Ruteamiento Multicast en Redes de Computadoras.

Lic Héctor A. Sánchez Tores
hasanche@iinfo.unsj.edu.ar

Instituto de Informática
Facultad de Ciencias Exactas, Físicas y Naturales
Universidad Nacional de San Juan

Resumen.- Muchas aplicaciones como video-conferencia o sistemas colaborativos, requieren soporte multicast desde la capa de red. Multicast involucra el envío de mensajes sobre un árbol, con raíz en el nodo fuente, y cuyos caminos llegan hasta varios receptores. Uno de los objetivos de los algoritmos de ruteo es encontrar un árbol con costo mínimo. El establecer o encontrar ese árbol puede ser modelado como un problema del árbol de Steiner NP-completo. Muchas heurísticas han sido propuestas para encontrar en forma eficiente el árbol más cercano al óptimo. En este trabajo se propone y evalúa un algoritmo para encontrar el árbol multicast en redes de datos punto a punto. Este nuevo algoritmo fue implementado y probado mediante simulación, comparándolo con otros dos algoritmos. Los resultados obtenidos demuestran que el algoritmo propuesto, encuentra un buen árbol multicast utilizando en forma eficiente los recursos disponibles de la red. El aprovechar en mejor forma los recursos permite aceptar un mayor número de conexiones posibles, por cuanto se reducen los posibles puntos de congestión.

I - Introducción.

La transmisión de audio y video digital por una red de datos se puede entender como un reflejo de nuestro deseo de comunicarnos, usando formas más naturales y expresivas que un texto escrito. Los sistemas de comunicación construidos teniendo en cuenta el modelo de referencia OSI (Open Systems Interconecction) o la arquitectura de *Internet*, fueron diseñados para soportar servicios o comunicaciones *punto a punto*. En este tipo de redes, rápidamente se detectó la necesidad de proveer también comunicaciones multipunto. Por algún tiempo esto constituyó el impulso básico tendiente a proporcionar un soporte multipunto en redes de datos.

Uno de los tópicos en el cual las redes actuales están poniendo mucho énfasis, proviene de las aplicaciones multipunto o de grupo. Estas involucran más de dos usuarios (estos usuarios definen el grupo) los cuales desean intercambiar información; por el contrario, en las aplicaciones *punto a punto* la información fluye entre dos usuarios.

Las aplicaciones multipunto cubren un amplio espectro, incluyendo distribución de software, trabajo colaborativo, tele-educación, actualización de bases de datos duplicadas, comando y control de sistemas, video-conferencias, juegos y simulación distribuidas, entre otros.

Recientes avances en los medios de transmisión como la fibra óptica y el equipamiento para lograr la interconexión de la red (router o *switch*), han logrado velocidades del orden de gigabits por segundo. Lo cual sumado al trabajo en las nuevas tecnologías para almacenar datos, audio y video, han permitido el desarrollo de nuevas aplicaciones de tiempo real distribuidas.

Los requerimientos de las aplicaciones como retardo fin a fin, variaciones del retardo y taza de pérdida, vienen expresados por el término *Calidad de Servicio* o *QoS* (por su sigla en ingles). Dichos parámetros deben ser garantizados por la red. Actualmente muchas de estas aplicaciones pueden involucrar múltiples usuarios, por eso la importancia de la comunicación multipunto.

Uno de los problemas centrales que es necesario resolver para ofrecer comunicaciones multipunto, es establecer las rutas que debe seguir la información para conectar los nodos participantes. Esto se conoce con el nombre de *ruteamiento multipunto*. La solución de este problema es esencial para explotar eficientemente los recursos de la red y ofrecer la calidad de servicio requerida para los usuarios.

El diseño de algoritmos de *ruteamiento multipunto* es un problema complejo, de hecho, se demostró que el tiempo necesario para

encontrar una solución óptima crece en forma exponencial con el tamaño de la red [HUANG 92]. Más aún, la complejidad del problema aumenta debido a que los miembros de un grupo pueden cambiar, la topología de la red puede modificarse (por fallas en los enlaces o nodos) y al hecho de que normalmente las rutas deben ser establecidas en tiempo real [BAUER 97]. El algoritmo de ruteamiento debería ser capaz de entregar las rutas óptimas, tomando en consideración diferentes funciones de costo que incluyan los recursos disponibles, tales como: ancho de banda, número de enlaces (optimización de grafos), conectividad de nodos, precio a pagar, y retardo extremo a extremo [NORONHA 94].

En la próxima sección se presenta una clasificación de algoritmos y protocolos de ruteo multicast. En la sección III veremos los algoritmos evaluados, como así también el algoritmo propuesto. La sección IV muestra el modelo empleado y los resultados de la simulación, mientras en la sección V se presentan las conclusiones y los futuros trabajos.

II – Ruteo Multicast.

1 Definición.

Para nuestros propósitos, la red de comunicación es modelada usando un grafo no dirigido $\boldsymbol{G}=(\boldsymbol{V},\boldsymbol{E})$, donde $\boldsymbol{V}$ es un conjunto de host o routers y $\boldsymbol{E}$ es un conjunto de enlaces de comunicación. Dado un host (fuente) $\boldsymbol{s} \in \boldsymbol{V}$ y un conjunto de destinos $\boldsymbol{D} \in \boldsymbol{V}$, tal que $\boldsymbol{s} \notin \boldsymbol{D}$, una ruta multicast es un árbol del grafo $\boldsymbol{G}$, con raíz en $\boldsymbol{s}$, siendo sus hijos todos los nodos de $\boldsymbol{D}$. Se puede notar que $\boldsymbol{s}$ puede o no ser el único emisor, ya que el algoritmo produce un árbol multicast compartido, donde cualquier miembro de $\boldsymbol{D}$ puede utilizarlo para transmitir.

El problema ha sido estudiado tanto en el contexto de comunicaciones multicast [WAXMAN 88], y planteado también como un problema de la teoría de grafos [WINTER 87].

El objetivo perseguido por varios algoritmos de ruteo multicast podríamos decir que cae dentro de dos categorías generales [BHARATH 83]. En la primera se trata de minimizar el retardo a cada destino, lo cual es justamente lo que tratan de hacer los algoritmos *SPT* (camino más corto). Mientras en la otra categoría, se trata de minimizar el costo total del árbol que conecta la fuente con los nodos destino. Esto ultimo es conocido como el *problema del árbol de Steiner* [WINTER 87]. Cuando el conjunto de nodos destino abarca todo el conjunto de nodos de la red, se lo denomina *problema del árbol de mínima expansión* [CORMEN 90].

