Propuesta de un protocolo de enrutamiento multidifusión basado en agentes: Multicast Antnet Routing Protocol*

Fernando Solano**, María Angélica Tirado** y Yezid Donoso**

Resumen

Este trabajo de investigación presenta un protocolo de enrutamiento multidifusión, MARP, que usa agentes móviles para hallar rutas eficientes de reenvío de paquetes desde una fuente hacia todos los miembros del grupo multicast. El mecanismo de reenvío de paquetes emplea balanceo de carga sobre las mejores rutas. El protocolo desarrollado usó como base el algoritmo de enrutamiento unidifusión AntNet, el cual se adaptó para trabajar en redes multicast.
Palabras clave: Protocolo de enrutamiento, redes multicast, AntNet, agentes móviles, balanceo de carga.

Fecha de recepción: 23 de octuber de 2002

Abstract

This research paper describes a multicast routing protocol, MARP, which uses mobile agents to find efficient forwarding paths for information packets from a source to all members in a multicast group. The packets forwarding mechanism works with load balancing, using the best paths. The developed protocol used as base AntNet, an unicast routing algorithm. MARP could be consider as a multicast version of AntNet.
Key words: Routing protocol, multicast networks, AntNet, mobile agents, load balancing.

INTRODUCCIÓN

El número de computadores conectados a la red crece de forma exponencial y la necesidad de transmitir la misma información a múltiples destinos, como es el caso de las multi-conferencias (videoconferencias a múltiples puntos), las estaciones de radio en Internet, la réplica de datos a múltiples puntos, juegos y simulaciones por

* Proyecto financiado por la Universidad del Norte según contrato Nº OJD-2002-00059 realizado a través de la Dirección de Investigaciones y Proyectos. Proyecto de Fortalecimiento de Grupos de Investigación.

** Universidad del Norte, Departamento de Ingeniería de Sistemas y Computación, Grupo de Investigación Redes de Computadores. Barranquilla (Colombia). *fsolanod@lycos.com, tiradom@unimail.uninorte. edu.co,* y *donoso@uninorte.edu.co*

la red [5]. Debido a lo anterior fue necesario adoptar una forma de transmisión eficiente a múltiples destinos, ganando recursos en la red; esta forma de transmisión es llamada multidifusión o *multicast* [7]. Para estas redes se definieron varios protocolos de enrutamiento, entre los cuales se tiene DVMRP, BGMP, PIM-DM, PIM-SM, MOSPF y CBT [18]. Ninguno de estos protocolos realiza balanceo de carga sobre las rutas seleccionadas, con el fin de mantener su condición de rutas óptimas por más tiempo, gracias a lo cual se logran mejores resultados en la transmisión. AntNet ha sido escogido como la solución para llevar a cabo el balanceo de carga a través de múltiples árboles.

Se escogió AntNet como base de nuestro trabajo debido a los buenos resultados obtenidos en simulaciones [3,8] frente a otros protocolos unicast, y a que el método heurístico de selección de rutas usado por éste es apropiado para un balanceo de carga en redes multicast.

1. PROPUESTA DE FUNCIONAMIENTO

1.1. **Conceptos básicos**

Un enrutador o nodo es un dispositivo, en una red de computadores, que se encarga de reenviar los paquetes (o información) de tal forma que lleguen a su(s) destino(s). Los enrutadores ejecutan continuamente un algoritmo de enrutamiento, el cual permite determinar la forma de reenvío de información para cumplir este objetivo. Estos algoritmos de enrutamiento normalmente almacenan su información en tablas o estructuras de datos en su memoria.

En multidifusión existe un equipo emisor de datos, el cual es llamado *host* fuente, S. Estos datos son enviados a un grupo de *hosts* receptores, G. Debido a que nuestro trabajo es a nivel de red, no se hablará de los *hosts* en sí sino de los nodos conectados a ellos directamente.

Para el funcionamiento del protocolo se han distinguido 3 tipos de enrutadores en una red funcionando bajo MARP: *nodo*[1] *fuente* (S), *nodos intermedios* (R_i) y *nodos inteligente*[2]*s* (I_i). Para el funcionamiento de MARP se emplearán cuatro tipos de hormigas: *Hormigas de exploración, Hormigas Join, Hormigas Prune* y *las Hormigas cíclicas.*

[1] En adelante se usará de forma indiferente las palabras *nodo* y *enrutador.*

[2] El término *inteligente* (nodo inteligente) se debe a que cada nodo aparenta tomar decisiones en el tiempo teniendo como criterio el estado de la red. No indica que se esté usando un método de inteligencia artificial en el momento. Sin embargo se emplean métodos heurísticos como primera aproximación a un método realmente inteligente.

Los *nodos fuentes (S)* son enrutadores que tienen conectados directamente a sus interfases computadores que envían tráfico a un grupo multidifusión de nodos *G*. Los *nodos intermedios* (R_i) son aquellos que no reenvían el tráfico a otros nodos, pero son candidatos a ser nodos inteligentes. Los *nodos inteligentes* (I_i) son aquellos que reciben el tráfico para reenviarlo, ya sea porque poseen miembros receptores conectados directamente o porque necesitan reenviar el tráfico a otros nodos inteligentes. Dichos nodos usarán lógica borrosa (o *fuzzy*) para analizar las mejores vías de transmisión del flujo y basarán su conocimiento en la información suministrada por los agentes de información en la red: las hormigas de exploración.

