

Evaluación de Algoritmo de Confiabilidad Multipunto Basado en Árbol de Recuperación.

Jeannette Galleguillos Bustamante
Departamento de Computación
Universidad de Valparaíso
Gran Bretaña 1041, Valparaíso. FAX: 32-508301
Alumna en tesis de magíster, Departamento de Electrónica
Universidad Técnica Federico Santa María.
j.galleguillos@uv.cl
janet@atmlab.utfsm.cl

Reinaldo Vallejos Campos
Departamento de Electrónica
Universidad Técnica Federico Santa María
reinaldo@elo.utfsm.cl

Marta Barría Martínez
Departamento de Computación
Universidad de Valparaíso
marta.barria@uv.cl

Resumen

En este artículo se presenta la evaluación de rendimiento de un algoritmo multicast confiable, llamado RMART, que está basado en la recuperación local de los paquetes perdidos. Las medidas de desempeño que se evaluaron fueron la latencia y la implosión. El método de evaluación de rendimiento esta compuesto de dos partes. La primera consiste en hacer un análisis matemático del algoritmo, fruto del cual se obtienen algunas ecuaciones que permiten evaluar la latencia media y la implosión media en base a las condiciones de operación del algoritmo. La segunda parte se encarga de generar un gran número de topologías de red aleatorias; y luego, para cada topología de red, se evalúa la implosión y la latencia usando las ecuaciones obtenidas.

Para efectuar en términos prácticos el proceso de evaluación de rendimiento descrito arriba, se desarrolló un módulo dentro de un software especializado. Este software posee varios atributos: genera grafos aleatorios en base a parámetros que son escogidos por el usuario, permite visualizar la topología de la red y la operación del algoritmo de recuperación de errores, y evidentemente, permite evaluar el rendimiento de RMART.

Palabras claves: multicast, confiabilidad, software.

1. Introducción

Las aplicaciones multicast están divididas en aplicaciones que requieren que la transmisión de información se haga en forma confiable, lo que significa, que se garantice que los paquetes transmitidos lleguen sin error a sus destinos; y aplicaciones que no necesitan de tal confiabilidad. Aplicaciones como videoconferencias, audio sobre internet, eventos multimediales, y otros, no requieren transmisión confiable, debido a que en estos casos puede ocurrir cierto nivel de pérdida de información sin que el usuario detecte un deterioro significativo en el servicio recibido. Por otro lado, aplicaciones como distribución de software, juegos interactivos, transacciones de cuentas bancarias, y otras, requieren transmisión confiable, lo que significa que todos los paquetes transmitidos deben ser recibidos sin errores por los receptores (miembros del grupo multicast).

Para el caso de las aplicaciones que requieren que la comunicación multicast sea confiable, es necesario diseñar un algoritmo que ofrezca dicha confiabilidad, es decir que permita recuperar los paquetes que se pierden. Para lograr esto, se han propuesto muchos algoritmos. Una cuestión evidente que surge entonces, es establecer algún procedimiento que permita evaluar las ventajas y desventajas de cada algoritmo, y consecuentemente, decidir cuál es el más conveniente.

Los mecanismos de recuperación de error están constituidos básicamente por una fase de detección y otra de corrección. Los algoritmos multicast confiable pueden usar esquemas de reconocimientos positivos (ACKs) o negativos (NACKs) para asegurar la confiabilidad de los mensajes transmitidos. Cuando un receptor envía un ACK confirma la correcta llegada del paquete, en cambio cuando envía un NACK solicita la retransmisión del paquete. La fase de detección normalmente se lleva a cabo en los receptores, ya que resuelven esta tarea en forma más eficiente que cuando la ejecuta la fuente. Para recuperar un paquete perdido los receptores envían al transmisor un paquete especial llamado NACK que le indica a la fuente que debe retransmitir el paquete perdido. Aunque el mecanismo de recuperación de errores recién descrito parece muy simple, en la práctica surgen algunas dificultades, ya que si cada uno de los receptores afectados por la pérdida de un paquete enviara un NACK a la fuente, en ella se produciría la llegada sincronizada de (posiblemente muchos) NACKs. Este fenómeno se conoce con el nombre de “implosión de NACKs”. La implosión de NACKs es indeseable porque produce congestión en la fuente y alrededor de ella, lo que obviamente causa degradación en las medidas de rendimiento globales de la red, tales como retardo y throughput.

En este artículo se presenta la evaluación de un algoritmo de recuperación de error, llamado RMART (Reliable Multicast Algorithm based on Recovery Tree), publicado en [1]. Este algoritmo realiza la recuperación de errores en forma local, lo que significa que únicamente los nodos que se encuentran en la vecindad del error participan de la recuperación. Una consecuencia de la forma específica en que se realiza la recuperación de errores, es que el algoritmo propuesto presenta una latencia (tiempo necesario para recuperar un error) y una implosión (número de mensajes que recibe la fuente solicitando la retransmisión del paquete perdido) que compiten favorablemente con los algoritmos actuales.