A continuación se verán sendas clasificaciones tanto para *algoritmos* como para *protocolos* de ruteo multicast.

2 Algoritmos de ruteo multicast.

Existen diversas clasificaciones [SALAMA 97][DIOT 97] para los algoritmos teniendo en cuenta diversas características.

2.1 Clasificación

Considerando la función de costo que utiliza el algoritmo, los podemos clasificar en dos grandes categorías. La primera categoría son los algoritmos de ***camino más corto***. Estos algoritmos construyen un árbol multicast que minimiza la longitud de cada camino desde el nodo fuente hasta cada uno de los nodos destino. La otra categoría son los algoritmos de ***árbol de steiner mínimo***. El objetivo de estos algoritmos es minimizar el costo total del árbol multicast. Siendo el costo total del árbol, la suma del costo de todos los enlaces que conforman el árbol. Sí el conjunto de receptores del árbol de Steiner incluye todos los nodos de la red, esto es llamado "***árbol de expansión mínima***".

Para poder soportar aplicaciones en tiempo real, los algoritmos y protocolos deben ser capaces de proveer una calidad de servicio garantizada. Los algoritmos de ruteo multicast propuestos específicamente para redes de alta velocidad, que construyen un árbol eficiente sin superar un límite superior de retardo establecido, son denominados ***algoritmos de ruteo multicast con retardo restringido.***

Se puede considerar el caso de nodos de la red con capacidad limitada de copia. Estos tienen un límite superior en cuanto a la copia de paquetes de entrada, que deben ser replicados a los próximos nodos vecinos. Esto es conocido como ***algoritmos de ruteo multicast con grado restringido***.

Teniendo en cuenta la dinámica del grupo multicast, podemos considerar algoritmos de ruteo multicast ***dinámicos***. Estos permiten a los nodos fuentes y receptores unirse y/o dejar la sesión multicast, y su correspondiente árbol multicast, en cualquier momento. En los algoritmos de ruteo multicast ***estáticos***, el grupo

multicast es fijo, y los caminos desde la fuente hasta los receptores son calculados en el mismo momento, cuando se inicia la sesión multicast.
Al considerar los nodos que realizan los cálculos de la ruta, podemos clasificarlos en: algoritmos multicast ***distribuidos***, en los cuales los cálculos son compartidos entre múltiples nodos. Esto reduce la sobrecarga en los cálculos pero requiere un mayor intercambio de mensajes entre los nodos. La complejidad de estos algoritmos es medida por el número de mensajes intercambiados. La información acerca del estado de la red, debe ser almacenado en cada nodo. Por otro lado tenemos los algoritmos ***centralizados***, donde la mayor parte de los cálculos y decisiones las realiza el nodo fuente o centro. Son usualmente más estables en comparación que los algoritmos distribuidos. La información acerca de la topología completa, debe estar disponible en cada nodo ejecutando el algoritmo centralizado.
La clasificación presentada anteriormente, establece ciertas características propias de cada algoritmo. Es posible encontrar algoritmos que posean más de una característica a la vez.

3 Protocolos de Ruteo Multicast.

3.1 Características.

Los refuerzos para desarrollar protocolos de ruteo multicast para redes de área amplia comenzaron sobre el final de la década de los ochenta. Motivado por el gran crecimiento de Internet y las aplicaciones multimediales con múltiples usuarios.
El protocolo de ruteo multicast describe como implementar, un algoritmo de ruteo multicast, en forma práctica. Los protocolos deben ser robustos y tolerantes a falla. Por ejemplo, un protocolo se espera reaccione rápido y seguro frente a fallas en un enlace o nodo, de forma de minimizar la inestabilidad resultante en la red. El protocolo debe ser diseñado con la capacidad de soportar información incorrecta o desactualizada, sin que esto provoque problemas en la aplicación. Los protocolos de ruteo usualmente residen en la *capa de red* del stack de protocolos del modelo propuesto por *OSI*.

3.2 Clasificación

En [DIOT 97] se presenta una clasificación de protocolos de ruteo múltcast de acuerdo a los siguietes tres grupos: protocolos para Internet, Hosts móviles y Arquitecturas ATM.
Los protocolos de transporte multicast son analizados y clasificados en [OBRACZKA 98], teniendo en cuenta que fueron diseñados para aplicaciones específicas. Se pueden mencionar las siguientes características:

- *Propagación de los datos:* el mecanismo usado para propagar los datos (sin información de control) entre los sitios participantes.
- *Mecanismos de confiabilidad:* Como los protocolos se recuperan frente a la pérdida de datos.
- *Control de Feedback (retroalimentación):* Como controla el protocolo el aumento de información de control generada por los receptores.
- *Retransmisión:* Como propaga el protocolo los datos que necesitan ser retransmitidos.
- *Flujo y control de Congestión:* El mecanismo usado para evitar que la fuente desborde a los receptores y cause congestión en la red.
- *Ubicación del lugar para Control*: Si el protocolo usa un sitio central para funciones de control tales como orden en los mensajes, miembros del grupo y retransmisiones.
- *Ordenamiento:* que garantía de orden ofrece el protocolo y como se consigue esa garantía.
- *Gerenciamiento del grupo:* Como administra el protocolo los miembros del grupo multicast.
- *Aplicación:* Si el protocolo fue propuesto o implementado para soportar una aplicación específica.

Dentro del diseño de protocolos de transporte se enfatiza en la *escalabilidad* como un objetivo fundamental. El cual traerá aparejado prestar atención a varias de las funciones vistas. Estos algoritmos deberían usar un esquema de confiabilidad basado en receptores, mediante el uso de NACK, con algún control para evitar la implosión. Al trabajar con grandes grupos de receptores, la fuente no puede estar al tanto de todos los receptores. El algoritmo debería ser distribuido y evitar cuellos de botella.

III - Algoritmos de ruteo multicast.

A continuación se detallan los algoritmos multicast simulados, como así también el algoritmo propuesto en este trabajo.

1 Algoritmo SPT (árbol de caminos más cortos)

El funcionamiento de este algoritmo se basa en el método *SPT* [CORMEN 90], generando distintas conexiones punto a punto, para formar una conexión multipunto. Es uno de los algoritmos más simples de ruteo multicast, siendo una versión distribuida del algoritmo de *camino más corto* de *Bellman-Ford* [CORMEN 90].

Dados los componentes de una conexión multipunto, es decir, el nodo origen y los nodos destinos, el algoritmo establece varias conexiones *punto a punto* independientes. En cada conexión une el nodo origen con uno de los nodos destino.