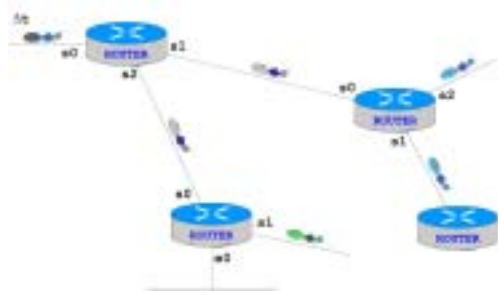
Estos enrutadores poseen para cada flujo (S,G) *un conjunto de interfases de salida*, un *conjunto de interfase de entradas candidatas* y una sola *interfase de entrada activa* o *funcional*. Las interfases de salida, *O*, son usadas por cada nodo *I* para reenviar paquetes **originados** por S con destino G (S,G). Las interfases candidatas a entradas, *I*, son aquellas que no son de salida pero serán utilizadas para recibir el flujo de S, y la interfase de entrada activa, *i*, pertenece al conjunto de interfases candidatas a entrada y es la seleccionada para recibir el flujo en un instante de tiempo. Es claro que se debe cumplir: $i \in I$.

Los enrutadores inteligentes poseen una tabla multicast donde se relaciona, para cada flujo de tráfico multicast (S,G), la interfase de entrada activa y candidatas con su respectiva probabilidad. Si una de estas celdas presenta un valor nulo, indica que es una interfase de salida

	Fuente 1 (S_1)	...	**Fuente n (S_n)**
Interface 1 (i_1)	$P_1(S_1,G)$ %		$P_1(S_n,G)$ %
Interface 2 (i_2)	$P_2(S_1,G)$ %		$P_2(S_n,G)$ %
⋮	⋮	⋮	⋮
Interface *i* (i_i)	$p_i(S_1,G)$ %		$p_i(S_n,G)$ %
⋮	⋮	⋮	⋮
Interface N_k (i_{Nk})	$p_n(S_1,G)$ %		$p_n(S_n,G)$ %

Cada celda puede contener un valor numérico entre 0 a 1 o ser nulo. Una celda nula indica que esa interfase es de salida, mientras que un valor numérico en la celda indica que la interfase es candidata a entrada. Los enrutadores inteligentes poseen una tabla de medias y varianzas (memoria) de los tiempos de recepción de cada fuente u origen del flujo multicast S, lo cual ayudará a determinar el valor de cada celda y, por ende, la mejor interfase para la **recepción** de la información.

1.2. **Las hormigas de exploración**

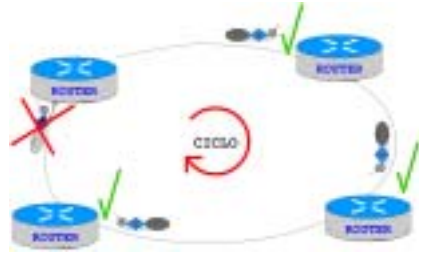


Estas hormigas tienen como propósito descubrir nuevas rutas o evaluar el estado de las ya existentes analizando el retardo en la red. El algoritmo de llenado o *flooding*, el cual actualiza la información de los enrutadores, se describe a continuación:

- A intervalos regulares cada nodo fuente S crea en *buffer*, en forma independiente de los otros nodos fuentes, hormigas de exploración *F* con destino los nodos G. Estas hormigas tienen como propósito descubrir nuevas rutas o evaluar el estado de las ya existentes analizando el retardo en la red.

- Para saltar de un nodo k[3] a otro, cada hormiga es clonada N_k-1 veces, para salir por cada una de las interfases por donde no entró. De esta forma se realiza un llenado en la red de las hormigas de exploración y se descubren y prueban todas las rutas.

- Las hormigas deben ser puestas en las mismas colas del flujo de información, con el fin de que las hormigas sean retardadas el mismo tiempo que los paquetes, lo cual ofrece datos confiables de los tiempo de retardo de la información.



- Al llegar a cada nodo la hormiga de exploración *F* actualiza el tiempo de envío, T, que ha transcurrido desde que fue lanzada. El tiempo es almacenado para un posterior análisis en los nodos I.

- Si la hormiga alcanza un enrutador inteligente, ésta ofrece la información recolectada al agente inteligente dentro del enrutador. En otro caso, la hormiga no es analizada.

- Si la hormiga regresa accidentalmente a un nodo que ya ha visitado, es destruida antes de cualquier análisis.

- Si un enrutador no posee una interfaz por donde reenviar la hormiga de exploración, ésta muere.

[3] Las notaciones y siglas usadas en este artículo corresponden de igual forma a las usadas en la documentación de AntNet en [9].

MARP calcula las rutas hacia los múltiples destinos en un solo sentido, desde S hasta G, ya que así será la orientación del tráfico fluctuante en la red. Asimismo, MARP no emplea hormigas de retorno o de actualización de datos, a diferencia de AntNet, ya que la información recolectada por las hormigas de exploración ya ha sido suministrada a todos los nodos involucrados.

