Implementación y Análisis de CBTv2 en el medioambiente Ns

Guillermo Rigotti
Universidad Nacional del Centro de la Provincia de Bs. As.
Instituto de Sistemas - San Martin 65 - Tandil(7000) - Argentina
Dirección electrónica: grigotti@exa.unicen.edu.ar

Resumen

Desde la definición del modelo de servicio multicast en IP [1], el uso de estas facilidades ha ido adquiriendo cada vez mayor importancia en el ámbito de la Internet. El número de aplicaciones que demandan transmisión multicast, tales como conferencias de audio, simulación interactiva distribuida, etc., ha aumentado en forma continua, viéndose reflejado en el crecimiento experimentado por el MBONE, experimental en sus inicios en 1992 y actualmente considerado indispensable para decenas de miles de usuarios.

Los protocolos de ruteo multicast son los encargados de construir y mantener los árboles de distribución por los que fluye la información multicast.

Estos protocolos deben ofrecer una performance adecuada a los requerimientos de una diversidad de aplicaciones (demora máxima acotada, diferencia entre demoras acotada, etc.), y a su vez realizar un uso eficiente de los recursos de la red. Es además de suma importancia su escalabilidad, ya sea en cuanto a la cantidad de grupos, número de emisores por grupo, dispersidad de los grupos y extensión de la red sobre la cual operan.

Mientras que los primeros protocolos fueron concebidos para grupos densos y áreas reducidas (DVMRP[2], MOSPF[3]), basándose en el paradigma source-group trees y en mecanismos de integración a los grupos por defecto y podas explícitas, otros como SM-PIM[4] y fundamentalmente CBT versión 2[5][6][7], han sido diseñados para lograr escalabilidad, basándose en el paradigma shared trees para la construcción del árbol de distribución y en la solicitud explícita de integración a un grupo por parte de los routers interesados.

Para poder estudiar las características de estos protocolos, es necesario recurrir a la simulación, debido a que en algunas ocasiones un modelo analítico resulta en demasiadas simplificaciones que lo alejan de la realidad. Por otra parte, en la mayoría de los casos es imposible disponer de redes reales de las dimensiones necesarias para llevar a cabo la experimentación.

En este trabajo se presenta una implementación del protocolo CBT versión 2 en el medioambiente de simulación provisto por Ns[8], y un análisis de los costos de recuperación (PDUs transmitidas) y demoras que insume la reconstrucción de subárboles de distribución que quedan aislados como consecuencia de fallas en los vínculos de transmisión.

Palabras clave: Multicast, Ruteo, CBT, Shared trees, Simulación de protocolos.

1. Introducción

Los protocolos de ruteo multicast tienen la función de construir y mantener árboles de distribución para los distintos grupos. Dichos árboles están constituidos por información almacenada en las tablas de ruteo multicast de los nodos involucrados, siendo utilizada para la diseminación en la red de paquetes multicast.

Se puede distinguir dos aspectos de importancia en este tipo de protocolos: 1-de qué manera satisfacen los requerimientos de las diversas aplicaciones y 2-los recursos de red consumidos.

En cuanto al primero de ellos, puede citarse por ejemplo requerimientos de aplicaciones tales como simulación interactiva distribuida, que demanda rapidez en el proceso de integrarción a un grupo (join), conferencias de audio, que requieren demoras máximas acotadas, etc.

El segundo aspecto se refiere por un lado a los recursos que se consumen

como consecuencia de la actividad de control (tamaño de las tablas de ruteo y volumen de la información de control intercambiada entre los routers) y por otro a la cantidad de tráfico inyectado en la red como consecuencia de la distribución de los datos (datagrams multicast) generados por las aplicaciones.

Los dos aspectos mencionados se ven influenciados por las características que presenten los grupos multicast, que están determinadas por el tipo de aplicación y la distribución de los usuarios en la red. Es posible encontrar grupos con miembros conocidos y estáticos o grupos en los que los miembros cambian dinámicamente; grupos densos, en los cuales la mayoría de los nodos conforman el grupo, o grupos dispersos (sparse groups), conformados por una pequeña cantidad de nodos respecto de la totalidad de los que integran la red; grupos en los cuales uno o muy pocos de los participantes son emisores, o grupos en los que gran cantidad (o la totalidad) de los participantes son emisores.

Una característica importante a considerar es el grado de escalabilidad que posee un protocolo de ruteo multicast, es decir, cómo afecta a su comportamiento el tamaño de la red sobre la cual opera y el número de routers involucrados en el protocolo.

Un grupo de protocolos multicast, entre los que se encuentran DVMRP y DM-PIM, han sido concebidos para grupos densos en áreas reducidas con uno o pocos emisores por grupo. De acuerdo a estas características, se basan en el paradigma source-based trees para la construcción de los árboles de distribución (un árbol de distribución por cada emisor del grupo) y en un esquema de poda explícita del árbol de distribución por parte de aquellos routers que no estén interesados en pertenecer a un grupo.

Estas características afectan la escalabilidad del protocolo en dos aspectos.

El primero de ellos se refiere a los recursos consumidos en los routers, ya que el tamaño de las tablas de ruteo aumenta con el orden del producto de la cantidad de grupos por la cantidad de emisores de cada grupo (O (|S| X |G|). Como contrapartida se ofrece a las aplicaciones un árbol de costo mínimo entre cada origen y cada destino, lo que en ciertos casos es de importancia (por ejemplo, aplicaciones que requieren demoras mínimas).

El segundo aspecto se refiere al uso ineficiente de los recursos de la red, ya que en el caso de grupos dispersos que abarquen áreas considerables de la Internet, el esquema de poda explícita produce una inundación periódica de la red con paquetes multicast que no serán utilizados.

Otro grupo de protocolos ha sido concebido para adaptarse a grupos dispersos con un número considerable (o la totalidad) de emisores por grupo; entre ellos se encuentran CBT versión 2 y SM-PIM.

En particular, CBT versión 2, ha sido concebido en base al paradigma shared trees, que implica la creación de un único árbol de distribución por grupo, reduciendo de esta manera los recursos necesarios en los routers a un orden relacionado con la cantidad de grupos (O(|G|). Debe considerarse el posible efecto negativo de este paradigma, que consiste en el uso de árboles no óptimos para cada par fuente-destino, lo que resulta en mayores demoras.

CBT utiliza además un mecanismo de join explícito que evita que la información multicast sea diseminada periódicamente en la totalidad de la red y que los routers deban almacenar información respecto de interfaces podadas (prune), características que permiten un uso eficiente de los recursos de la red en el caso de grupos dispersos que cubren áreas considerables de la Internet.

A diferencia de SM-PIM, CBT versión 2 no provee la capacidad de conmutar del árbol compartido a árboles específicos para cada fuente, lo que lo hace más simple. Por otro lado, presenta la característica de considerar, a efectos de la distribución de los datos, al árbol compartido como bidireccional, lo que evita que los paquetes deban ser enviados al core para luego ser inyectados en el árbol.

2. Descripción de CBT versión 2

Como se mencionó antes, CBT construye un árbol de distribución por grupo centrado en un nodo designado como core. Un nodo core puede serlo para uno o

varios grupos, no siendo necesario que pertenezca a dichos grupos (concepto de centro o core administrativo y no de distribución).

