser analizado fácilmente.

En el diseño se consideró que el sistema tuviera una interfaz clara, rápida y sencilla para el usuario. Una manera de lograr lo anterior, es que la manipulación de las operaciones más comunes se pueda realizar usando el ratón como por ejemplo: la inserción de nodos, el borrado de nodos, la selección del nodo fuente o del nodo destino, el algoritmo para planarizar el grafo del disco unitario, la elección del tipo de encaminamiento, entre otros comandos.

Uno de los lenguajes de programación que nos permiten desarrollar las características descritas que debe tener el simulador, es el lenguaje *Java* por tener bibliotecas gráficas. Las bibliotecas gráficas que se disponen en Java nos proporcionan un conjunto de elementos utilizados para la comunicación con la aplicación, tales como: botones, ventanas, listas desplegables, etc.

Como el lenguaje Java está orientado para su uso en Internet, entonces se puede aumentar la capacidad del simulador, ya que se podría colocar en un servidor y usarse desde cualquier lugar usando tan sólo un navegador con soporte para ejecutar *applets*.

Otra característica del lenguaje Java es su orientación a la programación de objetos, satisfaciendo las tres características propias que son: encapsulación, herencia y polimorfismo, por lo que con la programación orientada a objetos se tienen más ventajas al momento de usar los componentes de los sistemas operativos con entornos gráficos.

También el lenguaje Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas cosas para mantener el objetivo de la simplicidad del lenguaje.

Desafortunadamente Java tiene inconvenientes, por ejemplo, las funciones disponibles en Java dependen en muchas ocasiones de la versión de la herramienta para generar el código, en particular, de la versión del *API* del núcleo de Java con el que deban guardar compatibilidad, teniendo como consecuencia, en ocasiones, la versión y el tipo de navegador que se puede utilizar. La compatibilidad con versiones anteriores en principio está garantizada.

También el aspecto final de la aplicación dependerá del navegador utilizado para visualizar la aplicación y de su configuración, por lo que será una desventaja si se pretende que el programa pueda ser empleado por la mayoría de los navegadores.

Por las características descritas previamente y por ser un lenguaje muy empleado por las comunidades de investigadores se decidió implementar el simulador con el lenguaje Java.

El simulador fue hecho como una aplicación independiente, es decir, una *aplicación de consola*, ya que los *applets* tienen medidas de seguridad tan estrictas que no permiten muchas de las opciones del lenguaje, como por ejemplo el acceso a archivos, que es una utilidad importante dentro del simulador.

7.3. Requisitos del sistema

Para el funcionamiento correcto y la visualización del simulador, el equipo de cómputo debe cumplir con una serie de requisitos mínimos, aunque se ha pretendido que dichos requisitos sean lo menos restrictivos para que el simulador se pueda emplear en la mayoría de los equipos. Una limitante importante es la capacidad para la ejecución de la máquina virtual Java, para el caso de una computadora personal *Intel* son:

- Procesador Intel Pentium III 500 Mhz. o superior.
- 64 Mb de memoria RAM.
- Tarjeta gráfica de 32 bits con resolución mínima de 800x600 pixels.

Los requisitos del software son los siguientes:

- Sistema operativo con interfaz gráfica.
- JVM 1.4.0 o superior

7.4. Algoritmos implementados

Los algoritmos implementados en el simulador fueron:

- Encaminamiento por caras.
- Encaminamiento GFG.

Los cuales garantizan la entrega del paquete de datos. Como se mencionó anteriormente, si se usa el algoritmo de encaminamiento por caras se requiere un grafo plano. Para crear un grafo plano los algoritmos implementados en el simulador fueron:

- Grafo de Vecindad Relativa.
- Grafo de Gabriel.
- Grafo de Morelia.
- Grafo Virtual Triangular.
- Grafo Virtual Cuadrado.
- Grafo Virtual Hexagonal.

7.5. Descripción de la Interfaz Gráfica

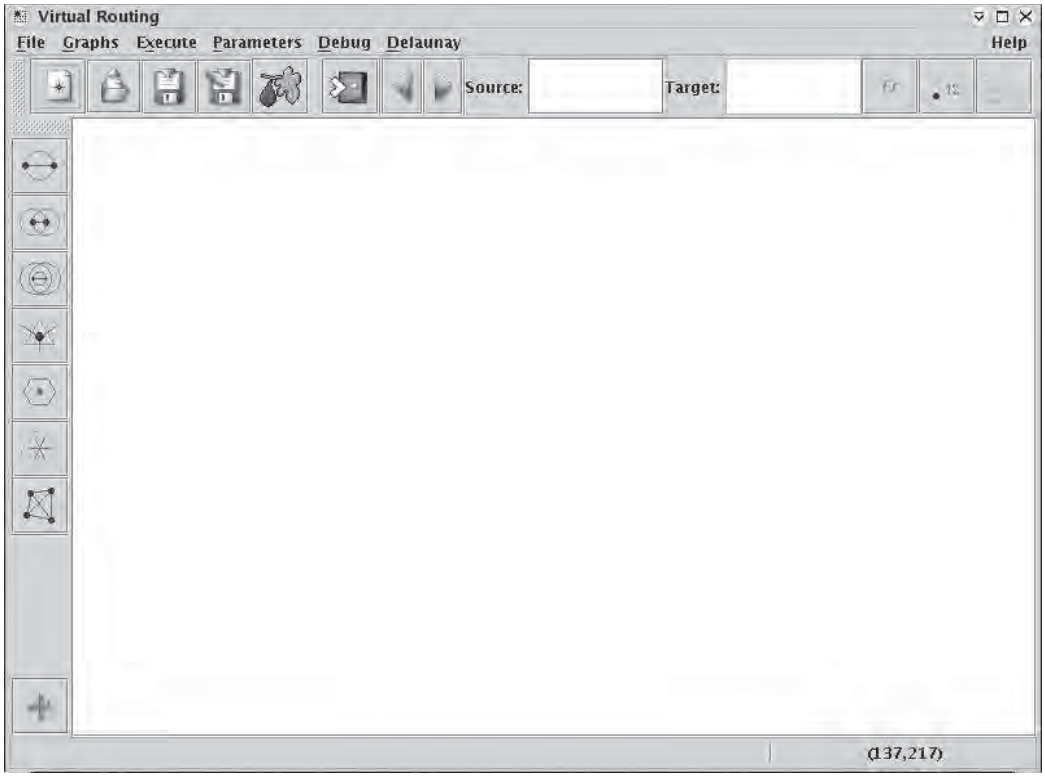


Figura 7.2: Simulador.

El simulador es una aplicación *JAVA* distribuida en un paquete *JAR* y por tanto ejecutable en un sistema que tenga instalado *JVM*. La distribución de los elementos no difiere de la empleada en otras herramientas gráficas como se muestra en la figura 7.2. El simulador se compone de una barra de menúes, de una barra de herramientas, de una barra para visualizar grafos (situada verticalmente), de una barra de estado y de la superficie gráfica o lienzo.

7.5.1. Superficie gráfica

La zona gráfica tiene varios usos para la manipulación de puntos, las cuales se describe a continuación:

- **Inserción de un punto**. Se ubica el apuntador del ratón en la posición que desea colocarse y se presiona el botón principal.