En [1] y [2] se presentó el análisis del desempeño del algoritmo RMART para una topología estrella y para una topología cadena. Como continuación de ese trabajo y para evaluar el algoritmo en una topología general, en este trabajo se presenta un análisis matemático que es útil para la evaluación del algoritmo en cualquier topología de red.

En lo restante, este artículo está organizado de la siguiente manera: en la Sección 2, y para hacer este trabajo autocontenido, se entrega una descripción del algoritmo multicast confiable RMART, que fue documentado en [1]. La Sección 3 presenta medidas de desempeño que son útiles para evaluar el algoritmo en una topología cualquiera. La Sección 4 describe el software que se desarrolló para hacer el análisis de rendimientos. La Sección 5 presenta algunos resultados, y por último, en la Sección 6, se presentan las conclusiones de este trabajo.

2. Descripción del algoritmo RMART

En una operación normal, que es, cuando no se han perdido paquetes, cualquier nodo del árbol de distribución (sea o no un miembro del grupo) transmite todos los paquetes que recibe desde su padre a todos sus hijos en el árbol de distribución. De esta forma, un paquete llega a todos los miembros del grupo. Adicionalmente, la primera vez que un

nodo del árbol de distribución que es además un miembro del grupo, recibe un determinado paquete, al transmitirlo a sus descendientes, éste almacena una copia del paquete en una memoria. Esta copia es usada para retransmitir el paquete en caso de que reciba de algún miembro una solicitud de retransmisión. Los nodos del árbol de distribución que no son miembros del grupo multicast, actúan solamente como transmisores intermediarios de los paquetes y NACKs que ellos reciban.

Si un determinado paquete se pierde en la transmisión, el algoritmo usado para la recuperación de error es el siguiente:

1. En este caso donde un nodo miembro del grupo, el nodo i , detecta la pérdida de un paquete, este inmediatamente transmite un mensaje a todos sus nodos hijos. El mensaje es tratado como un paquete normal por los nodos descendientes de i (y además llega a todos ellos). Este es un mensaje de inhibición el cual informa que el nodo i ha detectado la pérdida de un paquete, así impide que sus hijos intenten enviar su propio NACK. Después de transmitir el mensaje de inhibición, el nodo i continúa transmitiendo a sus descendientes los otros paquetes que llegan desde la fuente.
2. Luego, el nodo i decide si envía o no un NACK a través del árbol de distribución. Para decidir si enviar o no el NACK, i efectúa un experimento aleatorio con una distribución Bernoulli de parámetro $p_I(i)$, el cual se denomina $Be(p_I(i))$. El parámetro $p_I(i)$ del experimento Bernoulli es elegido como se explica en la sección 2.1 de este artículo. Solo si el resultado de este experimento es exitoso el nodo i envía el NACK en dirección al padre en el árbol de distribución.
3. Entonces, independientemente de tener que enviar el NACK, el nodo i inicia un "Timeout" (TO) para esperar la retransmisión del paquete perdido. Más adelante se explica cómo se elige el valor TO.
4. Si el TO expira antes de que el paquete solicitado sea recibido, el nodo i repite los pasos 2 y 3, pero esta vez el parámetro del experimento Bernoulli tiene una probabilidad $p_n(i)$ asociada con esto, donde n corresponde al número de veces que el nodo i ha repetido el paso 2 del algoritmo en su intento de recuperarse del error en cuestión. La manera en la cual $p_n(i)$ es asignada se explica más adelante.
5. Cuando el nodo i recibe el paquete solicitado, este retransmite el paquete hacia abajo en el árbol de distribución. De esta manera los miembros del grupo que son descendientes del nodo i recuperan el paquete perdido sin haber participado activamente en la recuperación.
6. Cuando un nodo del árbol de distribución que no es un miembro del grupo recibe un NACK desde uno de sus hijos en el árbol de distribución, lo retransmite en dirección al padre en el árbol de distribución.
7. Cuando un nodo que es miembro del grupo recibe un NACK con respecto a un paquete determinado, este retransmite el paquete a sus hijos y elimina el mensaje NACK de la red.
8. Finalmente, cuando un nodo que no fue afectado por la falla recibe el paquete retransmitido, este descarta el paquete de la red.

Al seguir se definen algunos conceptos para explicar la operación del algoritmo de una manera más intuitiva.

- Cualquier sub-árbol del árbol de distribución donde solo la raíz y las hojas son miembros del grupo multicast se define como un *árbol de recuperación*. Esto implica que el árbol de recuperación puede tener nodos que no son miembros del grupo. Esto también implica que un miembro del grupo que no es la raíz ni la hoja del árbol de distribución, es simultáneamente una hoja de un árbol de recuperación dado y es raíz de cualquier otro árbol de recuperación.
- Cada hoja del árbol de recuperación tiene un único árbol asociado a él llamado *árbol de espera*, el cual es un sub-árbol del árbol de distribución, cuya raíz es esa hoja y el cual contiene a todos los descendientes de ese nodo en el árbol de distribución.

