Evaluación de Métricas en Redes de Computadoras

Gagliardi, Edilma Olinda
Berón, Mario Marcelo
Departamento de Informática
Facultad de Ciencias Físico, Matemáticas y Naturales
Universidad Nacional de San Luis – Argentina
e-mail: {oli, mberon}@unsl.edu.ar
Fax: 54-2652-430224

Hernández Peñalver, Gregorio
Departamento de Matemática Aplicada
Facultad de Informática
Universidad Politécnica de Madrid – España
e-mail: gregorio@fi.upm.es
Fax: 34-91-3367426

Resumen

Los algoritmos de ruteo juegan un papel central en la transmisión de paquetes en una red de computadoras. En la actualidad existe un conjunto de algoritmos que funcionan adecuadamente con ciertas características de la red, mientras que en otras su desempeño decae significativamente. Estas situaciones pueden ser estimadas a través de la evaluación de métricas.

En este artículo se presenta un Simulador para la Evaluación de Algoritmos de Ruteo en Redes de Computadoras, cuya finalidad es la de proporcionar un medio automático para la evaluación de algoritmos de ruteo tradicionales como así también aquellos que usan información geográfica, sujetos a un conjunto de métricas. Esto permite la detección de estados y organizaciones de la red que conducen a un bajo desempeño de los mismos, posibilitando la creación de heurísticas que controlen esta situación.

Palabras Claves: Redes de Computadoras, Desempeño, Métricas, Algoritmo de Ruteo, Grafos, Geometría Computacional.

1. Introducción

Las redes de computadoras han adquirido gran importancia en la actualidad debido al amplio uso que se hacen de ellas. Un problema de interés en este ámbito es el estudio del comportamiento de los algoritmos usados en las redes de computadoras para enviar un paquete desde un servidor origen a otro destino. Existen diferentes razones para realizar este estudio, entre las cuales se destacan:

- La necesidad de mejorar el desempeño de los algoritmos de ruteo existentes, ya que éstos son eficientes para ciertas situaciones, mientras que en otras decae sustancialmente. Estas características son cuantificables a través de métricas que permiten determinar si el algoritmo de ruteo afecta positivamente o no el comportamiento de la red.
- El deseo de evaluar y comparar un conjunto de nuevos algoritmos de ruteo, cuyo estudio se halla enmarcado en el ámbito de la Geometría Computacional, y son aplicables a las redes estándares e inalámbricas, con los algoritmos de ruteo usados en la actualidad.

De este modo, al elegir e implementar estrategias de ruteo de paquetes que brinden un comportamiento adecuado en una red, considerando las métricas de interés[1], permitiría minimizar la memoria utilizada, mejorar la adaptabilidad a la ubicación de los nodos en la red y además obtener, en un tiempo aceptable, la ubicación del nodo destino, evitando problemas de pérdidas de mensajes y congestionamiento de la red. Los problemas de ruteo en redes se han estudiado en la Teoría de Grafos y Ciencias de la Computación, modelando las redes como grafos.

[1] tiempos, ancho de banda, recuperabilidad, escalabilidad, confiabilidad, congestionamiento de tráfico, entre otros.

Las investigaciones que se están realizando en el área de la Geometría Computacional brindan un marco importante para la incorporación de nuevos algoritmos de ruteo en redes [5,6,7,8,12,13,16,17]. La inserción de nuevos algoritmos puede llegar a minimizar la complejidad espacial y temporal en la problemática de las comunicaciones, produciendo un cambio en la forma en que actualmente se implementan.

A través de esta investigación se pretende aportar información que pueda ser utilizada en la creación de heurísticas que mejoren el desempeño de los algoritmos utilizados actualmente, y proporcionar una visión del comportamiento de las nuevas tendencias de algoritmos de ruteo.

La construcción de una herramienta que automatice la evaluación de algoritmos de ruteo en redes de computadoras teniendo en cuenta un conjunto de métricas, es de gran utilidad para los expertos en esta temática porque ayuda a la selección del algoritmo que a posteriori puede ser incorporado en un software de base como lo es un sistema operativo. Además permite una evaluación de los mismos a un bajo costo ya que se trabaja con una simulación sobre entidades virtuales.

El tema del ruteo en redes de computadoras ha sido tratado por diversos grupos de investigación, quienes en sus publicaciones proveen informes detallados de los algoritmos existentes y la definición de algoritmos [1] nuevos que utilizan a la Geometría Computacional [2,4,9,10,11,15,18,20,21], estos algoritmos pueden modelarse informalmente como sigue: supongamos que un turista llega a una ciudad y desea encontrar el camino hasta la Catedral, cuya torre es visible desde cualquier punto de la ciudad. El turista no tiene plano de la ciudad y se encuentra en una plaza desde la que parten varias calles. La elección natural será caminar por la calle cuyos “puntos” sean lo más cercanos a su destino. Repitiendo el proceso en cada cruce de calles, ¿alcanzará la Catedral?.