La elección de un core óptimo para un grupo y la distribución acerca de su ubicación a los routers interesados escapa a la definición de CBT.

Para el primer problema existen varias heurísticas propuestas para abordar problemas tales como número variable de emisores y conjunto variable de miembros del grupo, ya que los árboles de Steiner representan una solución para casos con un único emisor y un conjunto fijo de miembros del grupo.

La distribución de la información respecto a la ubicación de los cores, puede realizarse manualmente en cada router o a través de mecanismos de bootstrapping, que están siendo definidos actualmente en el contexto de SM-PIM [9].

Un router que desee incorporarse a un grupo (como consecuencia de haber recibido un requerimiento vía IGMP en su red local o de una aplicación local), iniciará un mecanismo de join explícito. Este consiste en el envío de un JOIN-REQUEST al próximo router con capacidad CBT camino al core, y en la creación de información transitoria respecto del subárbol de distribución que se intenta formar. El router que recibe el JOIN-REQUEST se comporta de igual manera que el anterior en caso de no pertenecer aún al grupo, mientras que si ya pertenece (estado on-tree) envía un JOIN-ACK al router inmediato anterior en la secuencia de requerimientos. Un router que recibe un JOIN-ACK, transforma la información transitoria creada en permanente, y pasa a integrar el árbol de distribución del grupo. La infomación almacenada de esta manera en la tabla de ruteo multicast, es de la forma <grupo, interfaz upstream, interfaz downstream>, donde “upstream” significa la interfaz a través de la cual se llega al core, y “downstream” la interfaz a través de la cual se accede a los nodos del árbol que se encuentran más alejados del core que el nodo en cuestión. Debe tenerse en cuenta que la diferenciación de la dirección, hacia el core o hacia las hojas del árbol, solo es utilizada por CBTv2 para el mantenimiento del árbol y no para la distribución de los paquetes multicast que arriban al router.

Un router que desee enviar información a un grupo multicast, puede o no pertenecer al grupo. En caso de pertenecer, enviará el paquete multicast por todas las interfaces asociadas al grupo. En el caso en que no pertenezca, deberá encapsular el paquete multicast y enviarlo a la dirección unicast del core. Este nodo desencapsulará el paquete y lo inyectará en el árbol de distribución. Esta manera de operar puede significar una carga considerable en los vínculos de transmisión cercanos al core.

CBT incluye mecanismos para mantenimiento del árbol de distribución. Un router en estado on-tree monitorea periódicamente el vínculo con su parent (ECHO-REQUEST); en caso de detectar alguna falla (ausencia del ECHO-REPLY correspondiente) se inicia un procedimiento de recuperación que consiste en la desconexión del nodo del árbol de distribución (eliminación de entradas en la tabla de ruteo multicast), aviso al parent (QUIT-NOTIFICATION) e indicación a todos los nodos child del grupo para que inicien el mismo procedimiento de desconexón (FLUSH-TREE). Debe notarse que el subárbol afectado es eliminado totalmente, es decir, no es posible mantener un subárbol conectado e intentar reconectarse al resto del árbol de distribución. Esta característica impide que se produzcan ciclos.

En redes multiacceso, no todos los routers pueden tener la misma visión del ruteo unicast, y por lo tanto, del mejor camino al core de un grupo (varias LIS en una red NBMA). Si esto ocurriera, podrían formarse ciclos en el árbol de distribución. Para evitar esta situación, se prevé un mecanismo por el cual se selecciona un router como “designated router” (para la totalidad de los grupos) en la red multiacceso. Este router será el responsable de efectuar las conexiones al árbol de distribución.

3. Medioambiente de simulación

El medioambiente utilizado para implementar la simulación fue Ns – Network Simulator – versión 2.0, un simulador orientado a objetos basado en eventos discretos cuyo objetivo es la simulación de redes de comunicación. Ns se encuentra actualmente en desarrollo en el marco del proyecto VINT (Virtual InterNetwork Testbed) (USC/ISI/, UC Berkeley y Xerox Parc) [http://netweb.usc.edu/vint].

Ns se encuentra implementado en c++ y en Otcl [10], extensión de Tcl orientada a objetos. Provee un soporte para la simulación compuesto de objetos implementados en c++ (objetos compilados) y objetos implementados en Otcl (objetos interpretados). Una característica importante es que pueden ampliarse los objetos compilados creando dinámicamente nuevos métodos en Otcl. Entre los objetos que conforman dicho soporte puede citarse vínculos de transmisión de diferentes tipos, nodos (routers), agentes de transmisión, funciones básicas para implementación de protocolos multicast, ruteo unicast, distribuciones y generadores de tráfico en la red, caída y recuperación de vínculos en forma dinámica, etc. Una característica de Ns es permitir la interacción de módulos desarrollados por diferentes grupos; en el caso del presente trabajo, se interactuó con el ruteo unicast (distance vector) para obtener los caminos hacia el core durante la construcción del árbol de distribución multicast en cada uno de los nodos. Esta característica posibilita también la comparación de diferentes protocolos desarrollados por distintos grupos en el mismo medioambiente de simulación.

4. Características de la implementación

La implementación del protocolo comprende todas sus características excepto el mecanismo para la elección del "Designated Router"; dicha configuración se realiza explícitamente desde el script de simulación del usuario. Se agregan facilidades de configuración de cores y de monitoreo de las acciones del CBT, que se detallan más adelante.

Un aspecto al que se dió importancia fué el de la portabilidad del código. Por ello se decidió implementar la totalidad del protocolo en forma de objetos interpretados, en Otcl, sin utilizar c++, característica que permite incorporar este código al simulador sin necesidad de recompilación. Debido a su total implementación en Otcl, debió adaptarse una parte del código encargado de la interacción entre las entradas de tablas de ruteo multicast y la función de reenvío en el nodo, debido a que esta última responde al paradigma source based trees.

El agente CBT fue definido formalmente como una máquina de estados finitos a partir de la descripción en lenguaje natural realizada en [6]. (ver Apéndice I).

La función del objeto CBT (derivado de la clase McastProtocol provista por Ns) consiste en crear, mantener y eliminar las entradas en la tabla de ruteo multicast a partir de la interacción con entidades pares en otros nodos, aplicaciones multicast locales y la entidad IGMP local.

Los eventos a los cuales responde el agente se dividen en eventos externos al nodo, consecuencia de PDUs generadas por entidades pares en otros nodos, eventos producidos en el nodo a nivel superior (join group, leave group) y eventos internos, producidos por los timers asociados al objeto CBT.

Los estados del agente CBT en un nodo, respecto de un grupo multicast, son los siguientes: estado on-tree, cuando existe una entrada permanente para el grupo (es decir, el nodo integra efectivamente el árbol de distribución); estado transitorio, cuando se ha recibido un JOIN-REQUEST de otro nodo o un requerimiento de integrar el grupo por parte de una aplicación local iniciándose el procedimiento para integrar el árbol, y estado sin entrada, cuando no se ha recibido aún ningún requerimiento de integrar el árbol correspondiente al grupo.