Primero, todo paquete que se pierde está asociado con el árbol de recuperación que contiene el enlace donde ocurrió la falla. Más específicamente, el hecho de que un miembro dado detecte una falla implica que el miembro es una hoja en el árbol de recuperación de esa falla (paso 1 del algoritmo). Si un miembro recibe un NACK respecto de un paquete dado que se ha perdido, ese miembro es la raíz del árbol de recuperación para ese paquete perdido (pasos 6 y 7 del algoritmo).

La ejecución del experimento Bernoulli (pasos 2 y 4 del algoritmo) tiene el doble objetivo de disminuir la latencia y la implosión que puede ocurrir en el nodo raíz del árbol de recuperación. La latencia disminuye porque el

experimento Bernoulli se ejecuta en el mismo instante en el cual la falla es detectada, y el resultado de este experimento es además obtenido inmediatamente. Por otro lado, la implosión disminuye debido al hecho de que solo algunas hojas del árbol de recuperación que ejecutan el experimento Bernoulli obtienen un resultado que las obliga a transmitir un NACK. Más precisamente, el tamaño de la implosión es igual al número de hojas del árbol de recuperación afectado por la falla que deciden enviar un NACK, siendo ese número mucho menor que el número de miembros del grupo afectados por la falla. Así, el algoritmo intenta ejecutar un tamaño de implosión aproximadamente igual a 1, y esto se da eligiendo un parámetro apropiado $p_n(i)$, $n \geq 1$, de la distribución Bernoulli.

Finalmente, los miembros del grupo que fueron afectados por la falla, pero no son parte del árbol de recuperación, deben necesariamente ser parte de un árbol de espera. Como una consecuencia, ellos reciben el paquete perdido cuando la raíz de su árbol de espera (la cual es una de las hojas del árbol de recuperación) se recupera de la falla y además retransmite el paquete perdido al árbol de espera (paso 5 del algoritmo).

2.1 Determinación del TO y $p_n(i)$

Notamos primero que cada árbol de recuperación tiene su propio valor TO, lo que quiere decir que árboles de recuperación diferentes pueden tener valores TO diferentes. Otra importante observación es que el TO de un árbol de recuperación dado puede cambiar en el tiempo. Esto puede ocurrir, por ejemplo, cuando un nuevo miembro se une al grupo y llega a ser parte del árbol de recuperación, o cuando un miembro del árbol deja el grupo. Abajo se describe cómo se determina un valor TO:

1. Si un miembro necesita actualizar un valor TO, por ejemplo cuando deja o se une a un grupo, este envía un mensaje de actualización de TO al árbol de distribución.
2. El primer miembro del grupo que recibe el mensaje (que es, la raíz del árbol de recuperación a la cual pertenece el miembro que envía el mensaje) lo elimina de la red, entonces envía un mensaje de evaluación de TO hacia abajo del árbol de distribución. Este mensaje contiene la información del instante en que fue transmitida.
3. Cada miembro que recibe el mensaje de evaluación de TO lleva a cabo las siguientes operaciones: calcula t_d , el tiempo que el mensaje toma en llegar a él; este elimina el mensaje de la red; y envía su propio mensaje de evaluación de TO hacia delante en el árbol de distribución. Este nuevo mensaje contiene el t_d calculado y el tiempo del momento en el cual el mensaje es enviado.
4. Cada mensaje del tipo explicado arriba, que recibe la raíz del árbol de recuperación, calcula el tiempo de propagación t_u , que el mensaje toma en llegar. Así, el retardo del viaje de ida y vuelta, t_r , entre la raíz del árbol de recuperación y el miembro que envía el mensaje anterior está dado por: $t_r = t_d + t_u$. El valor del TO está dado por el máximo valor t_r evaluado por la raíz del árbol de recuperación, más el tiempo de retransmisión del mensaje, más un tiempo de seguridad.
5. Una vez que la raíz del árbol de recuperación ha determinado el valor del TO, ésta envía un mensaje multicast hacia abajo en el árbol de distribución reportando este valor.
6. Finalmente, cada miembro que recibe el mensaje que contiene el valor del TO, almacena esta información en su memoria y elimina el mensaje de la red, bajo el cual el valor del TO queda establecido para todos los miembros que pertenecen al árbol de recuperación.