En términos matemáticos podemos modelizar el problema así: el sistema de calles de la ciudad representa un grafo geométrico plano, es decir, un grafo en el que los vértices se representan por puntos, y las aristas por segmentos de recta.

El algoritmo que aplica el turista, designado por el término “ruteo con brújula”[14,] se describe del siguiente modo: desde un vértice inicial *s* se desea viajar hasta un vértice destino *d* en un grafo geométrico. La información disponible en cada nodo es local, sólo conoce las coordenadas del punto y las de sus vecinos en el grafo. Cuando alcanza un nodo *v* continúa el camino por la arista incidente a *v* cuya pendiente es la más próxima a la pendiente del segmento que conecta *v* con el nodo destino *d*. Este algoritmo alcanza su objetivo en numerosas ciudades, pero en otras se necesita una estrategia diferente.

En los algoritmos de ruteo compacto (por ej. “interval routing” o “boolean routing”) cada nodo almacena una copia de un algoritmo distribuido, lo que puede llegar a ser muy costoso pues necesita almacenar gran cantidad de información en cada nodo.

Se está trabajando ahora en algoritmos de ruteo con las siguientes características:

- *Información local*: en cada nodo sólo se conoce la posición de sus vecinos.
- *Memoria limitada*: el turista sólo recuerda un número constante de nodos ya visitados, así como las coordenadas del origen y el destino.
- *Ecológico*: no se permite al turista dejar marcas en los nodos visitados.
- *Decisiones locales*: la elección del camino a seguir en cada nodo se basa exclusivamente en la información local almacenada en el nodo y en la información que lleva el turista (de tamaño constante).

Si bien los avances en este ámbito son de gran importancia, se carece de una herramienta que permita la evaluación y comparación de algoritmos de ruteo de paquetes en redes de computadoras en general de forma automática.

Teniendo en cuenta estas observaciones, construimos un *Simulador para la Evaluación de Algoritmos de Ruteo en Redes de Computadoras* con el fin de poder realizar un análisis detallado del funcionamiento de los mismos sujeto a un conjunto de métricas que el usuario del sistema considere importantes [3,19].

Para la construcción del simulador se utiliza el modelo de grafos, ya que provee una representación directa de una red de computadoras y un conjunto de operadores que facilita la implementación.

Las métricas incorporadas inicialmente en nuestro simulador son: distancia de enlace y sobrecarga de nodos. Se pretende que en el futuro se pueda ampliar el simulador para que permita la evaluación de métricas en general, es decir, que el usuario incorpore una nueva métrica al simulador y provea los algoritmos necesarios para su evaluación.

Se pretende que esta herramienta sea usada por los investigadores y usuarios en general que se dediquen a la creación de nuevos algoritmos de ruteo como así también a aquellos que estudien las redes en general.

2. El Simulador

En la figura 1 se puede observar la visión abstracta del simulador.

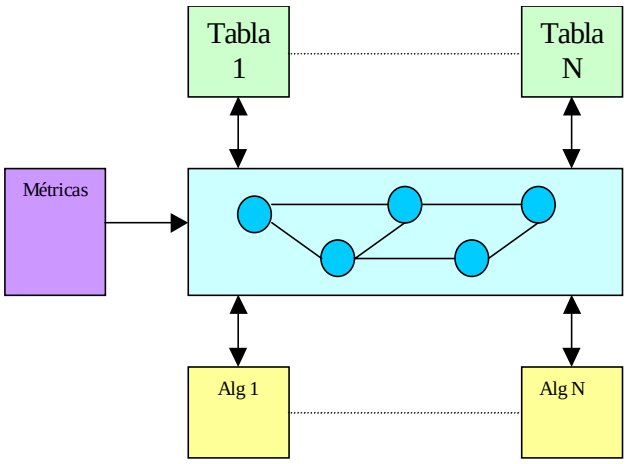

Fig.1: Visión abstracta del Simulador

El simulador posee dos modos de operación: *manual* y *automático*. El modo manual está pensado para la evaluación de algoritmos de ruteo para redes particulares que son ingresadas por el usuario a través de una interfaz gráfica. La interfaz gráfica muestra una representación visual de la red y, además, detalla el camino seguido por el paquete desde el nodo origen al destino, como así también la evaluación de las métricas del algoritmo de ruteo seleccionado por el usuario. El modo automático, tiene por objetivo la realización de ejecuciones sucesivas de los algoritmos de ruteo para diferentes tipos de red. La cantidad de ejecuciones y el tamaño de la redes son ingresadas por el usuario. Los resultados de la evaluación de las métricas para los diferentes algoritmos son plasmados en tablas que el usuario puede visualizar. La figura 2 muestra la visión interna del simulador, en donde se destacan las componentes internas como así también las métricas incorporadas actualmente. En las secciones siguientes, cada componente será explicada en detalle.