Una vez establecidas las diversas conexiones punto a punto, se obtiene un primer árbol de conexión que une la fuente con los nodos destino. Como cada conexión (fuente-destino) se establece en forma independiente una de otra, es posible que el árbol contenga enlaces duplicados o existan ciclos. Por lo tanto es necesario controlar y eliminar dichas situaciones. La duplicación de enlaces se produce cuando, dos o más caminos de conexión multicast utilizan el mismo enlace. Lo cual representa un mayor uso de los recursos de la red.

En el ejemplo de la figura *1* podemos observar el caso de la conexión del nodo 6 con los nodos 4 y 2. Donde el primer camino, entre los nodos 6 y 5, utiliza los enlaces 6-5 y 5-4; y el segundo, entre los nodos 6 y 2, utiliza los enlaces 6-5, 5-7 y 7-2. Por lo tanto el enlace 6-5, esta siendo utilizado por ambos caminos (duplicidad).

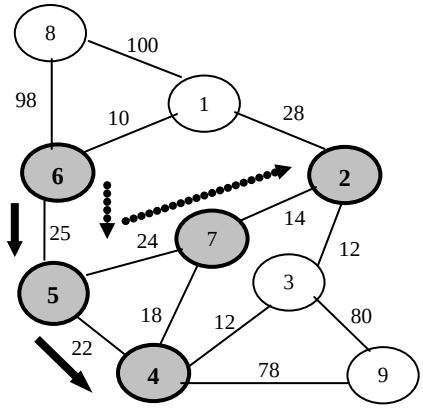

Fig. 1 Duplicación de enlaces.

Por otro lado un *ciclo* es cualquier camino que comience y finalice en el mismo nodo, sin haber pasado por algún otro nodo más de una vez. Esto se ve en el gráfico de la figura *2*.

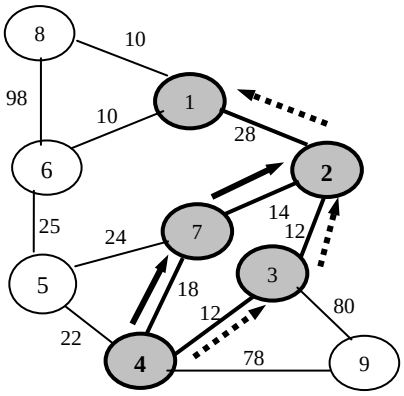

Fig. 2 Ciclo en una conexión

Siendo el nodo 4 la fuente y el grupo destino {2,1}, en un primero paso se conecta la fuente con el nodo 2, formándose el camino (4,7,2). Al conectar el nodo 4 con el 1, se forma el camino (4,3,2,1). Estos caminos se establecen en forma independiente uno del otro, por lo tanto se forma un *ciclo* entre los nodos {4,7,2,3} mediante el camino {4,7,2,3,4}.

En este trabajo se utilizo el algoritmo PRIM [CORMEN 90], como base para eliminar *enlaces duplicados* y *ciclos.*

2 Algoritmo Backbone Principal.

Este algoritmo representa el método utilizado para establecer conexiones multicast, en redes ATM. En la especificación UNI 3.1 y 4.0 del ATM Forum, se determina la forma de conectar los distintos nodos destino al árbol parcial, con o sin intervención del nodo fuente. En este trabajo se lo denomina *Backbone*. Una vez conocidos los componentes de la conexión multipunto (fuente y destinos), se determina el nodo más distante de la fuente, según el criterio de menor distancia, y se los conecta.

A partir de esta conexión, se forma un árbol extendido parcial. El algoritmo escogerá el próximo nodo a conectarse a dicho árbol o *Backbone* (de ahí su nombre). Este proceso se repite hasta conectar todos los nodos del conjunto destino. Se evaluaron tres criterios de selección del próximo nodo a conectar:

aleatorio, menor y *mayor distancia* al *Backbone* establecido.

La idea de elegir los nodos más distantes para formar el *Bakbone* inicial, es la de minimizar los recursos necesarios para conectar los restantes nodos destinos. Es importante notar que este proceso de creación del grafo esta libre de los fenómenos de duplicación de enlace y ciclos. Esto se debe a la manera en la cual los nodos destino se van conectando al árbol parcial o *Backbone*. El término "conectarse al árbol parcial" significa establecer una conexión con cualquiera de los nodos que conforman dicho árbol, sean nodos destinos o no.

En la figura 3 se observa un pedido de conexión multipunto, formado por los nodos {3,5,6,8}, siendo el nodo {3} la fuente. Una vez determinado el nodo más lejano, en este caso {6}, se lo conecta. Esta primera conexión forma el árbol parcial o *Bakbone*, como se muestra en la figura *3*, compuesto por los nodos {3,2,1,6}. Luego el algoritmo continúa conectando el resto de los nodos destino {8,5}. El orden de conexión dependerá de su distancia al *Backbone*. Siendo {8} el próximo nodo a conectar, el algoritmo determina el nodo del *Backbone* más cercano a él. Esto se repite hasta conectar todos los destinos.

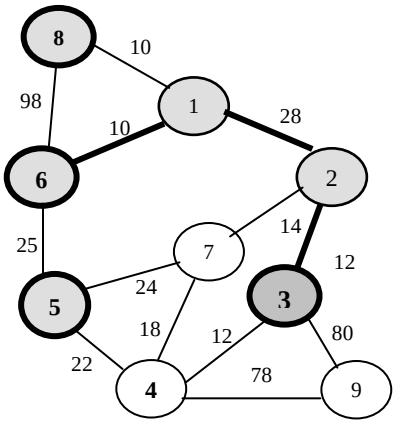

Fig. 3 – Árbol parcial

3 Algoritmo Árbol de Grupos.

Este es el algoritmo propuesto como parte de este trabajo y la diferencia básica con los algoritmos anteriores radica en conectar grupos de nodos entre sí, mientras en los dos casos vistos se intenta conectar nodos destino individualmente.

El algoritmo parte con la idea de dividir el grupo de nodos destino en subgrupos. Se define un grupo inicial que contiene el nodo fuente. En un primer paso se conectan los nodos dentro de cada subgrupo. Una vez finalizado esto, se comienzan a conectar dichos subgrupos con el grupo inicial. Al conectar todos los subgrupos se obtiene el árbol multicast que conecta el nodo fuente con todos los nodos del grupo destino.