La evaluación de $p_n(i)$ está dada de la siguiente forma. Sea $\alpha \geq 1$ un parámetro que es usado para ejecutar el equilibrio entre latencia e implosión. Entonces, el valor de $p_1(i)$, la probabilidad de que una hoja i de un árbol de recuperación A envíe un NACK la primera vez que el paso 2 del algoritmo es ejecutado con respecto a una falla determinada, está dada por:

$$p_i(i) = \min \left[1; \frac{\alpha \cdot t_{d\ i}}{S} \right], \quad 1 \leq i \leq R-1 \quad (1)$$

donde $t_{d\ i}$, $1 \leq i \leq R-1$, corresponde al retardo que existe entre la raíz del árbol de recuperación y su hoja i (este retardo es una aproximación de un número de enlaces entre la hoja i y la raíz de ese árbol de recuperación). El valor de S , por otro lado, está dado por: $S = \sum_{\forall \text{ hoja } j \in A} t_{d\ j}$. El valor de S es evaluado para cada nodo raíz de un árbol de

recuperación en una forma análoga a la evaluación del valor de TO. Una vez que S es determinado, este es transmitido por la raíz del árbol de recuperación hacia abajo en el árbol de distribución (hacia las hojas). El mensaje

que contiene S también contiene información del instante en que este fue transmitido. Lo cual permite a cada hoja receptora evaluar el t_d correspondiente. El valor de $p_1(i)$ dado en la ecuación (1) está designado de tal forma que el menos un NACK sea generado con una alta probabilidad en la primera iteración del algoritmo (para que se produzca una latencia baja), y también para que el número de NACKs enviados sea bajo (baja implosión). Sin embargo, si después del primer *timeout* una hoja del árbol de recuperación está esperando un paquete retransmitido que no ha sido recibido, es altamente probable que el NACK no haya sido enviado porque la falla no afectó a las hojas del árbol de recuperación. Se asume que el árbol de distribución es r -ario, $p_n(i)$ está aumentado por el factor r en relación a la probabilidad usada en la previa iteración. Que es:

$$p_n(i) = \min \begin{cases} 1 \\ r \cdot p_{n-1}(i) \end{cases}, n > 1 \quad (2)$$

donde i es la hoja del árbol de recuperación afectada por la falla.

3. Análisis de Desempeño de una Topología Estrella

En [1] se analizó la latencia y la implosión en una topología estrella (ver Figura 1) con distancias iguales entre la raíz y cada uno de los receptores. En esta topología hay R miembros del grupo multicast, de los cuales $R-1$ son receptores. Existe un nodo intermedio que no pertenece al grupo multicast, cuya única tarea aquí es retransmitir los paquetes que llegan desde la fuente a los nodos miembros del grupo multicast, y retransmitir los NACKs a la fuente.

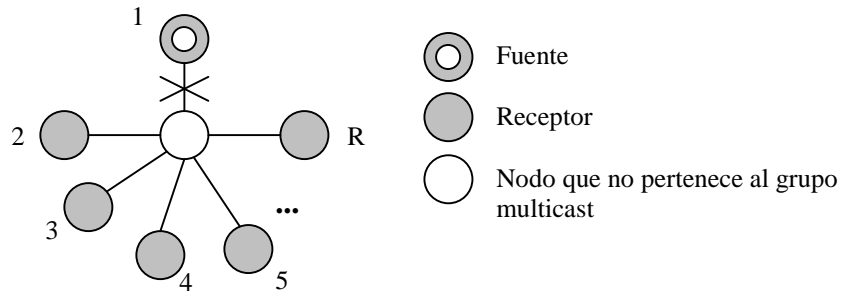


Figura 1: Topología Estrella. Posee R miembros del grupo, $R-1$ receptores y R enlaces.

Se asumió que el retardo que existe entre cada uno de los enlaces es el mismo, igual a 1. Además, de acuerdo a la Ecuación (1), la probabilidad que una hoja envíe un NACK la primera vez que la pérdida de un paquete determinado sea detectada está dada por:

$$p_1(i) = \min \left[1; \frac{\alpha}{R-1} \right], 1 \leq i \leq R-1 \quad (3)$$

La probabilidad de que una hoja envíe un NACK en iteraciones subsecuentes ($n > 1$), de la Ecuación (2), y considerando que α es siempre mayor o igual que 1, se tiene que:

$$p_n(i) = 1; n > 1; 1 \leq i \leq R \quad (4)$$

Las Ecuaciones (3) y (4) determinan el valor de p_n , $n \geq 1$, lo cual completa la especificación de la operación del algoritmo. Luego, en [1] se encontró la evaluación de la latencia media ($E[T]$) y la implosión media ($E[I]$), las cuales se reproducen a continuación.

$$E[T] = \sum_{i=0}^{R-1} \left(\sum_{n=1}^N E[T | n, i] P(n | i) \right) q(i) \quad (5)$$

donde el índice i identifica el enlace en cual ocurre la falla: si $i=0$, la falla ocurre en el enlace incidente a la fuente; y si $1 \leq i < R$, ésta ocurre en el enlace incidente en el receptor i . El índice n identifica el número de iteración del paso 2 del algoritmo, y N corresponde al máximo valor de n , es decir, N corresponde al número de iteraciones en el cual $p_N(i)=1$ es ejecutada. $E[T/n, i]$ corresponde al valor medio de la latencia dado que la falla ocurrió en el enlace i y al

menos un NACK es enviado en la n -ésima iteración del paso 2 del algoritmo. La probabilidad de que la falla ocurra en el enlace i es $q(i)$. Finalmente, $P(n|i)$ es la probabilidad que al menos un NACK sea enviado la primera vez en la n -ésima iteración del paso 2 del algoritmo, dado que la falla ocurrió en el enlace i .