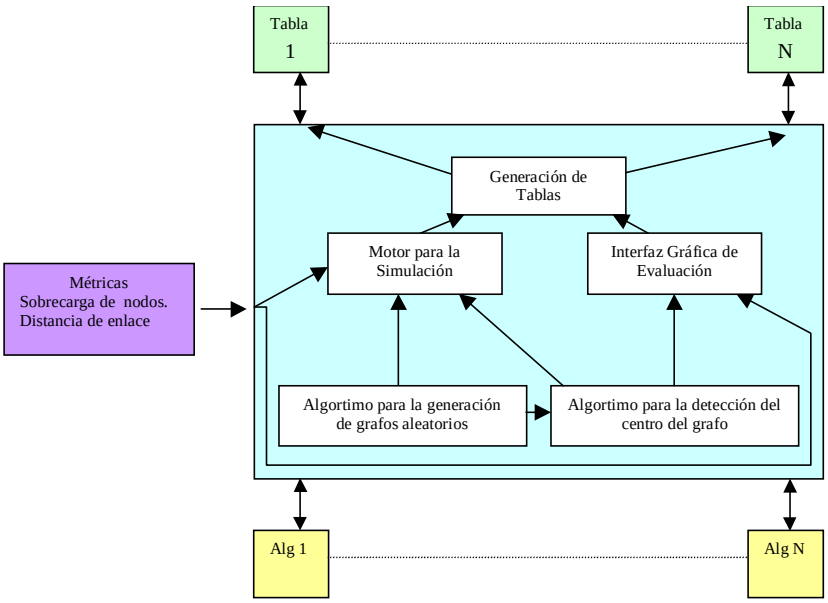

Fig.2: Visión Interna del Simulador

2.1. Descripción de las Componentes del Simulador

Se presenta aquí la descripción de los modos de operación del simulador, las componentes de cada uno de ellos y los algoritmos incorporados en el mismo para realizar el estudio de casos particulares.

2.1.1. Modo Automático

En objetivo del modo automático es generar las tablas que muestren el resultado de la evaluación de las métricas incorporadas en el simulador para grandes muestras de redes aleatorias. El modo automático consta de un módulo para la generación de grafos aleatorios y otro para la selección de los nodos origen y destino del paquete que se desea enviar. En esta sección se presenta la descripción de las componentes que forman parte del modo automático del simulador.

2.1.1.1. Generación de redes aleatorias

El simulador consta de dos métodos para la generación de grafos aleatorios. Ambos métodos asumen el siguiente modelo de grafos, según la figura 3:

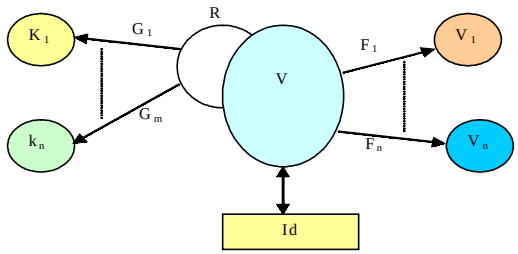

Fig.3 : Modelo de Grafos utilizado por los Generadores de Grafos Aleatorios

La entidad *V* representa el conjunto de nodos de la red. El atributo *Id* identifica a un nodo. Los atributos F_i son datos propios del nodo como por ejemplo: número de veces en que el nodo fue usado en el envío de paquetes, capacidad de procesamiento, etc.

La relación *R* definida en *V* x *V* representa la conexión entre los nodos de la red. Los atributos de la relación G_j, indican propiedades tales como: medio de transmisión, capacidad del medio de transmisión, etc..

De esta manera si *x* es un nodo de *V*, entonces:

- *Id(x):* retorna el identificador del nodo *x*.
- $F_i(x)$*:* retorna el valor el atributo F_i de *x*.

Lo mismo sucede con los arcos, si *w* es un arco de la relación entonces:

- *R(w):* retorna los nodos origen y destino del arco *w*.
- $G_j(w)$*:* retorna el valor del atributo $G_j(w)$ del arco *w*.

Además, éstos métodos tienen en común la generación de los nodos del grafo. Para ello, se utiliza el generador de números aleatorios del lenguaje de programación para obtener las coordenadas *x* e *y* de cada uno de los nodos de la red de computadoras, pero difieren en la construcción de la relación que vincula los nodos.

En las siguientes dos secciones se muestran en detalle ambos métodos de generación de grafos aleatorios.