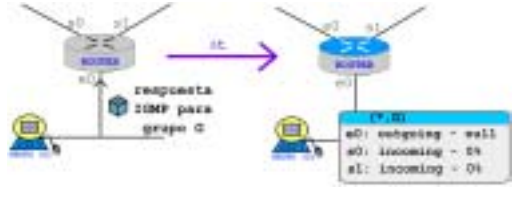
De esta forma se ha logrado que cada nodo en la red conozca los retardos de transmisión desde el nodo fuente S hasta él. Es función y criterio de cada nodo receptor de hormigas seleccionar la interfase por donde desea recibir la información. Esto será detallado más adelante.

1.3. **Nodos inteligentes y el agente inteligente**

Los nodos inteligentes son los encargados de reenviar la información del flujo (S,G) de la mejor forma posible. Estos nodos son los que analizan las hormigas de exploración para conocer su entorno y tomar decisiones.

Un nodo intermedio pasa a ser inteligente si y sólo si recibe una solicitud expresa de reenvío de tráfico. Estas solicitudes pueden ser mensajes IGMP [6,10] recibidos por interfases LAN u hormigas Join enviadas por otros nodos a través de interfases WAN.

Cuando un *host* desea recibir información de un grupo G, éste envía solicitudes IGMP al enrutador designado en su red LAN [13,14]. Estas solicitudes, arribadas por interfases LAN, no indican la unión de una fuente S explícita, por lo que el enrutador inteligente crea una entrada (*,G) indicando que debe analizar cualquier hormiga con destino G procedente de un origen cualquiera. Así, sólo los nodos que posean miembros conectados directamente tendrán entradas (*,G) en sus tablas.



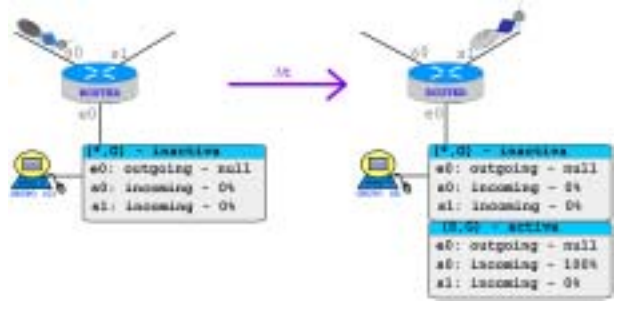
Los nodos inteligentes funcionan de la siguiente manera:

- Al recibir una solicitud IGMP por una interfaz LAN el enrutador pasa a ser inteligente (si no lo era anteriormente) y crea una entrada (*,G) para el grupo solicitado. Esta entrada se marcará como inactiva con lista de interfases de salida aquella por donde se recibió el paquete IGMP y lista de interfases de entrada el resto de interfases.

- Los nodos analizarán solamente las hormigas de exploración que arriben por las interfases que se encuentren como candidatas a entrada.

Como ya se mencionó, cada interfase de salida poseerá una probabilidad nula en la tabla; debido a esto, los cálculos efectuados en la tabla de G no afectarán las celdas que sean para interfases de salida impidiendo que una interfase de salida sea también de entrada.

Una hormiga de exploración será analizada si existe una entrada (S,G) o (*,G) en el nodo inteligente. Si el nodo inteligente posee a (*,G) y (S,G) no existe, el nodo debe crear (S,G) copiando la información de (*,G), como se muestra en la figura. Esto será detallado posteriormente.



1.4. **Hormigas Join y Prune**

El funcionamiento de las hormigas Join y Prune es muy semejante al de los mensajes Join y Prune usados en los protocolos Multicast [1,2,9,11,12,16,17]. Las hormigas Join se encargan de informar a su vecino que «les gustaría recibir» el tráfico a través de él. Por otro lado, las hormigas Prune indican a sus vecinos que no se necesita más sus servicios, es decir, que no se desea recibir más tráfico por esta interfase.

Las hormigas Join y Prune son generadas cada vez que un nodo inteligente considera que se deba cambiar la interfase de entrada del flujo de tráfico. Ambas hormigas serán puestas en colas con alta prioridad durante su transmisión, ya que ellas serán las encargadas de reformar el árbol de expansión.

La interfase de entrada activa, *i*, es seleccionada de forma cíclica[4] de la lista de interfases de entrada candidatas, *I*, la cual es funcional durante la cantidad de tiempo $T_f(i)$ dada por:

$$T_f(i) = T_r \cdot P_i(S,G)$$

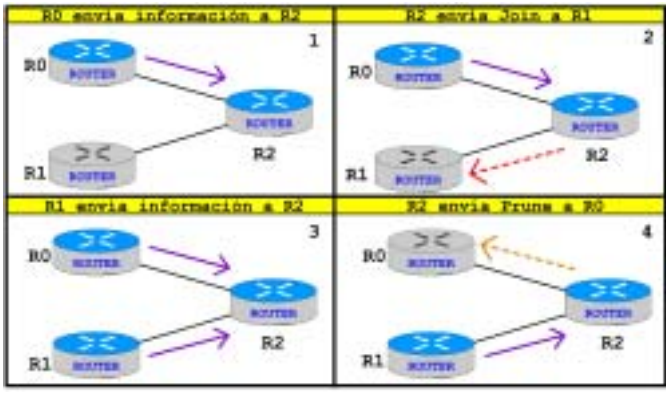
donde T_r es la magnitud en segundos[5] de un ciclo y $P_i(S,G)$ es la probabilidad de la interfase candidata a entrada *i* para el flujo (S,G).