Se usó el hecho de que la falla ocurre con igual probabilidad en cada enlace, es decir, $q(i)=1/R$, $0 \leq i < R$. Se asume también de que NACKs y paquetes retransmitidos no se pierden. Esto implica que $E[T]=n \cdot TO$.

Por otro lado, de forma similar, se calcula la media de la implosión está dada por [1]:

$$E[I] = \sum_{i=0}^{R-1} \left(\sum_{n=1}^N E[I | n, i] P(n | i) \right) q(i) \quad (6)$$

Las Ecuaciones (5) y (6) permiten evaluar la latencia y la implosión de RMART operando en una red con una topología estrella, sin embargo, se requiere hacer la evaluación en cualquier topología de red. En una topología de red cualquiera, una falla genera un árbol de recuperación con una topología cadena (compuesta de la fuente y un solo receptor), o genera una topología estrella de largo variable. El caso de un topología cadena ya fue analizado en [2]. Luego, analizar el rendimiento de RMART se traduce a analizar el rendimiento de una topología estrella de largo variable. Lo que se presenta a continuación.

3.1 Análisis de Desempeño de Topología Estrella de Distintos Largos

En esta sección se muestra el cálculo de $E[T]$ y $E[I]$ para una topología estrella con enlaces de distintos largos (ver Figura 2). Los enlaces con distintos largos en la topología estrella generan distintas probabilidades de envío de un NACK, en la primera iteración de la ejecución del algoritmo.

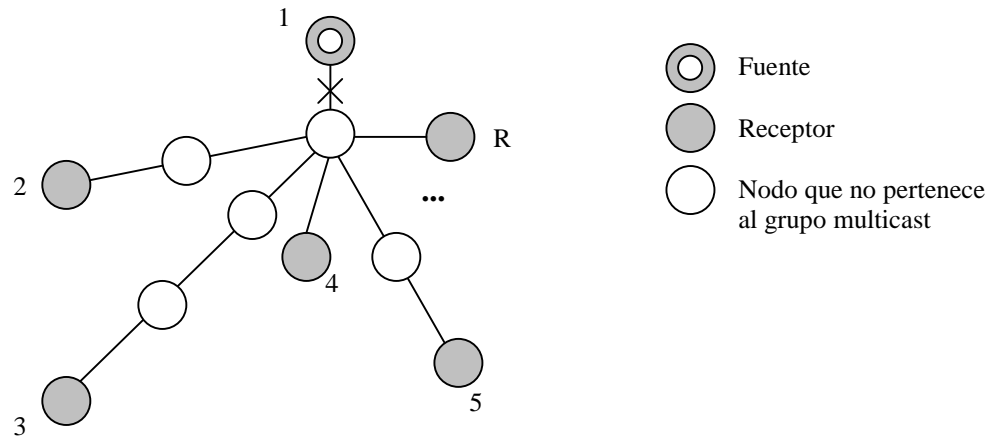


Figura 2: Topología estrella de distintos largos. Posee R miembros del grupo, $R-1$ receptores y $NL \geq R$ enlaces.

En [1] se calcularon las probabilidades $P(1/i)$ y $P(2/i)$. Se define $P(1/i)$ como la probabilidad que el receptor i responda con probabilidad $p_1(i)$ en la primera iteración del algoritmo, dado que la falla ocurrió en el nodo adyacente al receptor i . Para el caso bajo análisis, no se usa la Ecuación 2, sino que se supone que si el receptor i no responde en la primera iteración, entonces responderá en la segunda iteración con probabilidad 1, dado que no respondió en la primera iteración. En consecuencia:

$$\begin{aligned} P(1/i) &= p_1(i) & 1 \leq i < R \\ P(2/i) &= (1 - p_1(i)) \cdot 1 \end{aligned} \quad (7)$$

La probabilidad de que algún receptor responda en el primer intento dado que la falla ocurrió en el enlace adyacente a la fuente ($i=0$) es:

$$P(1/0) = 1 - \prod_{i=1}^{R-1} (1 - p_1(i)) \quad (8)$$

ya que, $\text{prob}(\text{algún receptor responda}) = 1 - \text{prob}(\text{ningún receptor responda})$.

La probabilidad de que algún receptor responda en el segundo intento, dado que la falla ocurrió en el enlace adyacente a la fuente, corresponde a la probabilidad de que ningún receptor transmita el NACK en el primer intento, multiplicado por la probabilidad de que cada receptor transmita un NACK en la segunda iteración, lo cual es 1. Entonces

$$P(2/0) = \prod_{i=1}^{R-1} (1 - p_1(i)) \cdot 1 \quad (9)$$

En base a los valores de probabilidades condicionales recién calculados, en la siguiente subsección se calculará $E[T]$ y $E[I]$.

3.2 Cálculo de Latencia Media e Implosión Media

El valor esperado de la latencia media dado que la falla se produjo en el enlace i , en la n -ésima iteración del algoritmo puede calcularse como $E[T/n, i] = n \cdot TO$.