2.1.1.2. Generación de grafos aleatorios con asignación de probabilidades a los arcos de la relación

Este método genera un grafo de *n* nodos con todos los arcos posibles asignándole una probabilidad a cada uno de ellos. Luego, se modifica el grafo, haciendo una selección de los arcos de la relación original cuya probabilidad se encuentra en un rango preestablecido por el usuario o aleatoriamente por el sistema, de esta forma se obtiene la relación definitiva. La figura 4, muestra el pseudo código de éste método.

Algoritmo: Generación de Grafos Aleatorios (n, $P_{mín}$, $P_{máx}$)
Entrada: *n* es la cantidad de nodos del grafo generado.
$P_{mín}$ *y* $P_{máx}$, representan el rango de probabilidades admisibles para la generación del grafo.
Salida: Un grafo *G=(V,R)* aleatorio, donde *V* es el conjunto de vértices que posee como funciones de asignación a puntos sus coordenadas x e y en el plano, y *R* el conjunto de arcos que posee como función de asignación a arcos una probabilidad de aparición denotada por Prob.
Método:

```
1.  V ← ∅
2.  R ← ∅
3.  Mientras n ≠ 0 hacer
4.    x ← random()
5.    y ← random()
6.    Si  coordenadasDistintas (V,x,y)  entonces
7.      id ← n; V← V ∪ {(id,x,y)}; n ← n – 1
8.    Fin Si
9.  Fin Mientras
10. Para todo vértice i ∈ V hacer
11.   Para todo vértice j ∈ V hacer
12.     R ← R ∪ {((i,j), generarProbabilidad(id(i),id(j)))}
13.   Fin Para todo
14. Fin Para todo
15.  Para todo  arco  a ∈ R   hacer
16.    Si  (Prob(a) < Pmín) ∨ (Prob(a) > Pmáx) entonces R ←R-{(R(a), P(a))}
17. Fin Para todo
18. (V,R) ← conectarComponentes(V,R)
19.  Retornar (V,R).
```

Fig. 4: Pseudo Código del Método 1 de generación de grafos aleatorios

Las funciones:

- *random():* genera un número aleatorio.
- *coordenadasDistintas(V,x,y):* retorna verdadero si en el conjunto de vértices *V* no existe un nodo con coordenadas *x* e *y*. En otro caso retorna falso.
- *generarProbabilidad(o,d):* asigna una probabilidad de aparición al arco con origen *o* y destino *d*.
- *conectarComponentes(V,R):* calcula las componentes conexas del grafo aleatorio. Si existen más de una las conecta .

2.1.1.3. Generación de Grafos Aleatorios por Cálculo de Distancia entre los nodos

Este método también involucra dos etapas; la primera consiste en la generación de los nodos de la misma forma que la técnica anterior y luego se calcula la distancia entre los diferentes nodos del grafo conectando aquellos cuya distancia es menor o igual que un parámetro *d* dado. La figura 5, presenta el pseudo código del método.

Algoritmo: Generación de Grafos Aleatorios (n, d)
Entrada: n es la cantidad de nodos del grafo generado.
d es la máxima distancia aceptada.
Salida: Un grafo G=(V,R) aleatorio, donde V es el conjunto de vértices y R el conjunto de arcos.
Método:

1. $V \leftarrow \varnothing$
2. $R \leftarrow \varnothing$
3. **Mientras** $n \neq 0$ **hacer**
4. $x \leftarrow$ **random()**
5. $y \leftarrow$ **random()**
6. **Si coordenadasDistintas (**V,x,y**) entonces** $id \leftarrow n$; $V \leftarrow V \cup \{(id,x,y)\}$; $n \leftarrow n - 1$**Fin Si**
7. **Fin Mientras**
8. **Para todo** i tal que $i \in V$ **hacer**
9. **Para todo** j tal que $j \in V$ **hacer**
10. **Si** distancia(i,j) $\leq d$ **entonces** $R \leftarrow R \cup \{(id(i),id(j))\}$ **Fin Si**
11. **Fin Para todo**
12. **Fin Para todo**
13. (V,R) $\leftarrow$ **conectarComponentes(**V,R**)**
14. **Retornar** (V, R)

Fig. 5: Pseudo Código para el Método 2 de generación de grafos aleatorios

Las funciones *random(), coordenadasDistintas(V,x,y), conectarComponentes(V,A),* realizan las mismas tareas que para el método 1. La función *distancia(i,j)* calcula la distancia euclidiana entre el nodo *i* y el nodo *j*.

2.1.2. Modo Manual

Como mencionamos, el simulador dispone de un modo manual de trabajo. El propósito de esta utilidad es poder realizar estudios de redes particulares y ver el funcionamiento de los algoritmos de ruteo disponibles en el simulador u otros que el usuario incorpore al mismo.

