UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

E. T. S. INGENIEROS DE TELECOMUNICACIÓN

PROYECTO FINAL DE CARRERA

DISEÑO DE UN PROTOCOLO DE COMUNICACIÓN INALÁMBRICA MEDIANTE EL USO DEL ANTS

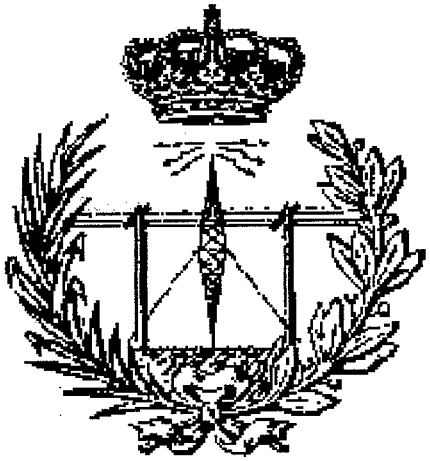
JAVIER JESÚS SÁNCHEZ MEDINA

Las Palmas de Gran Canaria, Julio de 2001

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN



BIBLIOTECA DE TELECOMUNICACIONES * UNIVERSIDAD DE LAS PALMAS DE G. C. * E.U.I.T.-E.T.S.I.T.

BIBLIOTECA UNIVERSITARIA
LAS PALMAS DE G. CANARIA
N.º Documento
N.º Copia 64182

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

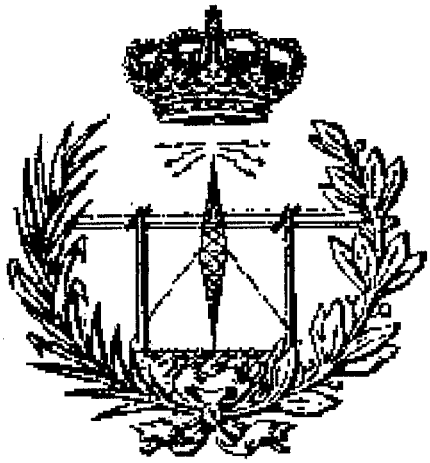
PROYECTO FIN DE CARRERA

Diseño de un protocolo de comunicación inalámbrica mediante el uso del ANTS

Autor: Javier Jesús Sánchez Medina
Tutor: Alvaro Suárez Sarmiento
Director: Alvaro Suárez Sarmiento
Fecha: Julio, 2001

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN



PROYECTO FIN DE CARRERA

Diseño de un protocolo de comunicación inalámbrica mediante el uso del ANTS

HOJA DE FIRMAS

Alumno/a

Fdo.: Javier Jesús Sánchez Medina

Tutor/a

Fdo.: Alvaro Suárez Sarmiento

Director/a

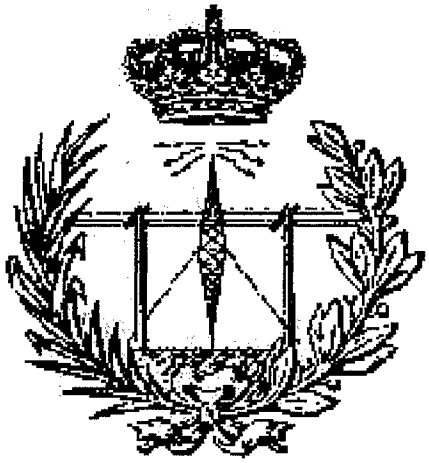
Fdo.: Alvaro Suárez Sarmiento

Fecha: Julio, 2001

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN



PROYECTO FIN DE CARRERA

Diseño de un protocolo de comunicación inalámbrica mediante el uso del ANTS

HOJA DE EVALUACIÓN

Calificación: SOBRESALIENTE (9)

Presidente

Fdo.: FRANCISCO JOSE GUERRA SANTANA

Vocal

Fdo.: Miguel A. Quintana S.

Secretario/a

Fdo.: Luis Hdez. Acosto

Tutor/a

Fdo.: Alvaro Suárez Sarmiento

Fecha: Julio, 2001

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

E.U.I.T. - E.T.S.I.T. BIBLIOTECA DE TELECOMUNICACIONES UNIVERSIDAD DE LAS PALMAS

A Yolanda

"Esto no puede ser no
más que una canción..."

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Agradecimientos

Sería muy injusto que asumiera como enteramente mío el mérito de la elaboración de este proyecto. Son muchas las personas que han contribuido a que todo el trabajo de un año haya dado su fruto al fin.

Quiero agradecerles a mis padres que me hayan enseñado desde niño que lo único que en verdad poseemos es lo que somos y como amueblamos nuestro espíritu con lo que aprendemos. De mi madre aprendí a ser luchador, y de mi padre a soñar. Sé que el mejor regalo que puedo hacerles es llegar a donde he llegado.

Gracias a mis hermanos Manolo, Rubén, Iván y Diego, con toda mi ilusión y apoyo para que ellos también luchen por sus sueños.

A mi novia Yolanda. La historia de este proyecto y la historia de lo nuestro estarán entrelazadas para siempre. Cada momento de este año de trabajo tiene hermosos recuerdos en los que Ella es siempre protagonista.

A mis compañeros-amigos de la carrera. A pesar de los duros que han sido algunos momentos, siempre tendremos el recuerdo de otros muchos momentos buenos y divertidos compartidos.

Gracias a mi tutor Álvaro por haber sabido alimentar mis ilusiones y delirios de grandeza de aspirante a científico.

Gracias a mi amiga y jefa Itzíar.

Gracias a Ernesto Robayna por su ayuda desinteresada en la cuestión inalámbrica.

Gracias a Yolanda Suárez por su apoyo y su impresora.

Gracias a Antonio Santiago del COIT por facilitarme información para la elaboración del presupuesto.

Gracias al resto de mi familia y amigos en La Palma y en Gran Canaria.

Espero que mi trabajo esté a la altura de tanto cariño y buenas intenciones en mí depositados.

- Javier

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Indice de partes del proyecto

Memoria

Capítulo 1. Introducción..... 1
Capítulo 2. Las Redes Activas..... 7
Capítulo 3. Protocolos de comunicación..... 19
Capítulo 4. Protocolos diseñados..... 29
Capítulo 5. Pruebas y resultados de los protocolos diseñados..... 93
Capítulo 6. Conclusiones y líneas futuras..... 133
Bibliografía. 137

Pliego de Condiciones

Pliego de Condiciones..... PC.1

Presupuesto

1. Introducción..... P.1
2. Descripción de cada fase..... P.1
3. Presupuesto..... P.3

Anexos

Anexo I AI.1
Anexo II AII.1
Anexo III AIII.1

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Memoria

En esta parte del proyecto se desarrolla la memoria. En primer lugar se expone una introducción a las redes actuales particularizando en las redes inalámbricas. Seguidamente se da una visión general de las Redes Activas y se explica la arquitectura del ANTS. Luego se clasifican los tipos de protocolo. Acto seguido se explican todos los protocolos diseñados, sus implementaciones y las pruebas realizadas. Por último se comentan las conclusiones.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Indice de la memoria

Capítulo 1. Introducción **1**
1.1 Estado actual de las redes de datos 2
1.2 Las redes de área local inalámbricas 4
1.3 Objetivos del proyecto 5
1.4 Estructura de la memoria 6
Capítulo 2. Las Redes Activas **7**
2.1 Introducción 8
2.2 Definición de Red Activa 8
2.3 Aproximaciones actuales 11
2.3.1 MIT 11
2.3.2 BBN 11
2.3.3 Georgia Tech 12
2.3.4 Universidad de Pennsylvania 12
2.3.5 Universidad de Arizona 13
2.3.6 Universidad de Columbia 13
2.3.7 ETH Zurich y Universidad de Washington 14
2.3.8 Universidad Carlos III 14
2.4 Arquitectura básica del ANTS 14
2.5 Programación de prototipos con el ANTS 16
Capítulo 3. Protocolos de comunicación **19**
3.1 Tipos de protocolo 20
3.2 Protocolos de Unicast 22
3.3 Protocolos de Multicast 22
3.4 Protocolos de Broadcast 24
3.5 Justificación del uso de Protocolos de Multicast 24
Capítulo 4. Protocolos diseñados **29**
4.1 Protocolos Unicast 30
4.1.1 Protocolo de Producto Distribuido de Matriz por Matriz 30
4.1.2 Protocolo de *PingBroadcast* 40
4.2 Protocolos Multicast 50
4.2.1 Protocolo de multicast en tiempo real 50
4.2.2 Protocolo de Multicast para la Edición Cooperativa, versión 1 67
4.2.3 Protocolo de Multicast para la Edición Cooperativa, versión 2 85
Capítulo 5. Pruebas y resultados de los protocolos diseñados **93**
5.1 Introducción 94
5.2 Protocolo de Matriz por Matriz 95
5.3 Protocolo de *PingBroadcast* v1 98
5.4 Protocolo de *PingBroadcast* v2 105
5.5 Protocolo de Multicast en Tiempo Real v1 107
5.6 Protocolo de Multicast en Tiempo Real v2 116
5.7 Protocolo de Multicast para la Edición Cooperativa v2 125
5.8 Resumen de todos los protocolos explicados 130
Capítulo 6. Conclusiones y líneas futuras **133**
6.1 Conclusiones 134
6.2 Posibles ampliaciones 136
Bibliografía. **137**

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Capítulo 1. Introducción

En este capítulo se expone una introducción al estado actual de las redes de datos, explicando el problema que supone la estandarización de los nuevos servicios y se enuncian dos nuevos paradigmas de programación de redes que pueden solucionar este problema. Seguidamente se habla de las Redes de Area Local Inalámbricas, y se explica el importante esfuerzo investigador que en tecnologías inalámbricas y móviles se esta dando. A continuación se comentan los objetivos de este Proyecto de Fin de Carrera y se concluye con la explicación de la estructura de esta memoria.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

1.1 Estado actual de las redes de datos

Actualmente asistimos a una evolución espectacular en las redes de datos. Podemos afirmar que nos encontramos en el esplendor de una nueva revolución industrial, una revolución de las comunicaciones.

La telefonía celular, Internet, las plataformas digitales vía satélite, la televisión por cable ... son ejemplos de redes que se encuentran en este momento en un proceso febril de evolución.

Internet es un buen ejemplo. Antes de los años 90 Internet era una tecnología al alcance solamente de universidades y centros de investigación americanos. Hoy en día se ha extendido enormemente, y se encuentra al alcance de casi cualquier persona en todo el mundo, duplicándose la tasa de tráfico cada cuatro meses [1] y apareciendo servicios nuevos y atractivos constantemente. Comercio electrónico, banca on-line, servicios multimedia, agencias de viajes, televisión y radio son solamente algunos ejemplos de las nuevas aplicaciones que han aparecido en la red. En [1] se enumeran las aplicaciones que tendrá en un futuro inmediato la que se ha dado a llamar *Next Generation Internet* (*Internet de la Próxima Generación*). *Voz sobre IP* (*VoIP*), que sustituirá a la telefonía tradicional; un uso más ampliamente extendido de la videoconferencia en el mundo laboral; sustitución de edificios físicos por sitios virtuales para las escuelas, universidades y demás centros de estudio; incluso hay noticias de un proyecto de la *NASA* (*Administración Aerospacial Nacional*) para adaptar *TCP/IP* (*Protocolo de Control del Transporte / Protocolo de Interconexión de redes*) para su uso en transmisiones por radio en misiones espaciales interplanetarias.

La telefonía móvil en general y *GSM* (*Sistema Global de comunicación Móvil*) en particular también han sufrido un avance importante. Existen aplicaciones que los diseñadores de *GSM* jamás imaginaron para su estándar, por ejemplo para comunicar los contenedores de una red de clasificación y reciclado de vidrio. También las plataformas digitales ven como aumenta mucho cada año el número de abonados a los servicios que ofrecen, cada vez de mayor calidad. La televisión por cable es una tecnología que también experimenta un aumento de la demanda muy esperanzador.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Sin embargo [2] existe un gran problema en las redes de datos actuales: la dificultad de implantar nuevos estándares en el modelo arquitectónico de red actual. Desde que se detecta la necesidad de un nuevo servicio, hasta que este se implanta a nivel global pueden transcurrir años en su proceso de estandarización. Este es un proceso extremadamente lento porque requiere el acuerdo de multitud de entidades científicas, económicas y políticas. Además, en la mayoría de los casos se requiere retrocompatibilidad.

Un ejemplo de este problema es lo que sucedió con el conjunto de protocolos de *OSI* (*Interconexión de Sistemas Abiertos*). Las empresas no pudieron esperar a la finalización del estándar de *OSI* y se decidieron por opción más rápida: el conjunto de protocolos *TCP/IP* que se convirtió en un estándar *de facto* arruinando años de trabajo en el diseño de *OSI*. Un ejemplo más cercano en el tiempo es el *UMTS* (*Sistema Universal de Telecomunicación Móvil*) que no se acaba de imponer por falta de acuerdo entre los grandes operadores de telefonía mundial.

En [3] se enumeran un conjunto de servicios, tales como videoconferencia, telefonía sobre *IP*, multicast, servicios de movilidad para comunicaciones inalámbricas, etc. Estos, o no han sido implantados totalmente, o los intentos de implantación que se han hecho no tienen el rendimiento esperado porque la utilización de la arquitectura tradicional de Internet les impone muchas restricciones de *Calidad de Servicio* (*QoS*) y de rendimiento.

En [4] se presentan los últimos desarrollos que se han dado en Internet como la aparición de dos nuevos paradigmas de programación de red: las redes programables y las redes activas. Ambos son posibles soluciones para el cuello de botella que significa el proceso de estandarización. Una red programable es aquella en la cual los nodos de la red están abiertos a la administración de sus recursos, procesado de protocolos y procesado de información por un conjunto de entidades de control de cada cliente. El objetivo es facilitar mecanismos de señalización abierta, control y gestión de aplicaciones y servicios multimedia de nivel más alto. Una red activa permite la ejecución de código contenido en los paquetes activos o cápsulas, en los nodos intermedios de la red. Para ello se provee de una *Interfaz de Programación de la Aplicación* (*API*).

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

1.2 Las Redes de Área Local Inalámbricas (WLAN)

Las comunicaciones [5] en el siglo 21 van a ser básicamente comunicaciones móviles. Los nuevos dispositivos de comunicaciones móviles permitirán que las personas del nuevo siglo sean *nómadas* en sus actividades cotidianas. Entre las tecnologías que van a provocar este cambio se encuentran la telefonía celular de cuarta y quinta generación, sistemas de acceso inalámbrico de banda ancha, Redes de Area Local (*LAN's*) de onda milimétrica, *Sistemas de Transporte Inteligente* (*ITS's*) y *Sistemas de Plataforma Estratosférica de Gran Altitud* (*HAPS's*). Estas tecnologías deben satisfacer básicamente tres necesidades: alta movilidad, alta tasa de bps. (bits por segundo) y cobertura global. Así, la *Internet Inalámbrica* (*Wireless Internet*) [6] deberá tener las siguientes cualidades:

- Debe dar todos los servicios de Internet a los usuarios móviles, y no un subconjunto. Es más, la telefonía de voz debe migrar también a ser servida en esta Internet Inalámbrica.
- Debe de ser razonablemente rápida (al menos un throughput de 100.000 bps. por usuario).
- Debe funcionar en entornos *indoor* y *outdoor*.
- Debe implementar un uso eficiente de la potencia, dado que la mayoría de los dispositivos funcionarán con baterías.
- Debe ser escalable para soportar millones de dispositivos activos.

Dentro de toda esta vorágine de nuevas tecnologías y servicios se encuentran las redes WLAN. El campo de las WLAN's está sufriendo una rápida expansión [7] como resultado de los avances en las comunicaciones digitales, los ordenadores portátiles y la tecnología de semiconductores. Las primeras aplicaciones que han sido implementadas son aquellas en las cuales la movilidad es una característica muy deseable. Por ejemplo control de inventario en almacenes, terminales en el punto de venta, y chequeo de coches de alquiler. Las redes WLAN también están siendo cada vez más usadas en entornos hospitalarios y universitarios, donde los usuarios requieren mucha movilidad, pero no demasiado ancho de

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

banda. Además existen entornos en los cuales una instalación alámbrica es impracticable: edificios históricos, infraestructuras alquiladas, convenciones, etc.

Con este desarrollo de las redes WLAN es necesario un proceso de estandarización que permita la interoperabilidad entre los productos para esta tecnología. Dos estándares relativamente nuevos para este tipo de redes son el IEEE 802.11 y el *HIP*ERLAN [7].

1.3 Objetivos del proyecto

Las redes WLAN tienen determinadas características que las hacen distintas a las redes LAN cableadas. Los principales problemas [8] de las redes WLAN son tres:

- los host móviles son pobres en recursos,
- los enlaces tienen un reducido y variable ancho de banda (latencias elevadas), y
- baja fiabilidad (los enlaces son susceptibles de perderse en un determinado momento).

Si nos fijamos en estas tres características es inmediato pensar que una red WLAN, salvando las distancias, debería presentar un comportamiento similar a la Internet. En la Internet los *hosts* tienen recursos muy diferentes, y por lo general escasos. En la Internet las latencias del tráfico son muy variadas y elevadas. La fiabilidad de los enlaces en la Internet es muy baja, dado que es muy común la pérdida de datos.

Por todo esto, a otra escala, una red WLAN se comporta exactamente igual que la Internet. Si además, en nuestro entorno de área local, combinamos segmentos alámbricos e inalámbricos, diversificaremos aún más las latencias.

Por lo tanto, en este proyecto fin de carrera vamos a utilizar entornos WLAN-LAN para evaluar nuestros protocolos con dos objetivos:

- Evaluar el rendimiento de los mismos sobre redes WLAN-LAN
- Evaluar la posibilidad de la implementación de estos protocolos en aplicaciones orientadas a Internet.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Existe una herramienta que utiliza los conceptos de Redes Activas que se llama *ANTS* (*Sistema de Transporte de Red Activa*). Su principal cualidad es que permite realizar prototipos de protocolos rápidamente. El uso de esta herramienta nos permitirá testear a fondo nuestros protocolos.

1.4. Estructura de la memoria

A continuación pasamos a explicar la estructura de esta memoria. En el capítulo 2 se presentan las Redes Activas. Se explican los conceptos que incluyen y se enumeran algunos proyectos de investigación en este tema. Además se explica como es la programación con el ANTS.

En el capítulo 3 se muestra una clasificación de los protocolos de comunicación. Cada tipo es explicado con detalle.

En el capítulo 4 se explican uno a uno y en orden de complejidad creciente, los cuatro de los protocolos diseñados, tres de los cuales tienen dos versiones. Además se expone brevemente como se implantaron sendos prototipos.

En el capítulo 5 se exponen todas las pruebas realizadas con los prototipos de estos protocolos. Así mismo se enumeran las conclusiones respecto de cada uno de los protocolos, a la luz de los resultados de las pruebas.

Por último, en el capítulo 6 se hace un balance del uso del ANTS en el diseño rápido de prototipos de nuevos servicios. Se expondrán las conclusiones de este Proyecto Fin de Carrera, y líneas futuras de trabajo.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Capítulo 2. Las Redes Activas

En este capítulo se explican las Redes Activas. En primer lugar se hace una introducción en la que se explica el contexto en el que aparecen. Luego se definen y se da una visión conceptual de un nodo activo y los elementos que lo componen. Seguidamente se exponen algunos proyectos relacionados. Por último se concluye el capítulo explicando la arquitectura básica del *ANTS* (*Sistema de transporte de red activa*) y como se programan los protocolos utilizando esta herramienta.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

2.1 Introducción

Durante la ya dilatada existencia de las redes de datos una constante ha estado presente en todas las aproximaciones: los datos portados son tratados transparentemente. De esta manera se garantizaba la integridad de la información y se simplificaba la arquitectura de la red de datos.

Otro aspecto común a todas las redes de comunicación de datos tradicionales es el hecho de que los nodos intermedios son sistemas cerrados programados de manera muy rígida con software ad hoc empotrado.

Estas dos características provocan que la implantación de nuevos protocolos se haga muy lenta. Además el hecho de que el software de los nodos intermedios esté empotrado es una limitación crítica de la posibilidad de realizar computación de manera distribuida en la red. Para liberar a las redes de estos dos serios problemas, científicos del MIT inventaron el nuevo paradigma denominado Redes Activas (*AN*) [9][2].

2.2 Definición de Red Activa

Una red activa es una red de datos en la cual:

Podemos inyectar programas a medida

Los datos portados por los paquetes se pueden manipular en los nodos intermedios

A través de una red activa fluyen *paquetes activos*. Estos presentan una peculiaridad respecto de los paquetes de una red de datos tradicional: Portan el código o referencias al código que los procesará al llegar a los distintos nodos de esta red.

Una red activa está formada por un conjunto de *nodos activos* interconectados. Llamamos *nodo activo* a aquel nodo con la facultad de procesar activamente los paquetes que reciben. Esto significa que los nodos activos procesan los paquetes activos ejecutando el código al que hacen referencia o que portan estos últimos. Esto es lo mismo que decir que ejecutarán los programas a medida inyectados por los usuarios. En general también los

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

nodos activos deben ser capaces de manipular la carga útil de los paquetes, modificándola si es necesario.

La aproximación discreta o fuera de banda [9], es aquella en la que los paquetes activos están arquitectónicamente separados del código que los procesará. La aproximación integrada o en banda es aquella en la que los paquetes activos portan ellos mismos el código o referencias a librerías de código que ha sido cargado utilizando un mecanismo dentro de la misma arquitectura. Esto es, si el administrador de cada nodo carga el código que procesa los paquetes activos en los nodos activos estáticamente, estamos ante una aproximación fuera de banda o discreta. En caso contrario, cuando se produce una carga del código de manera dinámica según se necesite, es una aproximación en banda o integrada.

Ejemplos de aproximaciones integradas son el ANTS (Active Network Transport System) [10], [11] y [12], el M0 [26], los SmartPackets [24]. Ejemplos de aproximaciones discretas son los conmutadores programables como el que se explica en [18]. Existen aproximaciones mixtas como el proyecto SwitchWare en [14] y [15].

En la figura 2.1 se muestran algunas abstracciones que nos ayudarán a entender el concepto de nodo activo. En primer lugar definimos **entorno de ejecución** (*EE, Execution Environment*) como el conjunto de recursos software y hardware que se asignan a un determinado flujo de paquetes en un nodo activo. Tanto los recursos que se asignan como el nivel de libertad para operar con ellos son en general, determinados de antemano, antes de la creación del entorno de ejecución por motivos de seguridad.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

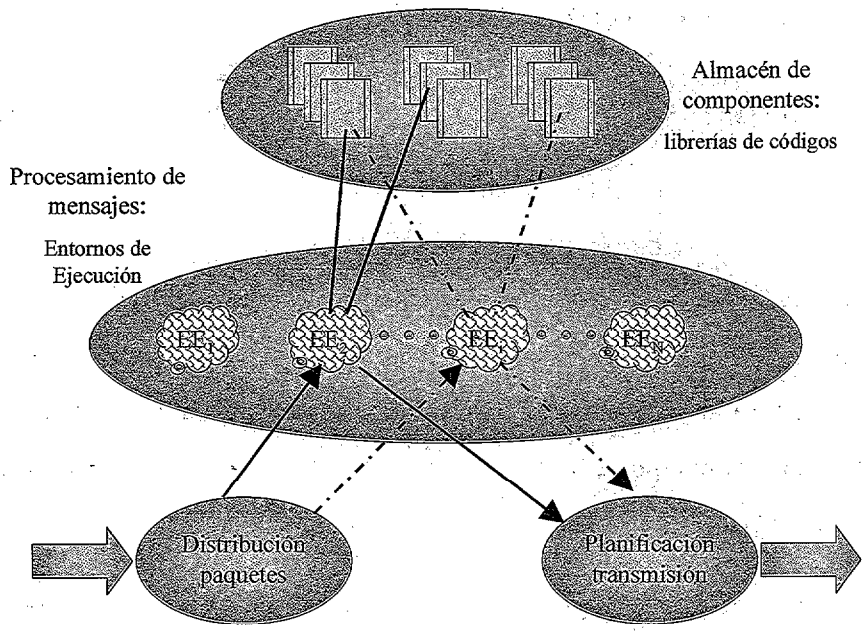


Figura 2.1: Visión conceptual de un nodo activo.

En un nodo activo se llevan a cabo cuatro actividades:

- **Distribución de paquetes**: Los paquetes entrantes según su tipo, que se indicará por medio de campos en los mismos, son distribuidos al EE que corresponda.

- **Procesamiento de mensajes**: Cada EE lleva a cabo la computación que procesada sobre los paquetes que le son asignados y haciendo uso de los códigos que corresponda. Estos códigos en general se encontrarán en el almacén de componentes o en los mismos paquetes.

- **Almacenamiento de componentes**: Se reservarán recursos para el almacén de librerías de código que podrán ser cargadas o descargadas de manera estática o dinámica. En general, para cada EE se especificará a qué componentes de este almacén puede acceder.

- **Planificación/transmisión**: Los flujos de paquetes de salida de cada entorno de ejecución son planificados y transmitidos.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

2.3 Aproximaciones actuales

A continuación presentamos una muestra de los trabajos relacionados con las ANs.

2.3.1 MIT

D. Tennenhouse y otros [9] [2] fueron los creadores del concepto de red activa. Crearon el concepto de cápsulas como datagramas que portan fragmentos de código ligeros. Para evitar fallas de seguridad la ejecución de las cápsulas se lleva a cabo de manera interpretada sobre una Máquina Virtual Java (*JVM*). Esto da lugar a que esta aproximación tenga problemas de rendimiento.

También en este grupo D. Wetherall con la colaboración de otros científicos construyó el ANTS [10], [11] y [12]. El principal éxito del ANTS es proveer de una arquitectura para el desarrollo de protocolos de red dinámicos. Introducen una optimización del concepto de cápsula. En vez de portar código, los paquetes activos portan punteros a código. Este código se almacena en una cache la primera vez que se necesite del nodo del que provenga el paquete. De esta manera se optimiza el ancho de banda necesario. El lenguaje de programación utilizado es el Java.