Para la especificación se mantendrá la notación definida en la sección II. Se asume que cada nodo conoce el camino más corto (medidos en cantidad de enlaces) a los otros nodos de la red, manteniendo esta información actualizada. Los caminos más cortos son simétricos. En este sentido los nodos *i* y *j*, comparten el mismo camino más corto entre ellos. Es decir, el camino (i,j) tiene la misma distancia y nodos intermedios, que el camino (j,i). Los grupos iniciales con los cuales comienza el algoritmo, se los denomina *grupos parciales* o *subgrupos*.

Este algoritmo se apoya en la idea propuesta en [GALLAGER 83], donde se da una solución al problema del árbol de expansión mínimo (*MST*) mediante un algoritmo distribuido. El objetivo de este algoritmo es incluir a todos los nodos del grafo o red en el árbol solución. En un instante inicial, algunos o todos los nodos de la red, participan en la conexión. Cada nodo intenta conectarse con alguno de sus vecinos. Al conectarse dos nodos, se forma un *fragmento*. Estos *fragmentos* se van conectando de a dos. El algoritmo finaliza al conectar los dos últimos *fragmentos*. Al unirlos se obtiene el árbol de expansión mínimo abarcando todos los nodos de la red.

Para nuestro caso, al trabajar con árboles multicast, los nodos destino son un subconjunto de los nodos de la red o grafo. Por lo cual la conexión entre dos nodos destino, implicaría utilizar nodos intermedios o nodos de la red. Si nos situamos en el problema de *Steiner*, estos nodos intermedios, serían los nodos *Steiner*.

3.1 Funcionamiento.

Inicialmente tanto el nodo fuente como cada nodo del grupo destino es un subgrupo. Con estos subgrupos formados, la fuente establece una lista ordenada según su distancia a cada uno de ellos. Hasta este momento, los conjuntos están compuestos de un solo elemento. De esa lista se generan grupos con pares de nodos

destino. Los grupos se forman tomando dos nodos destino sacados de la lista en orden de menor a mayor. La fuente forma un grupo con el nodo más cercano a ella, o sea el primer nodo de la lista. A continuación se escogen los próximos dos nodos de la lista para formar el próximo grupo(en este caso el segundo). Este proceso se repite hasta terminar con los nodos de la lista. El último grupo puede o no estar formado por un único nodo, dependiendo sí la cardinalidad de *Z* es par o no.

Una vez establecidos los grupos, se comienza a conectar los nodos de cada subgrupo, entre sí. Dentro de cada subgrupo (formado por dos nodos), se define al de menor distancia hasta la fuente como nodo *origen*, y el restante será el nodo *destino*. En caso de tener los dos la misma distancia, el primero en la lista ordenada, será el *origen*. Esta determinación es llevada a cabo por la fuente en el momento de generar los grupos. La fuente comunica a cada uno de los nodos origen de los distintos grupos, cual es el nodo a conectar (*destino*) para formar el grupo. El grupo con el nodo fuente, se denomina *grupo base*. A este grupo se irán conectando los demás grupos.

Mientras se conectan los nodos dentro de cada subgrupo, se pueden producir ciertas "*colisiones*" o "*intersecciones*" entre caminos de distintos grupos. Una *colisión* ocurre cuando dos caminos, de distintos grupos, pasan por un mismo nodo perteneciente a *S*. Detectada la colisión por ambos grupos, estos automáticamente quedarían conectados, formando un nuevo grupo. Una *intersección* se produce cuando el camino de un grupo *A* pasa por un nodo origen o destino de algún otro grupo *B*, no conectado todavía. El grupo *A* continua con su proceso de conexión. El grupo *B* al tratar de conectar su *origen* con el *destino*, detectara la intersección, quedando conectados ambos grupos. Una intersección se puede dar en varias situaciones.

- El nodo *origen* ya pertenece al conjunto base, con lo cual solo se debe conectar el nodo *destino* con el *grupo base* para tener conectados los dos grupos.
- El nodo *destino* ya pertenece al conjunto base, con lo cual solo se debe conectar el nodo *origen* con el *grupo base* para tener conectados los dos grupos.
- Los nodos, *origen* y *destino*, ya pertenecen al *grupo base*, con lo cual ya están los dos grupos conectados.
- Si al conectar los nodos *origen* y *destino* no se pasó por ningún nodo del *grupo base*, el grupo queda formado sin tener intersección alguna con el grupo base.
- En caso de no haber algún tipo de intersección o colisión, entre el nuevo grupo formado y el grupo base, se procede a conectar los dos grupos.

Una vez conectados los nodos de cada *grupo parcial*, estos grupos comienzan a conectarse con el grupo base. La *fuente* determinará el orden, según el cual, se van conectando los subgrupos con el *grupo base*. El *grupo base* va creciendo mientras los *grupos parciales* se van conectando a él. Tanto los nodos de ambos grupos (parcial y base) como los nodos intervinientes en el camino de conexión, pasan a formar parte del nuevo *grupo base*. Al conectar todos los grupos en el grupo base, este último representará el árbol multicast final que conecta el nodo fuente con todos los nodos destinos.

Debido a la forma de conectarse los subgrupos con el grupo base, no es posible la creación de ciclos. Esta imposibilidad esta dada por los controles efectuados en el momento de llevar a cabo la unión. En cada proceso de unión, por cada nuevo nodo que se visita, se controla si pertenece o no a un grupo ya formado. El método utilizado para resolver las intersecciones (presentadas anteriormente), lleva a cabo los controles necesarios para garantizar la ausencia de ciclos al conectar un nuevo subgrupo al grupo base.

El problema de ruteo multicast abordado en este trabajo, puede ser modelado como el problema del árbol de Steiner en redes. Encontrar soluciones explícitas en grandes redes es muy caro. El algoritmo propuesto en este trabajo, resuelve el problema de Steiner, mediante una heurística. La cual entrega como solución final un árbol de Steiner, que podría no ser de costo mínimo, conectando todos los nodos destino. Este es encontrado en un número polinomial de pasos en función de *n* y *m*. Siendo la complejidad $O(m.n^2)$, donde *m* es el número de nodos terminales y *n* el número de nodos de la red.