Una vez que ha expirado el tiempo funcional de la entrada activa actual se selecciona la siguiente interfase (de la lista de interfases candidatas a entrada, *I*)

[4] Este proceso es semejante a un Round-Robin, sin embargo se diferencia en que los cuantos de tiempo son variables en el tiempo para cada selección.

[5] Se optó por segundos debido a que medidas muy pequeñas (milisegundos) perjudican la estabilidad de la red, y magnitudes muy grandes (horas) pueden hacer que el protocolo no ofrezca información fiable en cuanto a balanceo de carga.

teniendo en cuenta que el tiempo funcional de la nueva interfase debe ser mayor que un valor umbral Δt. Con esto se logra un balanceo de carga en la red, ya que las interfases de entrada son rotadas en el tiempo y sólo serán usadas aquellas que muestren pequeños tiempos de retardo.



Cada vez que hay un cambio de interfase de entrada activa se envían hormigas Join por la nueva interfase activa. En este momento el enrutador espera paquetes tanto por la interfase activa antigua como por la nueva. Una vez que se haya recibido el primer paquete del flujo por la nueva interfase se envían hormigas Prune por la antigua interfase, eliminando así el flujo de información por la antigua ruta. Los enrutadores esperan el arribo del primer paquete del flujo durante un tiempo determinado. Si pasado este tiempo el paquete no llega, se selecciona otra interfase como entrada activa de la misma forma, como ya se describió. El tiempo de espera del paquete por la nueva interfase es proporcional al tiempo medio de recepción de paquetes para el flujo (S,G), μ, y a la probabilidad de esa interfase, p_i.

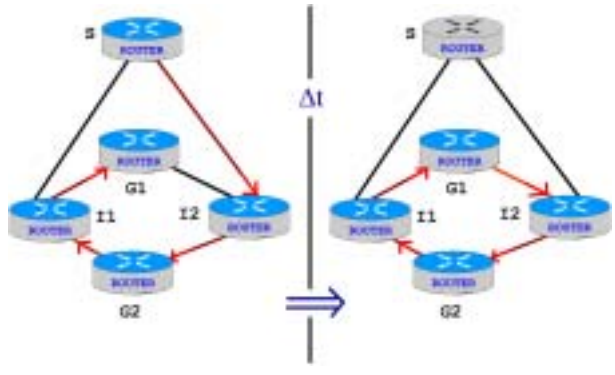
Los nodos intermedios que reciban mensajes Join se convertirán en nodos inteligentes de la siguiente forma:

- Si no existe una entrada (S,G) en el nodo inteligente, se crea inicialmente con interfaz de entrada nula y lista de interfases de salida vacía.
- La entrada (S,G) es marcada como activa.
- La interfase por donde se recibió la hormiga Join se incluye en la lista de interfases de salida.
- Si no existe una interfaz de entrada para el flujo (S,G), se crea una tabla de probabilidades y se inicia dándole mayor porcentaje de probabilidad a aquella interfaz por donde la tabla unidifusión indique que se encuentra S.
- La hormiga Join sigue su camino hacia S hasta que encuentre un nodo al cual no sea necesario crear la entrada (S,G).

El procedimiento que se debe seguir con las hormigas Prune es el siguiente:

- La hormiga Prune al llegar a un nodo inteligente solicita que se elimine la interfaz por donde llegó de la lista de interfases de salida para la entrada (S,G).
- Si la lista de interfases de salida llega a ser nula, es decir, a contener 0 miembros, entonces la hormiga Prune marca la entrada (S,G) como inactiva.
- Si se da el caso anterior, la hormiga Prune continúa su recorrido por la interfaz de entrada de ese nodo.
- La hormiga Prune continúa su recorrido hasta que encuentre un nodo al cual no pueda inactivar su entrada (S,G).

1.5. **Hormigas cíclicas**



Debido a que cada nodo es autónomo en seleccionar la interfase de entrada para el flujo (S,G), es posible que varios nodos en la red formen un ciclo, es decir, que un nodo reciba información indirectamente de otro nodo al cual el primero le suministra información. Las hormigas cíclicas son las encargadas de detectar estos ciclos en la red. Su funcionamiento se describe a continuación.

En la gráfica se observa que *I2* inicialmente recibe el tráfico a través de *S*, el cual lo retransmite a *G2*, *I1* y finalmente *G1* en cadena. Sin embargo, una vez que el tiempo activo de la interfase hacia *S* ha culminado, *I2* puede seleccionar recibir el tráfico desde *G1* creando un ciclo.

Se genera una hormiga cíclica cada vez que un nodo inteligente *k* realice un cambio satisfactorio en la interfase de entrada, esto es, una vez que se haya recibido el primer paquete por la nueva interfase activa.

La hormiga cíclica generada chequea la tabla de enrutamiento MARP y se reenvía por todas las interfases de salida almacenando en cada salto el identificador del nodo que visitó.

Si la hormiga cíclica detecta que ha llegado a un nodo donde previamente había estado, informa al agente inteligente del enrutador para deshacer la ruta y tomar una nueva interfase de entrada.

De esta forma se controla el reenvío de información de forma cíclica.