- **Borrado de un punto**. Se ubica el apuntador del ratón encima del punto que desea eliminarse y se presiona el botón principal.

- **Selección del nodo fuente y destino**. Se ubica el apuntador del ratón en el punto que será el nodo fuente o el nodo destino y se presiona el botón secundario. Si el nodo fuente ya fue seleccionado, entonces al repetir la operación con otro punto distinto será el nodo destino. Al seleccionar el punto origen o el punto destino aparecen encerrados por una circunferencia.

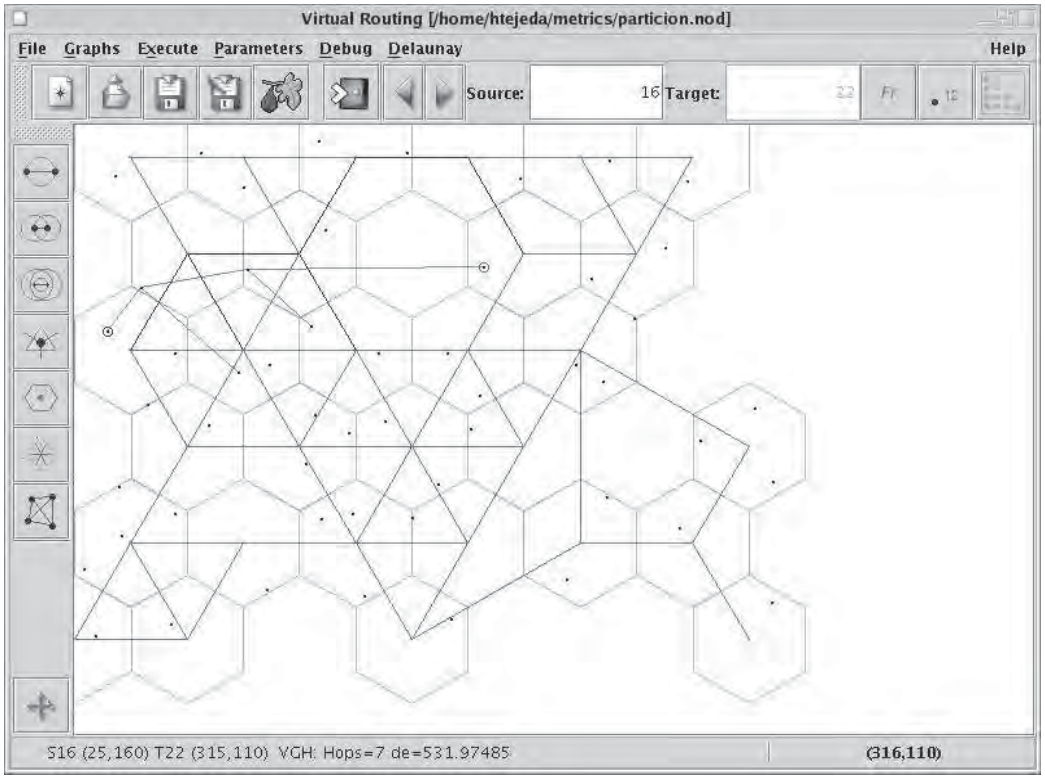


Figura 7.3: Ejemplo hecho con el simulador usando encaminamiento por caras en el Grafo Virtual Hexagonal.

En la figura 7.3 se muestra el resultado del encaminamiento por caras con Grafo Virtual Hexagonal. En la barra de estado se muestra información del nodo fuente, del nodo origen y el costo del encaminamiento.

En la parte derecha de la barra de estado se muestran las coordenadas donde se encuentra el apuntador del ratón.

7.5.2. Barra de menúes

La barra de menúes se compone de siete listas que son:

- *File*
- *Graphs*
- *Execute*
- *Parameters*
- *Debug*
- *Delaunay*
- *Help*

Los comandos de la barra vertical *File* también se encuentran presentes en la barra de herramientas. Los elementos que contiene son descritos a continuación:

- *New*, limpia la superficie gráfica para iniciar un nuevo conjunto de puntos.
- *Open ...*, carga en el simulador un conjunto de puntos que haya sido previamente almacenado usando un ventana de diálogo para seleccionar el archivo, la extensión por defecto es *.nod*.
- *Save*, permite almacenar en disco el conjunto de puntos que contenga la superficie gráfica si ya está relacionado con algún archivo, en caso contrario solicita que se proporcione el nombre usando una ventana de diálogo.
- *Save as ...*, almacena en disco el conjunto de puntos con algún nombre usando una ventana de diálogo, puede usarse también para guardar el conjunto de puntos con otro nombre.
- *Export fig ...*, permite generar un archivo en formato *fig* de lo que se muestra en la superficie gráfica. El formato *fig* puede ser manipulado por *Xfig* que es una herramienta interactiva para dibujar y que también permite generar archivos en otros formatos.
- *Quit*, termina el uso del simulador, en caso de que se hayan hechos cambios al conjunto de nodos y no se hayan almacenado pregunta si desea guardar.

Con las opciones del menú *Graphs* de la barra de menúes se pueden seleccionar los tipos de grafos que se desean mostrar para el conjunto de puntos, como también el tipo de encaminamiento que se desea hacer, las opciones son descritas a continuación:

- *Gabriel Graph*, dibuja el Grafo de Gabriel.
- *RNG*, dibuja el Grafo de Vecindad Relativa.
- *Morelia Graph*, dibuja el Grafo de Morelia.
- *Triangular Virtual Graph*, dibuja el Grafo Virtual con Triángulos.
- *Square Virtual Graph*, dibuja el Grafo Virtual con Cuadrados.
- *Hexagonal Virtual Graph*, dibuja el Grafo Virtual con Hexágonos.
- *Unit Disk Graph*, dibuja el grafo de discos unitarios.
- *Delaunay Graph*, dibuja el Grafo de Voronoi y la triangulación de Delaunay locales para el centro virtual del punto origen.
- *Euclidean Shortest Path*, dibuja el camino euclidiano más corto.
- *Hops Shortest Path*, dibuja el camino más corto en saltos.
- *Euclidean Shortest Path TVG*, dibuja el camino euclidiano más corto para el Grafo Virtual con Triángulos.
- *Hops Shortest Path TVG*, dibuja el camino más corto en saltos en el Grafo Virtual con Triángulos.
- *Euclidean Shortest Path HVG*, dibuja el camino euclidiano más corto para el Grafo Virtual con Hexágonos.
- *Hops Shortest Path HVG*, dibuja el camino más corto en saltos en el Grafo Virtual con Hexágonos.
- *Euclidean Shortest Path SVG*, dibuja el camino euclidiano más corto para el Grafo Virtual con Cuadrados.
- *Hops Shortest Path SVG*, dibuja el camino más corto en saltos en el Grafo Virtual con Cuadrados.

- *Euclidean Distance*, muestra en la barra de estado la distancia euclidiana entre el punto fuente y el destino.

- *Face Routing*, permite intercambiar los algoritmos de encaminamiento por caras y voraz-cara-voraz (GFG).

De la lista anterior de opciones que se encuentran disponibles en el menú *Grafos*, las primeras siete opciones pueden también accederse usando la barra de botones empleando la tabla 7.1.