A continuación veremos un ejemplo. Considerando el grafo de la figura 4, veamos

como funciona el algoritmo y las decisiones que va tomando en cada paso. El nodo fuente es el 8 y el conjunto de nodos destino es {2,3,6}. Según las distancias desde 8 hasta cada uno de los nodos destinos, la lista ordenada quedaría de la siguiente manera {6,2,3}. De acuerdo con la lista formada, el nodo fuente establece el primer grupo con el nodo 6, quedando el grupo {8,6}. El segundo grupo estaría compuesto por los nodos {2,3}. Una vez establecidos estos dos grupos iniciales, se procede a conectar los nodos de cada grupo. Para este caso, en el primer grupo el nodo fuente (8) se debe conectar con el nodo 6. Mientras en el segundo grupo, el nodo 2 se debe conectar con el nodo 3. Las conexiones establecidas entre los nodos de cada grupo, se puede apreciar en la figura 4. A continuación se deben conectar los dos grupos recientemente formados. El nodo fuente (8), en el grupo base, determina como se conectara el grupo (2,3).

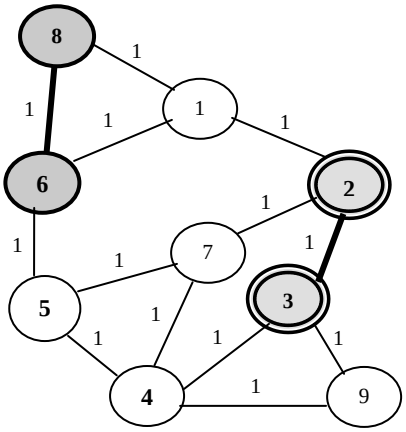

Fig. 4 Grupos formados.

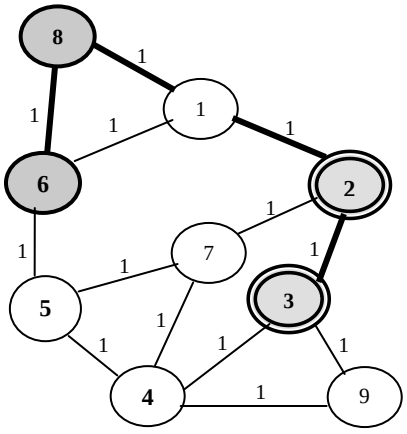

Fig. 5 Conexión final.

El nodo fuente (en el grupo base) determina cual de sus nodos es el más cercano (menor distancia) al grupo a conectar. El grupo 2 hace lo mismo, con respecto al grupo base. Los nodos a conectar entre los dos grupos son el 8 (por el grupo base) y el 2 (por el grupo 2). La conexión entre el grupo base y el grupo 2 se muestra en la figura 5, con lo cual termina la ejecución del algoritmo.

IV - Modelo y Resultados.

En esta sección se detalla el modelo utilizado para llevar a cabo las simulaciones de los algoritmos multicast vistos en la sección anterior. Junto con la descripción del modelo se presentan los resultados obtenidos, en forma de gráficos.

1 Simulación.

La simulación utilizada en este trabajo es orientada a eventos. Los dos eventos que intervienen son: *arribo* o *llegada* de un nuevo pedido de conexión representado por *llegada*(*pedido*); y *finalización* de una conexión existente en el sistema, *fin*(*pedido*). El evento *llegada*(*pedido*$_i$) viene acompañado de la siguiente información: nodo fuente, nodos del grupo destino, tiempo *t* y ancho de banda solicitado. Este evento representa para el sistema, el establecimiento del árbol multicast que conecta el *nodo fuente* con el grupo de nodos destino. Por cada evento *llegada* los enlaces (que forman parte de su árbol multicast) ven disminuido su ancho de banda disponible. Para nuestro caso, el ancho de banda solicitado por cada conexión es de una unidad. El evento *fin*(*pedido*$_i$) representa la expiración del tiempo ***t*** del *pedido*$_i$. Esta expiración indica la finalización de la conexión y por tanto la liberación de recursos, o ancho de banda, en los enlaces ocupados por el pedido *i-esimo*.

2 Redes

Se utilizaron 3 topologías de red diferentes:

- Red ARPANET: 21 nodos, 26 enlaces y conectividad con grado de 2,4 (fig. 6).
- Red USANET: 26 nodos, 29 enlaces y conectividad con grado de 3 (fig. 7).
- Red MANHATTAN: 64 nodos, 112 enlaces y conectividad con grado de 3,5 (fig. 8).

Para las tres topologías consideradas, se fijo la capacidad de cada enlace en 100 unidades, y cada pedido de conexión utiliza 1 unidad.
Se supone también, que cada nodo en el momento inicial de la simulación, dispone de la información acerca de la topología de la red, es decir, la distancia entre él y cualquier otro nodo. Esto se puede conseguir mediante la utilización de algoritmos fuera de línea como *Dijkstra* o *Bellman Ford*.

3 Carga.
Para cada simulación se realizaron dos mil (*2000*) pedidos de conexión. Cada uno de estos pedidos posee un tiempo de duración ***t***, o tiempo durante el cual la conexión permanece en el sistema. Este tiempo de conexión corresponde a una variable aleatoria con distribución uniforme, pudiendo tomar valores entre uno (1) y doscientas (*200*) unidades de tiempo, considerando que el tiempo de simulación total es de dos mil (*2000*).

4 Distribución.
La generación de pedidos de conexión puede ser de dos tipos de distribución:

- *Distribución uniforme*: Los conjuntos de nodos *origen/destino* son generados aleatoriamente por el sistema.
- *Distribución con puntos calientes*: Un porcentaje P de todas los pedidos de conexión son hechos para un conjunto determinado de *origen/destino*. El resto de los pedidos se distribuyen uniformemente entre los otros nodos posibles.

La idea de trabajar con una distribución con *puntos calientes*, fue para ver como era el comportamiento del algoritmo frente a una red con carga desbalanceada. Donde ciertos enlaces poseen una mayor carga, con lo cual en el futuro puedan convertirse en puntos de congestión.
Los experimentos se realizaron en tres tandas de 45 repeticiones cada una, es decir, en total 135 valores. Con esta cantidad se alcanzo un

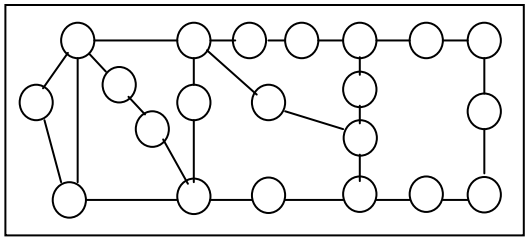

Fig. 6 Red Arpanet.