2. REENVÍO DE INFORMACIÓN POR LOS AGENTES INTELIGENTES

Al llegar un paquete multicast se realiza el siguiente procedimiento:

- Se busca la tabla apropiada para el flujo (S,G) del paquete arribado.
- Si no se encuentra ninguna entrada (S,G), se procede a buscar una entrada (*,G). Si es encontrada alguna, entonces se crea una entrada (S,G) copiando la información de la entrada padre (*,G) y se continúa el proceso con la entrada (S,G) recientemente creada.
- Si la tabla (S,G) está inactiva, el paquete es descartado.
- Se comprueba que el paquete haya llegado por la interfase de entrada activa. Si no llegó por ella, es descartado.
- El paquete finalmente es reenviado por todas las interfases de salida de la entrada (S,G).

3. ANÁLISIS DE LAS HORMIGAS DE EXPLORACIÓN - ANÁLISIS MATEMÁTICO

Cuando una hormiga de exploración llega a un nodo inteligente ofrece el tiempo de viaje, T, desde que fue lanzada desde el nodo S. El agente inteligente en cada nodo va relacionado al comportamiento de T con relación a la memoria que posee el agente. La formulación matemática presentada en esta sección fue extraído del algoritmo AntNet [3,8].

Para cada entrada (S,G) el agente inteligente poseerá tres variables ($\sigma^2_{(S,G)}$, $\mu_{(S,G)}$ y $W_{(S,G)}$), que serán la representación ¿aproximada? de su conocimiento respecto al comportamiento histórico de la red que tiene la interfase por donde llegó la hormiga i_k antes mencionada.

La variable σ[6] (varianza) mide qué tanto varían los datos observados en relación con la cantidad. La variable μ mide la media de los valores y la variable W es una ventana deslizante, la cual almacena los últimos valores de los viajes.

[6] De aquí en adelante se omitirán los subíndices (S,G) para las variables σ, μ y W debido a que los cálculos se harán sólo para una entrada (S,G) en el nodo.

A continuación se explicará el código heurístico, que surge de la lógica Fuzzy [4,15], y las formulaciones matemáticas que hacen que el agente inteligente tome sus decisiones.

Sea $P_i(S,G)$ la probabilidad en la tabla para el flujo (S,G) que mide la bondad de la ruta para recibir el flujo por la interfase i del nodo y k la interfase por donde arribó la hormiga de exploración con tiempo T. El siguiente cálculo es aplicado para actualizar las probabilidades no nulas:

$$p_k(S,G) = p_k(S,G) + r \cdot (1 - p_k(S,G))$$
$$p_{k'}(S,G) = p_{k'}(S,G) \cdot (1 - r), \quad k' \in \mathrm{I}, k' \neq k$$

Se ha optado por afectar el valor p_k con la razón de hormigas de exploración que han sido analizadas por la interfase *k* con las hormigas que han sido recibidas por la interfase *k* de la siguiente forma:

$$P_k = \left(\Delta H_k \cdot P_k\right) \Big/ \sum_{i \in} \left(\Delta H_i \cdot P_i\right)$$

donde es la razón entre el número de hormigas que han sido analizadas y el número de hormigas que se ha recibido.

Definimos el refuerzo *r* como función de T, el nuevo tiempo del agente, y M, la memoria del agente (definida por σ, μ, *W*). *r* es una magnitud adimensional, , es usada por el nodo actual como una cantidad de refuerzo positivo para la interfase por donde vino.

r está dado por:

$$r = c_1 \cdot \left(\frac{W_{mejor}}{T}\right) + c_2 \cdot \left(\frac{I_{\sup} - I_{\inf}}{\left(I_{\sup} - I_{\inf}\right) + \left(T - I_{\inf}\right)}\right)$$

W_{mejor} es el mejor tiempo observado en la ventana de *W*. Con esto, el primer término de la ecuación mide qué tan bueno fue el viaje con respecto al mejor observado en la ventana *W*. Los valores I_{sup} e I_{inf} estiman los límites de un intervalo confiable para μ. El mejor valor observado, W_{mejor}, es también el más pequeño del rango, por lo que

$$I_{\inf} = W_{mejor}$$

Mientras que una estimación del valor de I_{sup} es encontrada de acuerdo con

$$I_{\sup} = \mu + \frac{\sigma}{\sqrt{|W| \cdot (1 - \gamma)}}$$

donde γ es el nivel de confianza seleccionado[7] y $|W|$ es la magnitud actual de la ventana.

El primer término de la ecuación evalúa la razón entre el tiempo actual, *T*, y el mejor tiempo observado sobre la actual ventana *W*. Este término es corregido por el segundo, el cual evalúa qué tan lejos se encuentra *T* de I_{inf} en relación con la extensión del intervalo confiable, esto es, considerando la estabilidad en los últimos viajes realizados.

Los valores de γ, c_1 y c_2 deben cumplir estas reglas: $c_1 \in (0,1]$, $c_2 \in (0,1]$, $\gamma \in (0,1]$ y $c_1+c_2=1$.