Tipo	Botón
Grafo de Vecindad Relativa	
Grafo de Gabriel	
Grafo de Morelia	
Grafo Virtual Triangular	
Grafo Virtual Cuadrado	
Grafo Virtual Hexagonal	
Grafo del Disco Unitario	

Cuadro 7.1: Descripción de Botones para Grafos

Las opciones del menú *Execute* permiten hacer lo siguiente:

- *Graphs*, permite generar los grafos que hayan sido seleccionados con el conjunto de puntos que se encuentren en la superficie gráfica. Este comando también puede ser dado usando el botón que se encuentra al final de la barra de botonos para visualizar grafos.

- *Generate Points ...*, permite colocar un conjunto de puntos, que es conexo, en la superficie gráfica.

- *Grid*, realiza un conjunto de ejemplos variando el tamaño de la malla y manteniendo constante el número de puntos. Los distintos tamaños de las mallas son configurados en el menú *Parameters* de la barra de menúes.

- *Points*, realiza un conjunto de ejemplos variando el tamaño de un conjunto de puntos y manteniendo constante el tamaño de la malla. Los valores iniciales y finales, así como el incremento de puntos son configurados usando el menú *Parameters* de la barra de menúes.

- *Source Point ...*, esta opción se usa si se conoce cual es la etiqueta que tiene asociado el punto fuente. En la barra de herramientas se tiene un campo de texto para realizar la misma tarea.

- *Target Point ...*, con esta opción se puede introducir la etiqueta del nodo destino. Al igual que la opción anterior se tiene un campo de texto para introducir el nodo destino.

Con las opciones del menú *Parameters* de la barra de menúes se pueden dar valores para el comportamiento de la simulación, los cuales se describen a continuación:

- *Width ...*, sirve para indicar el ancho que tendrá la superficie gráfica usando un cuadro de diálogo.

- *Height ...*, sirve para indicar el alto que tendrá la superficie gráfica usando un cuadro de diálogo.

- *Transmission Range ...*, se usa para modificar el rango de transmisión de todo el conjunto de nodos.

- *Graphs ...*, modifica la cantidad de grafos que serán usados para alguna configuración particular. Por ejemplo, supongamos que se debe generar para una simulación un grafo de 500×500 con 100 puntos, entonces con este parámetro se indica el número de grafos aleatorios conexos con las características anteriores que serán generados.

- *Experiments ...*, se emplea para indicar en la simulación para un grafo la cantidad de casos o pares origen-destino, que se probarán.

Los siguientes parámetros se usan cuando se quiere automatizar un gran conjunto de pruebas aleatorias, variando el número de puntos y manteniendo fijo el tamaño de la malla.

- *Initial Points ...*, en el caso que se genere una simulación variando el tamaño del conjunto de puntos, con esta opción se puede indicar con cuantos puntos iniciará.

- *Final Points ...*, se emplea para indicar hasta que cuantos puntos se consideran al simular una variación con puntos.
- *Points Increment ...*, indica el incremento que tendrá al hacer la simulación con puntos.

Los siguientes parámetros se emplean cuando se decide automatizar un conjunto de pruebas aleatorias, variando el tamaño de la malla y manteniendo constante el número de puntos.

- *Initial Width ...*, modifica el valor inicial del ancho de la malla.
- *Final Width ...*, cambia el valor final del ancho de la malla.
- *Widht Increment ...*, modifica el incremento que tendrá el ancho de la malla.
- *Initial Height ...*, modifica el valor inicial del alto de la malla.
- *Final Height ...*, cambia el valor final del alto de la malla.
- *Height Increment ...*, modifica el incremento que tendrá el alto de la malla.

- *Generate Graphs*, indica si los grafos serán generados, o se usarán los que se tienen en disco.

En el menu *Debug* se tienen disponibles opciones para controlar el simulador, o bien, para mostrar información, se tienen las siguientes opciones disponibles:

- *Labels*, muestra u oculta las etiquetas de los nodos. También se tiene disponible mediante un botón en la barra de herramientas.

- *Debug*, se usa para generar información a la salida estándar del encaminamiento hecho en los distintos grafos. El comando también se encuentra disponible con un botón en la barra de herramientas.

- *Geographic Coordinates*, muestra en la salida estándar las coordenadas que tiene cada punto.
- *Max. Face Routing*, configura el número máximo de enrutamientos por cara que hará el simulador antes de abortar el envío para un par de puntos.
- *Max. Hops*, configura el número máximo de saltos que hará el simulador antes de abortar para ir de un nodo a otro.
- *Connection cells*, muestra en la salida estándar para cada celda virtual sus puntos correspondientes.

El menú de *Delaunay* de la barra de menúes permite indicar que objetos son mostrados al tener puesta la opción *Delaunay Graph* del menú *Graphs*. Los objetos que se muestran son locales al centro al que pertenece el punto origen, y los comandos son:

- *Delaunay Triangulation*, muestra o quita la triangulación de Delaunay para el centro del nodo origen.
- *Voronoi Regions*, muestra u oculta las regiones de Voronoi locales para el centro del nodo origen.
- *Delaunay Circles*, muestra u oculta los circulos inscritos de la triangulación de Delaunay para el centro del nodo origen.

El menú *Help* muestra información del simulador, el comando asociado al menú es:

- *About ...* muestra un cuadro de diálogo con información del simulador.

7.6. Resumen

La herramienta descrita en el capítulo permite realizar la simulación del encaminamiento de una manera sencilla y ágil. El usuario tiene la posibilidad de colocar su configuración de nodos utilizando solamente el ratón, o bien, puede emplear el comando que le permite generar un grafo conexo. También con el ratón se puede indicar cual es el nodo origen y el nodo destino al que se quiere encaminar. Una ventaja del simulador es que permite automatizar una gran conjunto de pruebas aleatorias y ejecutarlas como un proceso, para lograr lo anterior, se requiere dar algunos valores para controlar la simulación.

Capítulo 8

Resultados

Se muestra a continuación los resultados de los experimentos hechos usando el simulador. Como se describió en el capítulo anterior el simulador puede generar un conjunto de grafos de acuerdo a los valores de los parámetros dados. Los grafos generados son usados por el simulador para seleccionar pares de puntos, con los que se realiza el encaminamiento.

Se hizo la simulación construyendo un grafo aleatorio conexo dentro de un área cuadrada o malla; el simulador emplea el modelo del grafo del disco unitario, el cual considera que todos los nodos tienen el mismo rango de transmisión. Los dos tipos de experimentos que fueron hechos para variar la densidad de nodos en la malla, fueron:

- Variación del tamaño de la malla con un grupo fijo de puntos.
- Variación de la cantidad de puntos en una malla de tamaño fijo.

Con cada grupo de experimentos se aplicó el algoritmo de encaminamiento por caras exclusivamente y el algoritmo GFG. La razón de mostrar los resultados que se obtienen al usar solamente el algoritmo de encaminamiento por caras es por que son más notorias las ventajas que se presentan el usar el Grafo Virtual, además, cuando se tiene una densidad adecuada de nodos los grafos planos no son usados, ya que la mayor parte del encaminamiento hecho es del tipo voraz cuando es usado el algoritmo GFG.

En los experimentos hechos se midieron los siguientes dos tipos de métricas:

- El número de saltos o transmisiones. Este valor nos indica el número de retransmisiones hechas para mandar un mensaje de algún origen a un destino.