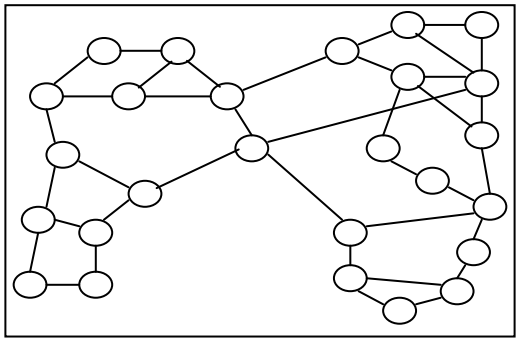

Fig. 7 Red Usanet.

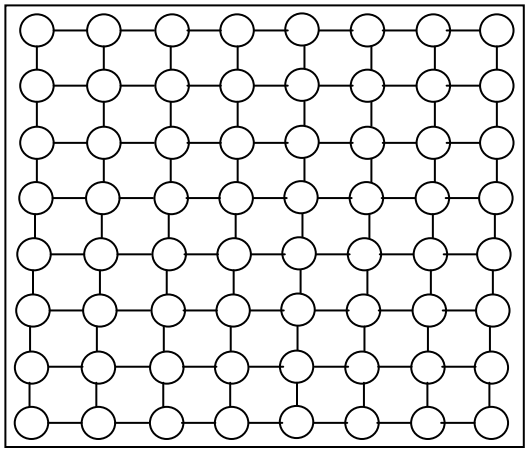

Fig. 8 Red Manhattan.

intervalo de confianza del 95 %. Este proceso se efectuó para cada ensayo realizado. Cada ensayo estuvo compuesto por: algoritmo multicast, topología de red y distribución y cantidad de nodos del conjunto destino.

5 Medidas de desempeño.
Dado un pedido de conexión $c_i = (s_i, D_i)$ ($1 < i \leq k$, k = total de pedidos) el algoritmo produce las siguientes salidas:

- *Acep.*(c_i) = 1 sí el pedido de conexión fue aceptado; 0 sí fue rechazado.
- *Distancia*(c_i)=Número de nodos usados en el establecimiento de la ruta, sí la conexión fue aceptada; 0 sí fue rechazada.

- *Tpo_establecimiento*(c_i)=Número de nodos visitados por el paquete de ruteamiento, sí la conexión fue aceptada; 0 sí fue rechazada. Cuando el algoritmo necesita buscar caminos alternativos frente a un enlace saturado, este parámetro será mayor que la *distancia*, por cuanto debe recorrer un número mayor de nodos para conectar el nodo destino.

Las siguientes métricas [HUANG 94] para medir el desempeño de los algoritmos, se obtienen como salida del modelo.

- *Tasa de bloqueo* (CBR - Call Blocking Rate): Mide la tasa de rechazo de los pedidos de conexión solicitados.

$$CBR = 1 - \left(\frac{n\_acep.}{Total} \right)$$

- *Distancia media de ruta* (RD – Average Route Distance): Mide la distancia promedio entre las rutas establecidas o aceptadas.

$$RD = \frac{RD * \left(n\_acepta - acepta \right) + acepta * distancia}{n\_acepta}$$

- *Distancia media del pedido* (CST – Average Call Setup Time): Mide la distancia promedio entre los pedidos aceptados. Considerando la cantidad total de nodos visitados para establecer la ruta.

$$CST = \frac{CST * \left(n\_acep. - acep. \right) + acep. * d\_setup}{n\_acep.}$$

- *Tasa de sobrecarga de la ruta* (STO – Setup Overhead): Es la razón entre la distancia promedio de las rutas establecidas y la distancia promedio de dichas rutas considerando el total de los nodos visitados.

$$STO \begin{cases} \frac{CST}{RD} - 1, RD \neq 0 \\ 0, RD = 0 \end{cases}$$

6 Resultados.

Las simulaciones se dividieron en dos grandes grupos, según las distribuciones de la carga. Luego en cada distribución se efectuaron simulaciones sobre los tres tipos de redes. Esto fue repetido para cada algoritmo de ruteo variando la cantidad de nodos del grupo destino.

1 Carga Uniforme

Primero analizaremos los casos con carga uniforme para las tres topologías utilizadas. Si bien hay diferencias entre los resultados obtenidos para cada topología las tendencias e cuanto a diferencias entre los algoritmos se mantuvieron.

Los gráficos de la figura 9 muestran el desempeño de los algoritmos implementados, donde *Grupo* obtiene los mejores resultados tanto para *CBR* como *RD*. La métrica *CBR* indica la taza de aceptación de llamadas, valores bajos en dichas métricas, representa una mayor posibilidad de aceptación de futuras llamadas o pedidos de conexión. La probabilidad de bloqueo para las llamadas siguientes, es menor.

La métrica *RD* indica la cantidad de enlaces que forman la ruta definitiva (árbol multicast) hasta los destinos. El algoritmo *Grupo* utiliza una menor cantidad de recursos de la red. Ligado con esto es lo que ocurre con *STO*. Este valor nos indica la sobrecarga promedio de las conexiones establecidas. Es decir, cuanto demás debió viajar el paquete ruteador, para alcanzar el destino. Tanto el algoritmo *Backbone* como *Group*, realizan una mayor búsqueda. Esto ocurre al intentar utilizar un enlace saturado. Acto seguido se buscan caminos alternativos que conduzcan hacia el destino.

También se realizaron simulaciones con grupos de 2, 10 y 15 nodos destinos. Los resultados obtenidos para 2, son similares para el caso de 5 usuarios, solo que las pendientes de las curvas, en los valores desde el 100 hasta el 400, son más suaves en su parte inicial. Para el caso de 10 y 15 se mantuvieron las tendencias de las curvas obtenidas en los casos anteriores. En el caso de *CBR* las diferencias entre *Group* y los otros dos algoritmos aumentan, mientras en *RD* (para grupo) los valores obtenidos se mantienen. *STO* para estos casos, disminuyo, al tener menor distancia que recorrer para encontrar el nodo destino, o algún nodo del árbol o parcial.