Paralelamente este grupo presentó el proyecto PAN [13], una aproximación similar al ANTS pero que en vez de utilizar Java utiliza Código Máquina para mejorar el rendimiento de la red activa. Esta última aproximación, si bien ha dado buenos resultados de velocidad presenta problemas de seguridad que, según las últimas noticias que tenemos, no han sido solucionados.

2.3.2 BBN

Este grupo de investigación implementa también la opción integrada. Ellos denominan a los paquetes activos Smart Packets [24]. El objetivo de esta aproximación es realizar funciones de diagnóstico extendido. El código procesado por los paquetes activos es portado en los mismos. Crearon un lenguaje compacto llamado Sprocket para ello. Los programas se autentifican antes de su ejecución y tienen una duración limitada temporalmente.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Este trabajo está claramente orientado a la gestión de red y suponen una gran mejora respecto al SNMP.

2.3.3 Georgia Tech

Zegura, Bhattacharjee y Calvert en [19], [20], [21], [22] y [23] implementan la opción discreta. Introdujeron una visión general del código de red como un conjunto de funciones que se llaman dependiendo de identificadores en los paquetes de datos. Implementan procesamiento de datos específico de la aplicación, por ejemplo para el control de la congestión en streams de vídeo MPEG.

Este grupo también trabaja en la construcción de caches de redes autoorganizativas construidas mediante redes activas. Utilizaron modelos analíticos y simulaciones para estudiar las mejoras obtenidas mediante el cacheo de código en la red.

2.3.4 Universidad de Pennsylvania

En este grupo de investigación se trabaja en el proyecto Switchware [14]. Es una mezcla entre las dos aproximaciones discreta e integrada. En este proyecto se utilizan tres componentes básicos que son los paquetes activos, los swichlets y una infraestructura de router activo segura. Los paquetes activos se programan en un lenguaje sencillo que se llama Lenguaje de Programación para Redes Activas (*PLAN*). Los programas PLAN son fuertemente tipeados y se les efectúa una comprobación de tipos estático antes de ser inyectados en la red. Además los programas PLAN se hacen seguros restringiendo sus acciones. Por ejemplo un programa PLAN no puede modificar el estado de un nodo.

Para compensar esas limitaciones los programas PLAN pueden llamar a los switchlets. Los switchlets se programan en CAML. Este lenguaje soporta metodologías formales para comprobar la seguridad de los programas en tiempo de compilación. Por esto no se necesita que sean programas interpretados, minimizando los recursos necesarios para ejecutarlos. El diseñador autentifica a los switchlets que carga explícitamente en los nodos.

En una capa más baja de esta arquitectura el *Entorno de Red Activa Segura* (*SANE*) se asegura de la integridad del entorno completo [15]. El SANE utiliza técnicas de criptografía para asegurar la integridad del sistema.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Otro grupo en esta universidad trabaja en el Pipeline de Procesos de Protocolo Programable [16]. Utilizan FPGAs reconfigurables para implementar la funcionalidad del procesado de los paquetes en hardware. En concreto se centran en la corrección de errores en los paquetes.

2.3.5 Universidad de Arizona

El Scout [27] y [28] es un sistema operativo orientado a las comunicaciones. El kernel es una composición a medida de primitivas de comunicación de bajo nivel que se implementan como módulos. Los módulos implementan funcionalidades distintas tales como los protocolos IP, UDP o TCP. Los módulos pueden ser combinados para formar Paths que constituyen canales lógicos sobre los que fluyen los datos de entrada/salida.

El Joust [29] es una implementación de la JVM que se ejecuta sobre el Scout. La API del Joust ha sido extendida para interactuar con el Scout y permitir que las aplicaciones accedan a recursos de bajo nivel. Todos los componentes fijos se escriben en C o Java y son compilados en código máquina estáticamente.

La implementación [29] del ANTS sobre Joust/Scout va entre dos y tres veces más rápido que la implementación sobre el JDK sobre Linux. De hecho es la implementación más rápida de la que hasta el momento tenemos noticia. Sin embargo sigue sin ser apropiado para aplicaciones de elevado volumen de trafico y requerimientos de gran ancho de banda.

2.3.6 Universidad de Columbia

El Netscript [25] es un *middleware* para programar las funciones que están en los nodos intermedios de la red. El lenguaje de programación Netscript permite procesar los flujos de paquetes (scripts) y está enfocado al routing, a analizadores de paquetes y a funciones de señalización. Los programas Netscript están organizados como un conjunto de agentes móviles que se envían a los sistemas remotos y son ejecutados bajo control local o remoto.

El principal logro de Netscript es simplificar el desarrollo de sistemas interconectados y habilitar la programación remota. La idea del proyecto Netscript es llegar a redes que soporten la programación flexible y el desarrollo dinámico de software en todos los nodos.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

2.3.7 ETH Zurich y Universidad de Washington

En [18] se presenta el trabajo conjunto de un grupo de la universidad de Zurich y otro grupo de la universidad de Washington que investigaron en la arquitectura hardware y software de un nodo de red activa para requisitos de alto volumen de tráfico y de ancho de banda necesario. En este trabajo se inserta el código fuera de banda. Las llamadas a códigos se disparan mediante referencias en los datagramas.

2.3.8 Universidad Carlos III

En la Universidad Politécnica de Madrid un grupo de investigadores que ahora trabajan en la Universidad Carlos III llevaron a cabo un proyecto [17] de un protocolo de multicast activo con tres niveles de fiabilidad distintos denominado RMANP. Lo implementaron sobre ANTS y de los resultados obtenidos llegaron a las siguientes conclusiones: Las redes activas proporcionan flexibilidad para el desarrollo de nuevos servicios de red y es necesario mejorar el rendimiento de las redes activas para tráfico en tiempo real.

Nosotros en este proyecto hemos utilizado el ANTS como plataforma de desarrollo. Por ello, a continuación describimos las líneas generales de este entorno.

2.4 Arquitectura básica del ANTS

Una red basada en el ANTS [11] consiste en un conjunto de nodos que ejecutan el núcleo del ANTS. Estos nodos pueden estar situados en una misma máquina, en una red de área local, o en una red de área extensa. El sistema está construido sobre UDP/IP y crea un nivel software que sirve como nivel de red para las aplicaciones distribuidas que se ejecutan sobre él.

A diferencia de IP, el servicio de red que crea el ANTS no es fijo, sino flexible. Diferentes aplicaciones pueden introducir nuevos protocolos por medio de la especificación de las rutinas a ejecutar en los nodos de la red sobre los mensajes enviados. Es posible balancear el procesamiento que se desee desde los nodos terminales a la red. Esto incluye operaciones que tradicionalmente se llevaron a cabo en los nodos terminales y operaciones nuevas que sólo tienen sentido en el ámbito de las Redes Activas.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Existen tres aspectos fundamentales del ANTS que permiten una rápida innovación de los protocolos:

- Los nodos de la red pueden soportar varios protocolos distintos ejecutándose simultáneamente y proporcionando cada uno servicios diferentes.

- La arquitectura permite la implantación de un nuevo servicio sólo con el acuerdo de las partes interesadas, sin necesidad de registrar nuevos protocolos de manera centralizada.

- La arquitectura facilita el desarrollo dinámico de nuevos protocolos "en caliente", sin necesidad de programar los nodos de la red uno por uno para dar el nuevo servicio, en especial en redes en las cuales su escala es grande.

Los tres mecanismos que permiten que el ANTS disfrute de esas cualidades son los siguientes tres componentes:

- Los paquetes de las redes tradicionales se sustituyen por **cápsulas** con referencias al código que las procesará.

- Los Routers y los nodos terminales se sustituyen por **nodos activos** que ejecutan las rutinas de procesamiento de las cápsulas que proceda y mantienen su estado asociado.

- Un mecanismo de **distribución de código** que asegura que las rutinas de procesamiento que son automática y dinámicamente transferidas a los nodos donde sean requeridas.

Para usar los elementos programables de esta red necesitamos un modelo para combinar las rutinas de encaminamiento en los nodos individuales en un patrón de comportamiento (protocolo) que defina el procesamiento que tendrá lugar en toda la red como un todo. Para ello se definen tres abstracciones: las cápsulas, los grupos de código y los protocolos (*capsule*, *code group* y *protocol*).

Algunas rutinas serán muy comunes y estarán disponibles en todos los nodos de la red. Otras en cambio, serán especificas de la aplicación y será necesario transferirlas a cada

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

nodo mediante el mecanismo de distribución de código antes del procesamiento de la primera cápsula de este tipo.

Un **grupo de código** es una colección de tipos de cápsula relacionados, cuyas rutinas de encaminamiento serán transferidas como un todo por el mecanismo de distribución de código.

Un **protocolo** es una colección de grupos de código relacionados que son tratados como una unidad de protección por los nodos activos. Las cápsulas que pertenecen a un mismo protocolo compartirán información y recursos en los nodos.

En la figura 2.2 se muestra el formato de una cápsula.

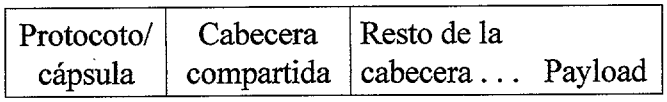


Figura 2.2: Formato de una cápsula

Por último enumeramos los cuatro tipos fundamentales de primitivas de nodo que soporta en ANTS:

- *Acceso al entorno*: estado de los enlaces, tablas de enrutamiento, tiempo local, etc.
- *Manipulación de cápsulas*: acceso a todos los campos y carga útil de las mismas.
- *Operaciones de control*: creación y encaminamiento de otras cápsulas, desecho o copia de sí mismas, etc.
- *Almacenamiento en los nodos*

2.5 Programación de prototipos con el ANTS

En [10] se explica el modelo de programación con el ANTS. Los usuarios desarrollan los nuevos protocolos extendiendo un conjunto de clases virtuales proveídas por el sistema. Una vez programados, los nuevos protocolos se usan manipulando instancias de esas

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

nuevas clases en conjunción con la ejecución del núcleo del ANTS. El núcleo del ANTS facilita los servicios necesarios por cada nodo de la red. En concreto:

- Cada nuevo protocolo se desarrolla extendiendo la clase virtual *Protocol*. Esto requiere la identificación de todos los tipos de paquete. Cada tipo de paquete y su rutina de encaminamiento es especificado extendiendo la clase virtual *Capsule*.

- Cada nueva aplicación se programa extendiendo la clase virtual *Application*.

- Cada instancia de la clase *Node* representa un núcleo del ANTS.

- El nuevo protocolo y la nueva aplicación son utilizadas creando una instancia de cada uno y asociándolos a un nodo.

Este modelo permite que lo usuarios creen servicios a medida, asumiendo que la red de nodos activos ya existe y se está ejecutando. Además, y para facilitar la experimentación con diferentes topologías y servicios de nodo, el ANTS provee dos herramientas más:

- La gestión de la configuración, que permite crear una topología de red completa con aplicaciones para ser gestionada. Esto incluye el cálculo de rutas y la inicialización de la configuración de los nodos.

- Arquitectura de extensión de nodo, para incrementar los servicios básicos del nodo con componentes mayores que no son fácilmente transferibles con el mecanismo de distribución de código, o son demasiado especializados para ser necesitados por todos los nodos de la red.

Por último indicar que, desde un punto de vista pragmático, la programación de los prototipos con el ANTS se divide en dos aspectos: la programación en el nivel de red de la parte distribuida (que se programa en las rutinas de encaminamiento de las cápsulas) y la programación en el nivel de aplicación de la parte extremo a extremo (que se programa en las aplicaciones).

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Capítulo 3: Protocolos de comunicación

En este capítulo se explican los tipos de protocolo de comunicación que existen en las redes de datos actuales. Primero se da una clasificación de los mismos en protocolos unicast, multicast y broadcast. En segundo lugar se explica cada uno de los tipos y su utilidad. Por último se da una justificación del uso de protocolos de multicast mediante un modelo matemático sencillo que hemos creado.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

3.1 Tipos de Protocolo

Actualmente, las redes de datos se clasifican en dos tipos dependiendo de si existe o no un establecimiento de la conexión antes de la transferencia de datos. Estos dos tipos son los siguientes:

- *Redes orientadas a conexión* (o basadas en *circuito virtual*): aquellas redes en las cuales existe un establecimiento de la conexión antes de la transferencia de la información y una liberación de la misma al final. Durante el establecimiento de la conexión se fija el camino que los datos seguirán en todos los nodos intermedios hasta alcanzar el destino. Por lo tanto, en la fase de transferencia de datos no es necesario que los paquetes lleven información del destino u origen.

- *Redes no orientadas a conexión* (o basadas en *datagrama*): aquellas redes en las cuales no existe un establecimiento de la conexión. Por ello, cada paquete debe llevar información de la dirección de origen y destino de los datos. Además, es posible que los paquetes sigan diferentes caminos hasta alcanzar el destino.

Con el término "redes", nos referimos a cualquier nivel de la torre de protocolos de cualquier red de datos. Por ejemplo, IP es un nivel no orientado a conexión. Los datagramas IP portan las direcciones de origen y destino. Sin embargo, TCP si es orientado a conexión. Una vez que se establece un par de *sockets* entre dos aplicaciones, los datos no requieren dirección del destino.

Por lo tanto podemos decir que tenemos una *red no orientada a conexión* (red IP) bajo una *red orientada a conexión* (red TCP).

Las conexiones [30] efectuadas en una *red orientada a conexión* se dividen en:

- Conexiones *punto a punto*: El origen de los datos es un único *host*. Es destino de los datos también es único.

- Conexiones *punto a multipunto*: Aunque el origen de los datos es un *host*, el destino de los mismos son varios *hosts*.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

- Conexiones *multipunto a punto*: El origen de los datos es múltiple y el destino un solo *host*.

- Conexiones *multipunto a multipunto*: Tanto el origen como el destino de los datos son múltiples.

En [30] también se clasifican las conexiones en las redes no orientadas a la conexión. Dado que en estas redes no existe una relación que se prolongue en el tiempo entre origen y destino, la clasificación queda reducida a:

- *Conexiones unicast*: el origen y el destino de los datos son únicos.

- *Conexiones multicast*: el origen es un solo *host* y el destino son un conjunto determinado de *hosts*.

- *Conexiones anycast* (o *broadcast*): el origen es un único *host* y el destino es cualquier *host* que tenga acceso al medio compartido en el cual se han difundido los datos.

Nosotros, basándonos en esta última, clasificamos los protocolos en función del tipo principal de conexiones que se den lugar en él, en:

- Protocolos de Unicast

- Protocolos de Multicast

- Protocolos de Broadcast.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

3.2 Protocolos de Unicast

Son aquellos protocolos en los cuales el tipo principal de Conexiones tienen un origen y un destino. La mayoría de los protocolos de Internet son de este tipo. Por ejemplo en HTTP, un cliente en un navegador de Web, hace una petición a un servidor Web y este le responde.

Este ejemplo responde al esquema de comunicación Cliente-Servidor. Los protocolos Cliente-Servidor se caracterizan porque el servidor es el que se encarga de aceptar o denegar las peticiones, del control de flujo, etc. La mayoría de los protocolos de Internet siguen este esquema de programación. Por lo tanto, las redes activas no aportarían nada nuevo si sólo fuera posible programar protocolos de este tipo. Afortunadamente no es así.

3.3 Protocolos de Multicast

Los protocolos de Multicast son aquellos en los que el origen de los datos es único pero el destino son un conjunto de *hosts*. Estudiar [30] para entender claramente y con bastante detalle lo que es multicast.

Cuando queremos que un *host* distribuya la información a un conjunto definido de usuarios (grupo) hablamos de un protocolo de multicast o de grupo. Hay muchas aplicaciones que entran en este tipo de protocolos: distribución de software, actualización de bases de datos replicadas, sistemas de control y comando, audio/video conferencia, juegos distribuidos, y *Simulación Distribuida Interactiva* (*DIS*).

Al contrario de los protocolos Cliente-Servidor, en un protocolo de multicast es el cliente, que en el caso de multicast se le suele llamar Receptor (*Receiver*), el que puede decidir cuando incorporarse a la sesión y cuando abandonarla. Por lo general la incorporación se produce mediante una petición de suscripción cursada desde el Receptor al servidor. Al servidor multicast se le suele llamar Transmisor (*Sender*). A partir del momento en que la petición del nuevo receptor es anotada, este recibirá todos los datos que el transmisor envíe al grupo.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Dado que la Internet fue diseñada pensando en el esquema Cliente-Servidor, ésta no permite que con los protocolos multicast se obtengan un rendimiento óptimo. Por lo tanto, la necesidad de implantar servicios multicast, es una de las motivaciones más importantes para que la comunidad científica estudie las Redes Activas como una alternativa a las redes tradicionales. En una Red Activa, la implementación de cualquier protocolo de multicast es inmediata.

Algunos de los problemas que presenta el diseño de estos protocolos de multicast son:

- La congestión que las peticiones de suscripción, las confirmaciones y las peticiones de retransmisión pueden incurrir.
- El desperdicio de ancho de banda que la distribución de paquetes con la misma información a muchos destinos.
- El desperdicio de ancho de banda que la transmisión de paquetes ya obsoletos hasta los suscriptores puede incurrir.

Las ANs pueden paliar estos problemas con técnicas tales como:

- Minimizar la transmisión de suscripciones hacia el nodo Transmisor, por medio de suscripciones "jerárquicas" de las distintas ramas de la red.
- Minimizar confirmaciones y retransmisiones mediante un cacheo distribuido de los datos recientes.
- Eliminar paquetes obsoletos descartándolos en los nodos intermedios.

Hay ya varios proyectos que avalan esta posibilidad. En [17], el grupo de investigación de María Calderón desarrolló un protocolo de multicast con tres niveles de fiabilidad sobre la plataforma ANTS desarrollada por el MIT. Otro ejemplo de protocolo multicast en una Red Activa es el trabajo explicado en [26]. En el se estudian dos protocolos de multicast. Uno es de multicast robusto de audio y otro de multicast de vídeo en capas. Ambos protocolos se implementan en dos plataformas distintas: el ANTS y el M0 [26].

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

3.4 Protocolos de Broadcast

Son aquellos protocolos en los que el tipo principal de conexiones son difusiones en un medio compartido de paquetes. Se caracterizan porque a priori se desconocen los *hosts* que recibirían esos paquetes. Se utiliza en aplicaciones en las cuales, aprovechando que no existe necesidad de privacidad en el tipo de información transmitida, se difunden los datos y el que le interesan los recoge.

La gran ventaja de este protocolo es que no hay *handshake* explícito. Es decir, el 100% del canal se utiliza para transmitir datos. No es necesaria ninguna confirmación por parte de los receptores, simplificando su diseño

3.5 Justificación del uso de Protocolos de Multicast

A continuación vamos a presentar dos sencillos modelos que hemos creado, que nos sirven para comparar el tráfico generado en una misma red por una aplicación multicast frente a una aplicación Cliente-Servidor.

La topología general en la que nos basamos se muestra en la figura 3.1. Para simplificar nuestro modelo hemos adoptado las siguientes restricciones en esta topología:

- Todos los nodos que no sean nodos hoja están conectados hacia un nivel inferior a un mismo número de nodos. Este número es M.
- Todos los nodos hoja se encuentran en el mismo nivel de jerarquía.
- El número de niveles de jerarquía de esta topología es L. No contamos el nivel en el que están los nodos hoja.

A partir de este modelo podemos calcular el número total de mensajes que se generarán cada vez que el nodo raíz desee difundir a todos los nodos hoja un mensaje, dependiendo si se trata de una aplicación Cliente-Servidor o multicast.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Si nos fijamos en la figura 3.2, es fácilmente demostrable que el número de mensajes generado será:

$$n_cs = L \times 2^L$$

Ecuación 3.1: Número de mensajes en una aplicación cliente servidor con M=2

$$n_mc = 2 \times (2^L - 1)$$

Ecuación 3.2: Número de mensajes en una aplicación cliente servidor con M=2

Siguiendo el mismo procedimiento es fácil llegar a las siguientes ecuaciones más genéricas:

$$n_cs = L \times M^L$$

Ecuación 3.3: Número de mensajes en una aplicación cliente servidor

$$n_mc = M \times \frac{(M^L - 1)}{M - 1}$$

Ecuación 3.4: Número de mensajes en una aplicación cliente servidor

A partir de estas dos últimas ecuaciones hemos realizado un conjunto de simulaciones que pueden ser muy ilustrativas de la disminución de mensajes en la red que supone el uso de un protocolo multicast frente a uno Cliente-Servidor. En las figuras 3.3 y 3.4 mostramos los resultados. En la figura 3.3 hemos puesto directamente el número de mensajes total que se generarían en la red. En la figura 3.4 hemos calculado la razón que indicamos en la ecuación 3.5.

$$Mejora = \frac{número_de_mensajes_.multicast}{número_de_mensajes_cliente - servidor} \times 100$$

Ecuación 3.5: Razón de mejora de multicast vs Cliente-Servidor

Es posible introducir un nuevo elemento en nuestros modelos que los hará más reales. Se trata de considerar que existe un porcentaje de paquetes fijo que será necesario retransmitir en todos los enlaces de la topología de la figura 3.1.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

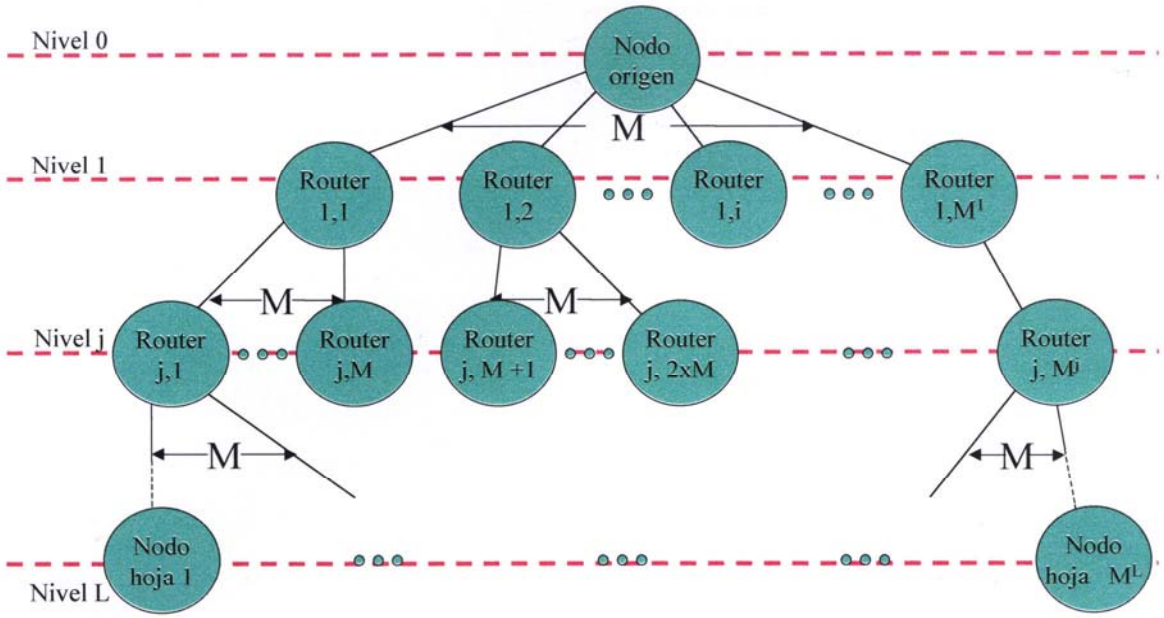


Figura 3.1: Topología general del modelo

Llamamos x al tanto por uno de los mensajes que será necesario retransmitir en cada enlace. Si transmitimos P mensajes por un enlace, el número real de mensajes que será necesario transmitir sería el que se indica en la ecuación 3.6.

$$P_{total} = P + P \times x + (P \times x) \times x + (P \times x^2) \times x + \cdots = \\ = P \times (1 + x + x^2 + x^3 + \cdots) = P \times \frac{1}{1-x}$$

Ecuación 3.6: Número de mensajes que se transmiten por cada enlace teniendo en cuenta x pérdidas.

Esto es lo mismo que decir que el número de paquetes a transmitir a través de cada enlace se verá ponderado por el factor 1/(1-x). Si esto lo extrapolamos a las ecuaciones 3.3 y 3.4 nos quedan las ecuaciones 3.7 y 3.8.

$$n_cs = L \times M^L \times \left(1/_{1-x}\right)$$

Ecuación 3.7: Número de mensajes en una aplicación cliente servidor teniendo en cuenta pérdidas

$$n_mc = \frac{1}{1-x} \times M \times \frac{(M^L - 1)}{M - 1}$$

Ecuación 3.8: Número de mensajes en una aplicación cliente servidor teniendo en cuenta pérdidas

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

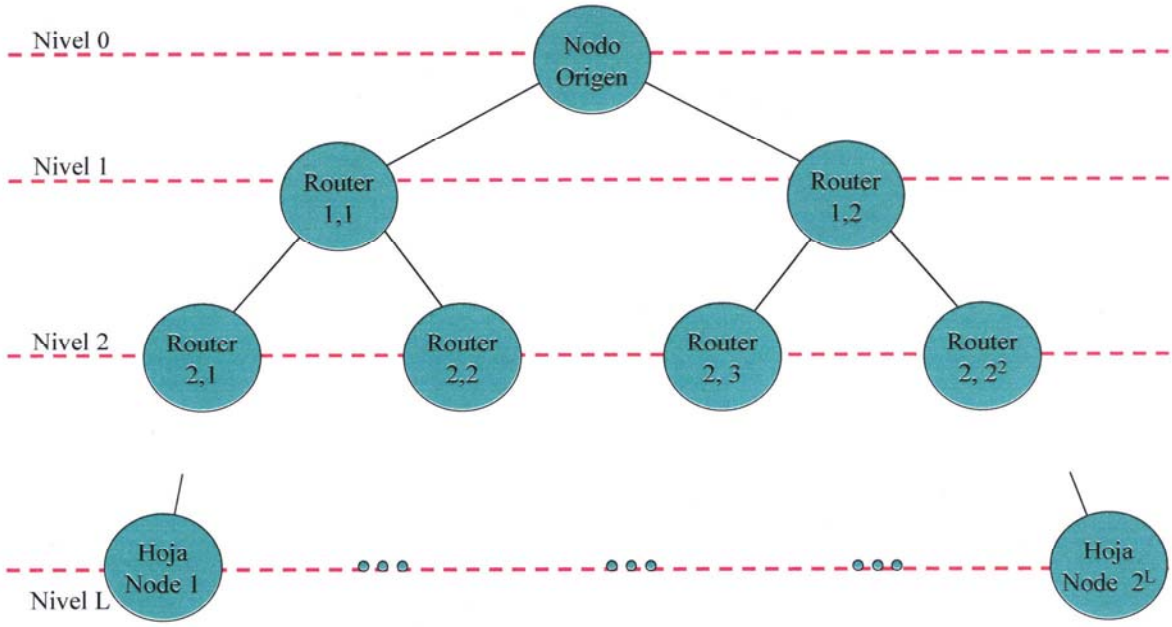


Figura 3.2: Caso de M=2

Dado que en ambos casos solamente añadimos el mismo factor, aunque el número de mensajes variará en ambos modelos, la razón de mejora del protocolo multicast será la misma.