- La distancia euclidiana. Indica la longitud total del recorrido hecho para encaminar del nodo origen al destino.

Los Grafos Virtuales con triángulos, cuadrados y hexágonos fueron comparados contra los siguientes grafos planos cuando se usa el *encaminamiento por caras*:

- El Grafo de Gabriel.
- El Grafo de Vecindad Relativa.
- El Grafo de Morelia.
- El Camino más corto.

La decisión de escoger el *camino más corto* usando el algoritmo de Dijkstra obedece a que es una cota inferior de los grafos anteriores pero, el algoritmo requiere de la información global para conocer el camino más corto.

8.1. Tamaño de Malla Variable

En este grupo de experimentos fueron puestos 50 nodos en una malla de tamaño variable. El tamaño inicial de la malla fue de 500 u^2 hasta 2000 u^2 con incrementos de 100 u^2. Se consideró el rango de transmisión para los nodos de 250 u.

Para cada tamaño de malla se generaron 50 grafos aleatorios, de los cuales se tomó el 4 % como tamaño de la muestra del total de pares ordenados. El total de pares es igual al número de combinaciones de 50 nodos tomando 2 a la vez, por lo que se obtiene:

$$\binom{50}{2} = \frac{50!}{(50-2)!2!} = \frac{50 \cdot 49}{2} = 1225$$

Entonces el tamaño de la muestra fueron 49 pares ordenados, los cuales fueron tomados aleatoriamente de una estratificación hecha del universo ordenado lexicográficamente $((0,1),(0,2),\ldots(49,50))$, es decir, el universo fue dividido en 49 grupos y de cada grupo se escogió aleatoriamente un par ordenado.

En las gráficas 8.1 y 8.2 se muestran el número de saltos promedio al variar la densidad de nodos, en este caso modificando el tamaño de la malla. Los incrementos promedio de los distintos grafos son comparados respecto al camino más corto empleando el encaminamiento por caras y el esquema GFG.

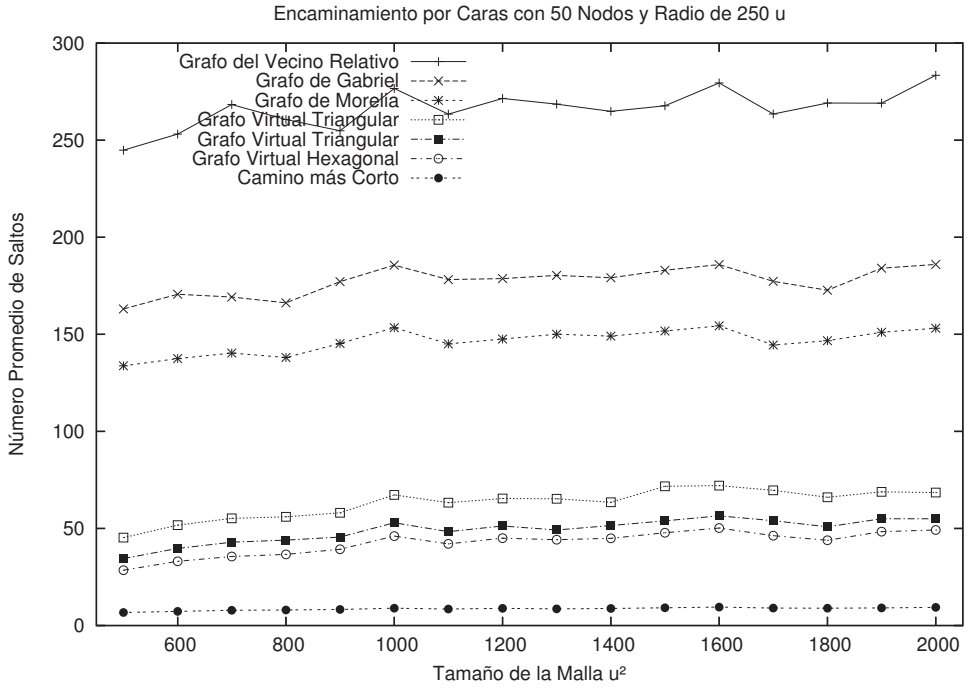


Figura 8.1: Encaminamiento por caras variando el tamaño de la malla vs transmisiones.

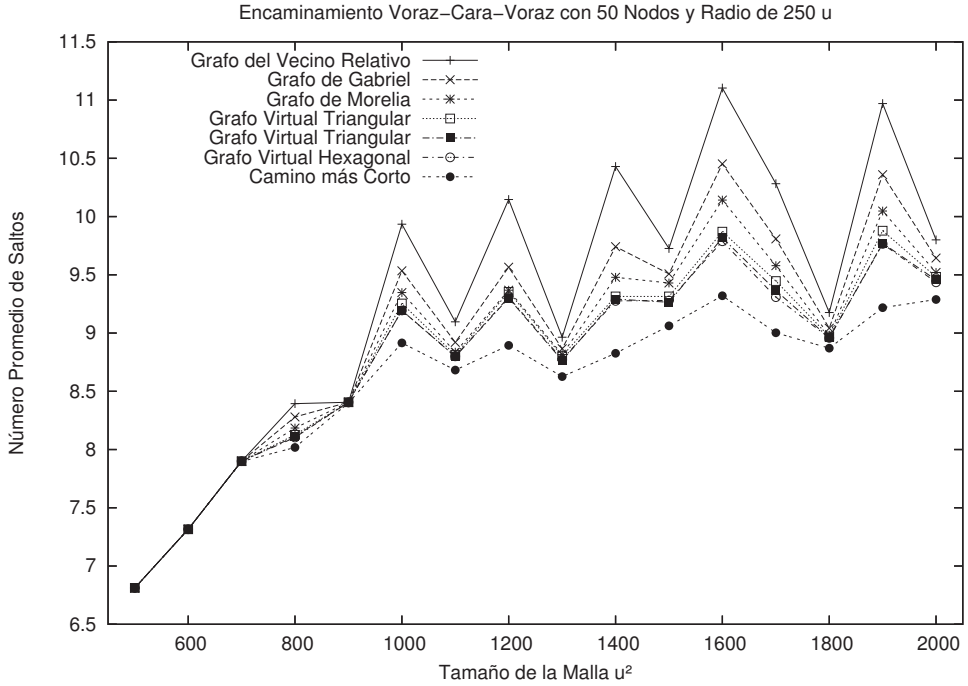


Figura 8.2: GFG variando el tamaño de la malla vs número de transmisiones.

Para comparar analíticamente el resultado de los experimentos cuando se mide el número de saltos, al emplear el algoritmo de encaminamiento por caras y GFG, en la tabla 8.1 se muestra los incrementos que se tienen respecto al camino más corto. Los resultados obtenidos muestran el incremento porcentual promedio respecto al camino más corto. Con las distintas configuraciones de los grafos virtuales se aprecia que los incrementos son menores respecto al grafo de Morelia, de Gabriel y el de vecindad relativa. El que logra el mejor desempeño de los seis grafos, es el grafo virtual hexagonal. La razón por la cual, los incrementos son tan grandes al hacer el encaminamiento por caras obedece a la cantidad de aristas que deben ser revisadas, por una parte es el recorrido en toda la cara para determinar la mejor arista para hacer el cambio, y después el recorrido para llegar nuevamente a la mejor arista.