Las acciones externas realizadas por el objeto CBT consisten en el envío de PDUs a sus entidades pares, a través de invocaciones a métodos de su agente de emisión/recepción asociado.

Las acciones internas se corresponden con métodos de objetos relacionados con CBT, que representan los elementos manipulados por el protocolo (tabla de entradas transitorias multicast,

tabla de entrada permanente, interfaces, timers).

Además de la clase CBT, fueron implementadas clases para completar la funcionalidad del protocolo: agentes de transmisión/recepción, agentes de encapsulación y desencapsulación, interfaces, timers, tablas de ruteo permanentes y transitorias.

4.1 Agentes de transmisión/recepción

Son objetos derivados de la clase Agent/Message provista por Ns y su función es el envío de PDUs CBT como consecuencia de requerimientos del objeto CBT, y la recepción y decodificación parcial de PDUs recibidas por el nodo. En base al tipo de PDU recibida, el agente de transmisión/recepción invoca al método correspondiente del objeto CBT.

Tienen a cargo determinar la forma de envío de las PDUs CBT, de acuerdo con la especificación del protocolo: en casos de vínculos con capacidad multicast, utilizan la dirección multicast que indica el grupo correspondiente a los routers CBT, con ttl igual a 1, y en caso contrario la dirección unicast del agente de transmisión/recepción asociado a la entidad CBT en el nodo adyacente (o al "próximo" nodo, si se trata de routers separados por subredes sin capacidad multicast).

4.2 Agentes de encapsulación y desencapsulación

Tanto en los vínculos que no ofrecen capacidad multicast como en los casos de envío a un grupo (a su core[1]) por parte de un nodo no miembro del grupo, es necesario realizar una encapsulación del paquete multicast en el nodo que lo envía (transmisión unicast) y la respectiva desencapsulación en el nodo que lo recibe para entregarlo a los destinatarios (aplicaciones locales o elementos de reenvío). Estas funciones están a cargo de agentes de encapsulación y desencapsulación respectivamente.

Existe un agente de desencapsulación por nodo, creado al iniciar CBT. Su dirección unicast es accedida por agentes de encapsulación remotos a través de mecanismos provistos por Ns. Cuando el agente recibe un datagram (unicast), desencapsula el datagram multicast contenido en él y lo inyecta en el nodo en su forma original (datagram multicast), con sus direcciones de origen (unicast) y de destino (multicast) correspondientes[2]. De esta manera, los componentes del nodo perciben la llegada de un datagram multicast y actúan de la manera habitual.

Los agentes de encapsulación pueden residir en un nodo sin capacidad multicast (por ejemplo, en el caso de envío a un grupo por parte de un nodo no miembro) o formar parte de los elementos CBT (en los casos de routers con capacidad CBT e interfaces sin capacidad multicast). En el primer caso, dichos agentes podrían ser creados al inicializar la aplicación correspondiente o en el momento en que sea necesario el envío multicast. En el segundo caso, los agentes serán creados dinámicamente por los elementos de reenvío del nodo al recibir un datagram multicast para una interfaz unicast. En ese momento, los agentes serán asociados a las interfaces correspondientes de acuerdo a las entradas existentes en la tabla de ruteo multicast.

4.3 Interfaces

Los objetos de la clase Interfaz tienen por función independizar al protocolo de las interfaces provistas por Ns, y lograr de esta manera la operación del protocolo sobre vínculos con y sin capacidad multicast en la misma simulación[3]. De esta manera, es posible la

[1] La especificación de CBT versión 2 determina que en caso de que un router que no sea miembro de un grupo desee enviar información a dicho grupo, debe encapsular el datagram multicat en uno unicast dirigido al core del grupo.

[2] Además de las direcciones de origen y destino, debe conservarse la interfaz de arribo del datagram, ya que los elementos de reenvío del nodo se basan también en ella (grupo, origen, interfaz de arribo).

[3] Ns permite dotar a la totalidad de los nodos con capacidad multicast, pero no es posible en

operación del protocolo en vínculos punto a punto con y sin capacidad multicast, en y vínculos multiacceso con (p. Ej. Ethernet) o sin capacidad multicast (redes NBMA).

La interacción del protocolo con los objetos que representan las interfaces adquiere importancia respecto a la portabilidad del código, ya que actualmente se está planificando para la próxima versión de Ns el rediseño de la codificación interna del soporte multicast provisto por el simulador, que afectará el código relacionado con las interfaces.

4.4 Timers

Los timers son elementos indispensables para la operación de un protocolo. Ns provee una clase implementada en c++. Sin embargo, se optó por la creación de una nueva clase (CBTTimer) implementada totalmente en Otcl, por razones de portabilidad, ya que se decidió, para simplificar el código del protocolo, que los timers tuvieran la capacidad del manejo de reintentos (lo que implica la modificación del código c++).

La clase base implementada, a partir de la cual se derivan los timers específicos, provee la posibilidad de especificar un tiempo luego del cual se producirá el timeout conjuntamente con un procedimiento asociado a ejecutar, y un número de reintentos al cabo de los cuales la instancia del objeto será eliminada ejecutándose a su vez un procedimiento de cancelación también especificado por el protocolo.

4.5 Tablas de ruteo multicast

Para simplificar el código CBT, se implementaron objetos de las clases MFC_Table y Transient_Table, que representan respectivamente la información permanente y transitoria respecto al árbol de distribución que posee el nodo para los diferentes grupos.

Estos objetos están compuestos de entradas, cada una de ellas conteniendo información respecto del grupo, interfaces child e interfaz parent hacia el core. Las operaciones realizadas sobre estas entradas (add_entry, add_child, etc.) permiten mantener actualizadas las tablas.

En algunos casos, producen la creación de timers asociados (por ejemplo, la creación de una entrada para un nuevo grupo, implica la creación de un timer para su eliminación para el caso de que el parent no responda al ECHO_REQUEST).

De acuerdo con los requerimientos locales (join-group) o de otros nodos (JOIN-REQUEST) para integrar el árbol de distribución de los distintos grupos, se crean las entradas en la tabla transitoria. Al recibirse el JOIN-ACK correspondiente a un conjunto de entradas, se transfiere la información de la tabla transitoria a la permanente, pasando el nodo a estado on-tree.

Por ultimo, en base al contenido de la tabla permanente (MFC_Table), son creados los elementos de reenvío (forwarding) en el nodo, al producirse el correspondiente handle-cache-miss cuando se recibe un datagram para el grupo (grupo, origen, interfaz de entrada).

4.6 Soporte para comprobación de la actividad del protocolo

Tanto en la etapa de prueba del código implementado como cuando se desea monitorear la actividad del protocolo en los diferentes nodos, es importante contar con soporte para registrar las acciones de las entidades que conforman el protocolo.

En la presente implementación se proveen facilidades que permiten producir mensajes por la salida standard, cuando se producen ciertos eventos. Los eventos que producirán mensajes pueden controlarse desde el script de la simulación, en forma dinámica desde las aplicaciones (agentes) o desde el protocolo e involucran: emisión de mensajes por recepción y/o envío de determinadas PDUs, por realización de operaciones sobre las tablas de ruteo, por operaciones y eventos relacionados con los timers y por PDUs recibidas y descartadas.