Finalmente, el valor de *r* es ajustado mediante una función s(x), la cual permite al sistema ser más sensible en favorecer los buenos valores de *r* y a la vez saturar los malos (cercanos a cero) valores de *r*: la escala está compresa para pequeños valores y expandida para los altos.

$$s(x) = \left(1 + e^{\frac{a}{x\cdot|I|}}\right)^{-1}, x \in (0,1], \quad a \in R^+, \qquad r = \frac{s(x)}{s(1)}$$

donde $|I|$ es el número de interfase de entrada candidatas en el nodo para la entrada (S,G).

Las magnitudes de μ y σ son actualizadas mediante la aproximación[8]:

$$\mu \leftarrow \mu + \eta \cdot (T - \mu),$$
$$\sigma^2 \leftarrow \sigma^2 + \eta \cdot \left((T - \mu)^2 - \sigma^2\right)$$

Donde *h* mide el número de muestras recientes que afectarán a μ y σ. Entre más pequeño sea *h* mayor número de muestras serán tenidas en cuenta.

Se considera el tamaño máximo de la ventana *W*, $|W|_{max}$, como:

$$|W|_{\max} = 5 \cdot (c/\eta),\ c < 1$$

donde *c* es una pequeña fracción de *h*.

---

[7] Esta expresión es obtenida usando la desigualdad de Tchebychef, que permite la definición de un intervalo de confianza para una variable aleatoria siguiendo una distribución. Normalmente, para especificar densidad de probabilidad los límites de Tchebychef son muy altos, pero aquí se convino usarlos debido a que (i) queremos impedir el hecho de asumir la distribución de **m** y (ii) necesitamos sólo una pequeña cantidad para estimar el intervalo confiable [9].

[8] El modelo presentado aquí fue extraído de los algoritmos de Jacobson/Karels, que estiman los tiempos de espera de la red TCP de Internet.

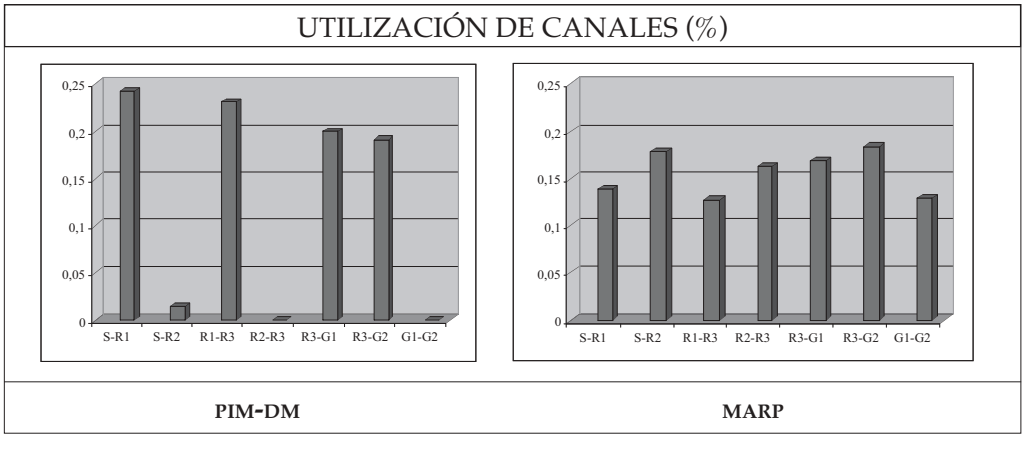
4. RESULTADOS

Se construyó un simulador en C++ Builder 5.0, el cual tiene como propósito recolectar estadísticas acerca de *throughput*, *delay*, utilización de canales, utilización de *buffers* en redes multicast que usan MARP y en redes multicast que usan PIM-DM. Para el tiempo en que se escribió este documento no se había recolectado información acerca de *throughput* y *delay*. Sin embargo, se posee información acerca de la utilización de canales en la red.

Se realizaron simulaciones con la topología del pescado mostrada a continuación:

Como se observa, tanto para G1 como para G2 existen 6 formas de recibir el tráfico multicast desde S. Todas 6 fueron aprovechadas en conjunto por los nodos del grupo. Claro está, cada ruta fue mantenida durante un lapso de tiempo proporcional a la calidad de la ruta.

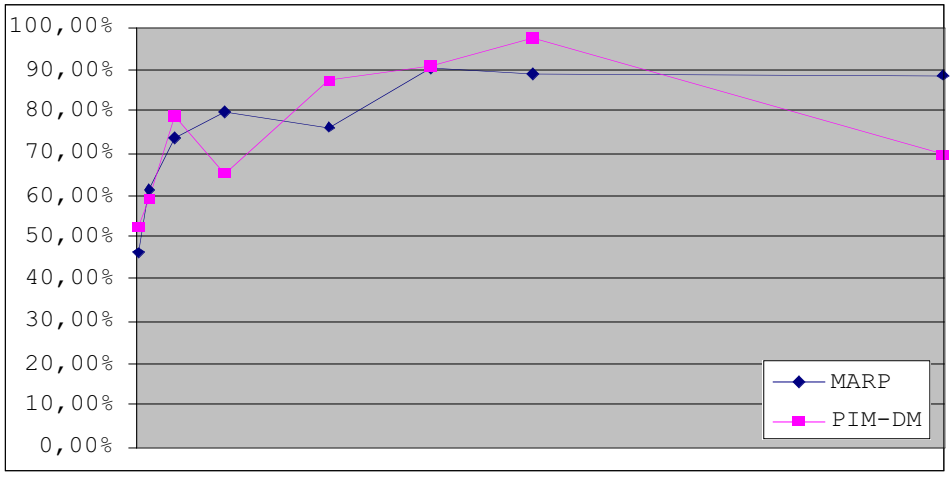
El simulador desarrollado fue probado cambiando valores de tamaños de *buffer*, *delay* en canales y tiempo de generación de paquetes. A continuación se presenta una gráfica que ilustra la utilización promedio de los canales durante las simulaciones.