Tipo	Encaminamiento por caras	Voraz-Cara-Voraz
Grafo de Vecindad Relativa	3010.51 %	8.25 %
Grafo de Gabriel	1972.21 %	5.11 %
Grafo de Morelia	1610.31 %	3.66 %
Grafo Virtual Triangular	636.16 %	2.85 %
Grafo Virtual Cuadrado	473.49 %	2.46 %
Grafo Virtual Hexagonal	397.50 %	2.39 %

Cuadro 8.1: Incremento porcentual en saltos respecto al camino más corto

Las gráficas 8.3 y 8.4 muestran los resultados obtenidos al usar como métrica la distancia euclidiana. Se muestra la distancia euclidiana promedio para llegar de un nodo fuente a un nodo destino variando el tamaño de la malla y considerando un número fijo de nodos.

En la tabla 8.2 se muestran los incrementos porcentuales en la distancia euclidiana que tienen los grafos al ser comparados con el camino más corto, usando el encaminamiento por caras y el esquema voraz-cara-voraz, los resultados de la tabla muestran que usando los grafos virtuales los incrementos son menores respecto a los otros grafos pero, en el caso del encaminamiento GFG, columna derecha de la tabla, los incrementos son casi idénticos porque la mayoría de las veces el tipo de encaminamiento fue del tipo voraz, y en esa situación no se hace uso del algoritmo que extrae un subgrafo plano. Se espera el encaminamiento GFG, tenga una mejor tendencia cuando se consideren más ejemplos, semejante a la que se da cuando se mide el número de transmisiones, observar la columna derecha de la tabla 8.1.

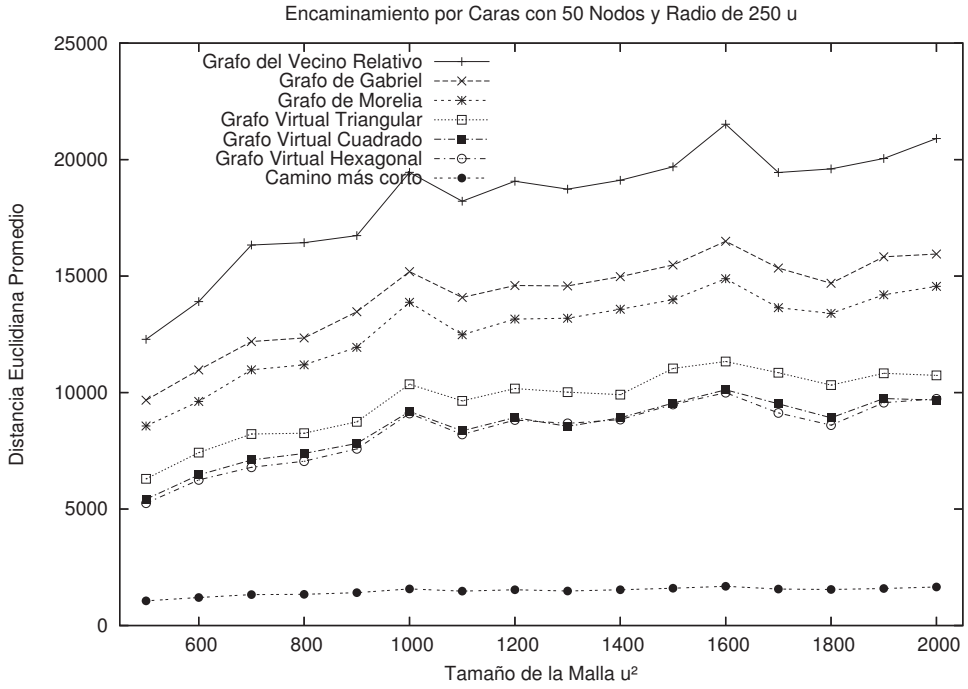


Figura 8.3: Encaminamiento por caras variando la malla vs distancia euclidiana.

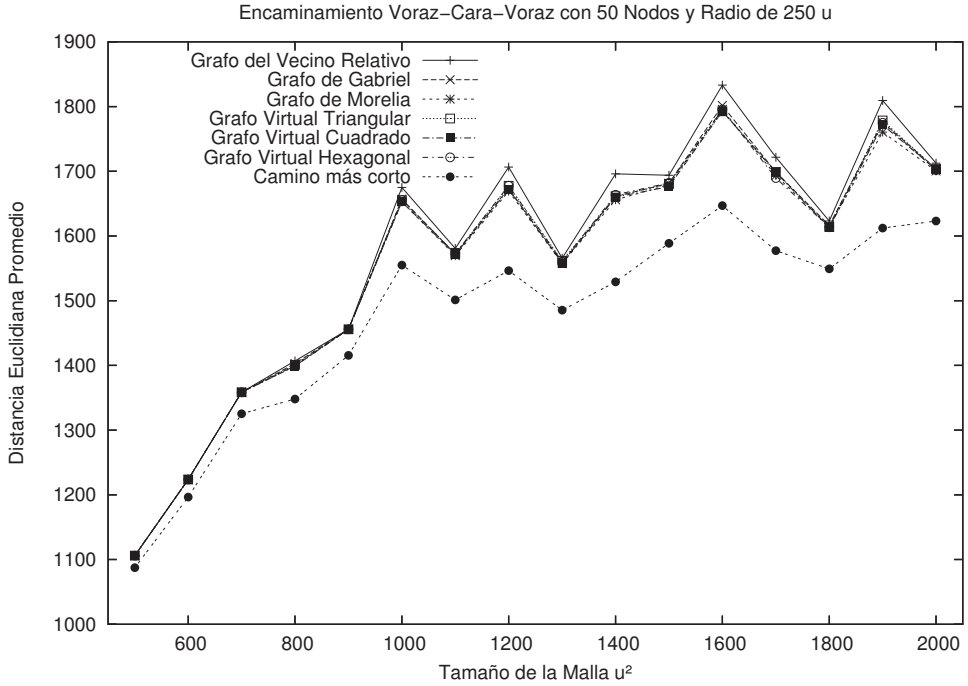


Figura 8.4: GFG variando tamaño de la malla vs distancia euclidiana.

Tipo	Encaminamiento por caras	Voraz-Cara-Voraz
Grafo de Vecindad Relativa	1136.38 %	6.71 %
Grafo de Gabriel	857.95 %	5.77 %
Grafo de Morelia	762.02 %	5.54 %
Grafo Virtual Triangular	553.83 %	5.74 %
Grafo Virtual Cuadrado	475.34 %	5.62 %
Grafo Virtual Hexagonal	464.50 %	5.69 %

Cuadro 8.2: Incremento porcentual en distancia respecto al camino más corto.

8.2. Número de Puntos Variable

En esta sección se hicieron experimentos en donde se varía el número de puntos mientras el tamaño de la malla permanece constante. El tamaño escogido para la malla fue de 5000 u^2; el número de nodos iniciales fue de 500, con incrementos de 100 hasta llegar a los 2000. El tamaño de la muestra tomado fue de 0.0005 %, de esta forma se tomaron 62 pares ordenados cuando se tiene 500 puntos y 999 pares ordenados con 2000 puntos. El total de pares es igual al número de combinaciones de n puntos tomando 2 a la vez. Para obtener el conjunto aleatorio de pares ordenados, el universo de pares fue ordenado lexicográficamente y estratificado; de cada conjunto fue tomado un elemento de forma aleatoria.