Estas alternativas se discriminan por cada objeto CBT (por ejemplo, podría activarse determinadas características sólo en los cores de ciertos grupos o en uno o varios nodos en particular).

---

el estado actual definir subconjuntos de nodos con capacidad multicast y el resto sin ella.

4.7 Soporte para configuración

Se implementó un soporte para configurar aspectos relativos al protocolo y al simulador. Los primeros se refieren a la configuración de cores para los diferentes grupos y designated routers para redes de acceso múltiple. Respecto del simulador, es posible configurar algunos de los vínculos sólo con capacidad unicast.

La configuración de cores se realiza a través de métodos agregados a la clase Simulator y métodos de la clase CBT. El simulador almacenará la información respectiva a grupos y nodos cores de manera tal que los diferentes objetos CBT puedan acceder a ella (get_core). Debe notarse que esta característica corresponde con la opción de configuración manual de cores propuesta en la especificación del protocolo.

En vinculos de acceso múltiple, CBT requiere que se seleccione un nodo como designated router (DR). En la presente implementación, el DR se selecciona manualmente para cada nodo de la subred multiacceso (set_des_router).

Como fue mencionado antes, Ns ofrece la opción de capacidad multicast pero en la totalidad de los vínculos de la topología simulada. A efectos de posibilitar el funcionamiento de CBT en caso de vínculos que no soportan multicast, se provee la opción de indicar al objeto CBT respectivo que algunos de los vínculos no pueden ser utilizados con direcciones multicast (set_unicast), logrando que el protocolo trate la correspondiente interfaz como unicast. De esta manera, es posible simular redes NBMA y vínculos punto a punto sin capacidad multicast.

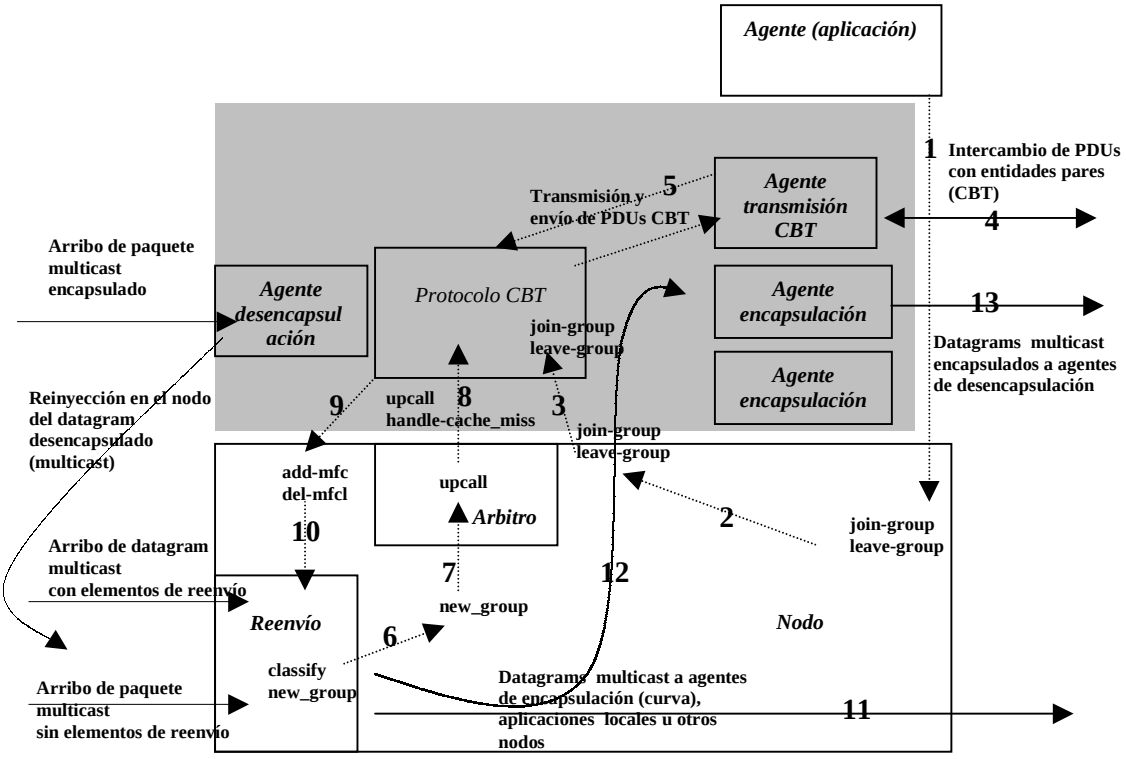

Fig. 1. Esquema simplificado de las interacciones entre el objeto que constituye la entidad CBT y otros elementos en el nodo. Los objetos en la parte sombreada corresponden a la implementación. Las líneas llenas representan pasaje de datos (datagrams); las punteadas interacciones sin paso de datos.

4.8 Interacción entre los componentes

En la figura 1 se muestran algunas de las componentes CBT implementadas (parte sombreada), la interacción entre ellas y con los demás elementos del nodo. Los intercambios de datos se muestran con línea llena y las demás interacciones con línea punteada.

Se distingue por un lado las acciones que producen la actualización de las tablas de rutas (Multicast Forwarding Cache) que maneja el nodo. Estas acciones pueden originarse en requerimientos de aplicaciones locales (o IGMP) que realizan un join-group o leave-group (1, 2, 3) o de PDUs recibidas (a través del agente de transmisión asociado) – JOIN-REQUEST JOIN-ACK – (4, 5).

En base al contenido de la MFC, y disparado por la llegada de datos, se produce (cuando es necesario) la creación de los elementos de reenvío que posibilitarán que el paquete multicast sea entregado a las aplicaciones locales, a los vínculos hacia otros nodos con capacidad multicast, o luego de la creación del agente de desencapsulación correspondiente, a vínculos a nodos sin capacidad multicast o a túneles.

El arribo de un datagram multicast para el cual ya fueron creados los elementos de reenvío, es entregado por el nodo a su(s) correspondiente(s) destinos(s) (11, 12, 13): vínculos a otros nodos, agentes de encapsulación y aplicaciones multicast locales (no mostrado en la figura).

El arribo de un datagram multicast sin elementos de reenvío creados y con entrada en la tabla MFC desencadena las interacciones que se muestran en el gráfico 6, 7, 8, 9 10), las que provocan la creación de los elementos de reenvío y, en los casos en que el próximo destino del paquete sea accedido através de un vínculo no multicast, la creación del agente de encapsulación correspondiente. Inmediatamente después, se provocan las acciones descriptas en el párrafo anterior para efectuar el envío del paquete.

El arribo de un datagram encapsulado (unicast) origina que el agente de desencapsulación lo desencapsule y lo reinyecte en el nodo, como si fuera un paquete multicast. Luego se tomarán las acciones que correspondan según lo descripto arriba.