El modo manual consta de una interfaz gráfica, amigable al usuario, para la visualización de las redes de computadoras. Las funcionalidades provistas por la interfaz son las siguientes:

- *Ingreso de redes, por parte del usuario, en forma manual:* ésta es una característica importante ya que si bien el simulador está preparado, para realizar el estudio de redes en general. En ciertas situaciones para poder establecer resultados generales, es necesario partir de estudios

particulares; esto es equivalente a decir, en el ámbito de las redes de computadoras, que el estudio del comportamiento de los algoritmos de ruteo para redes regulares es una base para inferir el comportamiento de redes en general.

- *Inicialización de los parámetros para los algoritmos de ruteo:* a través de la experiencia en el entendimiento e implementación de los algoritmos de ruteo se detectó que ciertos algoritmos requieren parámetros adicionales a los ya conocidos (red, nodo origen, nodo destino), como por ejemplo: cantidad de memoria utilizada, longitud del camino, etc.
- *Ejecución parcial: e*sta funcionalidad permite ver paso *a paso* los nodos que el algoritmo de ruteo que se está ejecutando selecciona. Es útil para aquellos algoritmos en donde la selección del nodo no es determinística.
- *Ejecución total:* ejecuta el algoritmo de ruteo en su totalidad y muestra el camino correspondiente.
- *Visualización de las tablas con los resultados de las métricas*: es necesaria para el análisis de los resultados.

2.1.3. El Motor para la Simulación

El motor para la simulación puede operar de dos maneras. En la primera se generan los nodos origen y destino aleatoriamente y luego se ejecutan los algoritmos de ruteo disponibles en el simulador. En la segunda se ejecutan los algoritmos de ruteo para todo par de nodos del grafo. En la figura 6, se presenta el pseudo código correspondiente a este módulo.

Algoritmo: Motor para la Simulación (G,m,c, met)
Entrada: G = (V,R) un grafo aleatorio, donde V es el conjunto de nodos y R el conjunto de arcos.
m un conjunto de métricas.
c el conjunto de nodos que forman el centro del grafo.
método indica el tipo de ejecución del motor para la simulación.
Salida: Consiste en un conjunto de tablas con los valores de las métricas para cada uno de los algoritmos.
Método:

```
1.  Si método = "todos" entonces
2.    Para todo i tal que i ∈ V hacer
3.      Para todo j tal que j ∈ V hacer
4.        Para todo A tal que A es un algoritmo de Ruteo disponible en el simulador hacer
5.          Ejecutar el algoritmo A con origen i y destino j
6.          Sea C el camino entre i y j retornado por el algoritmo A entonces
7.          evaluarDistanciaDeEnlace(C)
8.          evaluarSobrecargaDeNodos(C)
9.        Fin Para Todo
10.     Fin Para todo
11.   Fin Para Todo
12. sino
13.   n = númeroDeEjecuciones();
14. Mientras n ≠ 0 hacer
15.   o= nodoAleatorio();
16.   d = nodoAleatorio();
17.   Para todo A tal que A es un algoritmo de Ruteo disponible en el simulador
18.     Ejecutar el algoritmo A con origen o y destino d.
19.     Sea C el camino entre o y d retornado por el algoritmo A entonces
20.     evaluarDistnaciaDeEnlace(C,o,d)
21.     evaluarSobrecargaDeNodos(C,o,d)
22.   Fin Para todo
23. Fin Mientras
24. Fin Si
```

Fig. 6: Pseudo Código para el motor de la Simulación

Las funciones:

- *NúmeroDeEjecuciones():* retorna como resultado un entero que indica la cantidad de ejecuciones que se desean realizar.
- *NodoAleatorio():* retorna un identificador de nodo en forma aleatoria.
- *EvaluarDistanciaDeEnlace(Camino,Origen,Destino):* cuenta la longitud del *Camino* encontrado entre el nodo *Origen* y el *Destino*.
- *EvaluarSobrecargaDeNodos(Camino, Origen,Destino):* contabiliza la cantidad de veces que los nodos del *Camino* fueron utilizados en los caminos seleccionados por los algoritmos de ruteo disponibles en el simulador.

2.1.4. Casos Especiales

En este apartado cabe mencionar que el simulador consta de un conjunto de rutinas cuya finalidad es la de estudiar un caso especial, como lo es el análisis de la sobrecarga del centro del grafo, entendiéndose por este concepto al conjunto nodos que se encuentran más cercanos a los vértices más distantes en el grafo. El procedimiento implementado en el simulador, para este caso, puede ser visto en detalle en [1]. Esta funcionalidad fue incorporada en el simulador debido a que se desea probar, en principio, empíricamente y, luego, de ser posible, analíticamente si los algoritmos de ruteo disponibles en el simulador sobrecargan esta clase de nodos.

3. Los Algoritmos de Ruteo

El simulador está preparado para recepcionar cualquier algoritmo de ruteo. Para los algoritmos que usan información geográfica, el simulador genera grafos geométricos con las siguientes características: los nodos pueden ser localizados por sus coordenadas en el plano. Cada nodo conoce las coordenadas de si mismo y la de sus vecinos.