A la vista de los resultados obtenidos en nuestros sencillos modelos podemos afirmar que la utilización de un protocolo multicast para la difusión de mensajes a un conjunto de nodos hoja supone una mejora significativa frente a un protocolo Cliente-Servidor.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

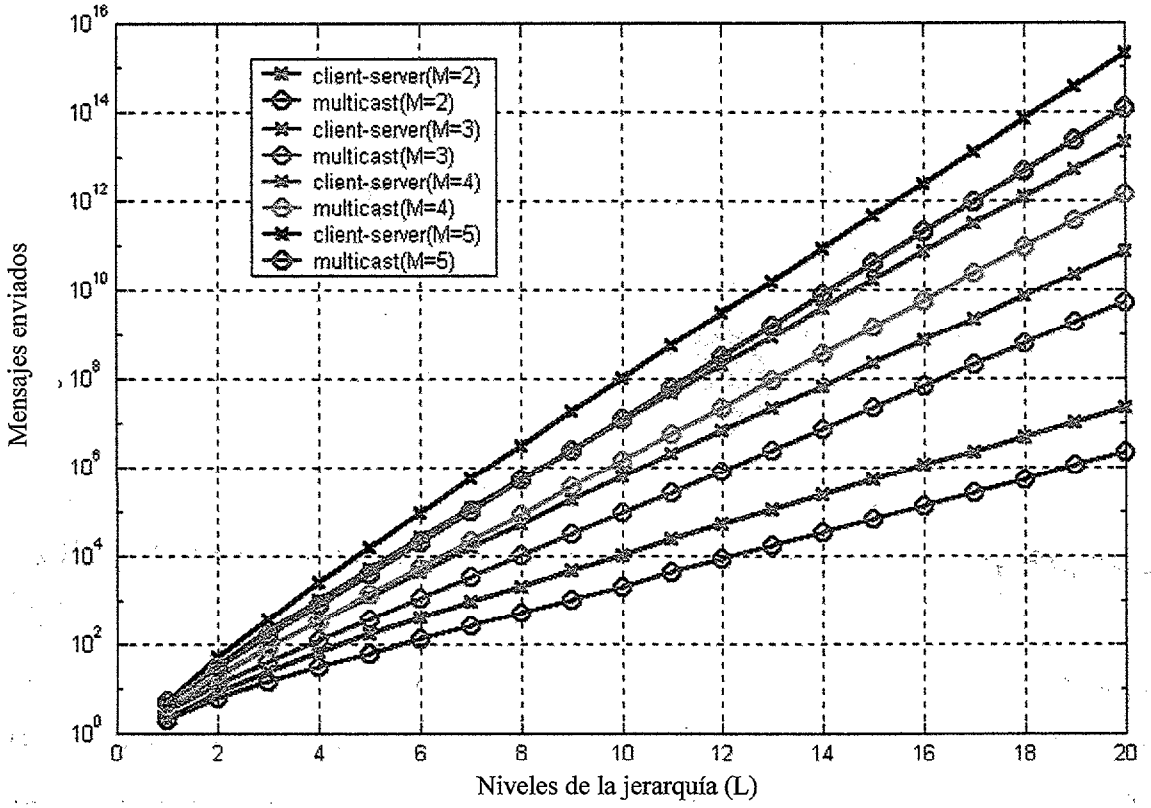


Figura 3.3: Número de paquetes con M=2 hasta M=5

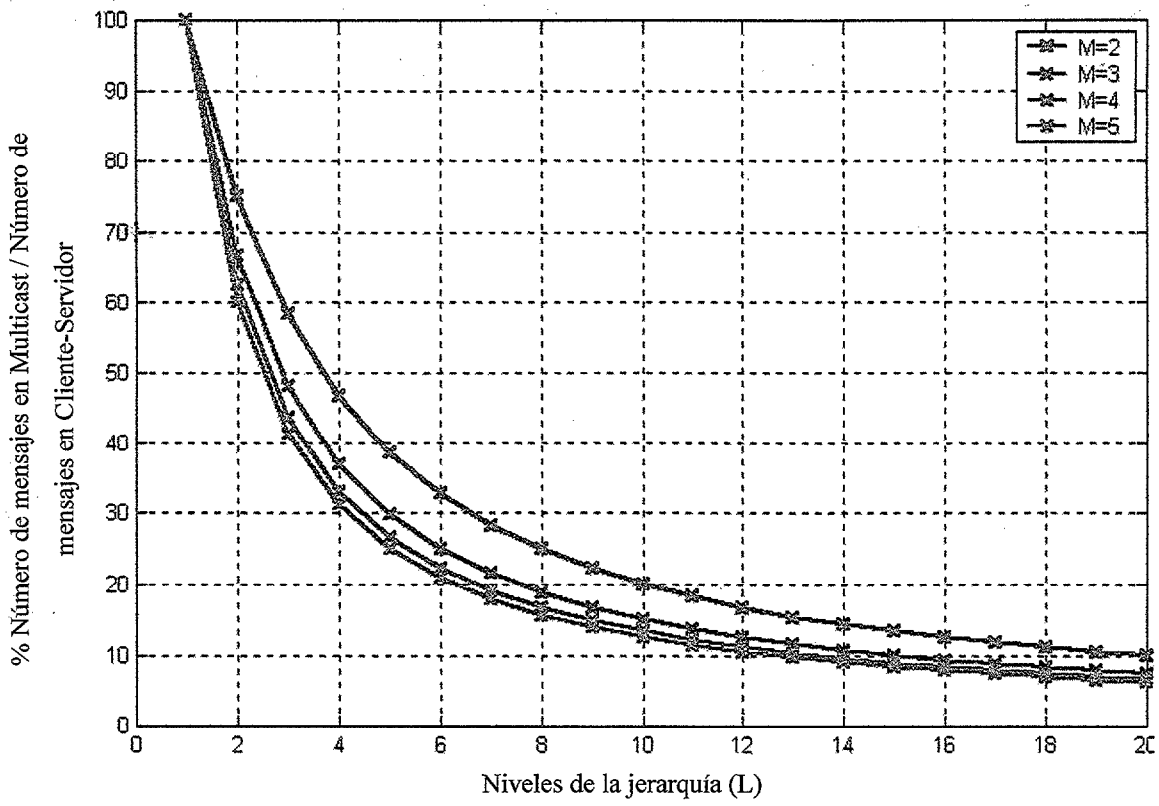


Figura 3.4: Mejora relativa con M=2 hasta M=5

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Capítulo 4. Protocolos diseñados

En este apartado vamos a explicar los protocolos diseñados y como se han implementado con el ANTS. En primer lugar se explican los protocolos de broadcast diseñados, que son el de producto distribuido de matriz por matriz y el de Ping con broadcast. Acto seguido se explican los dos protocolos de multicast diseñados, que son el de Multicast en Tiempo Real y el de Multicast para Edición Cooperativa. Para cada uno se explica primero el protocolo y luego se dan algunos aspectos interesantes de la implementación.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

4.1 Introducción

En este capítulo presentamos el conjunto de protocolos diseñados, en los que se ha empleado fundamentalmente la técnica de broadcast y de multicast. A continuación resumimos los nombres de estos protocolos y las acciones que llevan a cabo, clasificándolos por unicast y multicast.

- Protocolos broadcast. Matriz por Matriz: Multiplica dos matrices disponiendo de un nodo maestro y n nodos esclavos. PingBroadcast: Protocolo que testea las estaciones presentes en una red poco fiable.

- Protocolos multicast: Multicast en Tiempo Real: Protocolo multicast con restricciones de tiempo real para el descarte de los paquetes obsoletos. Multicast para Edición Cooperativa: Protocolo multicast para el mantenimiento de la coherencia en la Edición Cooperativa de un documento.

Para cada protocolo explicamos las ideas generales de como se diseñó, presentamos los gráficos de tiempo, explicamos varias versiones de los mismos y al final explicamos los aspectos interesantes de la implementación de los mismos usando el ANTS.

4.2 Protocolos broadcast

El propósito de este apartado es explicar los dos protocolos unicast diseñados y dar gradualmente, una idea de cómo se programa con el ANTS. Por lo tanto, los protocolos están ordenados en orden creciente de complejidad, tanto en este apartado como en el 4.2.

4.2.1 Protocolo de Producto Distribuido de Matriz por Matriz

Uno de los ejemplos más utilizados en arquitecturas paralelas es el cálculo del producto de dos matrices. Por eso, y como primer ejemplo de protocolos de unicast, vamos a explicar el protocolo que hemos realizado para afrontar este ejemplo.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

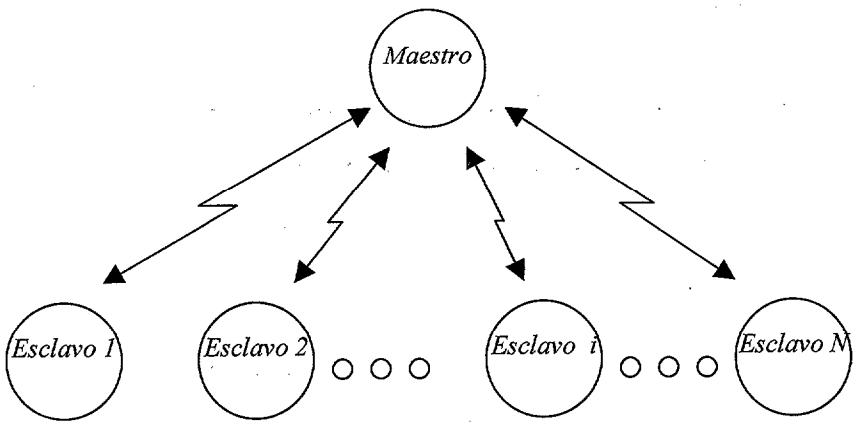


Figura 4.1: Arquitectura del sistema

La arquitectura que se va a utilizar para realizar este cálculo distribuido se muestra en la Figura 4.1. Como puede verse, se trata de una estación *maestro* que distribuye el trabajo y recolecta los resultados. Un conjunto de estaciones *esclavos* que se encargan de realizar el cálculo que se les ha encomendado.

En el *maestro* centraliza el control y la planificación del cálculo distribuido. Por lo tanto, el *maestro* asignará tareas a los *esclavos*, recogerá resultados de éstos y gestionará la utilización de la arquitectura.

En el diseño de este protocolo se ha mantenido siempre presente la premisa de la aplicabilidad a una red de nodos inalámbricos, con la problemática que explicamos en el apartado 1.3. Por lo tanto, en el capítulo de conclusiones se hará mención a este aspecto.

Explicación del protocolo

Partimos de que vamos a realizar el producto de una matriz A por una matriz B, ambas matrices cuadradas de dimensión *lado x lado*. El resultado de dicho producto es una matriz de igual dimensión que llamamos C. En la figura 4.2 se muestra esto gráficamente.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

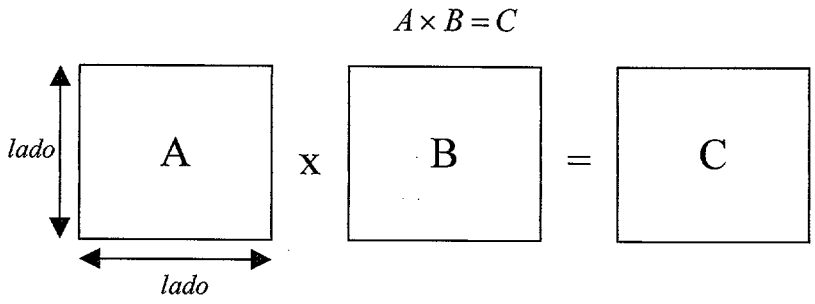


Figura 4.2: Producto de matriz por matriz

Para realizar este cálculo de manera distribuida se siguen los siguientes pasos:

Inicialmente se supone que el maestro contiene los datos de las matrices A y B y finalmente almacenará el resultado en C.

1) El *maestro* hace un broadcast de la matriz B a todos los *esclavos*.

2) El *maestro* va repartiendo entre todos los *esclavos* las filas de la matriz A de manera ordenada.

3) Cada *esclavo*, cuando ha finalizado el cálculo que se le ha solicitado, envía la fila resultado al *maestro*.

Hay que resaltar los siguientes detalles. El *maestro* envía a cada *esclavo* una fila de A. Sigue repartiendo filas hasta que las haya repartido todas. Como el número de *esclavos* (*nesclavos*) puede ser menor que el número de filas de la matriz A (*lado*), cada *esclavo* puede calcular más de una fila de la matriz C. El caso contrario también es posible, es decir que *lado* sea menor que *nesclavos*, en cuyo caso algunos *esclavos* no serán utilizados para realizar dicho cálculo.

Una vez que un *esclavo* recibe la matriz B, cada vez que reciba una fila de la matriz A realizará el cálculo y retransmitirá la fila de C correspondiente. Por lo tanto sólo es necesario difundir la matriz B una vez porque todos los *esclavos* ya la tendrán en lo siguiente. De esta manera sería posible realizar múltiples productos de diferentes matrices A, A', A'', etc., con la misma matriz B enviando sucesivamente las filas de aquellas con solo un broadcast inicial de B.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

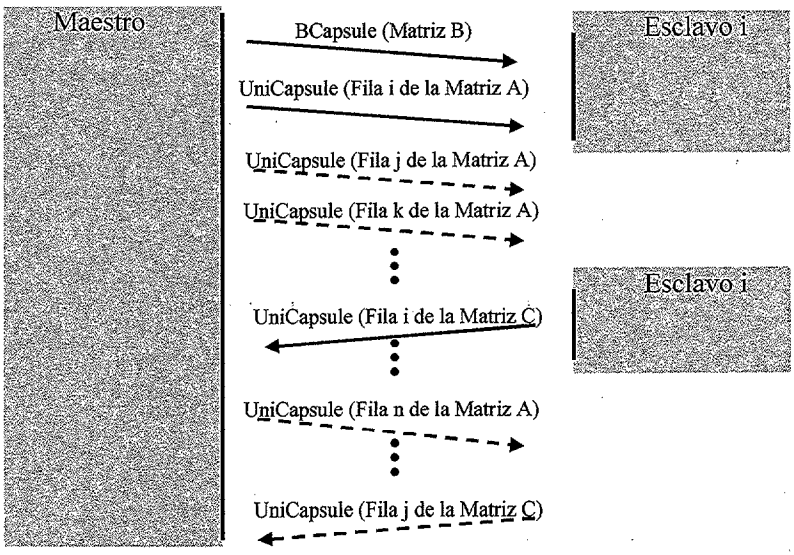


Figura 4.3: Diagrama de paso de mensajes en Matriz por Matriz

A priori, aunque la transmisión por parte del *maestro* de las filas de la matriz A se realiza en orden, no hay restricción de recibir en orden las filas de la matriz C, por lo que es posible tener una configuración de *esclavos* con diferentes capacidades de cálculo cada uno.

En la figura 4.3 se muestra el diagrama de paso de mensajes de este protocolo.

Detalles de la implementación

Ahora procedemos a la explicación de la implementación de este protocolo utilizando el ANTS. Como explicábamos en el capítulo 2, la implementación del protocolo en el entorno ANTS consta de dos niveles de programación: El nivel de enrutamiento y nivel de aplicación. El nivel de enrutamiento se programa básicamente en el diseño de las cápsulas utilizadas para realizar este protocolo. El nivel de aplicación se programa en las aplicaciones que se diseñen para la realización de este protocolo.

A continuación y a modo de ejemplo comentamos algunos fragmentos de código del prototipo programado.

En primer lugar hay que destacar que para la realización de este protocolo se llegó a las siguientes conclusiones en la etapa de diseño:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

1) En lo que al nivel de enrutamiento se refiere se crearán dos tipos de cápsula, una para el broadcast de la matriz B y otra para el unicast de las filas de la matriz A y de las filas resultado (matriz C).

2) Respecto al nivel de aplicación se diseñarán dos aplicaciones, una aplicación para el nodo *maestro* y otra para cada uno de los *esclavos*.

A continuación pasamos a describir cada uno de estos cuatro elementos sin entrar en mucho detalle. En el Anexo I se muestra el código fuente Java de todos los prototipos diseñados. Además en el CD adjunto a este Proyecto Fin de Carrera se muestran en formato de hipertexto, toda la documentación, incluyendo los ficheros fuente.

Cápsulas

En las Figuras 4.4 y 4.5 se muestra un fragmento de código de cada uno de los dos tipos de cápsula que hemos diseñado. En concreto son los fragmentos correspondientes al método *evaluate*. Este método programa el enrutamiento del protocolo. Cuando una cápsula llega a un nodo activo, éste la enruta ejecutando su método *evaluate()*.

Bcapsule. Cápsula de broadcast que se utiliza en el primer paso del protocolo para difundir la matriz B a todos los nodos *esclavos*. Su enrutamiento es sencillo (Figura 4.4).

```
public boolean evaluate(Node n) {
     if (n.getAddress()==getSrc())
          return n.sendToNeighbors(this);
     else return n.deliverToApp(this, dpt);
  }
```

Figura 4.4. Método *evaluate()* de la cápsula Bcapsule

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Si la dirección del nodo actual coincide con la dirección origen de la cápsula se usa un enrutamiento (se envía esta cápsula a todos los nodos *vecinos* del actual), en caso contrario se usa otro diferente (se pasa la cápsula a la aplicación). Esto es, cuando llega una cápsula del tipo BCapsule a un nodo activo, puede pasar dos cosas. La primera es que aún esté en el nodo origen del broadcast (*maestro*), en cuyo caso se procede a difundir la cápsula a todos los *esclavos*. Si no es así, debe estar en algún nodo *esclavo* y hay que pasar la cápsula a la aplicación para que extraiga la información que porta, la matriz B en este caso. Nótese que en este protocolo no hay nodos activos intermedios entre los nodos *esclavos* y el nodo *maestro*.

UniCapsule. Cápsula de unicast que utiliza tanto el *maestro* como el *esclavo,* el primero para transmitir las filas de la matriz A y el segundo para transmitir las filas de la matriz C (resultados parciales). El enrutamiento de esta cápsula está sujeto también a una condición. En este caso la condición es que la dirección del nodo actual coincida con la dirección destino de la cápsula, en cuyo caso se pasa la cápsula a la aplicación para que extraiga la información. En el caso contrario se enruta la cápsula de manera estándar hacia su destino.

En la Figura 4.5 se puede apreciar claramente este detalle. Nótese que en este primer protocolo la programación en el aspecto de enrutamiento es muy sencilla. Es decir, que la mayor carga de este protocolo es extremo a extremo, a nivel de aplicación.

Aplicaciones

A continuación procedemos a explicar la programación extremo a extremo de este protocolo. En la etapa de prototipado de este protocolo con el ANTS se llegó a la conclusión de que eran necesarias dos tipos de aplicación: La que ejecuta el nodo activo del *maestro* y la de los nodos activos de los *esclavos*.

Las dos aplicaciones disponen de una interfaz gráfica que se utiliza para la presentación de resultados, y para asegurar que el funcionamiento del sistema es correcto. En las Figuras 4.8 y 4.9 se muestran sendas imágenes con el aspecto de las dos interfaces.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Cada aplicación en el ANTS tiene tres aspectos programables:

- La inicialización del thread (método *start()*) que configura la interfaz gráfica e inicializa todo lo que sea necesario.
- La rutina del thread propiamente dicho (método *run()*).
- El método que se ejecuta cuando se da un evento de llegada de una cápsula (método *receive()*) y soporta la mayor parte de programación.

MApplication. Esta es la aplicación que se ejecuta sobre el nodo maestro. Es la aplicación que se encarga de distribuir el trabajo entre los esclavos, recolectar los resultados y comprobar que son correctos. Esta última función tiene sentido en un experimento. En una aplicación real de cálculo distribuido, lo que se pretende es precisamente evitar hacer los cálculos localmente.

En las Figuras 4.6 y 4.7 se muestran dos organigramas del funcionamiento de esta aplicación. El primero corresponde al método *receive()*. El segundo al método *run()*. Obviamos el método *start()* por no aportar nada interesante dado que es sólo la configuración de la interfaz. En el Anexo I se puede consultar el código fuente Java.

CApplication. Esta es la aplicación que se ejecuta en los nodos esclavos. Es la aplicación encargada de recoger la matriz B cuando es difundida por el maestro, y a partir de entonces, en el momento que reciba una fila de la matriz A, calculará la fila resultado y la enviará al maestro.

En la Figura 4.10 se muestra el organigrama del método *receive()*. Los métodos *start()* y *run()* se obvian porque carecen de interés. Como siempre, el método *receive* se ejecuta cuando se da el evento de llegada de una cápsula. Esta cápsula puede ser de broadcast de la matriz B (BCapsule) o de unicast de una fila de la matriz A (UniCapsule).