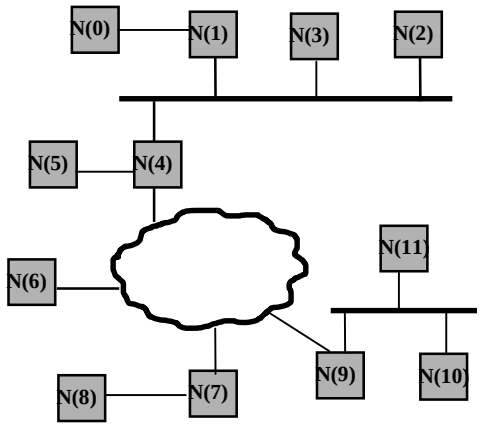

Fig 2. Topología compuesta de dos vínculos multiacceso multicast, uno multiacceso no multicast y tres punto a punto multicast.

4.9 Ejemplo

En la figura 2, se muestra un ejemplo de una topología simple y el correspondiente script de simulación. Se definen doce nodos (desde el punto de vista de Ns todos tienen la funcionalidad de routers), conectados por una topología consistente en tres vínculos punto a punto y tres vínculos de acceso múltiple.

El código CBT implementado, es accedido a través de la lectura del archivo cbtmain.tcl.

Las características propias de la implementación son la instanciación de los agentes CBT, la configuración de cores (en este caso nodo n(0) es core para el grupo 65490) y la definición de los nodos que actúan como DR en cada vínculo de acceso múltiple (los parámetros consisten de la identificación de un nodo adyacente que determina el vínculo multiacceso y la identificación del nodo DR).

El mecanismo de configuración de cores tiene alcance global (para todos los nodos de la simulación) mientras que el DR debe especificarse para cada nodo involucrado. Esta desición se basa en el alcance más reducido en este último caso (sólo los nodos conectados al vínculo de acceso múltiple).

Se muestra en este caso la habilitación, en todos los nodos de la

```
set ns [new Simulator]
Simulator set EnableMcast_ 1
Simulator set NumberInterfaces_ 1
source cbtmain.tcl
#Instanciacion de nodos
for {set i 0} {i <=11} {incr i 1} {set n($i) [$ns node]}
#topologia
$ns duplex-link $n(0) $n(1) 1.5Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 1.5Mb 10ms DropTail
$ns duplex-link $n(7) $n(8) 1.5Mb 10ms DropTail
$ns multi-link-of-interfaces $n(1) $n(2) $n(3) $n(4)
10Mb 1ms
$ns multi-link-of-interfaces $n(4) $n(6) $n(7) $n(9)
1.5Mb 10ms
$ns multi-link-of-interfaces $n(9) $n(10) $n(11) 10Mb
1ms
#Instanciacion de agentes CBT en los nodos
for {set i 0} {i <=11} {incr i 1} {set cbt($i) [new CBT
$ns $n($i)]}
#Definicion de core
$cbt(0) set-core 65490
#Nodos 2, 7 y 9 Drs
$cbt(2) set_des_router 1 2
$cbt(1) set_des_router 2 2
$cbt(3) set_des_router 2 2
$cbt(4) set_des_router 2 2
$cbt(7) set_des_router 6 7
$cbt(6) set_des_router 7 7
$cbt(9) set_des_router 7 7
$cbt(4) set_des_router 7 7
$cbt(9) set_des_router 10 9
$cbt(10) set_des_router 9 9
$cbt(11) set_des_router 9 9
#Configuracion de vinculos unicast
$cbt(4) set_unicast 6
$cbt(6) set_unicast 4
$cbt(7) set_unicast 4
$cbt(9) set_unicast 4
#Arranque de los agentes CBT en tiempo 1.0
for {set i 0} {i <=11} {incr i 1} {$ns at 1.0 “cbt($i)
start”}
#Debugging
for {set i 0} {i<=11} {incr i 1} {
    $cbt($i) enable_pdus JOIN_REQ
    $cbt($i) enable_pdus JOIN_ACK
    $cbt($i) enable_tables
}
#Definicion de agentes multicast
set e(7) [new Agent/Message/Emisor 0.01]
$e(7) set dst_ 65490
$ns attach-agent $n(7) $e(7)
set r(9) [new Agent/Message/Receptor]
$ns attach-agent $n(7) $r(7)
set r(5) [new Agent/Message/Receptor]
$ns attach-agent $n(5) $r(5)
#Join de grupos, arranque y parada de emisor
$ns at 2.5 “$r(9) join-group 65490”
$ns at 2.8 “$r(5) join-group 65490”
$ns at 4.0 “$e(7) start”
$ns at 8.0 “$e(7) stop”
#Visualizacion del arbol
$ns at 8.0 “$ns map-tree”
$ns run
```

Fig. 3. Script de simulación definiendo y utilizando la topología de la figura 2.

emisión de mensajes por recepción y emisión de JOIN-REQUEST y JOIN-ACK y por modificación de tablas, transitorias y permanentes.

Se definen luego agentes multicast de emisión y recepción en diferentes nodos, y se solicita, en el tiempo de simulación 5.0, un mapa del árbol de distribución producido (map-tree).

5. Análisis de tiempos de recuperación y costos de transmisión

El análisis realizado abarcó un aspecto simple del protocolo y constituyó una primera etapa en la que se testeó la implementación. En futuras pruebas se analizarán otros aspectos del protocolo, relacionándolo con otros tales como SM-PIM.

El objeto de las pruebas fue analizar el comportamiento del protocolo en casos de falla de vínculos. Se determinó las demoras y costos (PDUs de control intercambiadas) que demandó la reconstrucción del árbol de distribución como consecuencia de fallas en los vínculos.

Los experimentos realizados se basaron en una red con una topología similar a Arpanet (figura 4), consistente de 47 nodos y 69 vínculos. Cada vínculo se supuso de 1.5 Mbps de capacidad (T1) con una demora de propagación de 1 ms.

Para realizar las pruebas se seleccionó un conjunto de diez nodos pertenecientes al grupo multicast de manera que los grupos constituidos al considerar a cada uno de los nodos de la red como core resulten, en promedio, con un grado medio de densidad[4] .

[4] Se considera el grado de densidad de un grupo como el cociente entre la totalidad de los vínculos de la red y la cantidad de vínculos que conforman el árbol de distribución.