Manteniendo la suposición que todos los enlaces del árbol tienen la misma probabilidad de falla, la probabilidad de que la falla se produzca en el enlace i es $q(i) = 1/NL$, ya que existe NL enlaces en total.

Reemplazando las Ecuaciones (7), (8) y (9) en la Ecuación (5) se tiene que:

$$E[T] = \frac{TO}{NL} \left(2R - 1 - \prod_{i=1}^{R-1} (1 - p_1(i)) - \sum_{i=1}^{R-1} p_1(i) \right) \quad (10)$$

que representa la latencia media en una topología estrella con enlaces de distintos largos.

Para calcular la implosión media (Ecuación [6]) se necesita, además de algunos cálculos de probabilidades anteriores, el cálculo de $E[I/1,0]$, $E[I/2,0]$, $E[I/1,i]$ y $E[I/2,i]$. Estos fueron obtenidos en [1] para una topología estrella:

$$E[I/1,0] = \frac{\sum_{i=1}^{R-1} i \cdot \Pr[i \text{ nacks sean transmitidos en la primera iteración}]}{1 - \Pr[0 \text{ nacks sean transmitidos en la primera iteración}]}$$

Entonces, considerando que la topología analizada ahora tiene distintos largos en sus enlaces, se tiene que las probabilidades son también distintas para cada nodo (no interesa el número de enlaces), y de esto se obtiene que:

$$E[I/1,0] = \frac{\sum_{i=1}^{R-1} i \cdot (1 - p_i(i))}{1 - \prod_{i=1}^{R-1} (1 - p_i(i))} \quad (11)$$

$$E[I/2,0] = R - 1 \quad (12)$$

La Ecuación (12) quiere decir que en la segunda iteración, todos los receptores responden con probabilidad 1, lo que hace que todos transmitan su NACK, llegando $R-1$ NACKs a la fuente (despreciando la probabilidad de error en el envío del NACK).

$$E[I/1,i] = E[I/2,i] = 1, \quad 1 \leq i < R \quad (13)$$

En el caso de la Ecuación (13), solamente la hoja afectada por la falla envía el NACK a la raíz de árbol de recuperación. Luego, reemplazando las Ecuaciones (11), (12) y (13) en la Ecuación (6) se llega a:

$$E[I] = \frac{1}{R} \left(\sum_{i=1}^{R-1} i \cdot (1 - p_i(i)) + R - 1 \right) \quad (14)$$

que corresponde a la implosión media para una topología estrella de distintos largos. Las Ecuaciones (11), y (14) no han sido presentadas en otro artículo.

4. Descripción del Software

Para desarrollar el software que permitiera implementar el análisis matemático precedente, bajo cualquier condición de operación de RMART, se utilizó como base un software que había sido programado en Delphi 4.0, llamado Bestway, y que fue descrito en [3]. Bestway es capaz de desplegar gráficamente redes y ejecutar algoritmos de ruteamiento unicast y multicast. Para el caso de ruteamiento unicast, para encontrar los caminos más cortos entre los nodos, utiliza el algoritmo de Dijkstra. En el caso de ruteamiento multicast, dada la fuente, los receptores y la política de ruteamiento a utilizar, Bestway encuentra el árbol de distribución de información desde la fuente a cada uno de los receptores, que es necesario para la transmisión de la información. Por ejemplo, en la Figura 3 se muestra la red USAnet con la fuente elegida y los receptores que forman el árbol de distribución. Bajo cada nodo se muestra una etiqueta que muestra el tipo de nodo: fuente o destino.

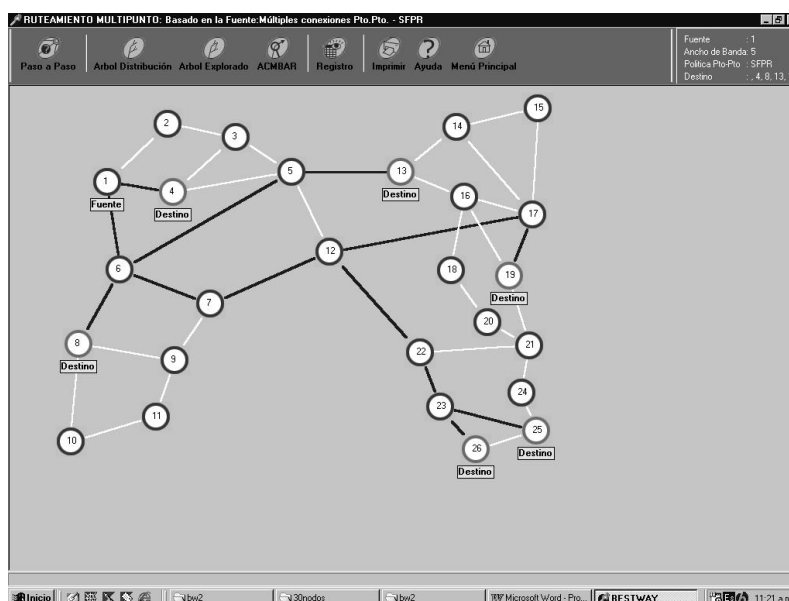


Figura 3: Red USANET con árbol de distribución y fuente en el nodo 1.