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
public boolean evaluate(Node n) {
    if ( n.getAddress() == getDst())
      return n.deliverToApp(this, dpt);
    else
      return n.routeForNode(this, getDst());
  }
```

Figura 4.5: Método *evaluate* de la cápsula UniCapsule

Dos variables definen si se ha recibido la matriz B y una fila de A. Cuando es así se realiza el cálculo y se envían los resultados. En el código del Anexo I se ha dispuesto que sea necesario transmitir de nuevo la matriz B para cada nueva multiplicación. Sin embargo se deja indicado como hacer que sea necesaria sólo una transmisión de la matriz B para sucesivas multiplicaciones. Se trata de simplemente, no poner a falso la bandera de Matriz B recibida una vez que se transmiten los resultados.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

ULPGC ETSIT BIBLIOTECA DE TELECOMUNICACIONES

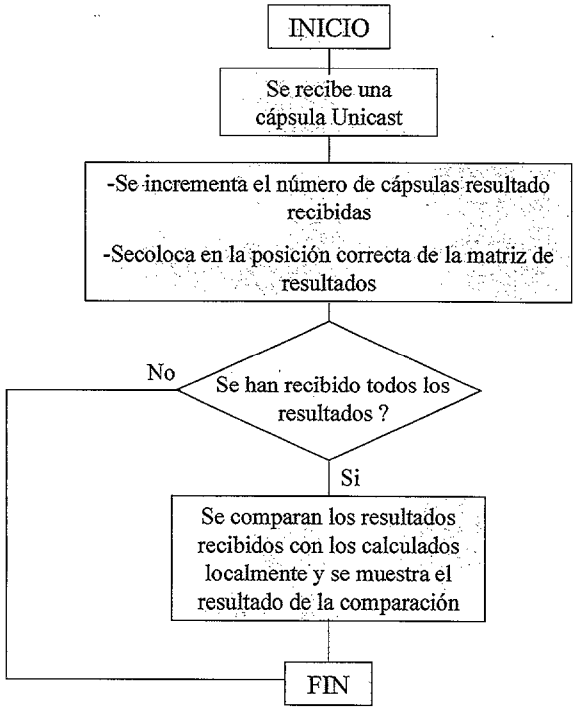


Figura 4.6: Método receive() que se ejecuta como resultado de la recepción de una cápsula en la MApplication

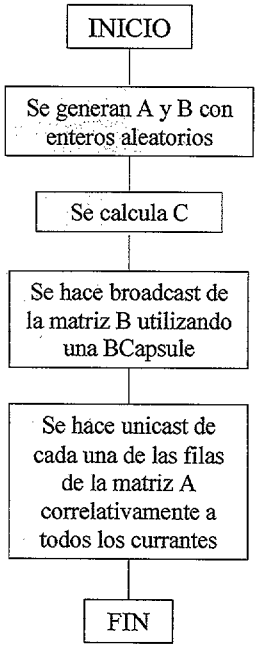


Figura 4.7: Método run() que se ejecuta cuando se lanza el thread de la MApplication

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

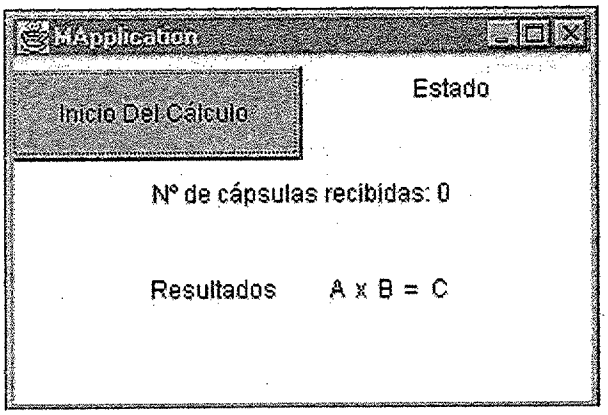


Figura 4.8: Interfaz gráfica de la MApplication

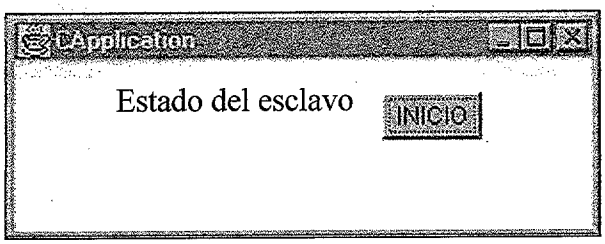


Figura 4.9: Interfaz gráfica de la CApplication

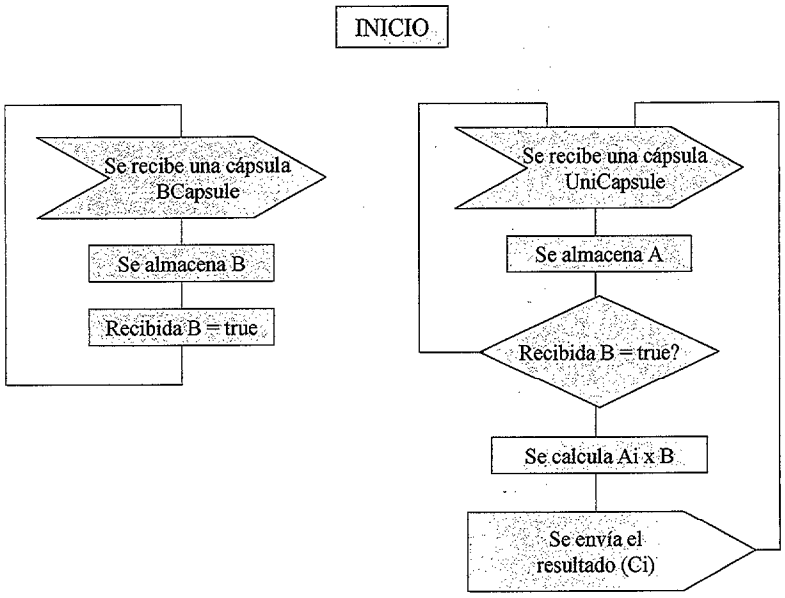


Figura 4.10: Metodo receive() de la CApplication

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

4.2.2 Protocolo de *PingBroadcast*

Actualmente se lleva a cabo un esfuerzo importante en las Redes de Area Local Inalámbricas (WLAN's). Las características de estas redes requieren un tratamiento diferente de las redes cableadas.

La comunidad científica pretende aprovechar las importantes ventajas de estas redes, tales como movilidad y configurabilidad dinámica. Al mismo tiempo se buscan soluciones a los problemas que presentan. Estos son el pequeño y variable ancho de banda disponible, la susceptibilidad de la pérdida temporal de los enlaces y la general escasez de recursos en los equipos inalámbricos.

En este contexto, presentamos un nuevo protocolo que puede ser de gran utilidad. Se trata de un protocolo de *ping* para la detección de las estaciones (e indirectamente de los enlaces) disponibles en una WLAN. Aunque en principio ha sido diseñado para este tipo de LAN, nada impide su utilización en redes de cualquier tipo, en las que la fiabilidad de los enlaces no sea muy buena.

Nuevamente hemos procedido a la implementación de este protocolo utilizando el ANTS. Por lo tanto estamos en el nivel de aplicación de la torre de protocolos TCP/IP. Según los resultados que se obtengan de este experimento podría ser recomendable la implementación del mismo protocolo en un nivel mucho más bajo. Incluso podría estudiarse la posibilidad de realizarlo en hardware, lo que daría rendimientos superiores.

Protocolo de* PingBroadcast *versión 1

En la Figura 4.11 se aprecia una representación de la topología de la red usada. En principio, dado que se comparte el medio de comunicación es equivalente a una red Ethernet como la de la Figura 4.12.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Figura 4.11: topología típica de una WLAN

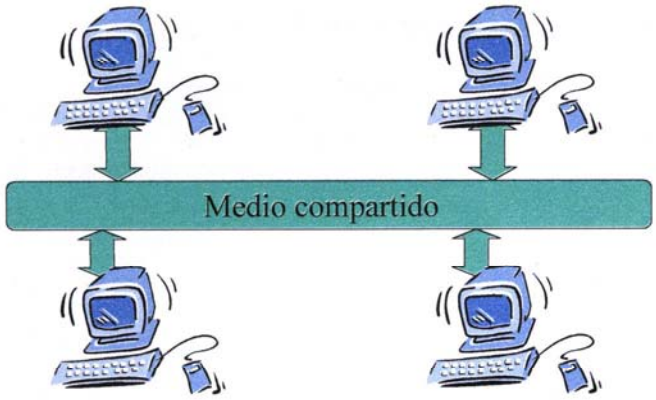


Figura 4.12: Equivalente a la topología de la WLAN

A continuación pasamos a describir la primera versión del protocolo diseñado. Suponemos una situación en la cual a cualquier nodo inalámbrico le puede interesar conocer, en un momento determinado, qué estaciones están presentes en la red en la que se encuentra. Otras aplicaciones pueden requerir este servicio, por ejemplo, una aplicación de cálculo distribuido a la que interesa saber en todo momento con qué estaciones cuenta.

A priori, la estación que desea realizar este *ping* no tiene confianza en la disponibilidad de ninguna de las otras estaciones. Por lo tanto es necesario que inicialmente ese *ping* se dirija a todas las estaciones presentes en la red que puedan *oírlo*. Es decir, al principio de este protocolo se hace necesario realizar un *broadcast* del *ping*.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

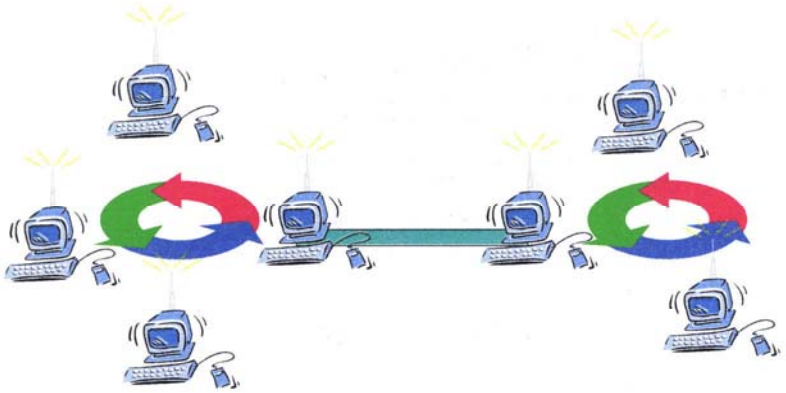


Figura 4.13: Topología con varios segmentos

Además, podemos imaginar configuraciones como la de Figura 4.13, en las cuales, no sería suficiente con un único broadcast del *ping*. Esto es así porque hay estaciones que actúan como puente con otros segmentos de esa LAN.

Partiendo de estos requisitos hemos diseñado el protocolo que denominamos de *PingBroadcast* cuyo funcionamiento se explica seguidamente, en base a una serie de pasos:

Una estación inicia la petición de *ping* haciendo un broadcast de la misma en el medio de comunicación.

1) Las estaciones presentes en el medio compartido (incluida la estación *origen del ping*) escuchan la petición de *ping*.

2) Comprueban no haber recibido una petición de *ping* con el mismo origen en un tiempo muy cercano.

   2.1) Si se cumple la anterior condición, retransmite la petición de *ping* haciendo un broadcast nuevamente a todos los vecinos. Además envía una cápsula unicast de ACK al *origen del ping* con su dirección de nodo.

   2.2) Si ya existía una petición de *ping* recibida en un tiempo muy cercano al actual, esa nueva petición se descarta.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

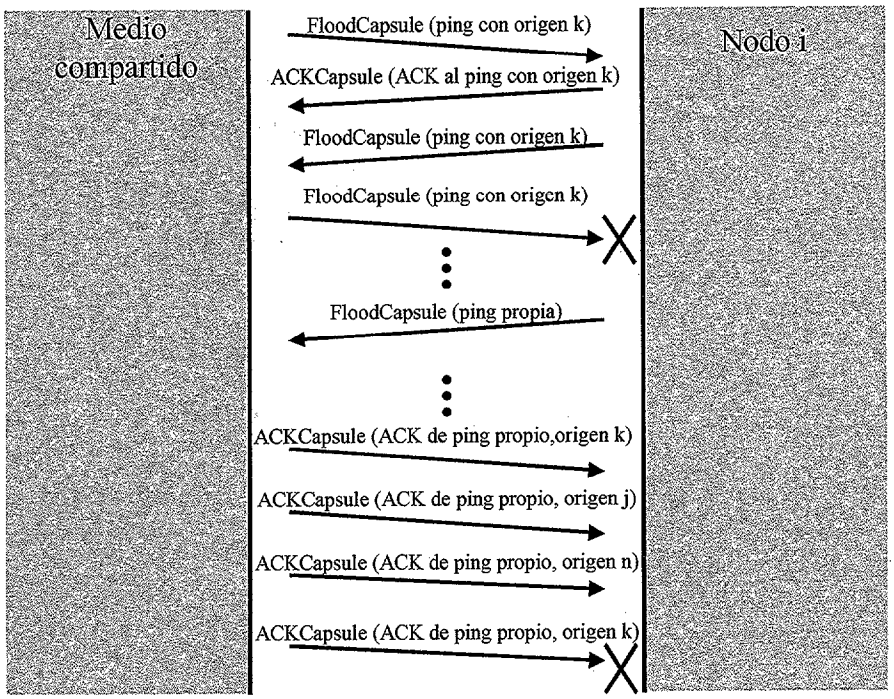


Figura 4.14: Diagrama de paso de mensajes del protocolo de *PingBroadcast*, versión 1

Este protocolo tiene dos etapas en su funcionamiento. En la primera etapa se inunda la red con un conjunto de paquetes de petición de *ping*, todos con la información del origen del mismo. En la segunda etapa se produce la respuesta de cada nodo que recibió la petición de *ping*. De esta manera, el nodo *origen del ping* adquiere el conocimiento de qué nodos están presentes en la red en ese momento. En la figura 4.14 se muestra un diagrama de paso de mensajes explicativo.

Por sencillez y ligereza, el mecanismo de inundación funciona de manera que cuando un nodo recibe un paquete de petición de *ping*, lo difunde además a todos sus vecinos. Para evitar el colapso de la red por la reproducción de estos paquetes de petición de *ping* se creó el siguiente mecanismo:

1) Cada cápsula de *ping* que se recibe se pasa al nivel de aplicación.

2) Se observa la diferencia entre el tiempo actual y el tiempo en que se recibió un paquete de *ping* del mismo origen.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

2.1) Si la diferencia es inferior a un margen de tiempo previamente fijado (*periodoRefresco*) se descarta la cápsula. Debe de tratarse de un *eco* de un *ping* ya recibido.

2.2) En caso contrario se acepta como válido, se anota su origen y el tiempo y se retransmite a los vecinos.

Ese margen de tiempo es función de la red en la que nos encontramos. Debe ser lo suficientemente pequeño como para que un mismo nodo pueda realizar peticiones de *ping* con relativa frecuencia, y lo suficientemente grande como para que la misma petición replicada por los nodos vecinos ("ecos" de una petición) no se confunda con una nueva petición. La Figura 4.15 trata de aclarar esta problemática.

Implementación del protocolo de* PingBroadcast *versión 1

Tras la etapa de diseño de este protocolo se llego a las siguientes premisas antes del prototipado del mismo:

- Se conciben dos tipos de cápsula, un tipo para la inundación (*FloodCapsule*) y otro para la respuesta al *ping* de cada nodo (*ACKCapsule*).
- Será necesario un solo tipo de aplicación (PingBApplication) dado que todos los nodos deben tener la capacidad de iniciar el *PingBroadcast*.

Seguidamente pasamos a explicar con un poco más de detalle cada uno de estos elementos. Para consultas más intensivas en el Anexo I se provee los ficheros fuente Java.

Cápsulas

En las Figuras 4.16 y 4.17 se muestra un fragmento de código, de cada uno de los dos tipos de cápsula. Son los fragmentos correspondientes al método *evaluate*. Es en este método precisamente donde se captura la programabilidad del enrutamiento en el protocolo. Cuando una cápsula llega a un nodo activo, éste la enruta ejecutando el método *evaluate* de la misma.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

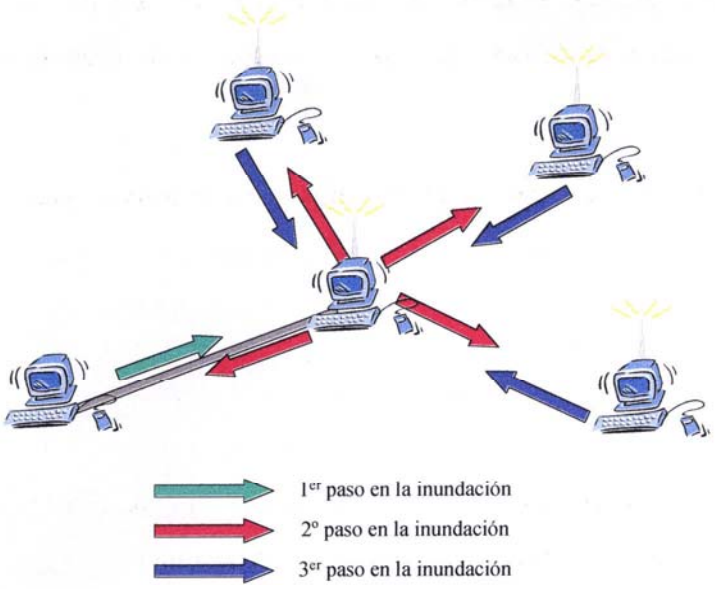


Figura 4.15: Inundación de la red por una petición de ping

```
public boolean evaluate(Node n) {
     if (n.getAddress()==getSrc())
          return n.sendToNeighbors(this);
     else return n.deliverToApp(this, dpt);
  }
```

Figura 4.16: Método evaluate de la cápsula *FloodCapsule* v1

```
public boolean evaluate(Node n) {
    if ( n.getAddress() == getDst())
      return n.deliverToApp(this, dpt);
    else
      return n.routeForNode(this, getDst());
  }
```

Figura 4.17: Método evaluate de la cápsula *ACKCapsule* v1

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Como se puede apreciar, también en este caso la carga de este protocolo en el nivel de enrutamiento es mínima, es decir, casi todo el protocolo se programa en el nivel de aplicación en este caso.

FloodCapsule. Es el tipo de cápsulas que se emplea en la primera fase del protocolo de *PingBroadcast*. La rutina de encaminamiento es sumamente sencilla. Se trata de una condición: si nos encontramos en el nodo origen de esa cápsula entonces se clona y se envía a todos los nodos vecinos del nuestro; en caso contrario se pasa la cápsula a la aplicación del nodo activo.

ACKCapsule. Es el tipo de cápsula por medio de la cual, un nodo activo que haya recibido una petición de *ping*, responde al *origen del ping*. El enrutamiento en este tipo de cápsulas es aún más sencillo que en el anterior. Es un enrutamiento estándar IP y se muestra en la Figura 4.17.

Aplicaciones

Dado que existe una simetría en cada nodo activo de la función que debe desempeñar esta aplicación, es necesaria sólo una clase: *PingBApplication*. En la Figura 4.18 se muestra la interfaz gráfica de esta aplicación.

PingBApplication. Es la aplicación que se ejecuta en cada nodo. El protocolo se programa en esta en los siguientes métodos:

1) *receive()*: Esta rutina se dispara cuando se produce un evento de llegada de una cápsula al nodo activo. En la Figura 4.19 se muestra un organigrama del funcionamiento de este método cuando la cápsula que se recibe es una *FloodCapsule*.

2) *run()*: Método que se ejecuta cada vez que se lanza el thread de la aplicación. Lo único que se hace en este método es enviar una cápsula de *ping* cada vez que se inicia el thread.

3) *HandleEvent()*: método que recoge la respuesta a los dos posibles eventos de esta aplicación. Estos dos eventos son el acto de activar alguno de los dos botones presentes en la interfaz de esta aplicación. El botón de inicio sirve para inicializar la tabla de

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

origen del ping conocido. El otro botón simplemente lanza el thread de la aplicación, que enviará una cápsula de inundación.

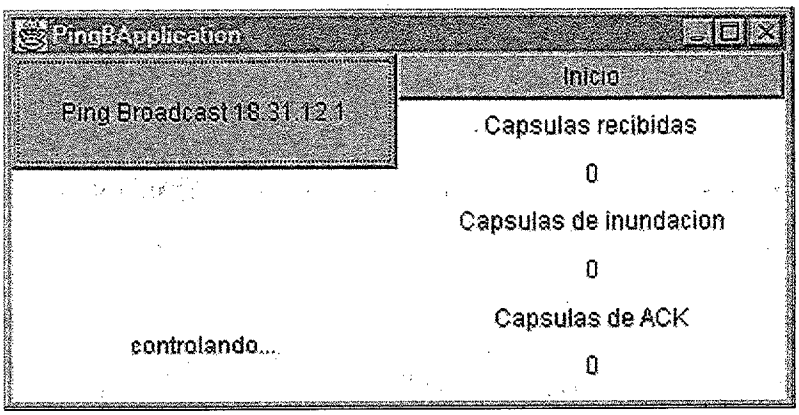


Figura 4.18: Interfaz de la aplicación

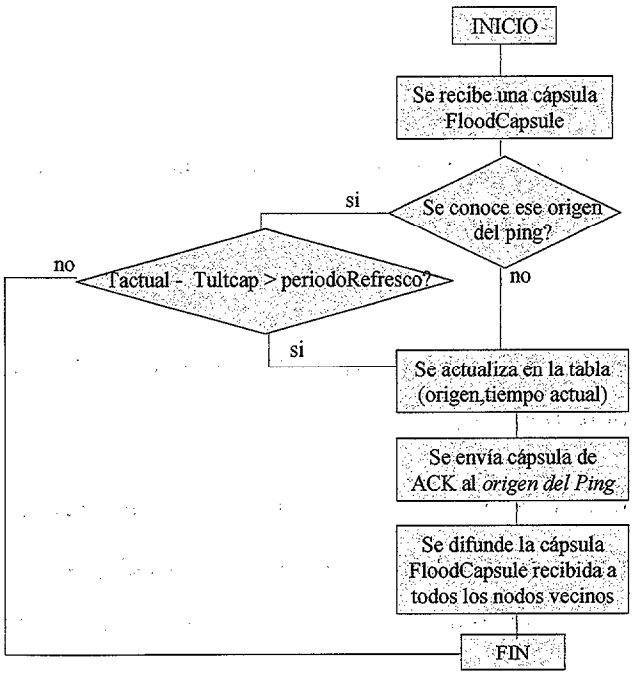


Figura 4.19: Método receive() cuando se recibe una *FloodCapsule*

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Protocolo de PingBroadcast versión 2

La versión del protocolo de *PingBroadcast* explicada arriba demuestra ser útil. Sin embargo, durante las pruebas realizadas con estos prototipos comprobamos que es posible realizar una mejora significativa en el rendimiento del mismo, mejorando su velocidad. Se trata de aprovechar la posibilidad de procesamiento a nivel de red que las redes activas nos brindan y trasladar la discriminación de los *pings* no válidos al nivel de red.

Exactamente, el nuevo esquema del protocolo es el siguiente:

1) Cada *ping* que lanza una aplicación lleva un *sello*, un entero aleatorio que lo distingue de los demás.

2) Todos los nodos cuentan con una cola FIFO de longitud MAX_SELLOS. En la cola se guarda información de los últimos *pings*. Cada elemento almacena el *sello* de un *ping* determinado.

3) Cuando un nodo recibe un *ping* se busca en la cola si hay información almacenada de algún *ping* de ese *origen del ping*.

   3.1) Si no existe información de *ping* alguno con ese *sello* se da por *ping* válido. Se desplaza la lista y se introduce su información, se difunde a todos los vecinos y se pasa a la aplicación.

   3.2) En caso de que si exista en la lista algún *ping* con ese *sello* se descarta el *ping* por tratarse de un eco de un *ping* ya registrado.

Esta versión del protocolo permite ahorrarnos tener que pasar cada *ping* al nivel de aplicación para decidir su validez. El mismo nivel de red lo hace, con el consiguiente ahorro en tiempo que eso supone.

Existe un detalle de esta versión que es importante cuidar: el tamaño de la cola FIFO (MAX_SELLOS). Si este valor es pequeño, *pings* ya recibidos y todavía en el medio pueden ser desplazados fuera de la cola, dándolos por nuevos *pings* si vuelve a recibirse algún "eco". Esto podría desembocar en una congestión de la red con los paquetes de inundación.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Por otra parte, si el tamaño de la cola es muy grande, aumentamos el tiempo necesario en el nivel de red para comparar cada nuevo *ping* recibido con todos los sellos de pings previos. Esto provocaría una disminución en el rendimiento óptimo del protocolo.

Implementación del protocolo de* PingBroadcast *versión 2

La implementación de esta versión del protocolo de *PingBroadcast* es muy similar a la anterior, con la única excepción de las cápsulas *FloodCapsule*. La rutina de encaminamiento de estas es diferente. El procesamiento que se realiza en las aplicaciones *PingBApplication* a las cápsulas *FloodCapsule*, también es ligeramente diferente, ya que no son las aplicaciones quienes se encargan ahora de descartar los *pings* no válidos.