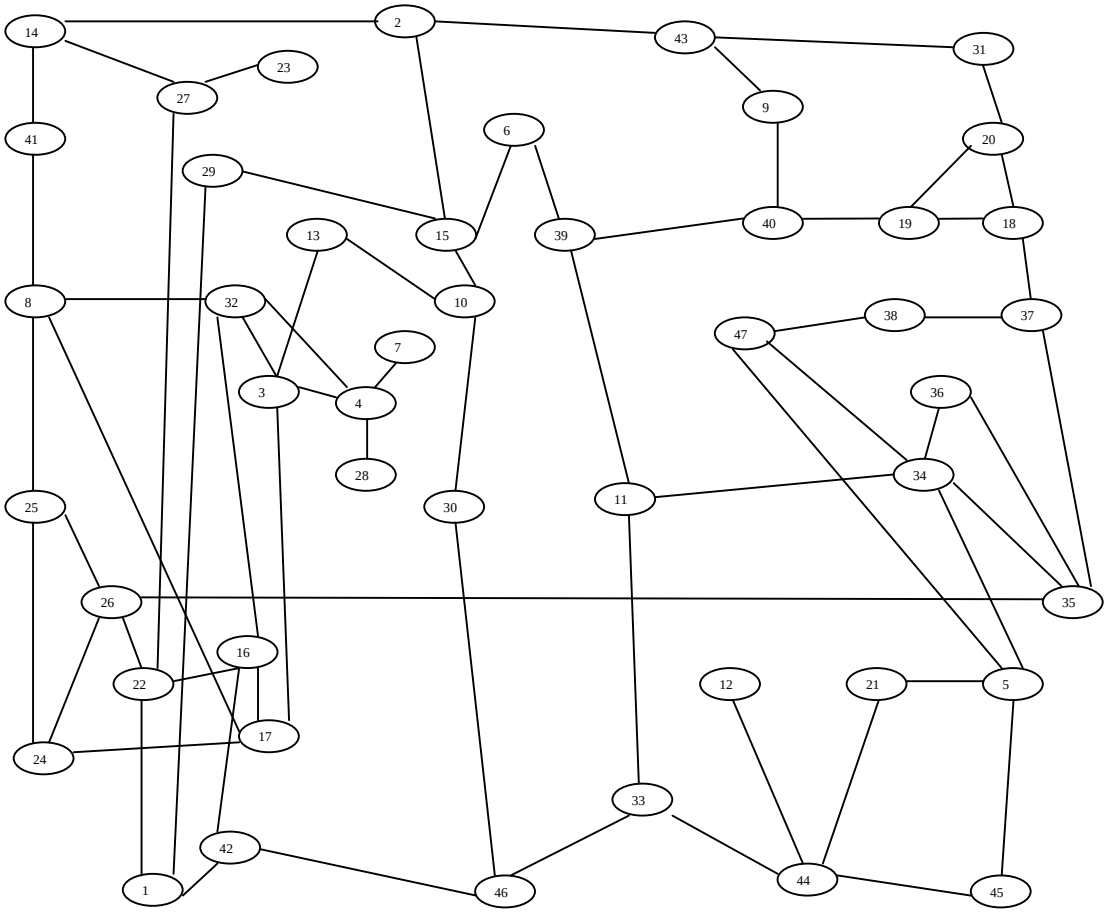

Fig 4. Topología utilizada para el análisis de costos y demoras

Se supuso que la totalidad de los routers tiene capacidad CBT y los vínculos son punto a punto con capacidad multicast.

Para simular la caída de los vínculos se utilizó la capacidad de caída y recuperación de vínculos provista por Ns, en este caso especificada manualmente desde el script de simulación a tiempos prefijados. Para la recuperación de los datos necesarios, se utilizó la capacidad de debugging implementada en los agentes CBT.

Las pruebas consistieron en provocar la caída de cada uno de los vínculos que integran el árbol de distribución para cada core, y medir el tiempo de recuperación y el costo del armado del nuevo subárbol.

Al comienzo de la simulación, se activaron los agentes CBT en cada nodo de la red, y luego los procesos multicast específicos (agentes emisores y receptores desarrollados) en cada uno de los diez nodos elegidos, lo que provocó el agregado (join) de cada uno de ellos al grupo.

Estas acciones provocaron el armado de un árbol de distribución basado en las tablas de ruteo unicast. Los nodos y vínculos que lo integran, fueron determinados a través de las facilidades de debugging implementadas en el agente CBT (método map-tree agregado a la clase Simulator).

Luego de un determinado tiempo, se produjo la falla en cada uno de los vínculos (uno en cada corrida de la simulación) componentes del árbol de distribución y se analizó el costo demandado y la demora producida para rearmar otro árbol.

Los resultados obtenidos (demoras en la construcción de un nuevo subárbol de

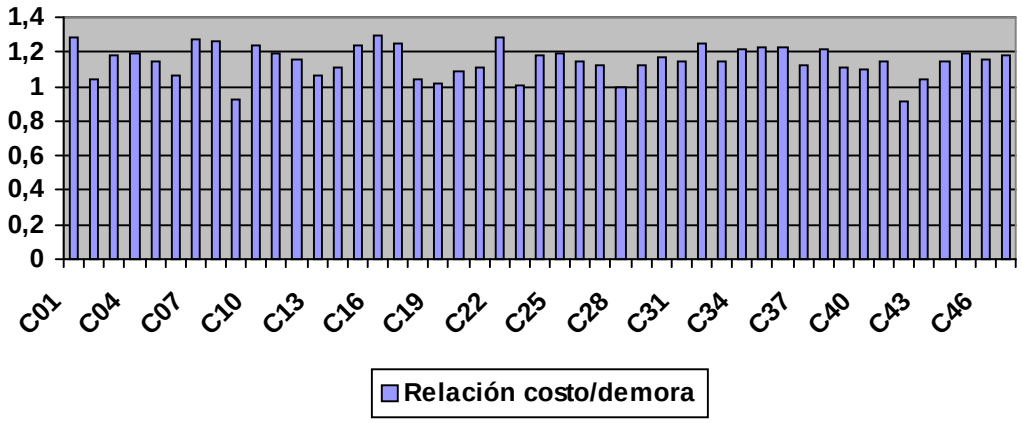

Fig 5. Relación entre costos y demoras promedio para cada nodo actuando como core

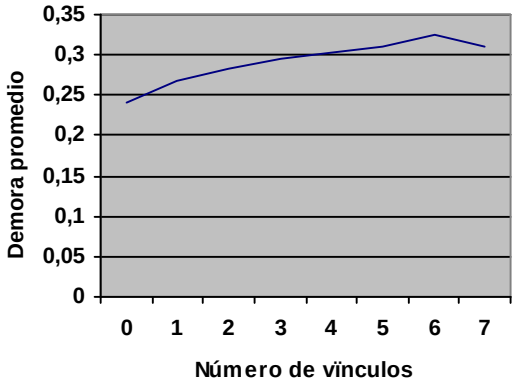

Fig. 6. Demora promedio para la reconstrucción del árbol en función del número de vínculos del subárbol aislado

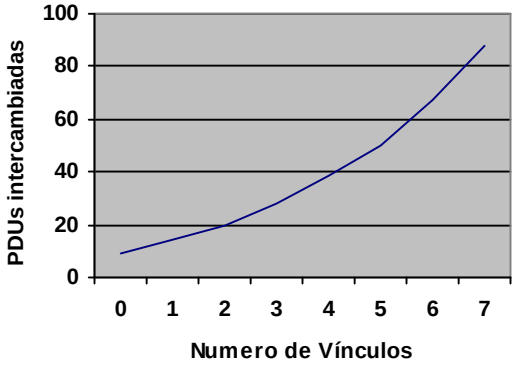

Fig. 7. Cantidad de PDUs intercambiadas para la reconstrucción del árbol en función del número de vínculos del subárbol aislado

distribución y costos de transmisión), se registraron por cada vínculo en falla, y luego se los agrupó por nodo core y de acuerdo al diámetro del árbol de distribución que quedó aislado en cada caso.

Con respecto a la reconstrucción del subárbol de distribución, se supuso que los nodos, cuando comienzan las acciones de recuperación, disponen de información actualizada de ruteo unicast y que éste converge, lo cual impide la producción de ciclos transitorios durante el armado del árbol.

Esto implica que a las demoras obtenidas debería agregarsele un tiempo mínimo fijo, consecuencia del tiempo de convergencia del algoritmo de ruteo unicast, tiempo a partir del cual deberían comenzar las acciones para recuperación del subárbol de distribución.