Entonces, en la primera fase de este trabajo se desarrolló un módulo para la generación de topologías aleatorias. incorporado en Bestway, que permite la generación aleatoria de topologías de red utilizando el método propuesto por Waxman[6] para generar grafos. Este método fue ampliamente utilizado en [4] y [5].

Posteriormente se desarrolló un módulo para la evaluación del algoritmo de confiabilidad multipunto RMART, en base a las Ecuaciones (11) y (14) presentadas en este artículo.

4.1 Generación de Topologías Aleatorias

Existen varios métodos para generar topologías aleatorias como por ejemplo: grafos aleatorios (Random), grafos exponenciales y grafos aleatorios de Waxman [6] (todos estos métodos fueron comparados en [4]).

En [6], Waxman propuso dos métodos de generación de grafos aleatorios que han sido ampliamente aceptados por investigadores. Uno de estos métodos se basa en generar distancias aleatorias entre cada par de nodos. Específicamente, la probabilidad de que exista un enlace entre los nodos u y v está dada por:

$$P(u, v) = \beta \exp\left(\frac{-d(u, v)}{\gamma \cdot L}\right) \quad (15)$$

donde $d(u, v)$ es la distancia entre los nodos u y v , y L es la máxima distancia entre dos nodos cualquiera. Los parámetros γ y β varían en el rango $(0, 1]$. Valores pequeños de γ generan enlaces largos y valores grandes de β producen grafos que, en promedio, presentan un alto grado (nivel de conectividad) en sus nodos.

En resumen, para definir la topología del grafo, el método de Waxman procede de la siguiente forma; para cada par de nodos (u, v) de un grafo de n nodos, se determina la probabilidad $P(u, v)$, de que exista un enlace entre ellos, de acuerdo a la Ecuación (15); posteriormente, en base a un experimento aleatorio (con distribución Uniforme) se determina en definitiva si el enlace existe o no.

Un problema de los grafos de Waxman es que entre los grafos resultantes se generan también grafos disconexos, a pesar de adecuar sus parámetros, los cuales no sirven para la evaluación de algoritmos. El segundo problema es que entre los grafos generados existen grafos con nodos terminales, o sea, nodos de grado uno. Esta última situación no representa bien la realidad, ya que los nodos de las redes reales poseen al menos dos conexiones a diferentes nodos. Esta propiedad se conoce como Two-Connected, y quiere decir que todo nodo en el grafo es de al menos grado dos. Noronha y Tobagi en [5] mostraron, usando simulación, que el rendimiento de un algoritmo de ruteamiento multicast cuando es aplicado a una red real es casi idéntico al rendimiento de un algoritmo que es aplicado a una red aleatoria Two-Connected. Un generador aleatorio así se utilizó en [7].

El módulo generador de grafos aleatorios de Waxman, descrito aquí, descarta automáticamente una topología cuando ésta es un grafo disconexo. Y si el grafo tiene nodos terminales (de grado uno), hace una modificación al grafo, generando otro enlace que se conecte desde el nodo terminal al nodo más cercano (o de menor costo). De esta forma, todos los grafos generados son conexos y Two-Connected.

4.2. Módulo ACMBAR

Utilizando como base el software BESTWAY se implementó un módulo denominado ACMBAR, que permite evaluar el algoritmo RMART. El método para la evaluación de rendimiento utilizado por ACMBAR, procede como se describe a continuación.

Utilizando el método de Waxman se generan diferentes topologías de n nodos, donde n es un parámetro que escoge el usuario. Para cada topología, se elige un grupo multicast de tamaño m ($m \leq n$) con su respectiva fuente. Luego, aplicando alguna política de ruteamiento multicast de las que posee Bestway, se encuentra el árbol de distribución entre la fuente y los receptores. A continuación, para cada árbol de distribución se evalúan las medidas de rendimiento.

Para evaluar las medidas en un árbol de distribución específico, se procede de la siguiente forma: se simula una falla en cada enlace del árbol de distribución. La falla en un enlace determinado da origen a un árbol de recuperación específico. En consecuencia, para cada árbol de recuperación generado se calculan las medidas de desempeño, las cuales finalmente se promedian, obteniéndose de esta forma las medidas de rendimiento para un árbol de distribución particular.

Una falla puede dar origen a dos tipos de árbol de recuperación. Un tipo de árbol de recuperación es el compuesto por la fuente y un solo receptor; y el otro es el compuesto por la fuente y dos o más receptores. En consecuencia, para obtener las medidas, el programa identifica el tipo de árbol de recuperación generado por la falla, y según ese tipo de árbol de recuperación, utiliza las ecuaciones correspondientes, que se describieron en la sección 3 de este trabajo.