El grafo total no se conoce, se descubre a través de la exploración. En el caso de los algoritmos sin memoria, la elección del próximo nodo al cual se el envía el paquete se realiza teniendo en cuenta la posición de sus vecinos. Para los algoritmos con memoria, el próximo nodo se selecciona tomando en cuenta la información disponible en la memoria y el conocimiento de los vecinos del nodo corriente.

En principio, la interfaz entre los algoritmos y el simulador está dada por el grafo que representa la red de computadoras, el nodo origen, el nodo destino, la cantidad de memoria requerida y la longitud del camino para los algoritmos que así lo requieran. De esta forma, el usuario que desee incorporar un nuevo algoritmo de ruteo debe implantarlo con estos parámetros de entrada.

El simulador consta actualmente de los siguientes algoritmos de Ruteo:

- *Dijkstra:* es el algoritmo óptimo en cuanto a la selección del camino ya que siempre encuentra el camino más corto de un nodo contra todos los demás y generalmente se toma como punto de referencia para comparar y medir la métrica distancia de enlace de otros algoritmos.
- *Breadth First, Depth First:* estos algoritmos difieren en la política de selección del próximo nodo a elegir en el camino, y es interesante estudiar ¿cómo dicha política afecta a la distancia de enlace y sobrecarga de nodos en la red?.
- *Ruteo por Brújula:* es uno de las nuevas propuestas de algoritmo de ruteo para el envío de un paquete desde un nodo origen a otro destino. Este algoritmo funciona de la siguiente forma: desde un vértice inicial *s* se desea viajar hasta un vértice destino *d* de un grafo. La información disponible en cada nodo es local, sólo se conoce las coordenadas del punto y las de sus vecinos inmediatos en el grafo. Cuando se alcanza un nodo *v*, continua por la arista incidente a *v* cuya pendiente es la más próxima a la determinada por el segmento que conecta a *v* con el nodo destino *d*. Este algoritmo junto con otros tales como: *Ruteo por Brújula Aleatorizado, Ruteo*

Ávido, brindan una forma diferente de concebir el envío de paquetes en una red de computadoras. Por lo tanto es importante poder analizar su desempeño para métricas tales como las presentadas en este artículo.

4. Experimentos y Ejemplos

Los experimentos realizados con el simulador estuvieron orientados a la comprobación de la siguiente hipótesis: el algoritmo de ruteo de Dijkstra sobrecarga el centro del grafo. La verificación de esta hipótesis es compleja, por lo tanto se decidió acotar el problema probando si ésta es verdadera para una clase particular de grafos: *las retículas cuadradas*.

La selección de este tipo de grafo se debe a que su forma hace más factible la elaboración de resultados debido a su simplicidad y, además, da la posibilidad de poder aplicar las conclusiones a redes en general.

Suponga una red de computadoras de cuatro nodos de la siguiente forma:

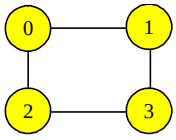

Fig. 7: Retícula de cuatro nodos

El centro de esta red está formado por los nodos {0, 1, 2, 3} por lo tanto cualquier mensaje que se pueda emitir entre los nodos recorrerán el centro del grafo.

Podemos considerar el mismo tipo de interconexión de computadoras pero con nueve máquinas, como lo muestra la siguiente figura:

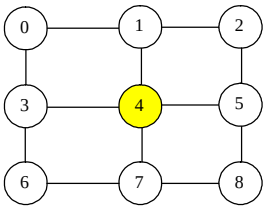

Fig. 8: Retícula de cuatro nodos

El centro del grafo está formado por el nodo 4; de esta forma toma interés la ejecución del algoritmo de Dijkstra para detectar si los mensajes emitidos pasan por el centro del grafo. Dicho algoritmo se ejecuto para retículas de tamaño: 3x3, 4x4, 5x5, 6x6, 7x7, 8x8, 9x9, 10x10, tomándose como nodo origen y destino aquellos que se encuentran en los límites de las retículas. La tabla 1, muestra los resultados que se obtuvieron.

Retícula	*Centro del Grafo*	*Utilización del Centro*
2x2	{0,1,2,3}	100%
3x3	{4}	36%
4x4	{5,6,9,10}	37%
5x5	{12}	20%
6x6	{14,15,20,21}	33.3%
7x7	{24}	14.28%
8x8	{27,28,35,36}	25%
9x9	{40}	7%
10x10	{44,45,54,55}	40%

Tabla 1: Resultados de la Simulación

La conclusión fue que el algoritmo de Dijkstra no sobrecarga los nodos de la red. La razón de esta aseveración se debe a que este algoritmo selecciona siempre el mismo camino más corto, independientemente que si existen otros distintos de la misma longitud. La selección del camino depende de la política de elección del próximo nodo a tratar, por lo tanto este algoritmo tenderá a sobrecargar a aquellos nodos que cumplan con las condiciones de la política de selección.