Como es de esperarse, MARP usa mejor el ancho de banda de los canales, repartiendo la carga en el tiempo. Esto nos lleva a conjeturar que MARP debe ser mejor en *throughput*, ya que en el tiempo comparte la carga del flujo con el resto de enrutadores usando de esta forma todos los *buffers* de la red y reduciendo así la cantidad de paquetes descartados.

4.1. **Pruebas de *delay* variable**

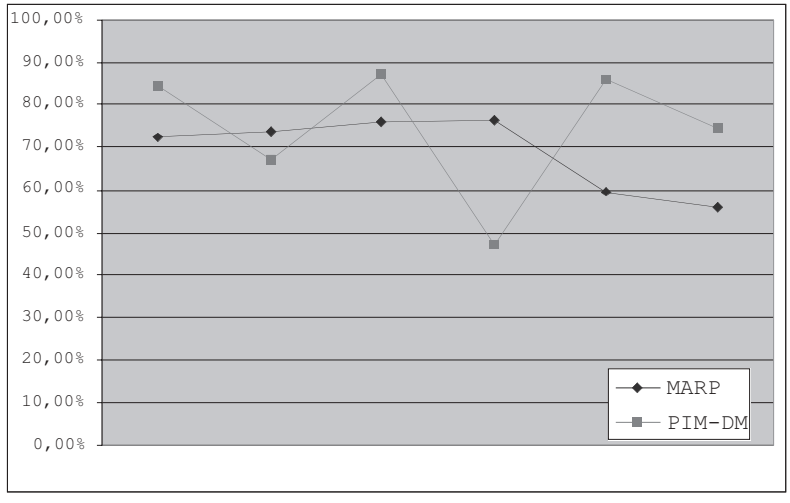
En esta sección se confrontan los protocolos analizando cómo varía la cantidad de paquetes que arriban a los destinos cuando el *delay* varía. Se realizaron varias simulaciones de MARP y PIM-DM, cada una de ellas manteniendo todos los parámetros iguales, excepto el *delay* en la línea de transmisión. Las pruebas comprendidas variaron el *delay* desde 0 hasta 100 milisegundos. Las redes actuales poseen *delays* ubicados en la primera mitad del rango probado por las simulaciones. El eje x representa la unidad tiempo.



Promediando los valores, se encontró un comportamiento semejante para los dos protocolos. Como se observa, el comportamiento de PIM-DM es oscilante, mientras que el de MARP es constante o menos oscilante que el de PIM-DM.

4.2. **Pruebas con *buffer* variable**

Las pruebas con *buffer* variable muestran resultados semejantes a las pruebas de *delay* variable descritas en la sección anterior. A continuación se presenta una tabla promedio de muestras donde se indica para cada *buffer* el número de paquetes que arribaron correctamente a los destinos para cada protocolo.



Los resultados son los mismos que los de la sección anterior. El comportamiento es en promedio igual, sin embargo se observa mayor estabilidad para el protocolo MARP.

4.3. **Validación de Resultados**

Es esta sección se detallan los pruebas estadísticas realizadas para validar los resultados obtenidos con las simulaciones.

La simbología empleada es la siguiente:

- Media poblacional en cuanto a *throughtput* para MARP(μ_{tM})
- Media poblacional en cuanto a *throughtput* para PIM-DM(μ_{tP})
- Media muestral en cuanto a *throughtput* para MARP($\bar{x}_{tM}$)
- Media muestral en cuanto a *throughtput* para PIM-DM($\bar{x}_{tP}$)
- Varianza muestral ()

4.3. 1. *Formulación de la Hipótesis*

$$H_0 = \mu_{(tM)} \geq \mu_{(tP)} \; ; \; \mu_{(tM)} - \mu_{(tP)} \geq 0$$

$$H_1 = \mu_{(tM)} < \mu_{(tP)} \; ; \; \mu_{(tM)} - \mu_{(tP)} < 0$$

4.3.2. *Especificación del nivel de confianza*

Siendo tan necesario comprobar que los resultados obtenidos en las simulaciones son válidos, se establecerá un nivel de confianza (α) igual al 0.03, lo cual se expresa a través de

$$1 - \alpha = 97\%$$

4.3.3. *Estadístico de Prueba*

El estadístico de prueba que se va a emplear es

$$z_{\Delta\bar{x}} = \frac{\bar{x}_{tM} - \bar{x}_{tP}}{\hat{S}_{\Delta\bar{x}}} \qquad \text{donde} \quad \hat{S}_{\Delta\bar{x}} = \sqrt{\frac{\hat{S}_M^2}{n_M} + \frac{\hat{S}_P^2}{n_P}}$$