En las figuras 8.5 y 8.6 se muestra el comportamiento cuando es medido el número de transmisiones promedio contra el número de nodos. Se observa en la figura 8.5 que conforme la densidad se va incrementando, los incrementos que sufren los grafos virtuales son pequeños comparados contra los otros grafos, lo cual no sucede se aprecia en la figura 8.2, porque al ser los grafos más densos, la mayoría de las veces se hace un encaminamiento voraz, el cual se aproxima bastante al camino más corto, por lo anterior los datos de la figura 8.2 fueron divididos por el camino más corto para observar mejor los detalles.

Debido a que no se logran apreciar diferencias al usar GFG al variar el número de puntos, se decidió normalizar los valores obtenidos con los grafos planos respecto al camino más corto. El resultado anterior se muestra en la figura 8.2. La razón por la cual no hay diferencias significativas, obedece a que en la mayoría de los casos fue suficiente con aplicar solamente el algoritmo voraz para encaminar entre un origen y un destino, lo anterior también puede apreciarse en la tabla 8.3 que muestra los incrementos porcentuales respecto al camino más corto.

En la tabla 8.3 se muestran los incrementos promedio del número de transmisiones

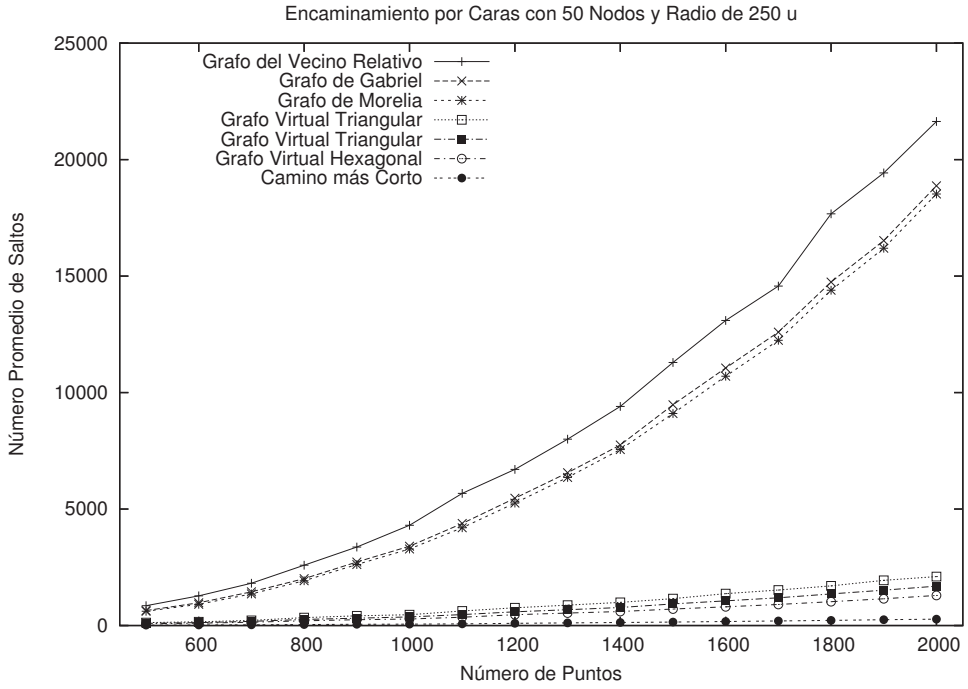


Figura 8.5: Encaminamiento por caras variando el número de puntos vs transmisiones.

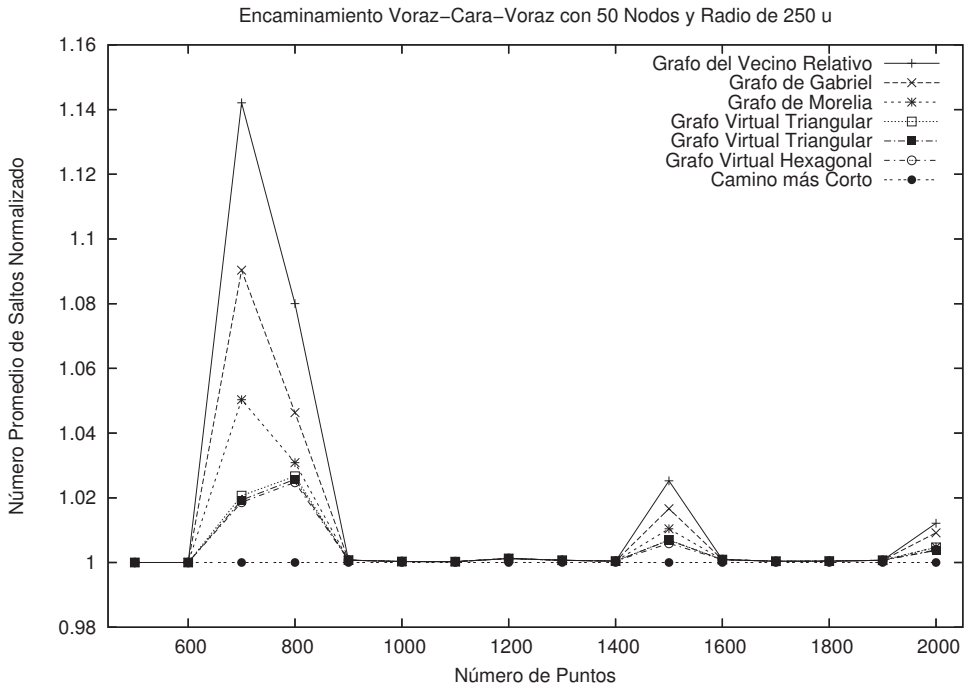


Figura 8.6: GFG variando el número de puntos vs número de transmisiones.

que se tienen al emplear los grafos respecto al camino más corto. Se observa que para el encaminamiento por caras y para GFG los grafos virtuales tienen un mejor desempeño respecto a los otros grafos. El grafo virtual hexagonal es el que logra los mejores resultados de los grafos virtuales.

Tipo	Encaminamiento por caras	Voraz-Cara-Voraz
Grafo de Vecindad Relativa	7353.36 %	0.82 %
Grafo de Gabriel	6141.34 %	0.55 %
Grafo de Morelia	5962.98 %	0.34 %
Grafo Virtual Triangular	677.51 %	0.26 %
Grafo Virtual Cuadrado	509.89 %	0.24 %
Grafo Virtual Hexagonal	368.29 %	0.23 %

Cuadro 8.3: Incremento porcentual en saltos respecto al camino más corto.

Para este grupo de experimentos se realizó también la medición de la distancia euclidiana promedio entre un par de nodos cuando transmiten usando el algoritmo de encaminamiento por caras y el esquema voraz-cara-voraz. En las figuras 8.7 y 8.8 se muestran los resultados obtenidos.