Un punto a destacar es que la clase de grafos estudiados aquí, presentan como característica la existencia varios caminos más cortos; esto se puede observar trivialmente, en la retícula de la figura 8 cuando se desea computar el camino más corto entre los nodos 0 y 8.

Esta observación conduce a pensar que el algoritmo de Dijkstra puede ser modificado para que seleccione un camino más corto distinto, cada vez que envía un paquete entre un mismo par de nodos. Esta modificación, del algoritmo de Dijkstra, debería ser acompañada por el conocimiento de cuántos caminos distintos entre los pares de la retícula existen, de esta manera cada vez que se desee enviar más de un paquete entre dos nodos, se puede seleccionar para cada uno de ellos un camino distinto. En la figura 9 se muestran retículas de tamaño 2x2 y 3x3, en donde cada nodo esta rotulado con el número de caminos más cortos distintos.

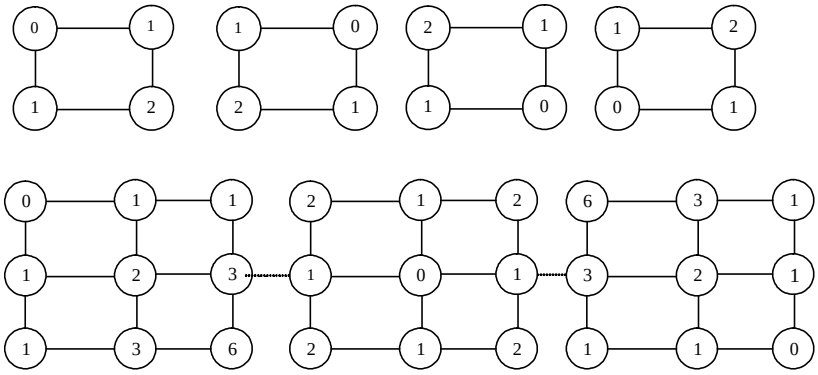

Fig.9: Cálculo de la cantidad de pasos entre los nodos de una retícula

El lector, puede observar el procedimiento para llevar a cabo éste computo. Por lo tanto, el algoritmo de Dijkstra puede ser modificado para que cuando se presente la necesidad de enviar más de un paquete entre un origen y un destino se seleccione un camino distinto. Este algoritmo debería contar con la información de cuántos caminos más cortos existen entre un par de nodos ya que a medida que se envían los paquetes esta cantidad disminuye en uno y cuando el mismo se recibe se incrementa en la misma magnitud, de esta manera se prevé de que cuando no existen más caminos para utilizar, el origen debe esperar hasta que uno de ellos se desocupe, lo cual indica que en el emisor debe existir un mecanismo para la retención de mensajes, un razonamiento similar se aplica al receptor y a los nodos intermedios.

Esta última observación es la base para la incorporación de una nueva métrica en el simulador que contemple la longitud de la estructura que retenga los paquetes. Esta estructura en el ámbito de las redes generalmente es una cola.

En esta sección se mostró la utilidad del modo manual de simulador para el análisis de los algoritmos de ruteo para redes particulares, en la sección siguiente se presentan las conclusiones y trabajos futuros.

5. Conclusiones y Trabajos Futuros

El grupo de trabajo en Geometría Computacional de la UNSL, con asesoramiento de docentes de la UPM dio inicio a un proyecto de investigación conjunto, *Proyecto de la UPM de AL2002-1010-2.43 Geometría Computacional*, con el objetivo principal de consolidar la línea de trabajo en la

UNSL, aportando nuevos enfoques y técnicas algorítmicas a las líneas de investigación ya establecidas en su Departamento de Informática.

Como parte de esta línea de investigación se presentó los avances realizados en la construcción de una herramienta para la evaluación de algoritmos de ruteo de paquetes en redes de computadoras, cuyo objetivo es automatizar la evaluación del comportamiento de los algoritmos de ruteo tradicionales como así también los que usan información geográfica para un conjunto de métricas.

Como experimentos, se utilizó el simulador para verificar la siguiente hipótesis: "*El algoritmo de Dijkstra sobre carga el centro del grafo"*, por simplicidad, se restringió el estudio a una clase de grafos particulares: "*Las retículas cuadradas*", ya que el estudio de casos particulares es una ayuda para el estudio de casos generales. A través del uso del simulador, con éste tipo de grafos, se detectó que el algoritmo de Dijkstra no sobrecarga el centro del grafo sino más bien a aquellos nodos que cumplen con la política de elección del próximo nodo a tratar, esto se debe a que, el algoritmo de Dijkstra, siempre selecciona un único camino más corto, aunque en el grafo existan más de uno. Las retículas cuadradas presentan esta característica. Esto condujo a analizar la posibilidad de construir un algoritmo que compute la cantidad de caminos más cortos en éste tipo de grafos, como así también modificar el algoritmo de Dijkstra para que seleccione un camino más corto distinto cada vez que se rutea un paquete entre un mismo par de vértices.