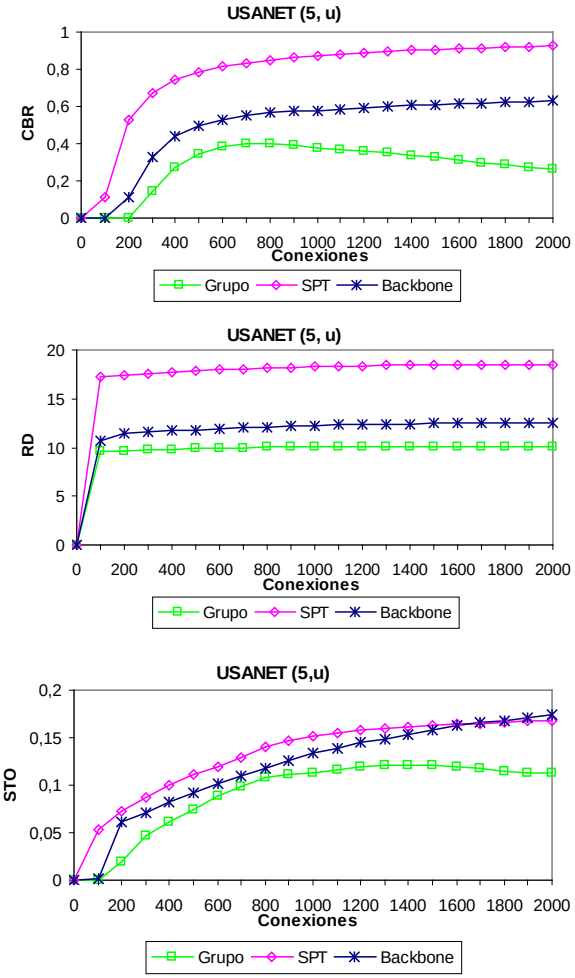

Fig. 9 Usanet, con carga uniforme.

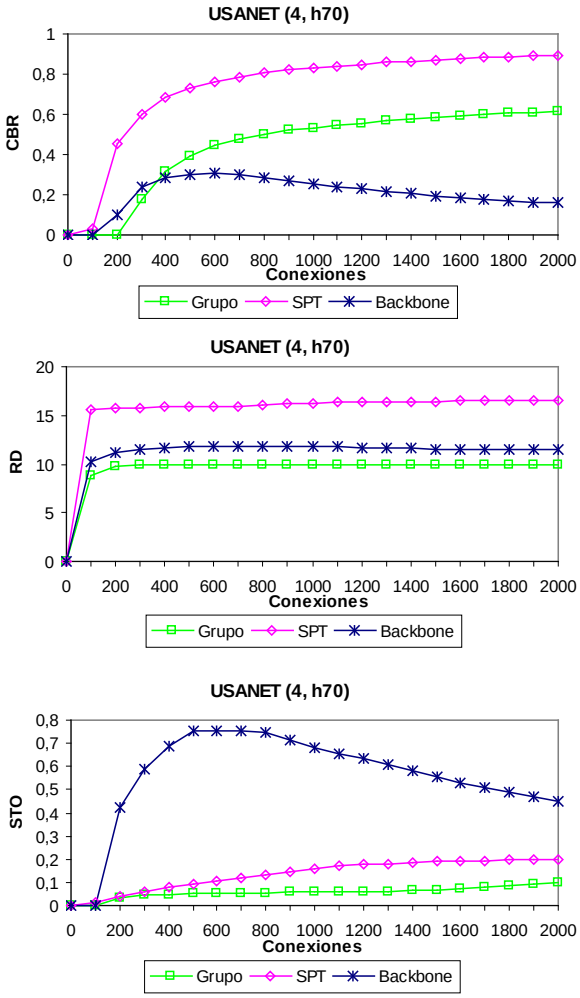

Fig. 10 Usanet, con puntos calientes.

2 Carga con puntos calientes.

Los gráficos de la figura 10 muestran los resultados obtenidos al ejecutar la simulación utilizando la topología USANET, con distribución de los usuarios según *Puntos Calientes*.

Los gráficos muestran como el algoritmo *Grupo* presenta para *CBR* un valor por encima de *Backbone*, para el caso de cinco nodos destino. Observando los valores de *CBR* podemos determinar la taza de aceptación de llamadas.

La métrica *RD* indica la cantidad de enlaces que forman la ruta definitiva hasta los destinos (árbol multicast). El algoritmo *Grupo* obtuvo (en todos los casos) el menor valor, lo cual indica una mejor utilización de recursos (en cuanto a cantidad de enlaces) de la red. *STO*, para este caso, es también bajo para el algoritmo *Grupo* pero no para *Backone* y *SPT*. Esto significa que el paquete ruteador de ambos, debió recorrer un mayor camino para conectar el nodo destino. Esto depende en gran medida de la política utilizada, para llevar a cabo la conexión punto a punto.

En la figura 11 se observan los resultados para la red Manhattan con puntos calientes. El valor de *CBR* para *Grupo* es muy bajo debido a la topología de esta red, lo cual permite un mayor número de caminos hacia el nodo destino.

En el caso de pocos nodos destino (dos), el algoritmo *Grupo* tiene buen desempeño, tanto en *CBR*, como *BRP*. Mientras que para cuatro nodos (con puntos calientes), se comporta bien en cuanto a *RD* y *STO*. Para una mayor cantidad de nodos destino (10), el algoritmo *Grupo* se mantuvo con los mejores resultados para las métricas *CBR* y *BRP*.

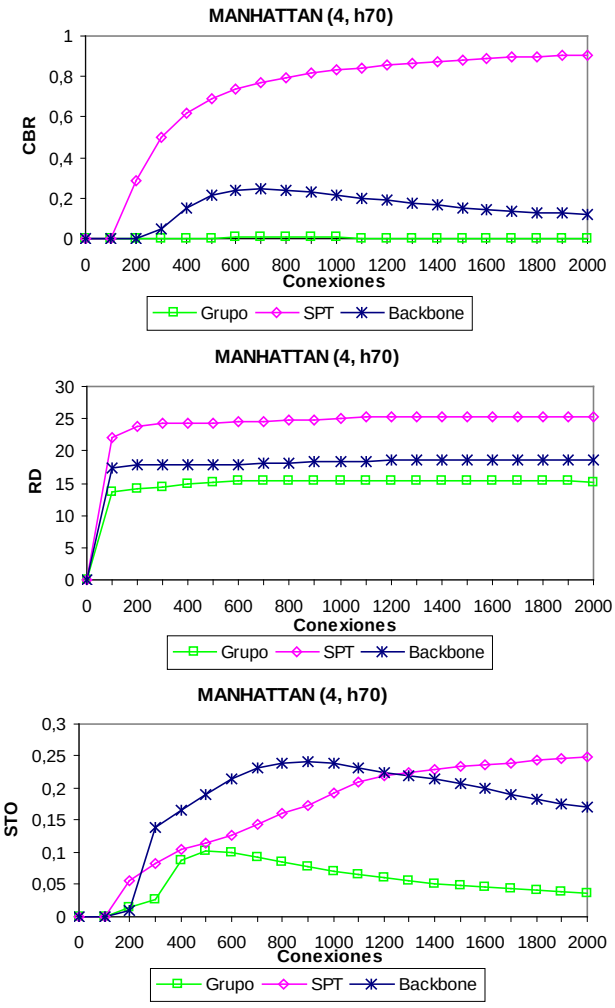

Fig. 11 Manhattan, con puntos calientes.