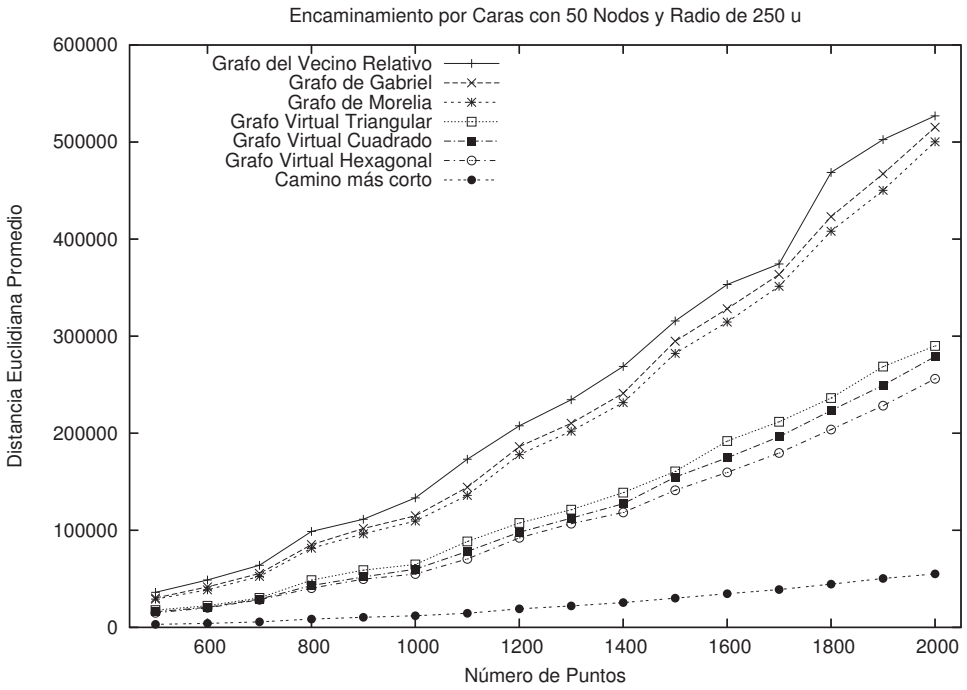


Figura 8.7: Encaminamiento por caras variando los puntos vs distancia euclidiana.

Los valores usados para la gráfica de la figura 8.8 fueron obtenidos normalizando los valores obtenidos de los grafos planos respecto al camino más corto, se tuvo que hacer de esa manera, ya que los cambios no hubieran podido ser apreciados, porque la mayoría de las veces se hace encaminamiento voraz, y poco encaminamiento por caras al usar GFG. Lo anterior también se puede apreciar en los incrementos porcentuales que se tienen con los grafos planos respecto al camino más corto, en la segunda columna de la tabla 8.4.

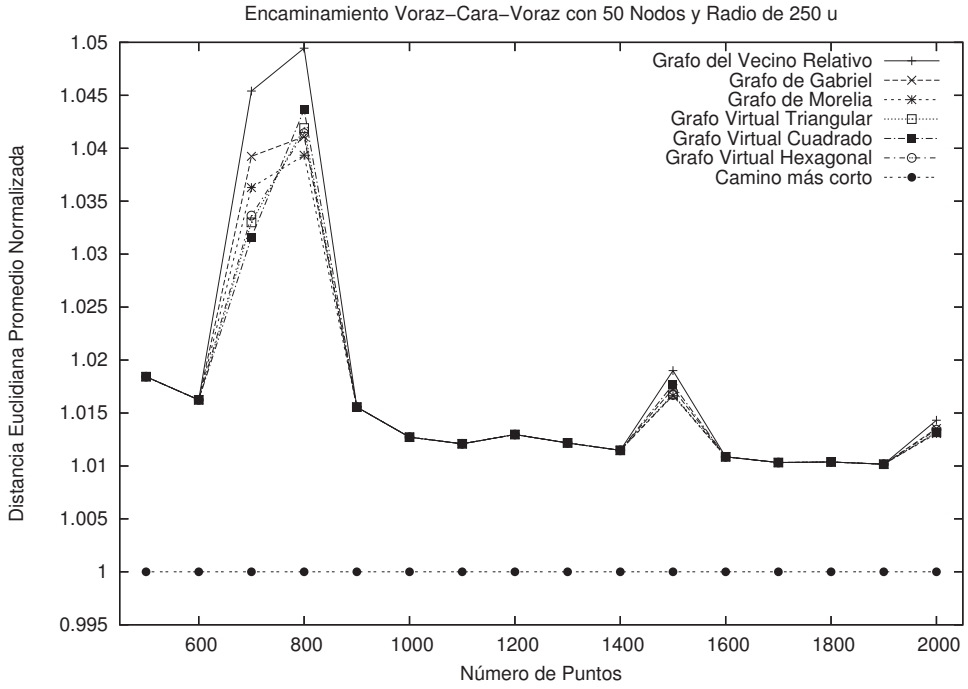


Figura 8.8: GFG variando el número de puntos vs distancia euclidiana.

En la tabla 8.4 se muestran los incrementos que tiene la distancia euclidiana promedio con los distintos grafos al ser comparados con el camino más corto. Los grafos virtuales logran tener el mejor desempeño, ya que su incremento es aproximadamente por la mitad del grafo de Morelia al usar encaminamiento por caras. En el caso del encaminamiento GFG no se aprecia lo suficiente las variaciones con los distintos grafos, debido a que la mayor parte de las veces el encaminamiento es voraz, por lo tanto no se requiere de un grafo plano. El encaminamiento voraz tiende al camino más corto cuando la densidad de la red se incrementa.

Tipo	Encaminamiento por caras	Voraz-Cara-Voraz
Grafo de Vecindad Relativa	933.26 %	1.37 %
Grafo de Gabriel	850.39 %	1.32 %
Grafo de Morelia	813.06 %	1.31 %
Grafo Virtual Triangular	442.85 %	1.30 %
Grafo Virtual Cuadrado	405.01 %	1.30 %
Grafo Virtual Hexagonal	365.77 %	1.30 %

Cuadro 8.4: Incremento porcentual de la distancia respecto al camino más corto

Capítulo 9

Conclusiones

Se ha mostrado que cuando se realiza encaminamiento basado en la posición con el Grafo Virtual en una red inalámbrica *ad-hoc* se tiene un mejor desempeño respecto a los algoritmos tradicionales que extraen subgrafos planos de la red. La razón de lo anterior, es porque el Grafo Virtual no discrimina o ignora aristas de la red, como lo hacen el Grafo de Gabriel o el Grafo de Morelia. Al considerar todas las aristas de la red para encaminar los resultados se mejoran notablemente de acuerdo a la métrica usada, ya que si se quiere cancelar el efecto multisalto para tener pocas transmisiones, el Grafo Virtual lo hace bastante bien, en cambio si se quiere usar el efecto multisalto para preservar la energía del transmisor, también se logra un rendimiento bastante aceptable.

Otra ventaja del Grafo Virtual es que la obtención es sencilla para un transmisor, con lo cual su implementación en algún dispositivo requiere de pocos recursos de cómputo y almacenamiento. Por lo general, los transmisores son aparatos con pocas prestaciones, por ejemplo los sensores.

Hay varios problemas abiertos como trabajo futuro en esta área. De los cuales podemos citar los siguientes:

- Generalizar el Grafo Virtual con romboides, ya que al variar los ángulos internos del rombo y su longitud se puede obtener el Grafo Virtual Cuadrado. Con la generalización se espera poder llegar a configuraciones en las cuales no se requieran tantas revisiones para evitar cruces entre nodos virtuales.