5. Resultados

Los resultados se obtuvieron para 300 grafos distintos, generados aleatoriamente. Los grafos generados poseían 40 nodos. El grupo multicast fue variando entre 5 y 39 miembros, y α varió entre los valores 1 y 5. Cada uno de los

grafos utilizados en la evaluación tiene distancia 1 en sus enlaces. Como política de ruteamiento para encontrar el árbol de distribución, se utilizó el algoritmo SFPR.

El gráfico de la Figura 4 muestra que cuando el grupo multicast aumenta, la latencia media disminuye. Además se observa que cuando el tamaño del grupo se acerca al número de nodos del grafo, la latencia es mínima, evaluada para distintos valores de α . Esto se debe a que cuando el tamaño del grupo multicast es mayor, la recuperación local del error tiene mayor influencia, acortando de esta manera el tiempo usado en la recuperación del error. En el gráfico de la Figura 5, puede observarse que la implosión crece levemente cuando el grupo multicast aumenta, manteniéndose entre 1.1 y 1.42, para cualquier α . Esto corrobora que el algoritmo previene del peligro de implosión y tiene una alta escalabilidad.

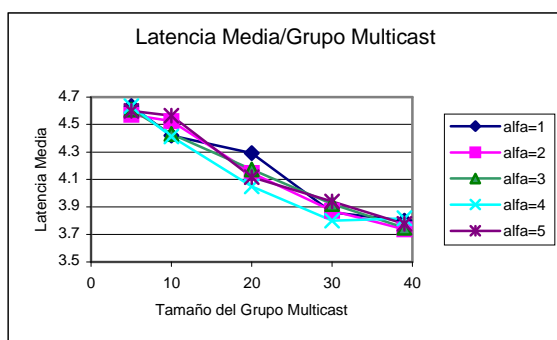


Figura 4: Latencia media v/s Tamaño del grupo.

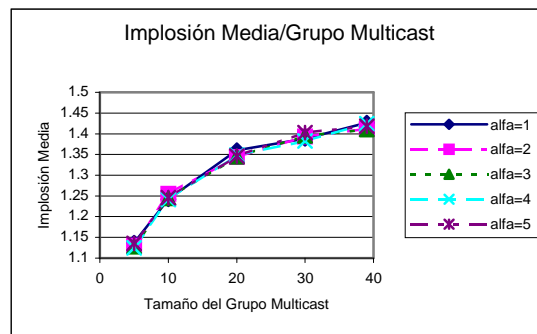


Figura 5: Implosión Media v/s Tamaño del grupo.

6. Conclusiones

En este trabajo se presentó la evaluación de rendimiento del algoritmo RMART para una topología estrella con enlaces de distintos largos. Las medidas de rendimiento (latencia media e implosión media) que permitieron evaluar el algoritmo RMART se obtuvieron en base a distintos tipos de grafos aleatorios. Para esto fueron desarrollados dos módulos que permitieron la evaluación del algoritmo. El primer módulo genera topologías aleatorias en base al método de Waxman. El segundo módulo calcula las medidas analíticas obtenidas en la Sección 3 de este trabajo. Según los resultados numéricos obtenidos, el algoritmo presenta una latencia mínima y baja implosión para grupos multicast grandes, lo que demuestra que es altamente escalable.

Agradecimientos

Este trabajo ha sido parcialmente financiado por Proyecto FONDECYT 1000055/2000 y por el Proyecto DIPUV19-2001.

Bibliografía.

- [1] M. Barría, R. Vallejos, L. F. G. Soares. "A reliable multicast algorithm based on recovery tree (RMART)". 9th Conference on performance modeling and evaluation of ATM & IP networks 2001, Budapest.
- [2] M. Barría, R. Vallejos, L.F.G. Soares. "Algoritmo de confiabilidad multipunto basado en árbol de recuperación (ACMBAR)". CLEI 2000, Ciudad de México, México, Septiembre 2000.
- [3] R. Vallejos, J. Olivares, P. Roncagliolo, A. Zapata. "BESTWAY: Una herramienta para visualizar la operación de algoritmos de ruteamiento en redes de computadores". IV Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica, 13-15 de Septiembre 2000, Barcelona, España.
- [4] E. Zegura, K. Calvert, M. Donahoo. "A quantitative comparison of graph-based models for internet topology". IEEE/ACM Transaction on Networking, 5(6):770-783. December, 1997.
- [5] C. Noronha, F. Tobagi. "Evaluation of multicast routing algorithms for multimedia streams". IEEE ITS 94 Proceedings, Rio de Janeiro, Brazil, August 1994.
- [6] B. Waxman. "Routing of multipoint connections". IEEE Journal on Selected Areas in Communications, 6:1617 – 1622, 1988.
- [7] H. Salama, D. Reeves y Y. Viniotis. "Evaluation of multicast routing algorithms for real-time communications on high-speed networks". IEEE Journal on Selected Areas in Communications, Vol 15, No. 3, p. 332-345, April, 1997.