Se analizaron dos aspectos, el primero referido a la comparación de cada uno de los posibles cores, y el segundo referido a la escalabilidad de CBT en cuanto a la distancia máxima entre los nodos que conforman el árbol de distribución.

De acuerdo con los resultados obtenidos, existe una alta correlación entre los costos y las demoras producidos en cada core (Figura 5), es decir, aquellos cores que producen costos más altos producen a su vez demoras mayores. Por otra parte, la elección del core demostró tener mayor peso respecto de los costos producidos, ya que la desviación standard de los costos resultó ser un 10,87% de su promedio,

mientras que la de las demoras fue de un 6,70%.

El segundo aspecto consistió en agrupar los resultados obtenidos por longitud (en vínculos) del subárbol que quedó aislado por la falla. Este aspecto permite determinar la escalabilidad del protocolo respecto al diámetro de la red.

Los resultados indican que las demoras en la reconstrucción del árbol aumentan lentamente (Figura 7), lo cual indica una buena capacidad de escalabilidad del protocolo al aumentar el diámetro de la red.

Los costos, en cambio (Figura 6) aumentan más rápidamente, pero debe tenerse en cuenta que éste escala con un orden O(|G|), y que además se ha considerado el costo total, que deberá distribuirse entre los vínculos afectados.

6. Conclusiones y continuación del trabajo

El trabajo realizado constituye una primera etapa que demostró la factibilidad y eficiencia de las simulaciones multicast en ambiente Ns, lográndose una implementación de CBTv2 a nivel detallado que insumió tiempos razonables de simulación.

El medioambiente de simulación resultó ser potente, fácil de utilizar y portable, permitiendo realizar el desarrollo completamente en Otcl a costa de sacrificar en algunos casos la simplicidad del código y la eficiencia de la implementación.

Si bien fueron implementadas clases que logran abstraer la operación del protocolo tanto del medioambiente de simulación como de los objetos con los que debe interactuarse (timers, tablas de ruteo multicast) será necesario actualizar la interfaz con los elementos del simulador debido a los cambios previstos en las próximas versiones de Ns y aumentar la funcionalidad de algunas clases para su adaptación a nuevas versiones del protocolo[11] (soporte de entradas (*,G), (*,Core), (S,G), soporte de prefijos, etc), lo que permitiría la simulación inter-dominio.

El análisis realizado cubrió un aspecto en particular del protocolo, restando comparar su performance con otros similares como PIM-SM.

Para obtener los datos necesarios, fueron de utilidad las facilidades de debugging implementadas

Referencias

[1] S. Deering, "Host Extensions for IP Multicasting", RFC 1112, August 1989. [2] S. Deering,C. Partridge,D. Waitzman, "Distance Vector Multicast Routing Protocol", RFC 1075, November 1988. http://www.isi.edu

[3] T. Moy, "Multicast Extensions to OSPF", RFC 1584, March 1994. http://www.isi.edu

[4] Estrin, D. et. al., "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", RFC 2117, June 1997. http://www.isi.edu

[5] A. Ballardie, "Core based Trees (CBT) Multicast Routing Architecture", RFC 2201, September 1997. http://www.isi.edu

[6] A. Ballardie, "Core Based Trees (CBT version 2) Multicast Routing. Protocol Specification", RFC 2189, September 1997. http://www.isi.edu

[7]A. Ballardie, "Core Based Trees (CBT) Multicast Routing MIB, Internet Draft, July 1997. http://www.isi.edu

[8] K. Varadhan, K. Fall, "Ns Notes and Documentation", (VINT Project), UC Berkeley, May 1998. http://www.isi.edu/~kannan/nsDoc.ps.gz

[9] D. Estrin, M. Handley, A. Helmy, P. Hang, D. Thaler, “A Dynamic Bootstrap Mechanism for Rendevouz-based Multicast Routing”, ftp://usc.edu/pub/csinfo/tech-reports/papers/97-664.ps.Z

[10] D. Wetherall, C. Lindblad, “Extending Tcl for Dynamic Object-Oriented Programming”, Proceedings of the Tcl/Tk Workshop 95, Toronto, Ontario, July 1995.

[11] A. Ballardie, B. Cain, Z. Zhang, “Core Based Trees (CBT version 3) Multicast Routing”, Internet draft, . http://www.isi.edu, (draft-ietf-idmr-cbt-spec-v3-00.txt), March 1998.

Apéndice I: Definición formal del protocolo CBT

	ON TREE (ES01)	TRANSITORIO (ES02)	SIN ENTRADA (ES03)
(EV21) Retransmit_Join_Request_Timer (T01) timeout	IGNORAR	P34	IGNORAR
(EV22) Delete_Leaf_Transient_Entry_Timer (T02) timeout	IGNORAR	P35	IGNORAR
(EV23) Delete_Transient_Entry_Timer (T03) timeout	IGNORAR	P36	IGNORAR
(EV24) Cache_Deletion_Timer (T04) timeout	P37 P38	IGNORAR	IGNORAR
(EV25) Resend_Join_Request_Timer (T05) timeout	P39	IGNORAR	IGNORAR
(EV26) Quit_Notification_Timer (T06) timeout	IGNORAR	IGNORAR	P40 P41
(EV27) Echo_Request_Timer (T07) timeout	P42	IGNORAR	IGNORAR
(EV28) Delete_Ontree_Timer (T08) timeout	P43 P44 P45	IGNORAR	IGNORAR
(EV29) Echo_Reply_Timer (T09) timeout	P46	IGNORAR	IGNORAR

Tabla 1. Estados/eventos para eventos internos (provenientes de timers) de la entidad CBT

	ON TREE (ES01)	TRANSITORIO (ES02)	SIN ENTRADA (ES03)
(EV01) Recepción JOIN_REQUEST	P01 P05 P02 P06 P03 P07 P04 P08	P09 P10 P11 P12	P13 P14 P15 P16
(EV02) Recepción JOIN_ACK	IGNORAR	P17 P18	IGNORAR
(EV03) Recepción QUIT_NOTIFICATION	P19 P20 P21 P22	IGNORAR	IGNORAR
(EV04) Recepción FLUSH_TREE	P23 P24	IGNORAR	IGNORAR
(EV05) Recepción ECHO_REQUEST	P25 P26	IGNORAR	IGNORAR
(EV06) Recepción ECHO_REPLY	P27	IGNORAR	IGNORAR
(EV10) Recepción JOIN-GROUP	P28	P29	P30
(EV11) Recepción LEAVE-GROUP	P31 P32 P33	IGNORAR	IGNORAR

Tabla 2. Estados/eventos para eventos externos (provenientes de otros nodos o de entidades en el nodo) a la entidad CBT