```
public boolean evaluate(Node n) {
      W_JAI m = (W_JAI)n.getCache().get(group);
      boolean encontrado = false;//leemos la cache
      if (m!=null){//si existe registro en la cache
        for (int i=0;m.ai1!=null&&i<m.ai1.length;i++)
          if (m.ai1[i] == sello)
            { encontrado = true; break; }
      }else{//si no existe registro en la cache
        m = new W_JAI(); //creamos uno
      n.getCache().put(group,m,TIEMPO_MAX_NODE_CACHE);
        int[] aux = new int[MAX_SELLOS];
        for (int j=0; j<MAX_SELLOS; j++)
        aux[j] = 0;
        m.ai1 = aux;
      }
      if (!encontrado){ //si no se tenía ese sello
desplazamos los existentes
        int[] aux = new int[MAX_SELLOS];
        System.arraycopy(m.ai1,0,aux,1,MAX_SELLOS-1);
        aux[0] = sello;
        m.ai1 = aux;
        n.deliverToApp(this,dpt);//lo pasamos a la ap.
        n.sendToNeighbors(this); //y lo propagamos
      }
    return true;
  }
```

Figura 4.20: Método evaluate de la cápsula *FloodCapsule* v2

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

En la figura 4.20 se muestra el código del método *evaluate()* del código de las cápsulas de inundación (*FloodCapsule*). Como puede apreciarse, el precio de la mejora introducida en el protocolo es un incremento sustancial en la complejidad del código de las cápsulas. El código completo de esta versión se encuentra en el Anexo I de esta memoria. Al igual que para todos los protocolos diseñados en este Proyecto Fin de Carrera, también se puede encontrar toda la documentación y código fuente en hipertexto en el CD adjunto.

4.3 Protocolos multicast

El propósito de este apartado es explicar los dos protocolos de multicast diseñados. Se explicarán dos versiones de cada uno. En estos protocolos la complejidad es mayor, explotando cada vez más en profundidad las posibilidades del ANTS como herramienta de prototipado de protocolos.

4.3.1 Protocolo de multicast en tiempo real

El propósito de este apartado es presentar un protocolo y su implementación en un prototipo, para el multicast de datos con restricciones de tiempo real. Se presenta en dos versiones. La primera requiere que todos los nodos estén sincronizados perfectamente, por ejemplo mediante el uso de un *GPS* (*Sistema de Posicionamiento Global*). La segunda evita esta limitación con un procesamiento a nivel de red más sofisticado.

También en este protocolo hemos tenido en cuenta la posibilidad de que sea implantado en una WLAN. Ello nos supone las restricciones arriba mencionadas.

Protocolo de Multicast en Tiempo Real versión 1

Tras reflexionar sobre los problemas enunciados en el capítulo 3, propios de las aplicaciones multicast hemos llegado a las siguientes soluciones:

1) Sería deseable concentrar los paquetes que porten información similar, para evitar congestionar la red (*fusing* [2]).

2) Utilizar una memoria cache para almacenar temporalmente los paquetes recientes en los nodos intermedios puede ser positivo por evitar que peticiones de retransmisión y

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

retransmisiones carguen innecesariamente todos los nodos intermedios entre el Receptor y el Transmisor.

3) Creemos necesario implementar una política de desecho de los paquetes que no cumplan las restricciones de tiempo real (*filtrado activo* [2]) en todos los nodos de la red, evitando ocupar ancho de banda con paquetes inútiles.

Partiendo de estas premisas iniciales hemos diseñado el protocolo que a continuación describimos esquemáticamente. Suponemos una situación inicial en la existe un Transmisor, un conjunto de nodos intermedios conectados y un conjunto de nodos Receptores, que serán los que se suscribirán al servicio cuando lo deseen. La figura 4.21 ilustra esta situación inicial.

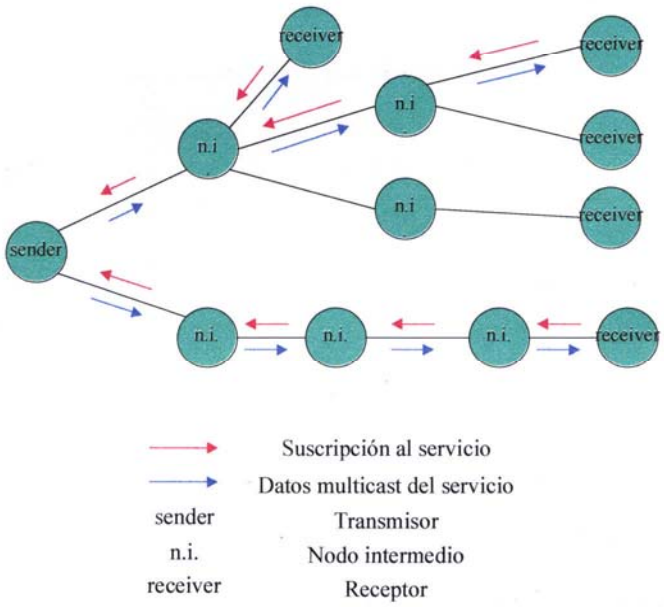


Figura 4.21: Configuración típica de un servicio multicast

Seguidamente exponemos el esquema básico del protocolo:

1) Durante el tiempo en el cual un Receptor quiera recibir los datos del servicio multicast, envía periódicamente peticiones de suscripción al nodo intermedio del que cuelga. En el momento en que cese de enviar las peticiones de suscripción, deja de recibir los paquetes multicast.

2) Los nodos intermedios recolectan las suscripciones y periódicamente envían una petición de suscripción al nodo anterior. Si un nodo intermedio no ha recibido

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

peticiones de ninguno de los nodos que cuelgan de él, él tampoco envía peticiones a niveles superiores.

3) Cada nodo dispone de una cache local en la cual tiene una lista de los suscriptores que cuelgan de él. El nodo Transmisor, una vez iniciado el servicio multicast, reparte los datos del servicio a todas las entradas de la tabla de suscriptores. Cada nodo intermedio hace lo mismo cuando le llegan paquetes del servicio multicast, los distribuye a sus suscriptores.

4) Tanto el Transmisor como cada nodo intermedio disponen también de una cache local en la que almacenan los índices y datos de los paquetes multicast enviados para volver a enviarlos en el caso de recibir una petición de retransmisión. Los almacenan mientras sean útiles teniendo en cuenta las restricciones de tiempo real.

5) Cuando un Receptor detecta la falta de un paquete multicast inmediatamente genera un paquete de *reconocimiento negativo* (NACK), es decir una petición de retransmisión que envía al nodo del que cuelga.

6) Cuando un nodo intermedio recibe una petición de retransmisión comprueba si tiene el paquete solicitado en su cache local. En caso afirmativo envía ese paquete al peticionario haciendo unicast. En caso contrario pone la petición de retranmisión en una lista de paquetes pedidos. Periódicamente cada nodo intermedio revisa esa lista, retransmite los paquetes que tiene en su cache local (que a él mismo le han sido retransmitidos), y fabrica con el resto de las peticiones de retransmisión válidas (las que entren dentro del margen de tiempo real) un paquete NACK que envía al nodo del que cuelga.

7) Por último hay que señalar que en todos los nodos y a nivel de red se hace un *filtrado activo* [2] con los paquetes de datos multicast ordinarios, las peticiones de retransmisión y las retransmisiones. Esto significa que se descarta todo paquete que no entre dentro de un margen de tiempo real que se fija globalmente. Para comprobar esta condición, cada paquete lleva un sello de tiempo del instante en el que fue creado el paquete de datos multicast en el Transmisor.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Creemos que este protocolo es un protocolo multicast eficiente y adecuado para entornos inalámbricos o heterogéneos por las siguientes razones:

1) Las peticiones de suscripción al servicio se realizan entre los Receptores y el nodo del que cuelgan, evitando el congestionamiento de la red por peticiones de suscripción cursadas todas hasta el Transmisor.

2) También se evita congestión en caso de una caída temporal de algún enlace haciendo *fusing* con las peticiones de retransmisión. Si todo un sector de la red necesita la retransmisión de uno o varios paquetes, el nodo intermedio anterior a ese sector sólo necesita un paquete NACK para saberlo. Además retransmite el número de paquetes perdidos a los nodos intermedios que lo piden y no a todos los Receptores que han perdido el servicio temporalmente (los nodos intermedios se encargan de hacerlo a aquellos que lo pidan).

3) Por último, el filtrado activo también contribuye a no malgastar el ancho de banda con paquetes que al final serían desechados por inútiles.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

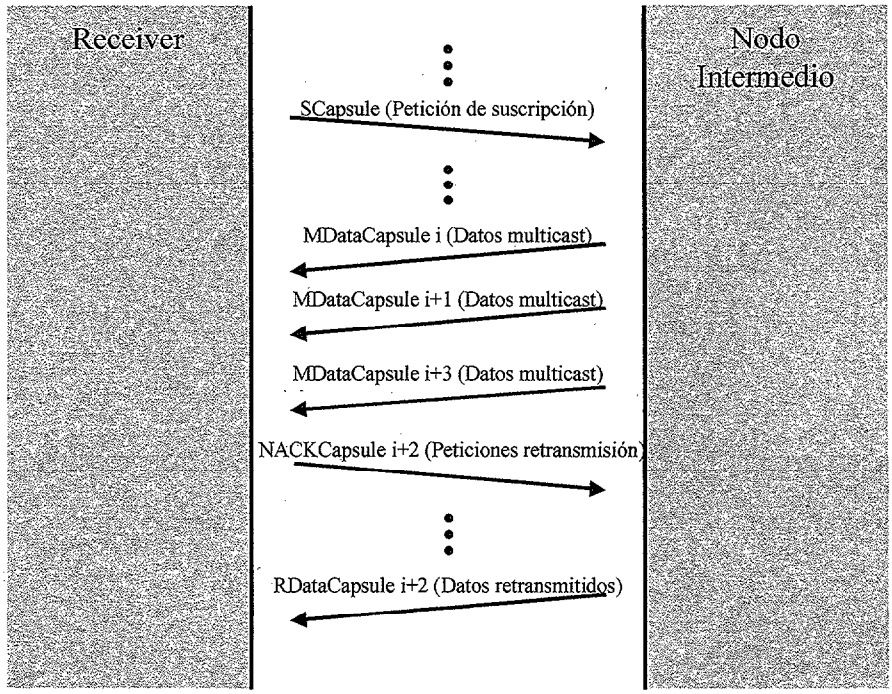


Figura 4.22: Diagrama de paso de mensajes del esquema básico de funcionamiento del Multicast RT v1

En la figura 4.22 se muestra el diagrama de paso de mensajes a modo ilustrativo del funcionamiento básico de este protocolo.

Implementación del protocolo de Multicast en Tiempo Real versión 1

Decidimos realizar el prototipo de este protocolo extendiendo cuatro tipos de cápsulas y tres tipos de aplicación.

Las cápsulas son las siguientes:

- *MDataCapsule*: cápsulas de datos multicast.
- *SCapsule*: cápsulas de suscripción al servicio.
- *RDataCapsule*: cápsulas unicast de retransmisión de datos.
- *NACKCapsule*: cápsulas de no-reconocimiento (petición de retransmisión).

Las aplicaciones son las siguientes:

- *SenderApplication*: aplicación suministradora de los datos multicast.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

- *NodeApplication*: aplicación de los nodos intermedios de la red
- *ReceiverApplication*: aplicación de los nodos terminales, susceptibles de suscribirse al servicio.

Cápsulas

En las Figuras 4.23, 4.24, 4.25 y 4.26 se muestra un fragmento de código, de cada uno de los cuatro tipos de cápsula. Son los fragmentos correspondientes al método *evaluate.* Es en este método precisamente donde se captura la programabilidad del enrutamiento en el protocolo. Cuando una cápsula llega a un nodo activo, éste la enruta ejecutando el método *evaluate* de la misma.

MdataCapsule. Es el tipo de cápsulas que se utiliza para distribuir los datos del sevicio multicast. Cuando se crean estas cápsulas en la aplicación Sender, se les incluye un sello de tiempo. A partir de ese momento, en cada nodo por los que pase esta cápsula, se evalúa la diferencia entre el tiempo actual y el sello de tiempo. Como puede verse en la Figura 4.23, en el caso de que esta diferencia no esté dentro del margen de tiempo real, que es una variable global a la que llamamos *margenRT*, la cápsula se descarta. En caso contrario se clona y se distribuye a todos los nodos suscritos al servicio. Para ello se dispone de una lista de los nodos suscritos en la cache de cada nodo. Además, toda cápsula válida se pasa a la aplicación del nodo por motivos que se explicarán después.

SCapsule. Este tipo de cápsulas se utiliza para mantener a un nodo suscrito al servicio multicast. Dicho de otra manera, se encarga de mantener la entrada del nodo propio en las caches de nodo correspondientes con el fin de recibir un clon de cada cápsula de datos multicast. En la Figura 4.24 se muestra el método *evaluate* de la misma.

Resumidamente podemos decir que el procesamiento que sufren estas cápsulas es el siguiente: En primer lugar se busca en la cache del nodo si existe una entrada para ese grupo de servicio multicast. Si no es así se introduce una nueva entrada. Después, si no nos encontramos en el nodo origen de la cápsula se observa si en la cache se tenía una entrada para el nodo que envía la suscripción. Si no se tenía se añade. Luego se pasan a la aplicación para su procesamiento.

En caso de que estemos en el nodo origen se enrutan hacia el nodo anterior en la red.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
public boolean evaluate(Node n) {
W_JAI m = (W_JAI)n.getCache().get(group, target);
    if (m == null) {
      m = new W_JAI();
      n.getCache().put(group, target, m, 5);
    }
    if (reverse != 0) {//no estoy en el nodo origen
      boolean found = false;
      for (int i = 0; m.ai1 != null && i < m.ai1.length; i++) {
        if (m.ai1[i] == reverse) { found = true; break; }
      }
      if (!found) { //no hay registro para el suscriptor
        int len = (m.ai1 != null) ? m.ai1.length : 0;
        int[] nn = new int[len+1];
        if (len > 0) System.arraycopy(m.ai1, 0, nn, 0, len);
        nn[len] = reverse;
        m.ai1 = nn;
      }
      return n.deliverToApp(this,dpt); //pasamos la cap. a la apli.
    }else {//estoy en el nodo origen
      if (System.currentTimeMillis() - m.j1 < vg.tiempoSuscripcion)
            return true;
      else m.j1 = System.currentTimeMillis();
      reverse = n.getAddress();
      return n.routeForNode(this, target);
    }
}
```

Figura 4.24: Método evaluate de la cápsula *Scapsule v1*

NACKCapsule. Este tipo de cápsulas se utilizan para pedir al nodo anterior en la topología cápsulas que se han recibido. Para ello las cápsulas *MDataCapsule* tienen un índice que permite detectar saltos en la secuencia. Este tipo de cápsulas se componen como una ristra de valores de índice y sellos de tiempo.

Los índices representan las cápsulas que se solicitan. Junto con cada índice se añade el sello de tiempo de la cápsula siguiente correctamente recibida. Los sellos de tiempo se utilizan para descartar las cápsulas NACK obsoletas (las que todos los sellos de tiempo están fuera del *margenRT*). En la Figura 4.25 se muestra el método *evaluate* de las mismas.

Rcapsule. Con este tipo de cápsula se retransmiten siguiendo el enrutamiento IP estándar las cápsulas que se habían solicitado una a una y con el sello de tiempo que puso

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

originariamente el *sender* con el objeto de descartarlas en el caso de que sean obsoletas. En la Figura 4.26 se muestra el método *evaluate* de la misma.

Aplicaciones

A continuación se muestran los tres tipos de aplicación que se ha utilizado para realizar el protocolo diseñado. Para ilustrarlos se presenta en la Figuras 4.27, 4.29, y 4.31 los organigramas simplificados de los métodos *receive()* y *run()*. El primer método sirve para implementar las rutinas que se activan al darse los eventos de llegada de los diferentes tipos de cápsula. El segundo incorpora la programación del thread de cada aplicación.

Hemos realizado el protocolo con tres tipos de aplicación:

- *SenderApplication.* Aplicación del nodo encargado de suministrar el servicio multicast.
- *NodeApplication.* Aplicación que corre en cada uno de los nodos intermedios de nuestra topología.
- *ReceiverApplication.* Aplicación que corre sobre los nodos *Receivers.*

SenderApplication. Esta es la aplicación que se ejecuta para suministrar el servicio multicast y tiene un funcionamiento muy sencillo. El método *run()* se encarga, una vez que se active el control correspondiente, de enviar sucesivamente cápsulas de datos del servicio multicast. A cada *MDataCapsule* se le dota de un índice que permitirá la detección de pérdidas por saltos en la secuencia, y de un sello de tiempo que permitirá detectar la obsolescencia de las cápsulas. Además se guarda una copia de cada cápsula enviada en una lista para permitir su retransmisión durante el periodo de tiempo de validez de las mismas. El método *receive()* contiene la programación que responde a los eventos de llegada de los diferentes tipos de cápsula. En el caso de esta aplicación sólo se reciben cápsulas del tipo *NACKCapsule*. En la figuras 4.27 se muestran los organigramas resumidos de estos dos métodos. En la figura 4.28 se muestra la interfaz de la aplicación.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
public boolean evaluate(Node n) {
  boolean obsoletos = true;
    for (int i = 0; i<npeticiones; i++){
      if (System.currentTimeMillis()-sellotiempo[i]<vg.margenRT){
        obsoletos = false; //vemos si hay peticiones útiles
        break;
      }//if
    }//for
    if (!obsoletos){
      if (n.getAddress() == getSrc())
        //si estamos en el nodo origen
        return n.routeForNode(this, getDst());
      else return n.deliverToApp(this, dpt);
    }
    else {
      return true; //desechamos la cápsula
    }
}
```

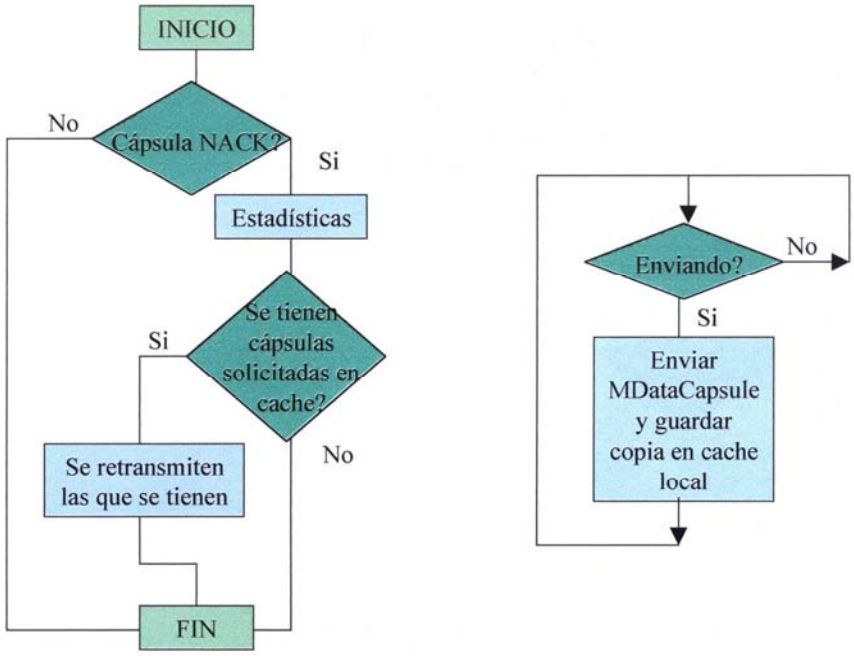
Figura 4.25: Método evaluate de la cápsula *NACKCapsule v1*

```
public boolean evaluate(Node n) {
  if ((System.currentTimeMillis()-sellotiempo)<vg.margenRT) {
  //si la cápsula es válida para RT
      if (n.getAddress()==getDst())
        return n.deliverToApp(this, dpt);
      else return n.routeForNode(this, getDst());
  }else return true;//si la cápsula es obsoleta se desecha
}
```

Figura 4.26: Método evaluate de la cápsula *RDataCapsule v1*

NodeApplication. Este es el tipo de aplicación que se instala sobre los nodos intermedios de la topología de red que utilizamos. En este tipo de aplicación se manipulan los cuatro tipos de cápsula. Además cada nodo dispone de una lista de cápsulas enviadas y de cápsulas perdidas. La de cápsulas enviadas guarda una copia de cada cápsula enviada durante el tiempo en el cual es válida para retransmitirla en caso de ser necesario. La lista de cápsulas pedidas almacena los índices de las cápsulas pedidas por nodos hijos para periódicamente fabricar una cápsula NACK con todas las peticiones que no sean obsoletas y enviarla al nodo anterior de la red.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004



(a) método *receive()* (b) método *run()*

Figura 4.27: Métodos principales de *SenderApplication* en el protocolo de Multicast RT v1

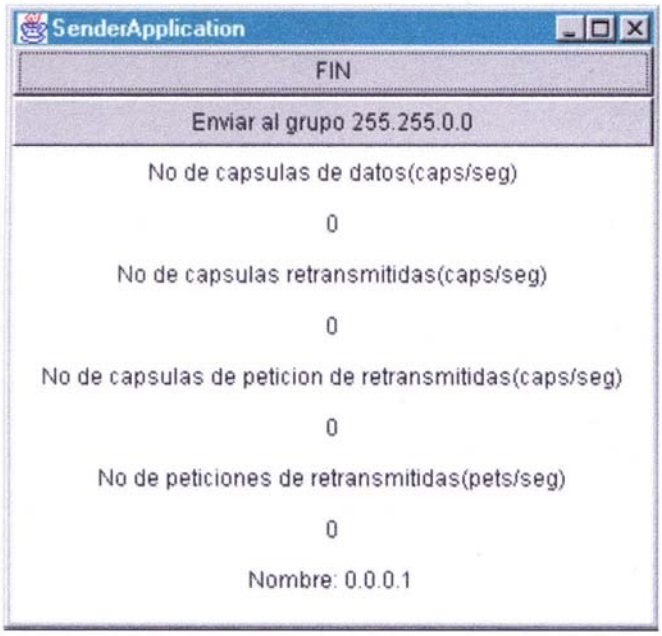
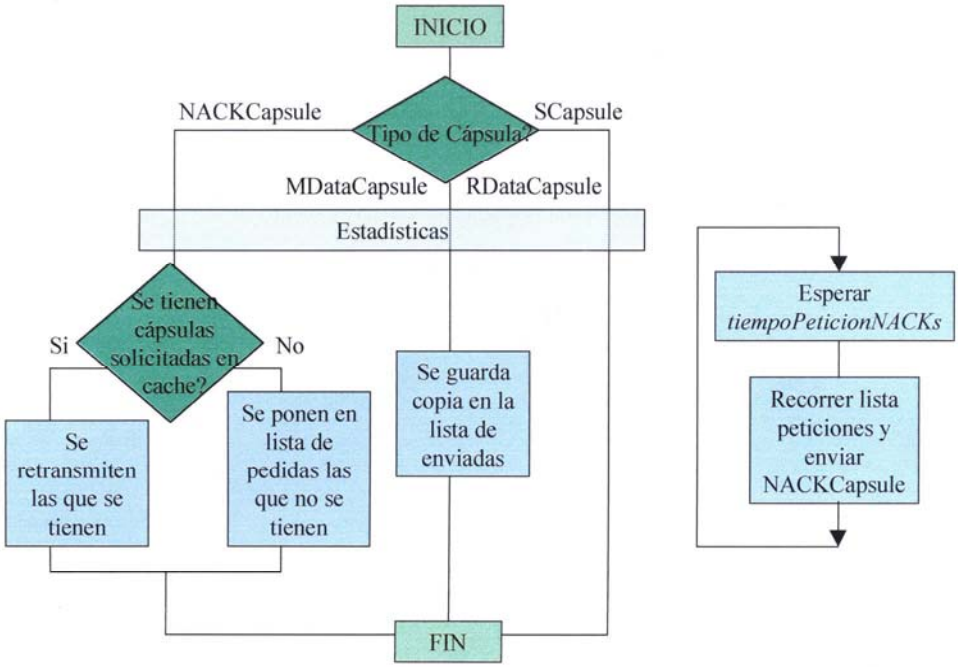


Figura 4.28: Interfaz de la *SenderApplication* en el protocolo de Multicast RT v1

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004



(a) método *receive()* (a) método *run()*

Figura 4.29:métodos principales de *NodeApplication* en el protocolo de Multicast RT v1

El método *run()* se encarga de la fabricación de las cápsulas *NACKCapsule* que son enviadas al nodo anterior. El método *receive()* se encarga de ejecutar las rutinas correspondientes a cada uno de los cuatro tipos de cápsula posibles. En la figura 4.29 se muestran los organigramas simplificados de estos dos métodos. Paralelamente se programa un thread que se encarga de mantener suscrito al nodo al servicio. Su organigrama es el mismo que el que ponemos en el método run() de la *ReceiverApplication* (figura 4.31)). La figura 4.30 muestra la interfaz de esta aplicación.

ReceiverApplication. Esta es la aplicación que se ejecuta en los nodos terminales de la topología de red utilizada, que permite que en un momento determinado el nodo se suscriba al servicio multicast. Envía periódicamente cápsulas *SCapsule* para suscribirse al servicio y a cambio recibe un clon de cada *MDataCapsule* generada por la *SenderApplication*. Cuando se detecta un salto en la secuencia de las cápsulas de datos se genera un paquete *NACKCapsule* con los índices de las cápsulas de datos que faltan y el sello de tiempo del siguiente paquete recibido correctamente, que es enviado al nodo anterior en la topología. Por último, es posible que se reciban cápsulas del tipo *RDataCapsule* retransmitiendo los datos que se han solicitado.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

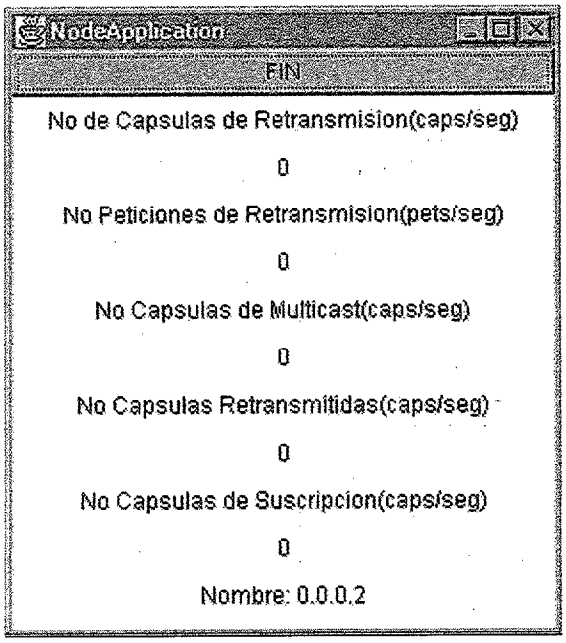


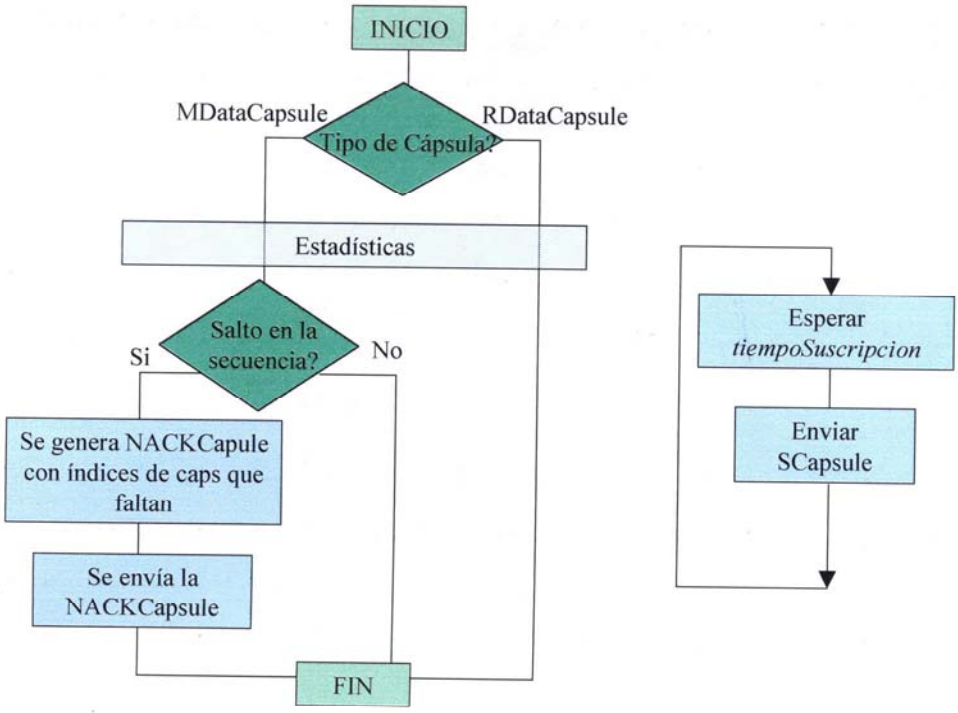
Figura 4.30: Interfaz de la *NodeApplicacion* del protocolo de Multicast en RT v1

En la figura 4.31 se muestran los métodos *receive()* y *run()* de esta aplicación. El método *receive()* se encarga de lanzar las rutinas correspondientes a los eventos de llegada de los tipos de cápsula *MDataCapsule* y *RDataCapsule*. El método *run()* se encarga de enviar periódicamente cápsulas de suscripción al nodo anterior para seguir recibiendo el servicio. En la figura 4.32 se muestra la interfaz de la aplicación.

Protocolo de Multicast en Tiempo Real versión 2

El protocolo anteriormente explicado fundamenta sus restricciones de tiempo real en una comparación entre el tiempo de llegada y el de salida de los paquetes de datos. Esta filosofía esta justificada en un entorno de área local, donde las latencias son relativamente pequeñas, incluso en redes inalámbricas. Sin embargo, si quisiéramos diseñar una aplicación que utilice este protocolo para funcionar en redes de mayor escala, como la Internet, no sería ni viable, ni útil dada la gran latencia de las comunicaciones en este caso.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004



(a) método *receive()*

(a) método *run()*

Figura 4.31: Métodos principales de *ReceiverApplication* del protocolo de Multicast RT v1

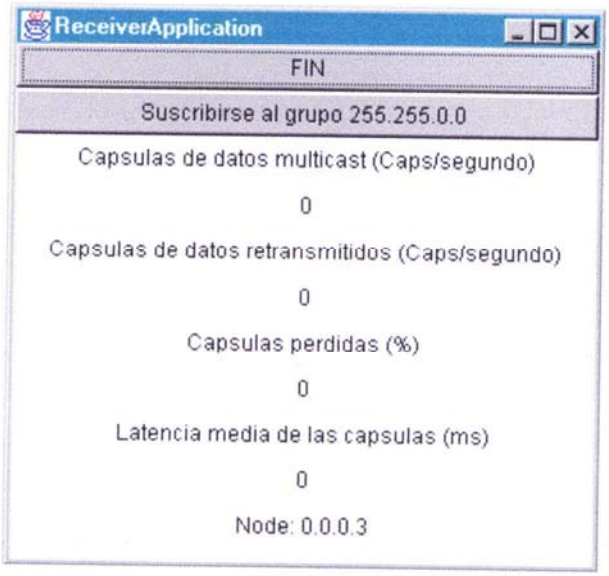


Figura 4.32: Interfaz de *ReceiverApplication* del protocolo de Multicast RT v1

En este tipo de redes no interesa mantener en un rango la diferencia entre el tiempo de salida y de llegada de los datos multicast. Más bien, lo interesante sería que entre paquetes con informaciones contiguas no existiera una latencia excesiva. Dicho de otra manera y con un ejemplo, no nos importa que las tramas del vídeo MPEG que estamos

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

viendo hayan sido transmitidas hace mucho tiempo, lo que necesitamos es que entre tramas consecutivas no exista un retardo que no nos permita procesarlas correctamente.

Ambos retardos pueden ser considerados el mismo en un entorno de área local. Sin embargo, aunque este proyecto ha sido desarrollado pensando en redes de área local cableadas-inalámbricas, esto no es un inconveniente para que diseñemos el protocolo contemplando este aspecto. De esta manera, su exportación a una aplicación para otro tipo de redes es inmediata.

Como debe de haberse entendido ya, se trata de modificar únicamente el criterio de tiempo real. O lo que es lo mismo, modificar los mecanismos de desecho de los paquetes obsoletos. Seguidamente explicamos esquemáticamente esta parte de la segunda versión del protocolo. El resto es idéntico al explicado en la versión anterior.

En la cache de cada nodo existe una tabla (Figura 4.33) en la que se almacenan índice de paquete esperado y tiempo máximo de llegada de este (*deadline*). Las entradas están ordenadas de menor a mayor en lo que a sus *deadline*s se refiere. El mecanismo de *filtrado activo* de los paquetes multicast es el siguiente:

1) Cuando llega un paquete multicast se eliminan aquellas entradas de la cache que estén obsoletas. Es decir, los índices que tengan un deadline que ya haya sido sobrepasado por el tiempo actual.

2) Se observa si el último índice de la tabla es menor que el índice del paquete nuevo, o si la tabla está vacía.

   2.1) En caso afirmativo:

      2.1.1) Se introduce en la tabla el *deadline* del índice siguiente al del paquete que ha llegado (tiempo actual + *margenRT*),

      2.1.2) Se distribuye el paquete a todos los suscriptores

   2.2) En caso contrario, si se da que exista una entrada para el índice del paquete (que no haya sido eliminada por obsoleta):

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

2.2.1) Se elimina la entrada de la tabla,

2.2.2) Se distribuye el paquete a todos los suscriptores

2.3) Si el índice del paquete no es mayor que el último indice esperado, ni la lista está vacía, ni hay entrada para ese índice, el paquete está obsoleto y se desecha.

En la Figura 4.34 se muestra el organigrama de este algoritmo.

El *filtrado activo* de los paquetes de NACK (petición de retransmisión) y de retransmisión es muy sencillo. Si las peticiones de retransmisión o las retransmisiones son obsoletas, es decir, no existe entrada para sus índices en las tablas de los nodos, son desechadas. En caso contrario siguen su rumbo con el mismo enrutamiento unicast de la versión anterior.

Implementación del protocolo de Multicast en Tiempo Real versión 2

En este apartado vamos a explicar los detalles interesantes de la implementación de esta segunda versión del protocolo de Multicast en Tiempo Real.

Lo único que diferencia a este protocolo de la versión anterior es como se discrimina qué paquetes son válidos y cuales no, para cumplir con las restricciones de tiempo real. Por lo tanto, la implementación del mismo es prácticamente igual. Tenemos el mismo conjunto de clases de aplicación y de cápsula. La diferencia radica fundamentalmente en las rutinas *evaluate* de las cápsulas de datos multicast (*MDataCapsule*), de petición de retransmisión (*NACKCapsule*) y retransmisión (*RDataCapsule*), que es donde se realiza el filtrado activo.

En las figuras 4.35, 4.36 y 4.37 se muestran dichos métodos. Con lo dicho en el apartado anterior creemos que estas figuras no requieren más comentario.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

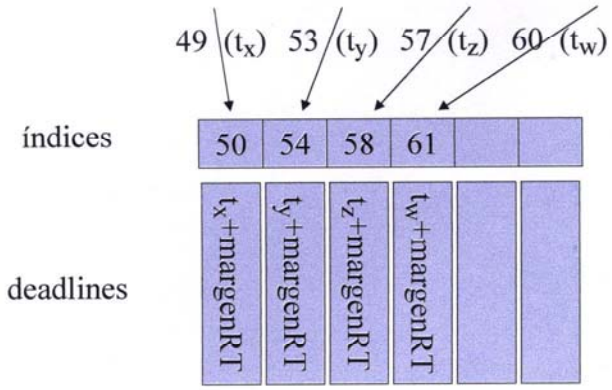


Figura 4.33: Tabla para RT en las caches de los nodos en el protocolo de Multicast RT v2

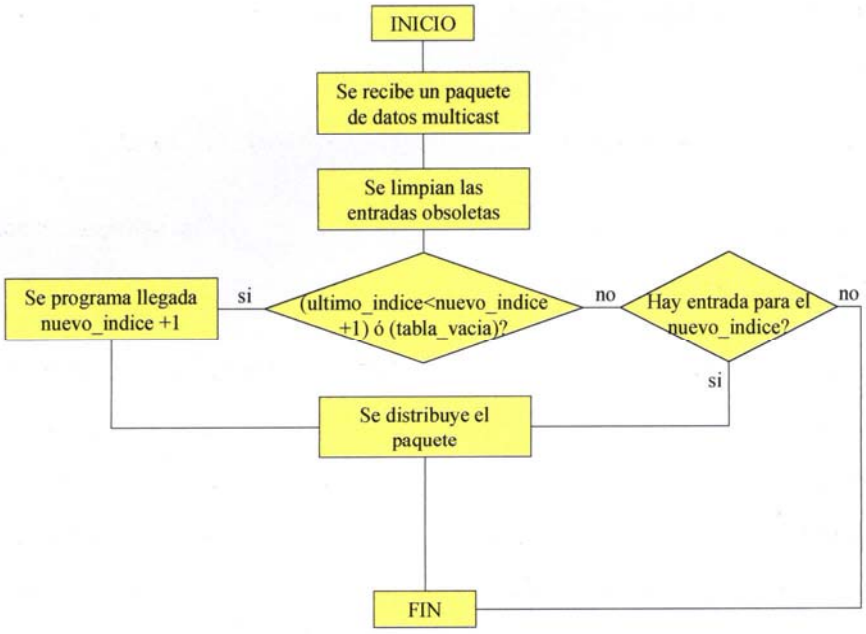


Figura 4.34: Organigrama del filtrado activo en el protocolo de Multicast RT v2

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
public boolean evaluate(Node n) {
    W_JAIAIAJ m = (W_JAIAIAJ)n.getCache().get(group, target);
    if (m != null) {
      //FILTRADO ACTIVO
      //limpieza de lo obsoleto
      int i = 0;
      while ((m.ai2.length>0)&&((n.time()-m.aj1[0])>vg.margenRT)){
          AuxTipo aux = (AuxTipo) eliminarEntrada(new AuxTipo(m.ai2,
m.aj1), 0); //eliminamos cada entrada obsoleta
        m.ai2 = aux.indices;
        m.aj1 = aux.sellos;
        }
      //fin limpieza de lo obsoleto
      //si la lista está vacía o el índice del último < indice nuevo+1
      if((m.ai2.length == 0)||(m.ai2[m.ai2.length-1]<indice+1)){
        AuxTipo aux = (AuxTipo) nuevaEntrada(new AuxTipo(m.ai2,
m.aj1),indice +1, n.time() + vg.margenRT);
        //programamos la llegada del proximo indice
        m.ai2 = aux.indices;
        m.aj1 = aux.sellos;
        //seguimos para la distribución multicast
      }else{
        int j;
        boolean encontrado = false;
        for (j = 0; j<m.ai2.length; j++)
          if (indice == m.ai2[j]){
            encontrado = true;
            break;
          }
        if (!encontrado){ //si no se encuentra el índice
          return true; //se desecha el paquete obsoleto
        }else{
        }
      }
      //Fin FILTRADO ACTIVO
      if (n.getAddress()!=getSrc()) //si no estamos en el nodo origen
le pasamos a la aplicación la dirección del nodo previo
          nodoPrevio = getPrevious();
      if (m.ai1 != null){
        for (int k = 0; k < m.ai1.length; k++)
          n.routeForNode(this, m.ai1[k]);
      }
    } return n.deliverToApp(this, dpt); //pasamos la cáp a la
aplicación
  }//evaluate()
```