- Aplicar la triangulación de Delaunnay como heurística para obtener aristas virtuales. Se espera que de esta forma el Grafo Virtual resultante tenga una valencia mayor y

lo anterior, resulte en un mejor desempeño al encaminar.

- Considerar la posibilidad de recorrer la cara simultáneamente en los dos sentidos posibles, es decir, usando la regla de la mano derecha y la de la mano izquierda. A la propuesta anterior le denominamos recorrer la cara usando dos agentes.

Bibliografía

[Barriére01] Barriére, L., Fraigniaud, P., y Narayanan, L. Robust position-based routing in wireless ad hoc networks with unstable transmission ranges. *En DIALM '01: Proceedings of the 5th international workshop on Discrete algorithms and methods for mobile computing and communications*, págs. 19–27. ACM Press, 2001.

[Basagni98] Basagni, S., Chlamtac, I., Syrotiuk, V. R., y Woodward, B. A. A distance routing effect algorithm for mobility (DREAM). *En MobiCom '98: Proceedings of the 4th annual ACM/IEEE International Conference on Mobile Computing and Networking*, págs. 76–84. ACM Press, 1998.

[Bondy76] Bondy, J. y Murty, U. *Graph Theory with Applications*. Elsevier North-Holland, 1976.

[Boone04] Boone, P., Chavez, E., Gleitzky, L., Kranakis, E., Opartny, J., Salazar, G., y Urrutia, J. Morelia test: Improving the efficiency of the gabriel test and face routing in ad-hoc networks. *Lecture Notes in Computer Science*, 3104:23–24, January 2004.

[Bose01] Bose, P., Morin, P., Stojmenovic, I., y Urrutia, J. Routing with guaranteed delivery in ad-hoc wireless networks. *ACM/Kluwer Wireless Networks*, 7(6):609–616, 2001.

[Camp02] Camp, T., Boleng, J., Williams, B., Wilcox, L., y W.Navidi. Performance evaluation of two location based routing protocols. *En Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, págs. 1678–1687. 2002.

[Gabriel69] Gabriel, K. y Sokal, R. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[Giordano01] Giordano, S., Stojmenovic, I., y Blazevie, L. Position based routing algorithms for ad hoc networks: a taxonomy, 2001.
URL `http://www.site.uottawa.ca/-ivan/routing-survey.pdf`

[Haas02] Haas, Z., Pearlman, M., y Samar, P. The zone routing protocol (ZRP) for ad-hoc networks IETF internet draft, July 2002.

[Hou86] Hou, T. y Li, V. Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications*, 34(1):38–44, 1986.

[IEEE97] IEEE. Computer society LAN MAN standards committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 802.11-1997*, 1997.

[IET] IETF manet charter. `http://www.ietf.org/html.charters/manet-charter.html`.

[Jacquet01] Jacquet, P., Mühlethaler, P., Clausen, T., Laouiti, A., Qayyum, A., y Viennot, L. Optimized link state routing protocol. *En IEEE INMIC'01, 28-30 December 2001, Lahore, Pakistan*. IEEE, IEEE, December 2001.
URL `http://hipercom.inria.fr/olsr/inmic2001.ps`

[Johnson96] Johnson, D. B. y Maltz, D. A. Dynamic source routing in ad hoc wireless networks. *En* Imielinski y Korth, eds., *Mobile Computing*, tomo 353. Kluwer Academic Publishers, 1996.

[Jubin87] Jubin, J. y Tornow, J. D. The DARPA packet radio network protocol. *Proceedings of the IEEE*, 75(1):21–32, January 1987.

[Karp00] Karp, B. y Kung, H. T. GPSR: Greedy perimeter stateless routing for wireless networks. *En Mobile computing and Networking*, págs. 243–254. ACM Press, 2000.

[Ko98] Ko, Y.-B. y Vaidya, N. H. Loaction-aided routing (lar) in mobile ad hoc networks. *En MOBICOM '98: Proceedings of the 4ht annual ACM/IEEE*

internacional conference on Mobile computing and Networking, págs. 66–75. ACM Press, 1998.

[Kranakis99] Kranakis, E., Singh, H., y Urrutia, J. Compass routing on geometric networks. *En Proc. 11 th Canadian Conference on Computational Geometry*, págs. 51–54. Vancouver, August 1999.

[Kuhn02a] Kuhn, F., Wattenhofer, R., y Zollinger, A. Asymptotically optimal geometric mobile ad-hoc routing. *En Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications*, págs. 24–33. ACM Press, 2002. ISBN 1-58113-587-4.

[Kuhn02b] Kuhn, F., Wattenhofer, R., y Zollinger, A. Geometric ad-hoc routing for unit disk graphs and general cost models. Inf. Téc. 373, ETH Zürich, 2002.

[Larson98] Larson, T. y Hedman, N. *Routing Protocols in Wireless Ad-hoc Network - A Simulation Study.* Proyecto Fin de Carrera, Luleå University of Technology, Stockhol, Sweden, 1998.

[Li03a] Li, X.-Y., G. Calinescu, P.-J. W., y Wang, Y. Localized delaunay triangulation with applications in ad hoc wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(9), 2003.

[Li03b] Li, X.-Y., Stojmenovic, I., y Wang, Y. Partial delaunay triangulation and degree limited localized bluetooh multihop scatternet formation. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, págs. 350–361, April 2003.

[Macker98] Macker, J. y Corson, M. Mobile ad hoc networking and the IETF. *ACM Mobile Computing and Communications Review*, 2(1):9–14, 1998.

[Mauve01] Mauve, M., Widmer, J., y Hartenstein, H. A survey on position-based routing in mobile ad-hoc networks. *IEEE Network Magazine*, 15(6):30–39, 2001.

[Morin01] Morin, P. R. *Online Routing in Geometric Graphs.* Tesis Doctoral, Carleton University, Ottawa, Canada, 2001.

[Navas97] Navas, J. C. y Imielinski, T. Geocast geographic addressing and routing. *En Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, págs. 66–76. ACM Press, 1997.

[Nelson87] Nelson, R. y Kleinrock, L. The spatial capacity of a slotted ALOHA multihop packet radio network with capture. *IEEE*, 32(6):684–694, June 1987.

[Perkins94] Perkins, C. y Bhagwat, P. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *En ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, págs. 234–244. 1994.

[Perkins99] Perkins, C. y Royer, E. Ad-hoc on-demand distance vector routing. *En Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, págs. 90–100. February 1999.

[Rao03] Rao, A., Ratnasamy, S., Papadimitriou, C., Shenker, S., y Stoica, I. Geographic routing without location information. *En MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, págs. 96–108. ACM Press, 2003.

[Royer99] Royer, E. y Toh, C. A review of current routing protocols for ad-hoc mobile wireless networks, 1999.
URL `citeseer.ist.psu.edu/royer99review.html`

[Takagi84] Takagi, H. y Kleinrock, L. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984.

[Toussaint98] Toussaint, G. The relative nighbourhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1998.

[Wang03] Wang, Y. y Li, X.-Y. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. *En DIALM-POMC '03: Proceedings of the 2003 joint workshop on Foundations of mobile computing*, págs. 59–68. ACM Press, 2003.

[Yao82] Yao, A. C. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.

Índice alfabético