4.3.4. *Criterio para la Decisión*

La hipótesis H_0 será válida si y sólo si $Z_{0.97} >= Z_{crítico}$

Para el nivel de confianza establecido la condición de validez es $Z_{0.97} >= -1.88$

5.3.5. *Cálculos*

Para *delay* variable sea:

$$\bar{x}_{tM} = 0.6805 \qquad \bar{x}_{tP} = 0.5979$$
$$\hat{S}_M^2 = 0.0942 \qquad \hat{S}_P^2 = 0.3003$$

entonces:

$$z = \frac{0.6803 - 0.5979}{0.11103} = 0.724141$$

Para *buffer* variable sea:

$$\bar{x}_{tM} = 0.7558 \qquad \bar{x}_{tP} = 0.75$$
$$\hat{S}_M^2 = 0.1530 \qquad \hat{S}_P^2 = 0.1605$$

entonces

$$z = \frac{0.7558 - 0.75}{0.098979} = 0.058598$$

4.3.6. Decisión

Para ambas hipótesis Z se encuentra en la zona de aceptación; por lo tanto, con base en la evidencia muestral no es posible rechazar la H_0, por lo que concluimos que las medias en cuanto *delay* y *throughtput* del MARP son mejores que las arrojadas por PIM–DM. Es decir, los resultados son válidos para un nivel de confianza (1-α) del 97%.

CONCLUSIÓN Y TRABAJO FUTURO

Este proyecto mostró un nuevo algoritmo multidifusión con balanceo de carga basado en la filosofía de AntNet. Se ha demostrado estadísticamente que el protocolo presenta mejores resultados en cuanto a *throughput*, ya que se hizo un balanceo de carga por las rutas óptimas de la red.

En simulaciones del algoritmo se ha observado un funcionamiento eficiente en cuanto a la selección de las mejores rutas para realizar el balanceo de carga. Actualmente se está comparando, mediante la simulación, el protocolo MARP con PIM–DM. Aunque no se haya realizado un estudio completo de su comportamiento frente a PIM–DM, se espera que los resultados sean mejores, en cuanto a *throughput*, por el simple hecho del empleo de balanceo de carga por parte de MARP.

En las situaciones estudiadas, el algoritmo funcionó adecuadamente durante el envió de paquetes, selección de rutas, detección de ciclos, búsqueda de nuevos caminos, entre otras funcionalidades del protocolo.

Se propone como trabajo futuro:

- Lograr que con los datos recopilado por las hormigas de exploración, los agentes inteligentes puedan predecir el comportamiento futuro de la red, brindando con esto otro factor importante al momento de calcular las probabilidades de cada enlace y, por consiguiente, de las rutas.

- Ampliar el número de variables de entrada, actualmente *delay*, para realizar mejores cálculos de la calidad de la ruta o interfase.

- Optimizar MARP para llegar a ser un protocolo Multipunto-Multipunto, es decir, que maneje una sola entrada para el grupo independientemente de la fuente.

- Llevar MARP al caso de *anycast* para su funcionamiento dentro de las especificaciones de Ipv6.

Referencias

[1] Ballardie, A., *Core Based Trees (CBT version 2) Multicast Routing*. RFC 2189. Septiembre de 1997.

[2] Ballardie, A., *Core Based Trees (CBT) Multicast Routing Architecture*. RFC 2201. Septiembre de 1997.

[3] Barán, B. & Sosa, R., *A new aproach for AntNet routing*. 1998.

[4] Bigus, J.P. & Bigus, J., *Constructing Intelligent Agents Using Java*, Second Edition.

[5] Comer, D., *Redes, Globales de Información con Internet y TCP/IP. Principios básicos, protocolos y arquitectura*, 3ª ed. Editorial Pearson, 2000.

[6] Deering, S., *Host extensions for IP Multicasting*. RFC 1112 de la IETF. Agosto de 1989.

[7] Deering, *Protocol Independent Multicast Version 2 Dense Mode. Specification*. draft-ietf-pim-v2-dm-01.txt. Noviembre de 1998.

[8] Di Caro, G. & Dorigo, M., *AntNet distributed stigmergetic control for comunications networks*. Diciembre de 1998.

[9] Estrin, D., *Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification*. RFC 2362. Junio de 1998.

[10] Fenner W., *Internet Group Management Protocol, Version 2* RFC 2236. Noviembre de 1997.

[11] Moy, J., *Multicast Extensions to OSPF*. RFC 1584. Marzo de 1994.

[12] Pusateri, T., *Distance Vector Multicast Routing Protocol*. draft-ietf-idmr-dvmrp-v3-10. Agosto de 2000.

[13] Slaterry, T. & Burton, B., *Advanced IP Routing in CISCO Networks*, 2ª ed. McGraw-Hill, 2000.

[14] Stevens, W. R., *TCPï©P Illustrated, Volume 1. The Protocols*.. Addison-Wesley, 2000.

[15] Tecuci, G., *Building Intelligent Agents*. Academic Press.

[16] Thaler, D., *Border Gateway Multicast Protocol (BGMP): Protocol Specification*. draft-ietf-bgmp-spec-02.txt. Noviembre de 2000.

[17] Waitzman, D., *Distance Vector Multicast Routing Protocol*. RFC 1075. 1988.

[18] Williamson, B., *Developing IP Multicast Networks. Volume I.* Cisco Press, 2000.