Figura 4.35: Método evaluate de la cápsula *MDataCapsule* en el protocolo de Multicast RT v2

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public boolean evaluate(Node n) {

    W_JAIAIAJ m = (W_JAIAIAJ)n.getCache().get(group, target);
    if (m == null){ //si estamos en un Receiver(no hay lista de
suscriptores)
      return n.routeForNode(this, getDst());
    }else{
      // FILTRADO ACTIVO
      if (m.ai2.length==0){
        return true; //todas las peticiones obsoletas o no suscrito
      }else{
        boolean encontrado = false;
        int posicion = -1;
        for (int j = 0; j < npeticiones; j++)
        for (int i = 0; i<m.ai2.length; i++)
            if (m.ai2[i] == indice[j]) {
              encontrado = true; break;
            }//if
        if (!encontrado){
        //si no hay registros no obsoletos desechamos la cápsula
          return true;
        }else {
        }
      }//if
      if (n.getAddress() == getSrc())//si estamos en el origen
      return n.routeForNode(this, getDst());
      else
        return n.deliverToApp(this, dpt); //si estamos en el destino
    }
  }
```

Figura 4.36: Método evaluate de la cápsula *NACKCapsule* en el protocolo de Multicast RT v2

4.3.2 Protocolo de Multicast para la Edición Cooperativa, versión 1

La Edición Cooperativa es hoy día un campo abierto de la investigación. El concepto de Edición Cooperativa es muy sencillo. Se trata de un conjunto de terminales que colaboran en la creación y procesamiento de un documento. Partiendo de esta premisa inicial, se nos plantean multitud de enfoques a la resolución de este problema. Con cada uno perseguimos el mismo objetivo: garantizar la coherencia entre la información a la que acceden de manera concurrente todos los colaboradores. Para ello buscamos un compromiso entre dos características deseables:

- Evitar la interferencia entre el trabajo de cada uno de los colaboradores
- Permitir el máximo posible de interacción con el documento de cada colaborador.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Un enfoque podría consistir en dividir el documento en diferentes segmentos. Cada colaborador podría leer todos los segmentos, pero tendría permiso para modificar solamente uno. Estos segmentos pueden ser resultado de la división lógica o física del documento, según se traten de segmentos de un mismo archivo, o archivos diferentes respectivamente.

En este enfoque sacrificamos la segunda característica para salvaguardar la primera. Existe otra posibilidad. Es posible que en nuestra aplicación concreta sea deseable potenciar la segunda característica a costa de la primera. Es decir, permitir una mayor interacción de cada colaborador con todo el documento, aún a costa de pérdida en la seguridad del trabajo de cada colaborador.

Si ponemos como premisa inicial que los colaboradores actuarán de manera cuidadosa evitando interferir unos con otros, es posible diseñar un protocolo en el cual, todos puedan modificar el documento completo y ver las modificaciones de los demás colaboradores. Asumimos que es probable que el trabajo de un colaborador se vea interferido por otro. Sin embargo nos parece sumamente interesante evaluar las ventajas y desventajas de una aproximación tan libre al problema en un prototipo real. No tenemos conocimiento de ninguna implementación de estas características hasta el momento.

Pretendemos extraer toda la información posible de un conjunto de pruebas con este prototipo. Esto nos permitirá comprender mejor el problema del procesamiento distribuido de texto y tal vez hallar nuevas ideas que podrán luego aplicarse a soluciones intermedias.

Protocolo diseñado

La idea inicial es la siguiente. Tenemos un conjunto de estaciones colaboradoras que llamamos Escritores (*Writers*). Son las entidades que van a crear y modificar el documento distribuido. Cada Escritor, una vez señalice su incorporación al servicio, tiene una copia local del documento en el cual se trabaja. Cuando uno realiza una modificación en el mismo, esta se verá realizada en todas las copias locales de todos los Escritores. Un nodo al que llamamos Transmisor (*Sender*) se encarga de que exista coherencia entre todas las copias locales del documento. Para ello tendrá una copia que denominamos documento global.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

En primer lugar, dado que en entornos inalámbricos disponemos de poco ancho de banda, hemos optado por la transmisión de los cambios realizados por cada Escritor, en lugar de transmitir copias de los documentos locales. Para ello debemos especificar el conjunto de cambios posibles que se pueden realizar sobre documentos locales. Es deseable que la granularidad de estos cambios sea pequeña, para hacer más ágil la actualización de los mismos en toda la red.

Los cambios que tenemos en cuenta en nuestro prototipo son los siguientes:

- Suprimir un carácter en una posición determinada
- Insertar un carácter en una posición determinada
- Sustituir un carácter en una posición determinada

Dado que todos los *Escritores* recibirán los cambios realizados por cada uno de ellos, decidimos utilizar una arquitectura multicast para la distribución de los mismos.

En la figura 4.37 se muestra la topología general de este servicio que hemos diseñado. Como sucede en cualquier servicio multicast, existen tres tipos de entidades:

- *Nodos Transmisor*: encargados de difundir los paquetes multicast a todos los miembros del grupo. En este caso es el elemento que se encarga de mantener la coherencia entre todas las copias locales del documento, mediante la transmisión de los cambios globales.
- *Nodos Intermedios* (*NI*): nodos que permiten la estructura jerárquica de la topología multicast, concentrando y distribuyendo paquetes.
- *Nodos Receiver* (*Nodos Escritores*): nodos que tienen la virtud de poder incorporarse al trabajo cooperativo cursando una suscripción hacia arriba, jerárquicamente hablando.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

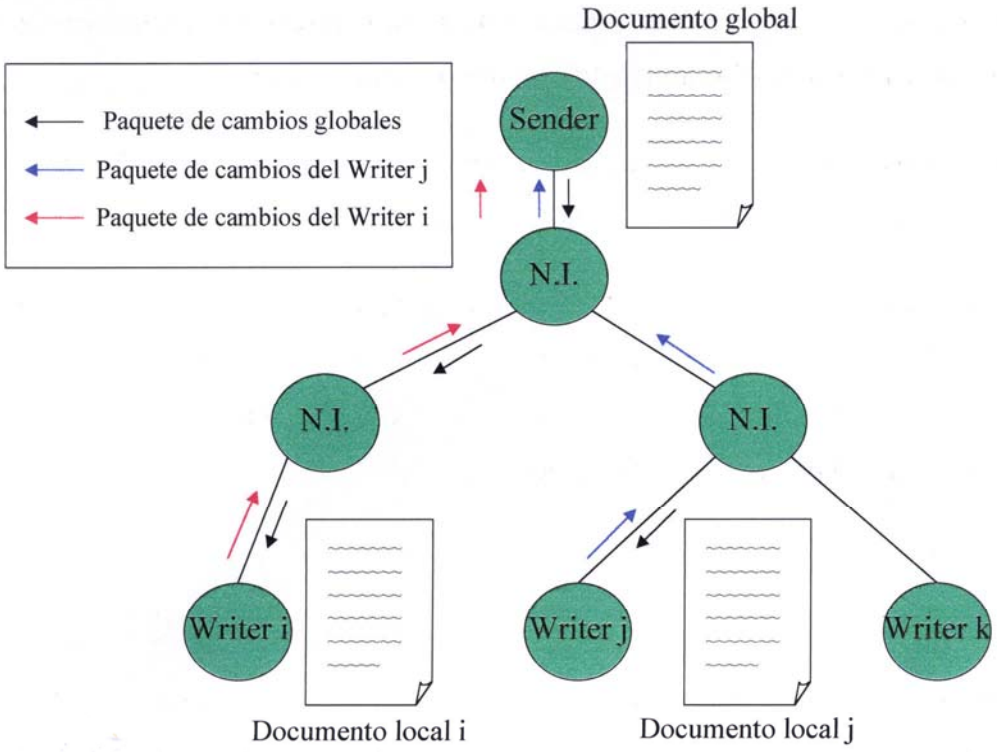


Figura 4.37: Topología general del servicio del protocolo de Multicast para Edición Cooperativa v1

Descripción general del protocolo. El esquema general del protocolo diseñado es el siguiente:

1) Un nodo Escritor cursa hacia el Transmisor una petición de incorporación al servicio. A continuación, cuando esa petición es aceptada, recibe de manera fragmentada una copia del documento global. Confirma cada fragmento que recibe.

2) Cuando el Escritor ha recibido todos los fragmentos del documento puede empezar a modificar su copia local. Cada modificación que realice es anotada con un índice. Periódicamente envía todas las modificaciones realizadas al nodo *Transmisor* por unicast.

3) A su vez el nodo Transmisor está recibiendo los cambios que le envían los nodos Escritor. Periódicamente, el Transmisor fabrica paquetes con todos esos cambios y los transmite a todos los Escritores.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

4) Los nodos Escritores actualizan los cambios que han realizado los otros Escritores y confirma los cambios propios transmitidos y deja de transmitirlos.

5) En el momento en el que un nodo Escritor desea finalizar su colaboración lo notifica al Transmisor y dejará de recibir los cambios de los demás.

Vamos a tomar como premisa inicial que nos encontramos en una WLAN. Por lo tanto cada señalización de este protocolo requiere de una confirmación, pues existe la posibilidad de pérdidas temporales de los enlaces. Así pues, a continuación detallamos un poco más como reforzamos este esquema general:

1)

1.1) Una vez que el nodo Escritor decide cursar una petición de incorporación al servicio, se queda a la espera del primer fragmento del documento global. Un thread se encarga de reenviar periódicamente peticiones de incorporación hasta que se reciba ese primer fragmento. En el momento que se recibe, tenemos la seguridad de que nuestra petición ha sido "oída".

1.2) Cuando el nodo Transmisor recibe una petición de incorporación al servicio por parte de un nodo Escritor detiene la distribución multicast de los cambios recibidos. Divide lógicamente el documento en fragmentos de un tamaño fijo (rellenando el último con ceros, si fuera necesario). Además crea para el nodo Escritor un registro en el que anotará las confirmaciones de cada fragmento enviado.

1.3) Si mientras está transmitiendo el documento recibe nuevas peticiones de suscripción de otros nodos Escritor, creará nuevos registros para éstos y les transmitirá el documento global. El Transmisor no reanudará la transmisión multicast de los cambios globales hasta que todos los nodos Escritor que hayan solicitado incorporarse confirmen estar servidos.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

2)

2.1) Cada nodo Escritor mantiene una lista con los cambios locales realizados. Cada cambio tiene un índice para identificarlo. Periódicamente transmite esos cambios al nodo Transmisor. Para quitar un cambio de esa lista, y dejar de transmitirlo, es preciso recibir confirmación de la recepción del mismo por parte del Transmisor.

3)

3.1) El Transmisor guarda el índice del último cambio recibido por cada nodo Escritor que está incorporado al servicio. Cuando se reciben cambios con índice posterior, se aplican al documento global y se anotan, junto con el nodo que los remitió, en una lista.

3.2) Periódicamente el nodo Transmisor recorre esa lista de cambios recibidos y con ellos va fabricando paquetes para la transmisión multicast de todos esos cambios. Con cada paquete que fabrica, extrae esos cambios de la lista. En estos paquetes van los cambios, sus índices y el nodo Escritor en el que se produjo cada uno. Además, a cada paquete de cambios globales se le añade un sello aleatorio que sirve para identificarlo.

3.3) Cada paquete de cambios globales se retransmite por multicast, hasta que se recibe confirmación de todos los nodos Escritores. La confirmación se realiza jerárquicamente, como se explicará luego. Una vez que se recibe esta confirmación, se hace el multicast del siguiente paquete de cambios globales etc.

4)

4.1) Cuando un nodo Escritor recibe un paquete de cambios globales, lo primero que hace es confirmar su recepción. Luego, observando su sello, comprueba si no ha sido recibido ya. Si lo ha recibido es descartado y no se aplican los cambios que porta.

4.2) El nodo Escritor aplica uno a uno todos los cambios que proceden de otros nodos Escritores. Si encuentra cambios propios, los utiliza a modo de confirmación. Es

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

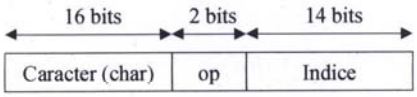
decir, deja de transmitir esos cambios hacia el Transmisor porque ya tiene la seguridad de que han sido recibidos.

5)

5.1) Cuando un nodo Escritor desea finalizar su colaboración envía un paquete para señalizarlo al nodo Transmisor. A partir de este momento, cada nuevo paquete de cambios globales recibido disparará una nueva señalización de fin de la colaboración. Así, la ausencia de más paquetes de cambios globales nos confirma implícitamente que nuestra petición ha sido cursada correctamente.

En las figuras 4.39 a la 4.43 mostramos la idea de cada una de estas fases del protocolo. En la figura 4.38, mostramos el formato de los cambios y de paquetes más relevantes que hemos utilizado.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004



Formato de los cambios

N° cambios	Cambio i	Cambio i+1	o o o	Indice del primero

Formato de los paquetes de cambios locales

N° nodos	Writer α	N° cambios	Ci	Ci+1	o o o	Indice del primero	Writer β	o o o	sello

Formato de los paquetes de cambios globales

Indice	N° fragmentos	Fragmento

Formato de los fragmentos del documento

Figura 4.38: Formato de los cambios y paquetes

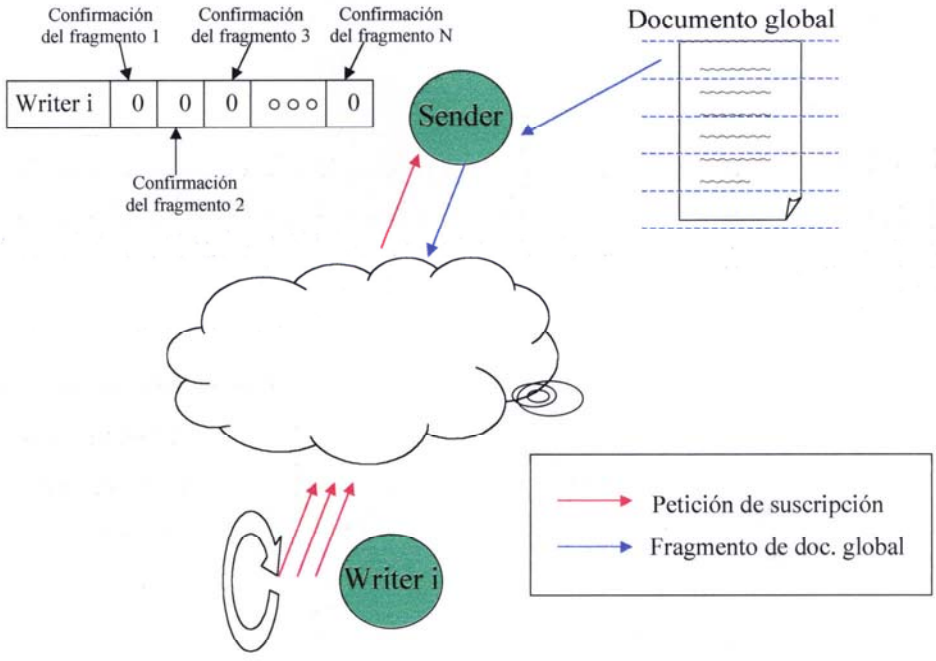


Figura 4.39: Petición de incorporación al servicio

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

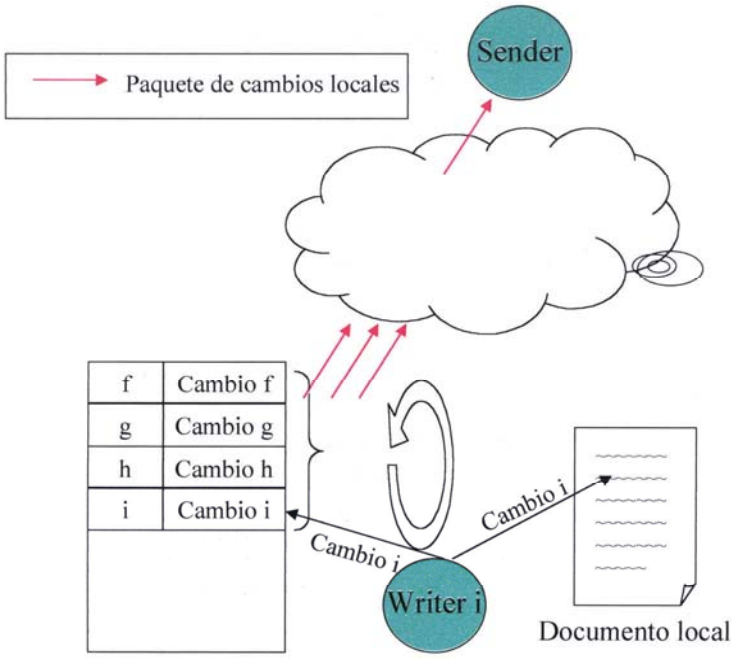


Figura 4.40: Mantenimiento de tablas locales

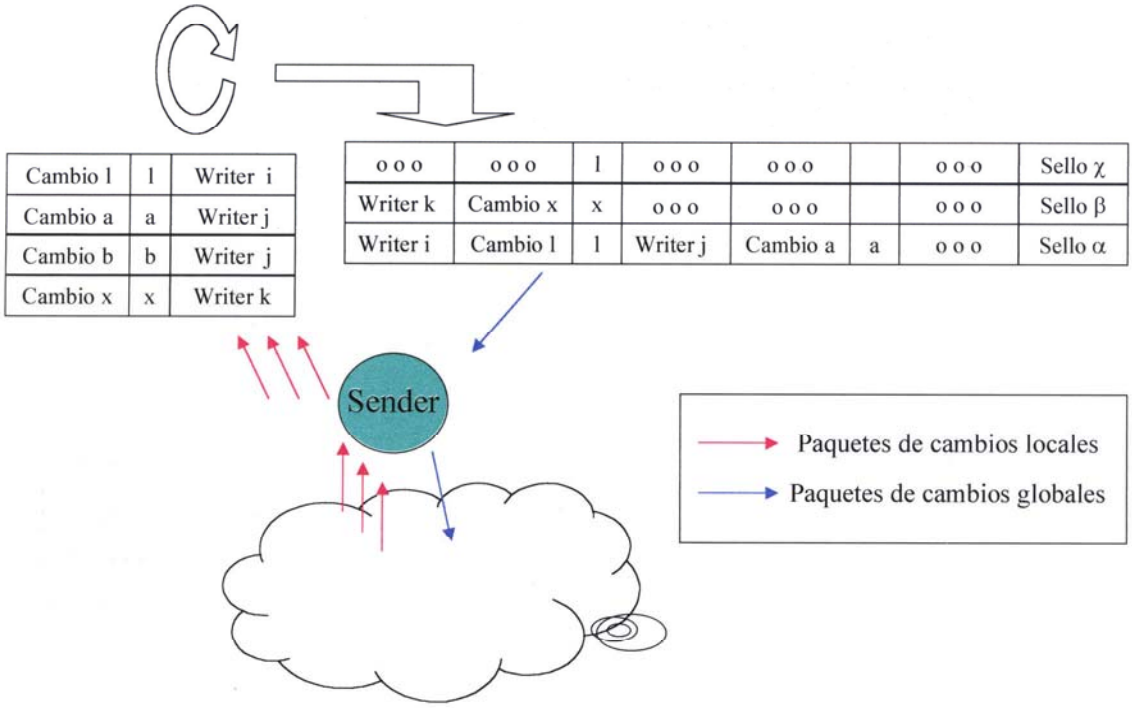


Figura 4.41: Creación de los paquetes de cambios globales

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

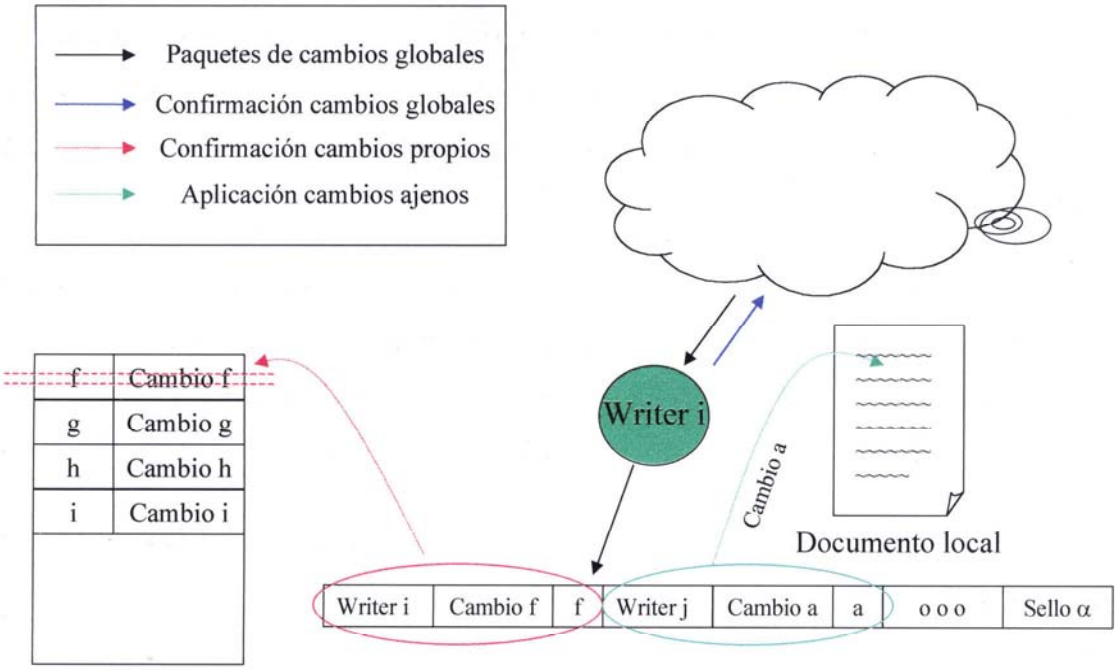


Figura 4.42: Recepción de paquete de cambios globales

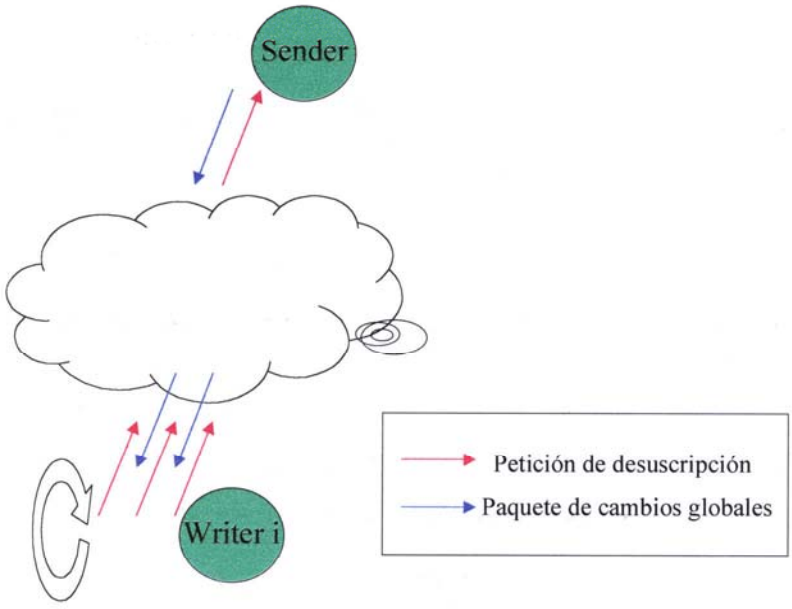


Figura 4.43: Desuscripción al servicio

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Atención a las peticiones de incorporación y de fin de colaboración

El objetivo de este apartado es detallar como se produce la incorporación de un nuevo suscriptor al servicio. En primer lugar vamos a enumerar que sucede a lo largo de la red cuando un nodo Escritor inicia una petición de incorporación y de fin de colaboración en la edición del texto.