Los trabajos futuros que se desean realizar en este ámbito son:

- Probar el comportamiento de algoritmos de ruteo como Ruteo por Brújula, Ruteo Ávido, etc. para las métricas incorporadas actualmente en el simulador.
- Simular el envío de paquetes con errores de recepción y proponer versiones de los algoritmos de ruteo con tolerancia a las fallas.
- Ampliar el simulador para que el usuario no ingrese solamente el algoritmo de ruteo, sino también nuevas métricas y el procedimiento para evaluarlas.
- Ampliar el simulador para que simule el comportamiento de redes móviles.

Referencias

1. Aho, A.V.; Hopcroft, J. E.; Ullman, J. *The desing and analysis of computer algorithms*, Addison-Wesley Series in Computer Science and Information Processing, 1974.
2. Berg, Kreveld, Overmars, Schwarzkopf. *Computational Geometry: algorithms and applications*, Springer Verlag, BH 1997
3. Beron, M.; Flores, S.; Gabliardi, O.. *Ruteo con Brújula en Redes sin Cables*. Argentina. CACIC 2001. 2001.
4. Boissonnat, J.D.; Yvinec, M. *Algorithmic Geometry*, Cambridge University Press, 1998.
5. Bose, P.; Brodnik, A.; Carlsson, S.; Demaine, E.; Fleischer, R.; López-Ortiz, A.; Morin, P.; Munro, J.. Online Routing in Convex Subdivision. http://ce.sharif.ac.ir/~ghodsi/archive/Course-Materials/Computational%20Geometry/papers /Online%20Routing%20in%20Convex%20Subdivisions.pdf Canada. 2001.
6. Bose, P.; Morin, P.. *Online Routing in Triangulations*. In Proceedings of the Tenth International Symposium on Algorithms and Computation (ISAAC'99), volume 1741 of Springer LNCS, pages 113-122, 1999.
7. Datta, S.; Stojmenovic, I.; Wu, J.. Internal node and shortcut base routing with guaranteed delivery in wireless networks. In Proc. IEEE Int. Conf. on Distributed Computing and Systems Workshops; Cluster Computing, to appear, pages 461 466, 2001. USA. 2001.

8. Gagliardi, O.; Taranilla, M.; Beron, M.. La Geometría Computacional a nuestro alrededor. III Workshop de Informática y Ciencias de la Computación. Argentina. 2002.
9. Gonnet G.;Baeza Yates, R. *Handbook of Algorithms and Data Structure,* Addison – Wesley, New York, 1991.
10. Goodman, J.; O'Rourke, J. *Handbook of Discrete and Computational Geometry*. CRC Press 1997.
11. Günther, O. *Efficient Structures For Geometric Data Management*. Springer Verlag 1988.
12. Heywood, S. Corneluis, S. Carver, *Geographical Information Systems*. Addison-Wesley Longman, New York, 1998.
13. Karp, B.; Kung, T.. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. Proc of 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking. 2000.
14. Kranakis E., Singh H., J. Urrutia. *Compass rotuing on Geometric network.* http://citeseer.nj.nec.com/kranakis99compass.html. *1999*.
15. Kurt Melhorn *Multidimensional Searching and Computational Geometry*. Springer-Verlag, Berlin, 1984.
16. Latombe J.C., *Robot Motion Plann*ing, Kluwer Academic Publisher, Boston, MA, 1991.
17. Mauve, M.; Widmer, J.; Hartenstein, H.. A Survey of Postion-Based Routing in Mobile Ad-Hoc Networks. http://citeseer.nj.nec.com/cache/papers/cs/25605/http:zSzzSzwww.informatik.uni-mannheim.dezSzinformatikzSzpi4zSzpublicationszSzlibraryzSzMauve2001f.pdf/mauve01survey.pdf Germany. 2001.
18. Pach, J. *New trends in Discrete and Gomputational Geometry*. Springer Verlag NY 1993.
19. Peñalver, G.; Gagliardi, O.; Beron, M.. Evaluación de Algoritmos de Ruteo en Redes de Computadoras. IV Workshop de Informática y Ciencias de la Computación. Argentina.2003.
20. Preparata, F.; Shamos,M. *Computational Geometry: an introduction,*. Springer Verlag, NY 1985.
21. Sack, J.R.; Urrutia, J.. *Handbook of Computational Geometry*. Elsevier Science B.V. 2000.