V – Conclusión y Futuros trabajos.

Los resultados mostrados en la sección anterior, se analizaron considerando los experimentos por separado según el tipo de carga, *uniforme* o con *Puntos Calientes*.

Considerando una carga uniforme, y con pocos nodos destinos, el algoritmo *Grupo* obtuvo buenos resultados comparándolo con los demás algoritmos. Los valores de la métrica *RD* obtenidos (en los tres tipos de topologías) para *Grupo*, fueron menores que para *SPT y Backbone*. Esto representa utilizar menos recursos de la red.

Con carga uniforme tanto para *ARPANET* y *USANET* se obtienen los valores más bajos en casi todas las métricas. Situación que se mantiene al aumentar el número de nodos destino. Lo cual indica que para una red no muy extensa con grado de conectividad de 2 a 3, el algoritmo *Grupo* funciona bien.

Para la red MANHATTAN mantiene su buen comportamiento, sobre todo al aumentar el número de nodos destino.

Al trabajar con un tipo de carga *no-uniforme*, lograda mediante *Puntos Calientes* (simulando una red desbalanceada), los algoritmos muestran (para los mismos experimentos) valores mayores en comparación con los obtenidos con distribución uniforme. Los valores de *CBR* aumentan al trabajar con la red desbalanceada, lo cual indica una mayor probabilidad de bloqueo para las futuras llamadas. El valor de la métrica *RD* aumenta para los algoritmos *SPT* y *Backbone*, siendo un aumento muy pequeño para *Grupo*.

Al crecer el número de nodos destinos, el algoritmo *Grupo* mantuvo la misma cantidad de enlaces promedio utilizados, tanto para redes desbalanceadas como balanceadas.

Teniendo en cuenta los resultados anteriores, podemos concluir que el algoritmo *Grupo* se comporta bien bajo distintas tipos de redes con diferentes condiciones de funcionamiento. Estas condiciones abarcan carga baja o alta y balanceada o no. Comportándose mejor cuando el conjunto de nodos destino aumenta.

Algo importante para destacar es cuando se consideran las condiciones de funcionamiento reales de una red, se debe tener en cuenta el grado de conectividad de los nodos. A medida que este grado es mayor, aumentan las conexiones que se pueden encontrar. Las opciones de elección aumentan. El problema se plantea al no poder agrandar mucho este valor. Es decir, los switches (nodos para nuestro caso) actuales no poseen la capacidad de llevar a cabo un gran número de copias de un mismo paquete. En [BAUER 95] se demuestra que con un grado de conectividad cercana a 3, es posible obtener resultados aceptables o buenas soluciones de conexión.

1 Futuros trabajos.

En este trabajo el algoritmo entrega como resultado un árbol de menor costo, pero es necesario tener en cuenta otras variables cuando trabajamos con aplicaciones multimediales. Hay aspectos que no fueron considerados en el desarrollo de este algoritmo, los cuales pueden formar parte de trabajos futuros. Estos aspectos serían los siguientes:

- Abarcar el problema de la *escalabilidad* para grandes redes, con grupos de destinos dispersos. Permitir que el algoritmo pueda funcionar correctamente mientras el grupo de nodos destino crece.
- Dar solución al problema relacionado con la dinámica del grupo de nodos destino (entrada y salida). El algoritmo debe soportar la *llegada* y *salida* de nuevos nodos al árbol multicast formado, sin que afecte el desempeño del algoritmo y/o las conexiones existentes.
- Considerar las restricciones de *Calidad de Servicio* (QoS) solicitada por ciertas aplicaciones, en cuanto al retardo desde la fuente a cada uno de los destinos, como así también las variaciones de retardo entre los nodos destino.
- Analizar la implementación del algoritmo propuesto, como parte de un protocolo de ruteo multicast.

Referencias.

[BAUER 95] Bauer, F., Varma A., "*Degree-Constrained Multicasting in Point to Point Networks*", IEEE INFOCOM 1995, pp 369-376.

[BAUER 97] Bauer F. and A. Varma, *"ARIES: A Rearrangeable Inexpensive Edge-Based On-Line Steiner Algorithm*", IEEE Journal on Selected Areas in Communications, Abril 1997.

[BHARATH 93] Bharath-Kumar K., Jaffe J. "*Routing to multiples destinations in computer networks*" IEEE Trans. Comm., vol 31, no. 2, pp. 343-353, Mar. 1983.

[CORMEN 90] Cormen T., Leiserson C. "*Introduction to Algorithms*" Cambrigde, MA: MIT Press., McGraw Hill 1990.

[DIOT 97] Diot Ch., Dabbous W., Crowcroft J., "*Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms*", IEEE Journal on Selected Areas in Communication, Vol 15, No 3, April 1997.

[GALLAGER 87] Gallager R., Humblet P., Spira P. "*A distributed algorithm form minimum weight spanning tree*", ACM Transaction on programing lenguages and systems, Vol 5, N 1, January 1983, pages 66-77.

[HUANG 94] Huang N., Wu C, y Wu Y. "*Some routing problems on Broadband ISDN*", Computer Networks adn ISDN Systems 27, 101-116, 1994.

[HWANG 92] Hwang F. and Richards D., *"Steiner tree problems*", IEEE Networks, vol. 22, pp 55-89, Jan. 1992.

[NORONHA 94] Noronha C. and Tobagi, F. *"Optimum routing of multicast streams*", in IEEE INFOCOM 94 vol. 2, Toronto, June 1994 pp. 865-873.

[OBRACZKA 98] Obraczka K. "*Multicast Transport Protocols: A survery and Taxonomy*" IEEE Communication Magazine, January 1998, pp. 94-102.

[SALAMA 97] Salama H., Reeves D., Viniotis Y., "*Evaluiation of multicast routing Algorithms for Real-Time Communication in High-Speed Network*", IEEE Journal on Selected Areas in Communication, Vol 15, No 3, April 1997.

[WAXMAN 88] Waxman B. "*Routing of multipoint connections*", IEEE J. Selected Areas Comm. vol. 6, no. 9, pp. 1617-1622, Dec. 1988.

[WINTER 87] Winter P. "*Steiner problem in networks:A survey*", IEEE Networks, vol.17, no.12, pp.129-167,1987.