1) Cuando un Escritor inicia una petición de incorporación el nodo intermedio anterior pone, en el caso de que no exista, una entrada en la lista de suscriptores que posee. Además señaliza la petición al nivel de aplicación del nodo intermedio. Por último cursa esta petición al siguiente nodo intermedio.

2) Cuando la petición llega al nodo Escritor, se pasa directamente al nivel de aplicación y sucede lo que se explica abajo.

3) Cuando el Escritor decide poner fin a la colaboración envía una notificación al nodo intermedio anterior. Este quita su entrada de la tabla de suscriptores.

4) Si la lista de suscriptores de este nodo intermedio está ahora vacía, se envía una petición de fin de colaboración al nivel anterior y se le notifica al nivel de aplicación para que retransmita esta petición de desuscripción periódicamente.

5) Cuando el nivel de aplicación deje de recibir paquetes de multicast se dará por recibida la petición de desuscripción.

En segundo lugar exponemos lo que sucede en el nodo Transmisor:

1) Inicialmente el nodo Transmisor tiene una variable entera que llamamos "enviando_doc_completo" en valor 0. Esto quiere decir que periódicamente transmite paquetes con los cambios recibidos a todos los suscriptores.

2) Cuando llega una petición de incorporación del nodo i se incrementa en 1 el valor de "enviando_doc_completo". Se paraliza la transmisión de cambios globales y se encolan los cambios locales que se reciban. También se paraliza su aplicación al documento global que tiene el Transmisor.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

3) El documento global se divide lógicamente en fragmentos de tamaño fijo, rellenándose con ceros en caso de que fuera necesario. Se crea un vector por cada nodo nuevo que solicite su incorporación. En este vector se anotarán las confirmaciones de cada fragmento del documento global que se reciban de dicho nodo.

4) Cuando se ha confirmado la recepción de todos los fragmentos del documento global se decrementa en 1 "enviando_doc_completo" y se retira la entrada de ese nodo. Cuando "enviando_doc_completo" vuelve a tomar el valor 0, es decir, cuando todas las peticiones de incorporación han sido servidas, se reanuda la distribución multicast de los cambios recibidos (y su aplicación al documento global).

En las figuras 4.44, 4.45 y 4.46 se muestran tres organigramas explicativos.

Distribución ordenada de los cambios globales y confirmación jerárquica

En este apartado explicamos con más detalle la parte del protocolo que se refiere al multicast de los cambios globales y la confirmación.

Partimos de las siguientes premisas:

- Cada paquete de cambios globales se transmite hasta que se reciba confirmación de la recepción del mismo por todos los suscriptores.

- Los cambios globales incluidos en estos paquetes no se vuelven a incluir en ningún otro paquete de cambios globales. Una vez que se recibe confirmación de la recepción de los mismos se da por hecho que se han actualizado en toda la red y se retiran de todas las tablas en las que habían sido memorizados.

- La creación y posterior transmisión de los paquetes de cambios globales está ordenada por los cambios locales recibidos. Es decir, si el Transmisor recibe un cambio local c1 en t1 y otro cambio local c2 en t2, y t1<t2, en la construcción de los paquetes de cambios globales y en el multicast de los mismos nunca un suscriptor recibirá primero el cambio c2 y después el cambio c1. De esta manera garantizamos que el resultado de la aplicación del cambio c1 y c2 sea el mismo en toda la red.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

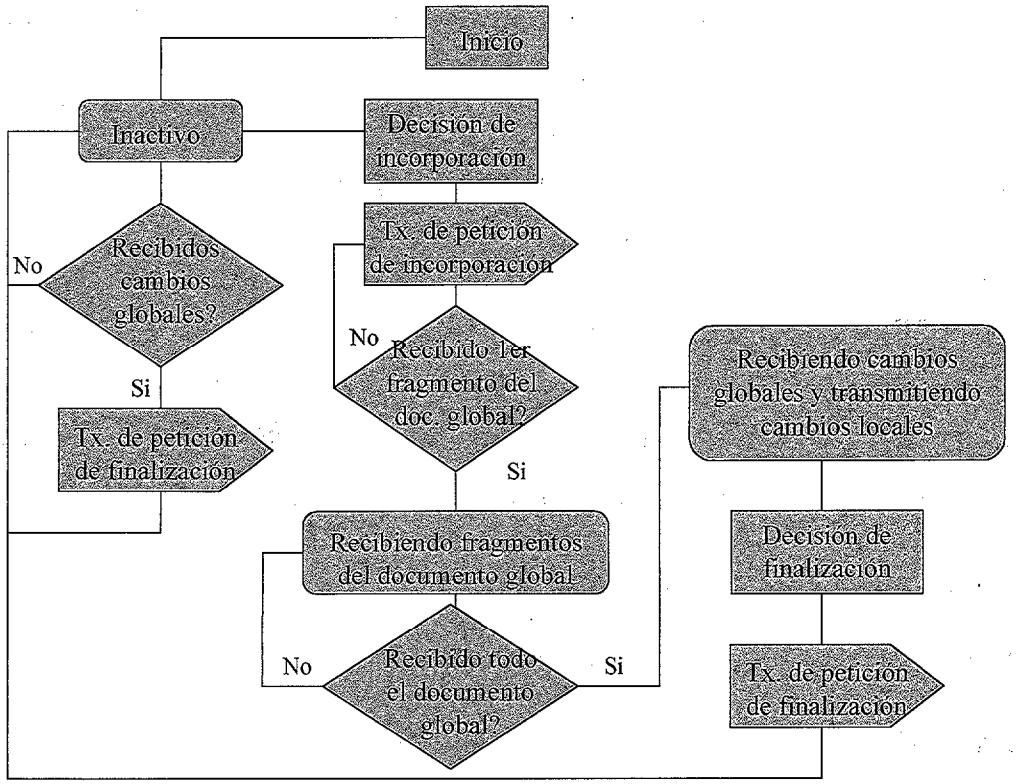


Figura 4.44: Organigrama de la suscripción al servicio. Nodo Writer

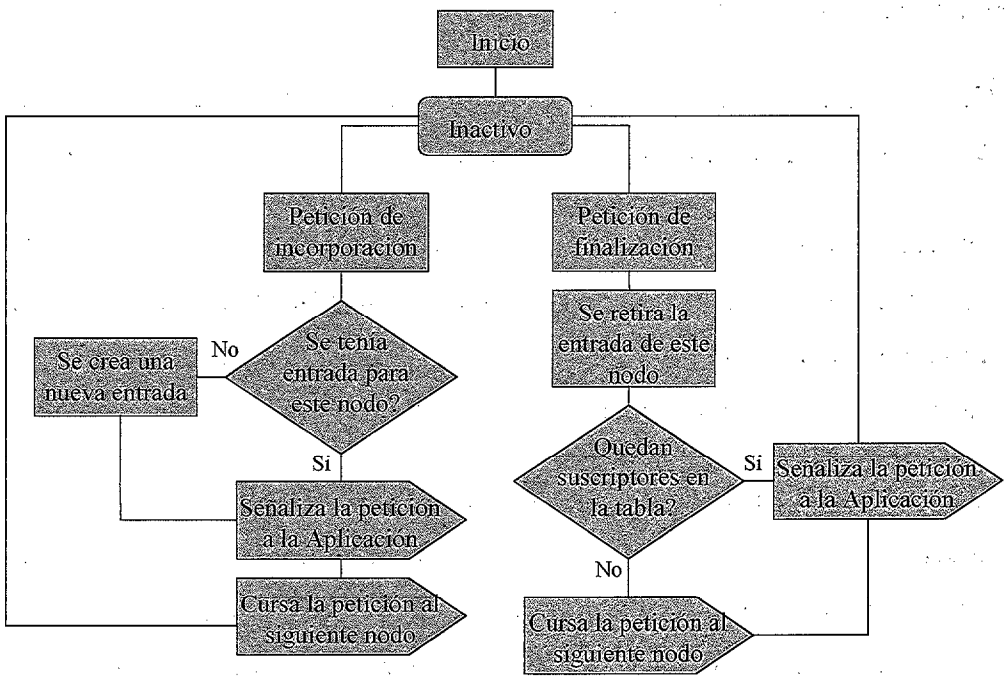


Figura 4.45: Organigrama de la suscripción al servicio. Nodo Intermedio

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

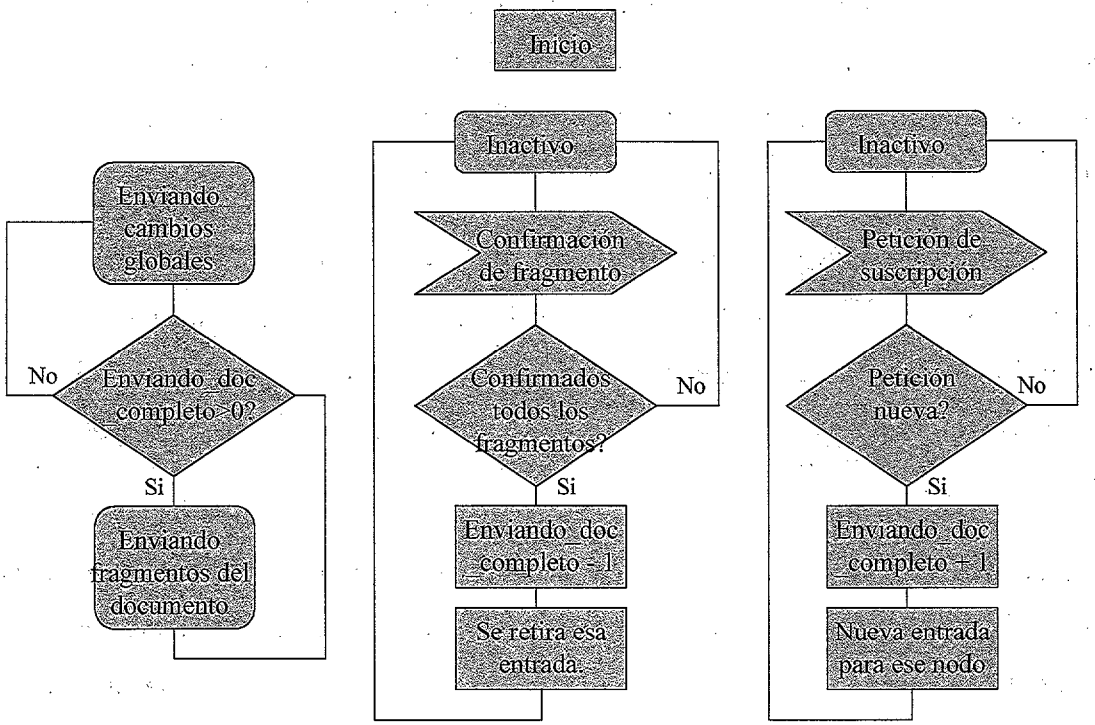


Figura 4.46: Organigrama de la suscripción al servicio. Nodo Sender

A continuación describimos los pasos que sigue el protocolo para la distribución de estos paquetes de cambios globales y la confirmación de los mismos. En todos los nodos de la topología distinguimos dos niveles del protocolo: el nivel de red y de aplicación.

1) El nodo Transmisor se dispone a transmitir un paquete de cambios globales. Para ello utiliza la tabla de suscriptores de que dispone en el nivel de red. Clona y envía este paquete para cada uno de los suscriptores.

2) En el nivel de red cada nodo intermedio de la red existe una variable lógica local que denominamos "esperando_confirmación". Su valor inicial es "falso". También existe en los nodos intermedios y en el Transmisor, una tabla con una entrada por cada nodo suscriptor, que nos indicará qué nodos han confirmado el paquete de cambios globales del cual se está haciendo multicast en este momento.

3) Cuando un nodo intermedio recibe una cápsula de cambios globales envía inmediatamente una confirmación al nodo emisor de la misma. Además pone la variable "esperando_confirmación" a "verdadero". La clona y envía a todos los

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

suscriptores que este nodo intermedio tiene en su tabla local, y pasa el paquete de cambios globales al nivel de aplicación. En estos nodos intermedios, el nivel de aplicación guardará una copia del mismo y lo retransmite periódicamente hasta la confirmación de su recepción por todos los suscriptores.

4) A partir de este momento y mientras "esperando_confirmación" sea "verdadero", cada nuevo paquete de cambios globales que se reciba se pasa al nivel de aplicación, pero no se confirma. El nivel de aplicación confirma los paquetes de cambios globales iguales al ya recibido e ignora los nuevos. Para identificarlos se utiliza el sello de estos paquetes de cambios globales.

5) Por cada confirmación recibida por un nodo intermedio o el nodo Transmisor, se anota en la entrada correspondiente a ese nodo suscriptor. Cuando todos los suscriptores han confirmado la recepción de este paquete de cambios globales se le notifica al nivel de aplicación para que cese de reenviar ese paquete. Por último se pone "esperando_confirmación" a "falso" y ya está listo para aceptar un nuevo paquete de cambios globales.

La figura 4.47 sirve para ilustrar esta parte del protocolo. En la figura 4.48 mostramos el organigrama.

Por último queremos explicar por qué decimos que es una confirmación jerárquica. Cada nodo intermedio confirma la recepción al nodo anterior y éste, cuando proceda a su nodo anterior y así jerárquicamente hasta que al nivel de aplicación del nodo Transmisor le llega una confirmación. En ese momento se tiene la seguridad de que todos los suscriptores han recibido el paquete de cambios globales.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

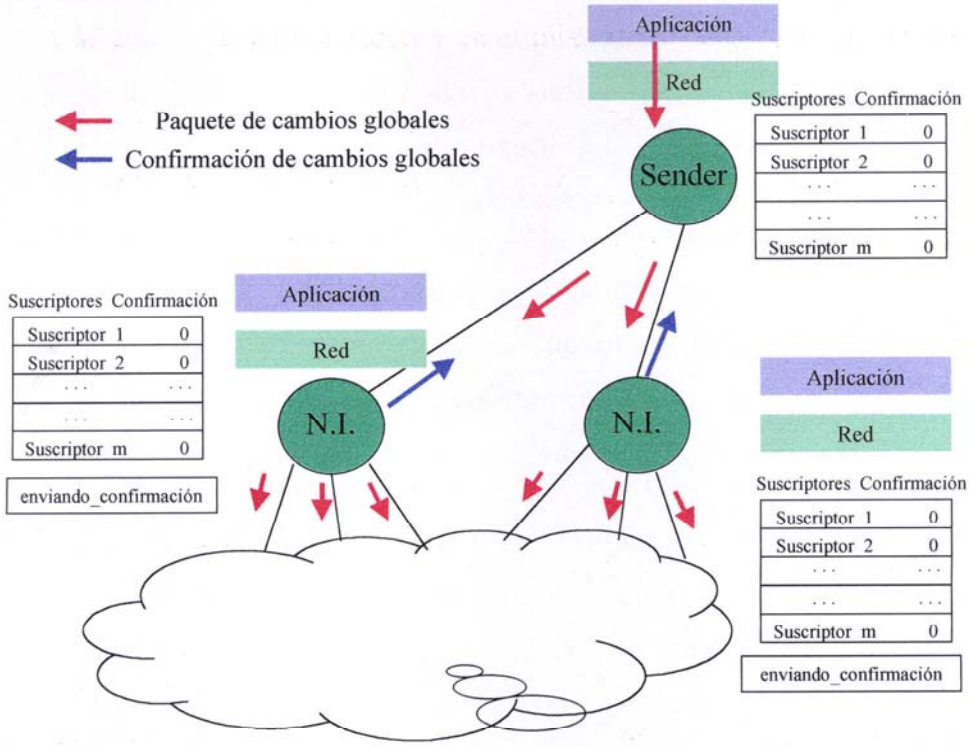


Figura 4.47: Distribución de cambios globales usando las tablas de suscriptores

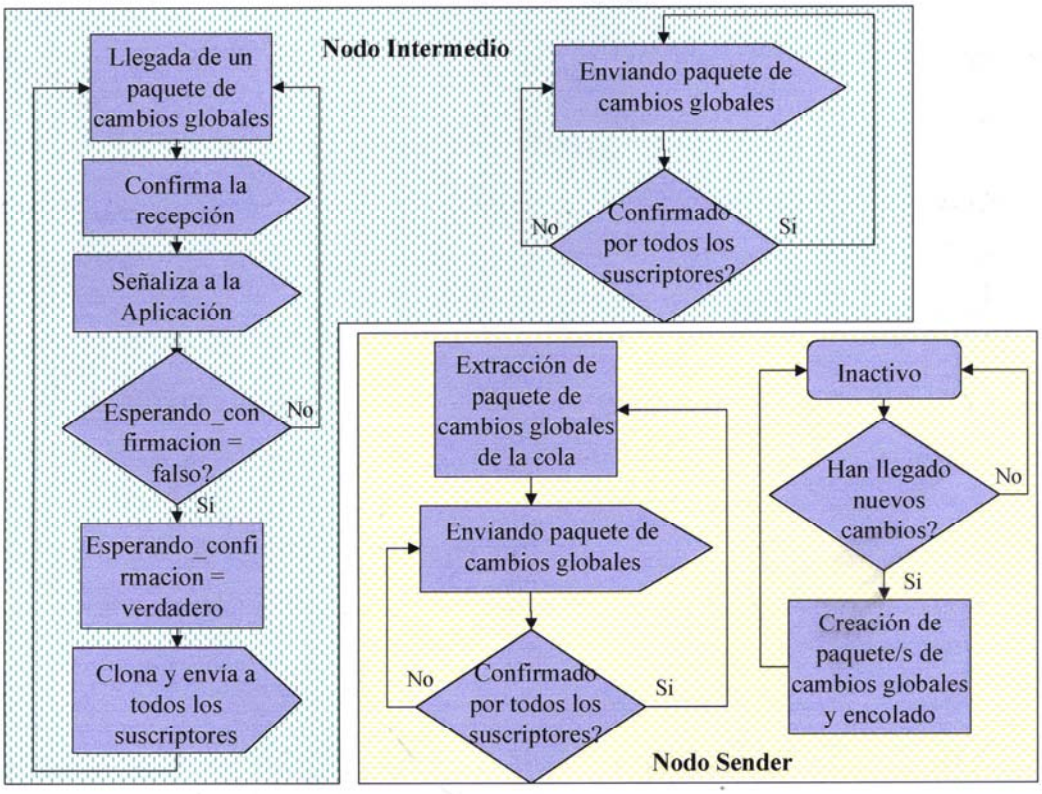


Figura 4.48: Organigrama de la distribución de los cambios globales

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Implementación del protocolo de Multicast para Edición cooperativa, versión 1

El protocolo se implementa en base a la extensión de las clases padre *Capsule* y *Application* del ANTS. Además y para separar los distintos tipos de cápsulas en grupos se extiende la clase padre *Protocol*. Las cápsulas se agrupan por motivos de seguridad.

Hemos implementado seis tipos de cápsula y tres tipos de aplicación.

Cápsulas

- ***DocRequestCapsule*:** Cápsula cuyo cometido es cursar una petición de incorporación al trabajo cooperativo, o señalizar el fin de la participación de un nodo Escritor.

- ***DocCapsule*:** Cápsula que porta un fragmento del documento global para permitir que un nodo Escritor reciba una copia actualizada del documento al suscribirse al servicio.

- ***DocACKCapsule*:** Cápsula de confirmación de recepción de *DocCapsule* que envía el nodo Escritor al Transmisor.

- ***ChangesCapsule*:** Cápsula que envía cada nodo Escritor con los cambios realizados en la copia local del documento, hacia el Transmisor con el objeto de que los actualice globalmente.

- ***GChangesCapsule*:** Cápsula con cambios de los nodos Escritor que se distribuye por multicast a todos los suscriptores para que los actualicen en sus copias locales.

- ***GCACKCapsule*:** Cápsula de reconocimiento de recepción de *GChangesCapsule*. Se encarga de la confirmación jerárquica de la recepción de los cambios globales.

Aplicaciones

- ***WriterApplication*:** Aplicación situada en los nodos Escritor que permite que un usuario pueda participar en la Edición Cooperativa cursando peticiones de suscripción, transmitiendo los cambios locales que se produzcan en el documento, confirmando la recepción de los mismos, actualizando los cambios ajenos y

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

cursando la petición de desuscripción cuando se desea finalizar. En la figura 4.49 se muestra la interfaz de esta aplicación.

- ***NodeApplication***: Aplicación situada en los nodos intermedios de la topología. Se encarga de gestionar la información de los nodos que esta´n suscritos al servicio y los que no, recoger y cursar cápsulas de peticiones y cambios locales, y distribuir las cápsulas de cambios globales.

- ***SenderApplication***: Aplicación situada en el nodo Transmisor que se encarga de recibir los cambios locales de todos los nodos Escritor que estén suscritos al servicio, construir cápsulas con todos los cambios recibidos y transmitirlas a través de una topología multicast. En definitiva, mantiene la coherencia entre todas las copias locales del documento.

Dado que la funcionalidad de cada uno de estos elementos ha sido suficientemente explicada en el apartado anterior nos hemos limitado a poner la interfaz de la aplicación de los nodos Escritores en el Anexo I el código fuente de cada aplicación y cada cápsula. Volvemos a recordar que tanto el código fuente como la documentación en formato de hipertexto se puede encontrar en el CD adjunto a este Proyecto Fin de Carrera.

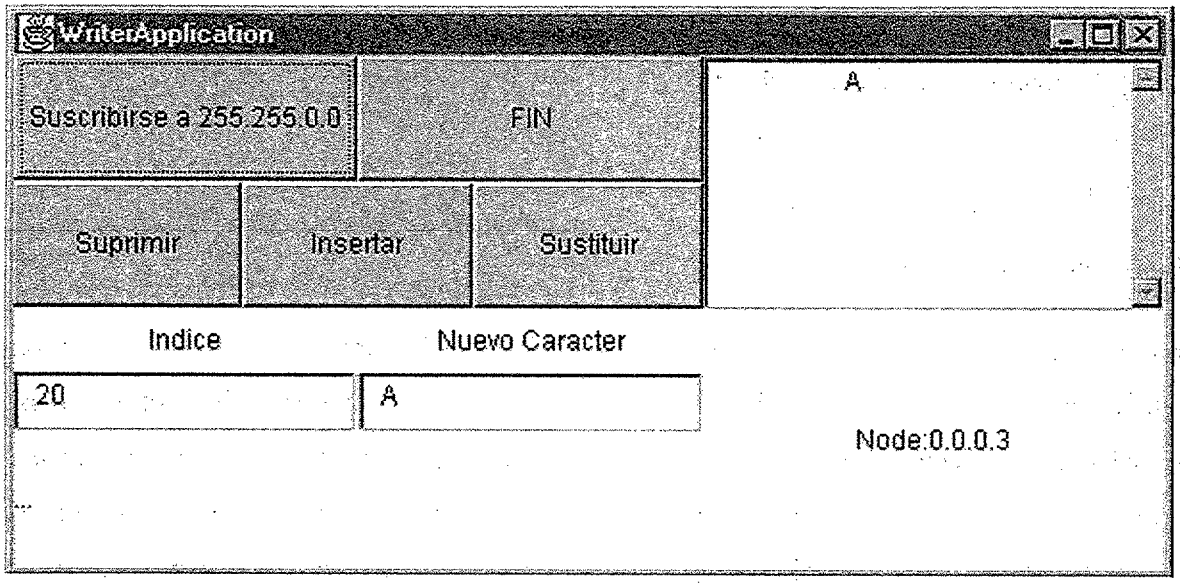


Figura 4.49: Interfaz de la WriterApplication del Protocolo de Edición Cooperativa v1

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

4.3.3 Protocolo de Multicast para la Edición Cooperativa, versión 2

El protocolo explicado en el apartado anterior es un ejemplo de la libertad que podemos tener, gracias al uso del ANTS como herramienta rápida de prototipado de protocolos para el testeo de nuevos servicios. Aún sabiendo que el planteamiento inicial podía ser problemático diseñamos el protocolo, realizamos el prototipo y lo probamos. La novedad es que el costo en tiempo fue muy bajo.

Sin embargo, desarrollar ese prototipo nos hizo ver con claridad cosas que no veíamos a priori sobre el papel. El protocolo explicado en el apartado anterior tiene un problema mucho más grave que la posible interferencia entre los Escritores. Se trata del siguiente problema: cuando un Escritor realiza un cambio local, la información que transmite para que ese cambio se aplique de manera global es el tipo de cambio, la posición en el documento global y un caracter (en el caso de que sea una sustitución). Sin embargo, la posición en el documento global sobre la que el Escritor quiere realizar un cambio puede variar desde el momento en que efectúa el cambio localmente, hasta que éste se aplica en el nodo Transmisor. Esto significa que un cambio local originaría otro muy diferente en el documento global y se perdería la coherencia entre las copias locales del documento.

Este defecto tan grave nos paso inadvertido durante el diseño del protocolo del apartado anterior. Gracias a la observación de las pruebas nos dimos cuenta de este grave problema y ello motivó la evolución del protocolo a la versión que explicaremos en este apartado.

Protocolo de Multicast para la Edición Cooperativa versión 2

En esta versión del protocolo hemos decidido que cada Receptor de la versión anterior puede actuar con dos roles distintos: Lector y Escritor. Para poder recibir los cambios actualizados del documento es necesario tener el rol de Lector. Para poder modificar un fragmento del documento global es necesario que el Receptor adquiera el rol de Escritor. A cualquier Receptor se le permitirá adquirir el rol de Lector. Para que un Receptor adquiera el rol de Escritor previamente debe tener el rol de Lector y se le debe conceder el fragmento del documento global que aspira a editar.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