Acciones Internas
IA000: Crear entrada transitoria para un grupo.
IA001: Eliminar entrada transitoria para un grupo.
IA002: Agregar child en entrada transitoria.
IA003: Eliminar child en entrada transitoria.
IA010: Crear entrada permanente (on-tree) para un grupo.
IA011: Eliminar entrada permanente (on-tree) para un grupo.
IA012: Agregar child en entrada permanente (on-tree).
IA013: Eliminar entrada child en entrada permanente (on-tree).
IA020: Activar Retransmit_Join_Request (T01)
IA021: Desactivar Retransmit_Join_Request_Timer (T01)
IA022: Reactivar Retransmit_Join_Request_Timer (T01)
IA030: Activar Delete_Leaf_Transient_Entry_Timer (T02)
IA031: Desactivar Delete_Leaf_Transient_Entry_Timer (T02)
IA032: Reactivar Delete_Leaf_Transient_Entry_Timer (T02)
IA040: Activar Delete_Transient_Entry_Timer (T03)
IA041: Desactivar Delete_Transient_Entry_Timer (T03)
IA042: Reactivar Delete_Transient_Entry_Timer (T03)
IA050: Activar Cache_Deletion_Timer (T04).
IA051: Desactivar Cache_Deletion_Timer (T04).
IA052: Reactivar Cache_Deletion_Timer (T04).
IA060: Activar Resend_Join_Request_Timer (T05).
IA061: Desactivar Resend_Join_Request_Timer (T05)
IA062: Reactivar Resend_Join_Request_Timer (T05)
IA070: Activar Quit_Notification_Timer (T06)
IA071: Desactivar Quit_Notification_Timer (T06).
IA072: Reactivar Quit_Notification_Timer (T06).
IA080: Activar Echo_Request_Timer (T07).
IA081: Desactivar Echo_Request_Timer (T07).
IA082: Reactivar Echo_Request_Timer (T07)
IA090: Activar Delete_On-tree_Entry_Timer (T08)
IA091: Desactivar Delete_On-tree_Timer (T08).
IA092: Reactivar Delete_On-tree_Timer (T08).
IA100: Activar Echo_Reply_Timer (T09).
IA101: Desactivar Echo_Reply_Timer (T09).
IA102: Reactivar Echo_Reply_Timer (T09).
IA200: Variable RTX = RTX + 1
IA201: Variable RTX = 0

Eventos externos
OE01: Enviar JOIN_ACK
OE02: Enviar JOIN_REQUEST
OE03: Enviar QUIT_NOTIFICATION
OE04: Enviar FLUSH_TREE
OE05: Enviar ECHO_REQUEST
OE06: Enviar ECHO_REPLY

Condiciones
C01: Interfaz de arribo de la PDU es parent (transitoria o permanente) para el grupo
C02: Entrada transitoria es leaf.
C03: RTX < MAX_RTX
C04: Vínculo asociado a la interfaz de arribo de la PDU es multiacceso multicast.
C05: Interfaz child es la unica del grupo.
C06: Interfaz de arribo es child para el grupo.
C07: Existe interfaz correcta a nodo upstream camino al core del grupo.

C08: Vínculo asociado a la interfaz parent del grupo es multiacceso multicast.
C09: Interfaz de arribo no es parent ni child para el grupo.
C10: Nodo es DR en el vínculo asociado a la interfaz.
C11: Vínculo asociado a interfaz de arribo no tiene capacidad multiacceso.
C12: Vínculo asociado a interfaz de arribo es multiacceso no multicast
C13: Red (interfaz parent) diferente Red (interfaz de arribo)
C14: Interfaz a nodo upstream diferente de interfaz de arribo

Predicados y acciones condición {(acciones internas), (eventos externos), (nuevo estado)}
P01: C09 & C11 {(IA012), (OE01), (ES01)}
P02: C09 & C12 & C10 {(IA012), (OE01), (ES01)}
P03: C09 & C12 & !C10 &C13 {(IA012), (OE01), (ES01)}
P04: C09 & C04 &C10 {(IA102), (OE01), (ES01)}
P05: C01 & C04 & !C10 {(IA061), , (ES01)}
P06: C06 & C11 {, (OE01), (ES01)}
P07: C06 & C12 {, (OE01), (ES01)}
P08: C06 & C04 & C10 {(IA051), (OE01), (ES01)}
P09: C09 & C11 {(IA002), , (ES02)}
P10: C09 & C12 & C10 {(IA002), , (ES02)}
P11: C09 & C12 & !C10 &C13 {(IA002), ,(ES02)}
P12: C09 & C04 &C10 {(IA002), , (ES02)}
P13: C09 & C11& C07 &C14 {(IA000, IA040, IA002, IA071), (OE02), (ES02)}
P14: C09 & C12 & !C10 & C07 & C14 {(IA000, IA040, IA002, IA071), (OE02), (ES02)}
P15: C09 & C12 & C10 &C07 & C13 {(IA000, IA040, IA002, IA071), (OE02), (ES02)}
P16: C09 & C04 & !C10 & C07 {(IA000, IA040, IA002, IA071), (OE02), (ES02)}
P17: C01 & C02 {(IA010,IA090,IA080,IA031,IA021, *C:(IA001,IA012)),(*C: OE01), (ES01)}
P18: C01 & !C02 {(IA010, IA090, IA080, IA041,*C: (IA001, IA012)), (*C:OE01),(ES01)}
P19: C04 & C05 & C06 {(IA050), ,(ES01)}
P20: !C04 & !C05 &C06 {(IA013), ,(ES01)}
P21: !C04 & C05 & C06 {(IA013, IA011, IA091, IA081, IA070, IA201), (OE03), (ES03)}
P22: C01 & C04 {(IA060), ,(ES01)}
P23: C01 & !C04 {((*C: IA013), IA011, IA081, IA091), ((*C: OE04)), (ES03)}
P24: C01 & C04 {((*C: IA013), IA011, IA081, IA091, IA021), ((*C:OE04)), (ES03)}
P25: C06 {(IA100), , (ES01)}
P26: C01 & C04 {(IA082), , (ES01)}
P27: C01 {(IA092), , (ES01)}
P28: V {(IA012), , (ES01)}
P29: V {(IA002), , (ES01)}
P30: C07 {(IA000, IA020, IA030, IA002), (OE02), (ES02)}
P31: !C05 {(IA013), , (ES01)}
P32: !C08 & C05 {(IA013, IA011, IA081, IA091, IA070), (OE03), (ES03)}
P33: C08 & C05 {(IA013, IA011, IA081, IA091, IA070, IA061), (OE03), (ES03)}
P34: V {(IA020), (OE02), (ES02)}
P35: V {(IA001, IA021) , , (ES03)}
P36: V {(IA001), , (ES03)}
P37: !C05 {(IA013), (OE04), (ES01)}
P38: C05 {(IA013, IA081, IA091, IA011, IA070), (OE04, OE04), (ES03)
P39: V { , (OE02), (ES01)}
P40: C03 {(IA200, IA070), (OE03), (ES03)}
P41: !C03 { , (OE03), (ES03)}
P42: V {(IA080), (OE05), (ES01)}
P43: !C04 {(*C:(IA013, IA081), IA011, IA070), (*C:(OE04), OE03), (ES03)}
P44: C04 & C10 {(*C:(IA013, IA081), IA011, IA070), (*C:(OE04), OE03), (ES03)}
P45: C04 & !C10 {(*C:(IA013, IA081, IA051), IA011, IA070), (*C:(OE04), OE03), (ES03)}
P46: V {, (OE06), (ES01)}