En esta nueva versión del protocolo existirán dos grupos multicast solapados. Un grupo será el grupo de Lectores (*Readers*) y el otro el grupo de Escritores (*Writers*). El grupo multicast de Lectores se encargará de distribuir los cambios locales a todos los Lectores. El grupo multicast de Escritores se encargará de distribuir las señalizaciones pertinentes a todos los Escritores.

Por lo tanto, el funcionamiento de protocolo es el siguiente:

1) Un Receptor (*Receiver*) solicita al Transmisor (*Sender*) que se le permita incorporar al grupo de Lectores.

2) El Transmisor cuando recibe alguna petición de incorporación como Lector paraliza la aplicación de los cambios locales de los Escritores, notifica el grupo de Escritores que cada uno haga temporalmente cacheo local de sus cambios, divide el documento global en fragmento de longitud finita y los envía a o los nuevos Lectores.

3) Una vez que un Receptor ha recibido el documento completo (adquiere el rol de Lector) este puede solicitar la incorporación al grupo de Escritores. Cuando así lo desee, el Receptor enviará una petición de asignación de un fragmento para la edición, es decir, que se le conceda el rol de Escritor.

4) El Transmisor decide las peticiones de fragmento y notifica la resolución a los aspirantes a Escritor.

5) Los aspirantes a Escritor cuya solicitud haya sido admitida pueden ya efectuar modificaciones en su fragmento. Los cambios que efectúen serán enviados al Transmisor.

6) El Transmisor distribuye los cambios locales de cada Escritor al grupo multicast de Lectores.

Como puede apreciarse, esta versión es muy parecida a la anterior, con la salvedad de que se separan en dos grupos multicast las actividades de lectura o edición del documento global.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

El mecanismo de distribución del documento completo, el de distribución multicast de los cambios globales y el de la confirmación jerárquica de los mismos son idénticos a los explicados en la versión anterior. La finalización de la pertenencia de un Receptor a cualquiera de los dos grupos del protocolo es explícita. Es decir, que para dejar de ser Escritor o Lector ha de indicarse explícitamente al Transmisor, por medio de un mensaje. El mantenimiento de la coherencia es el propósito de esta versión. Este se consigue por medio del mecanismo de sincronización, que lo explicamos a continuación.

Mecanismo de sincronización del documento global. En este apartado presentamos los mecanismos de sincronización para mantener la consistencia que han sido empleados.

Un Lector envía una notificación al Transmisor de que quiere adquirir el rol de Escritor. Esta notificación se anota con la dirección del Lector en el Transmisor.

Periódicamente el Transmisor analiza si se ha recibido alguna notificación de petición de incorporación al grupo de los Escritores.

Si es así, el Transmisor notifica al grupo de los Escritores que encolen temporalmente sus cambios locales y no los envíen. Así mismo, el Transmisor deja de aplicar al documento global los cambios que siga recibiendo.

Cuando el Transmisor recibe confirmación por parte de todos los *Lector*es de que han recibido los cambios globales previamente difundidos, notifica a todos los aspirantes a Escritores que indiquen qué fragmento del documento global desean editar. El Transmisor esperará un tiempo prudencial para que los aspirantes a Escritor lo indiquen.

Pasado ese tiempo el Transmisor resuelve las peticiones recibidas de la siguiente manera:

1)

1.1) Aquellas peticiones de fragmentos que estén actualmente siendo modificados por Escritores son denegadas.

1.2) Aquellas peticiones de un segmento que no ha sido reservado por ningún Escritor y que sólo pide un aspirante a Escritor, son concedidas.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

1.3) Aquellas peticiones de un segmento no reservado que piden varios aspirantes se resuelve concediéndolo aleatoriamente a un aspirante y denegándolo al resto.

2) A cada aspirante a Escritor se le notifica la resolución de su petición.

3) Cada aspirante a Escritor confirma la recepción de la resolución de su petición.

4) El Transmisor deja de encolar los cambios de los Escritores y reanuda su actividad normal. Los Escritores, pasado un tiempo dejarán de encolar los cambios locales y los enviarán al transmisor.

Gestión de los fragmentos ocupados en el transmisor. En el Transmisor se lleva un control de qué fragmento está editando qué Escritor y donde empieza y termina el mismo. Los Escritores envían sus cambios refiriéndose a un índice local relativo. Cuando el Transmisor recibe los cambios de los Escritores, antes de aplicarlos al documento global corrige los índices teniendo en cuenta la posición relativa del fragmento de Escritor que se trate. De esta manera estamos siempre seguros de que cada cambio se aplicará en la posición global de documento correcta.

En el Transmisor existe una tabla que llamamos *tabla de indices* en la cual se almacena la información necesaria para la gestión de los fragmentos ocupados. En la figura 4.50 se muestra esta tabla. Además de la tabla necesitamos un vector de valores lógicos que llamamos *vector de ocupación* y que nos dirá para cada índice del documento global, si está ocupado (reservado por algún Escritor) o libre. '1' es ocupado, '0' libre. También lo mostramos en la figura 4.50.

Cuando el Transmisor recibe un cambio local que afecta a la longitud del documento (inserción o supresión) se modifican tres cosas:

- El documento global
- El vector de ocupación
- La tabla de indices

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Vector de ocupación

Tabla de índices

Escritor	Indice inicial	Indice final
Escritor 0	10	15
Escritor 1	0	8
...	...	...
Escritor i	50	73
...	...	...
Escritor n	90	100

Figura 4.50: Estructuras de datos en el Transmisor para la gestión de los fragmentos en el protocolo de Multicast para la Edición Cooperativa

Cuando el Transmisor recibe una petición de reserva de un fragmento, que se apunta por medio de un índice al documento global, pasa lo siguiente:

1) Se comprueba el *vector de ocupación* para ver si dicha fragmento está libre

2) En caso de que se le conceda el fragmento, se le asigna un conjunto de índices limitado por un número máximo (*no_max_indices*)

3) Modificamos el *vector de ocupación* poniendo a '1' su fragmento en insertamos en la *tabla de indices* una entrada para ese Escritor con el índice inicial y el índice final de su fragmento.

Implementación del protocolo de Multicast para la Edición Cooperativa, versión 2

Hemos decidido realizar tres tipos de aplicación y doce tipos de cápsula. En la tabla 4.1 se indica brevemente la finalidad de cada una de esta nuevas clases. Como podrá apreciarse, muchos de las clases tienen una finalidad idéntica a las equivalentes en la versión anterior. Otras sin embargo, son nuevas.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Nombre de la Clase	Finalidad
SenderApplication	Nivel de aplicación del Transmisor del protocolo
NodeApplication	Nivel de aplicación de los Nodos Intermedios
ReceiverApplication	Nivel de aplicación de los nodos Receptores
ReaderSCapsule	Petición de incorporación al grupo de Lectores
DocCapsule	Fragmento del documento global para los nuevos Lectores
DocACKCapsule	Reconocimiento de la recepción de un fragmento del documento global
WriterSCapsule	Petición de incorporación al grupo de Escritores
StopSendingChangesCapsule	Notificación del Transmisor al grupo de Escritores para que encolen los cambios locales
SendYourRequestCapsule	Notificación del Transmisor a los aspirantes a Escritor para que envíen sus peticiones de fragmento
RequestCapsule	Petición de fragmento que el aspirante a Escritor envía al Transmisor
ResolutionCapsule	Resolución de las peticiones de incorporación al grupo de Escritores
ResolutionACKCapsule	Reconocimiento de la recepción de una ResolutionCapsule por parte del Transmisor
ChangesCapsule	Cambios locales que el Escritor envía al Transmisor
GChangesCapsule	Cambios globales que el Transmisor envía a todos los Lectores
GCACKCapsule	Reconocimiento de la recepción de una GchangesCapsule
WriterSACKCapsule	Reconocimiento de la recepción de una WriterSCapsule de un Escritor por parte del Transmisor

Tabla 1: Clases creadas del protocolo para la Edición Cooperativa versión 2.

La mayoría de estas clases fue explicada en la primera versión del protocolo, otras tiene semántica parecida a las explicadas. En cualquier caso, todas las rutinas de estas clases son fácilmente comprensibles si se han entendido los ejemplos explicados hasta este

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

momento. En el anexo I están todas las clases del prototipo que realizamos con esta versión del protocolo. En la figura 4.51 mostramos la interfaz de la aplicación del Receptor.

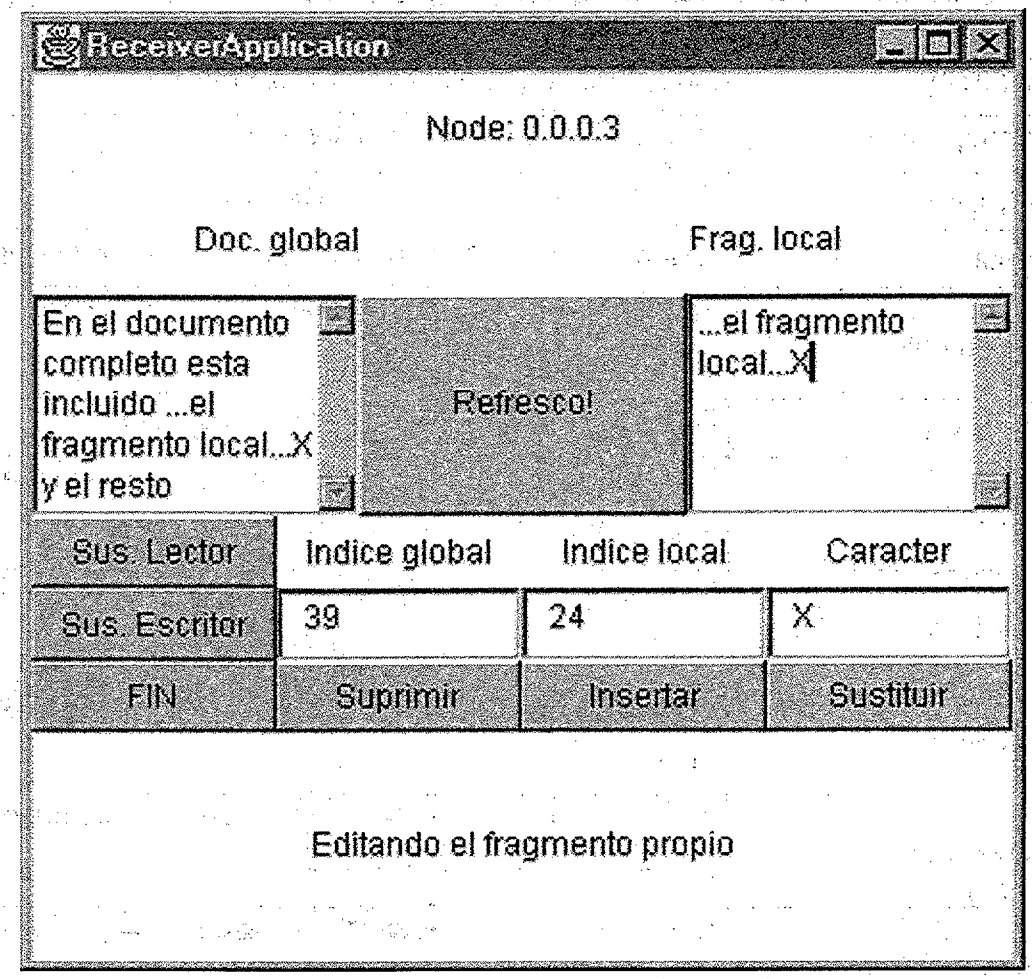


Figura 4.51: Interfaz del prototipo del protocolo de Edición Cooperativa, versión 2.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Capítulo 5. Pruebas y resultados de los protocolos diseñados

En este capítulo se muestra en primer lugar la infraestructura de red utilizada en casi todas las pruebas. Seguidamente y para cada protocolo se describen los mapeados de procesos a máquinas de todas las pruebas realizadas, se exponen los resultados de las mismas y las conclusiones.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

5.1 Introducción

En este capítulo se muestran las pruebas que se han realizado con cada uno de los prototipos diseñados. En cada uno de los apartados explicamos primero qué topologías de red se han utilizado, luego los resultados obtenidos y por último las conclusiones respecto de la viabilidad de cada protocolo.

Antes de explicar los resultados presentamos gráficamente (figura 5.1) un esquema de la infraestructura de red utilizada para realizar dichas pruebas.

Como puede apreciarse, se tratan de tres equipos de sobremesa y dos ordenadores portátiles. Las características hardware fundamentales se muestran en la figura 5.1. El software instalado es el siguiente: El sistema operativo instalado en todas las máquinas excepto en la Máquina C, es la distribución de Linux de Red Hat versión 7.0. El sistema operativo de la Máquina C es la distribución de Linux de Debbian versión 2.2. En todas las máquinas se instaló el jdk (*java developers kit*) versión 1.3, y el TCL/TK (tool command language) versión 8.0. Por último, se instaló en todas las máquinas el ANTS versión 1.0 puesta en la red en Noviembre de 1997 por los autores.

Para el clúster inalámbrico se ha utilizado una tarjeta ISA en la Máquina A y una tarjeta inalámbrica PCMCIA en cada uno de los ordenadores portátiles. La tasa de bits por segundo nominal es de 2 Mbps.

Algunas pruebas no han sido realizadas utilizando la infraestructura de la figura 5.1. En ese caso se explica qué infraestructura se utiliza.

Por último hay que explicar que los datos capturados en todas las pruebas son extraídos de ficheros ".log" a los que se ha volcado la información necesaria.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

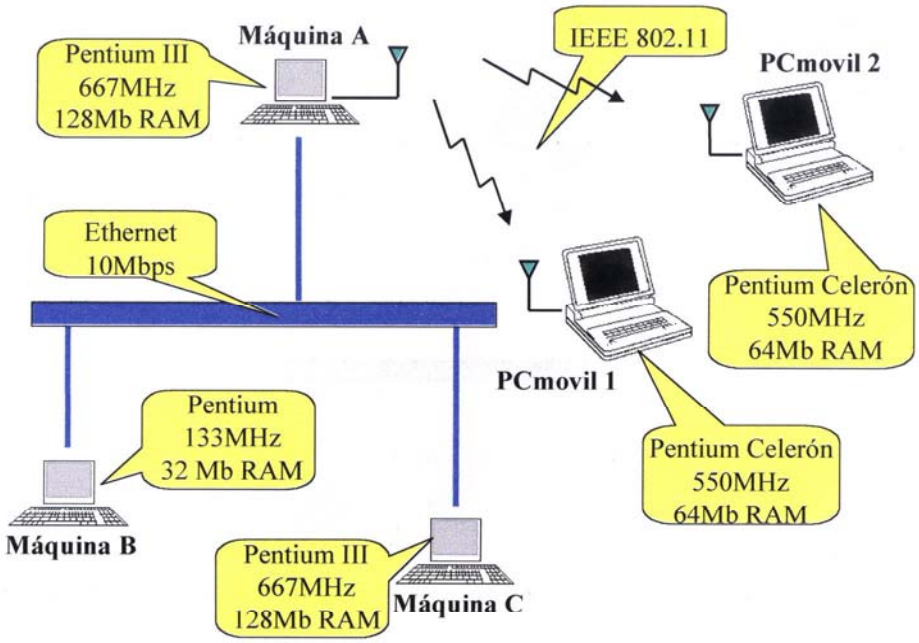


Figura 5.1: Infraestructura de red utilizada en la mayoría de las pruebas realizadas

5.2 Protocolo de Matriz por Matriz

En las figuras 5.2, 5.3 y 5.4 mostramos el mapeado de los procesos a máquinas de las tres pruebas realizadas con este prototipo. Como puede apreciarse, en las tres pruebas se ejecutan seis aplicaciones *esclavo* conectadas con una aplicación *maestro*. La diferencia estriba en que en la primera topología se utiliza el segmento cableado de la infraestructura de la figura 5.1 (Máquinas A, B y C), en la segunda topología se utiliza el cluster inalámbrico (Máquina A, Pcmovil1 y Pcmovil2), y en la tercera todos los procesos se ejecutan en la Máquina A.

El parámetro que hemos medido en las pruebas realizadas con este prototipo es la latencia del cálculo. Es decir, el tiempo desde que comenzamos a distribuir los datos entre todos los procesos hasta que recibimos todos los resultados.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

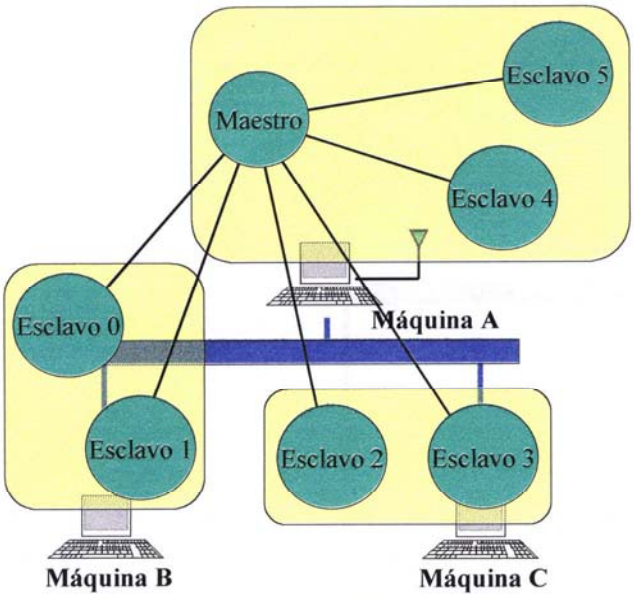


Figura 5.2: Procesos lanzados en la Configuración A del prototipo de Matriz por Matriz

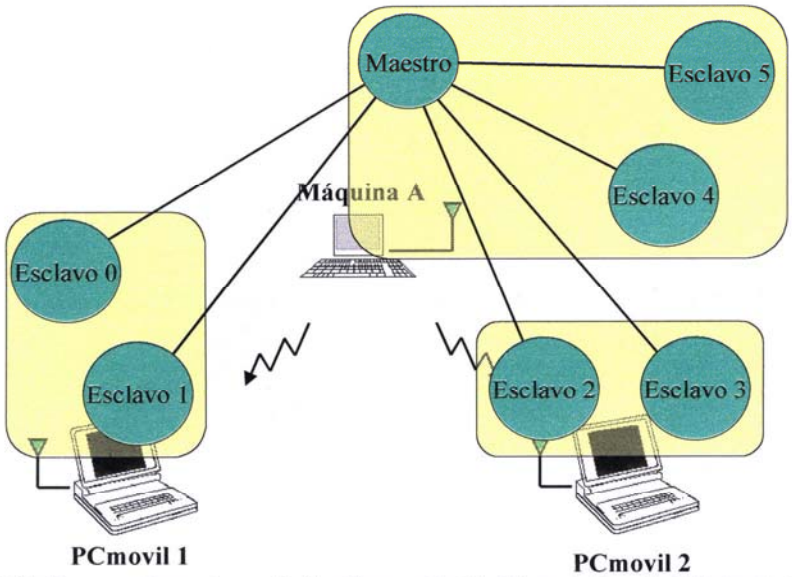


Figura 5.3: Procesos lanzados en la Configuración B del prototipo de Matriz por Matriz

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

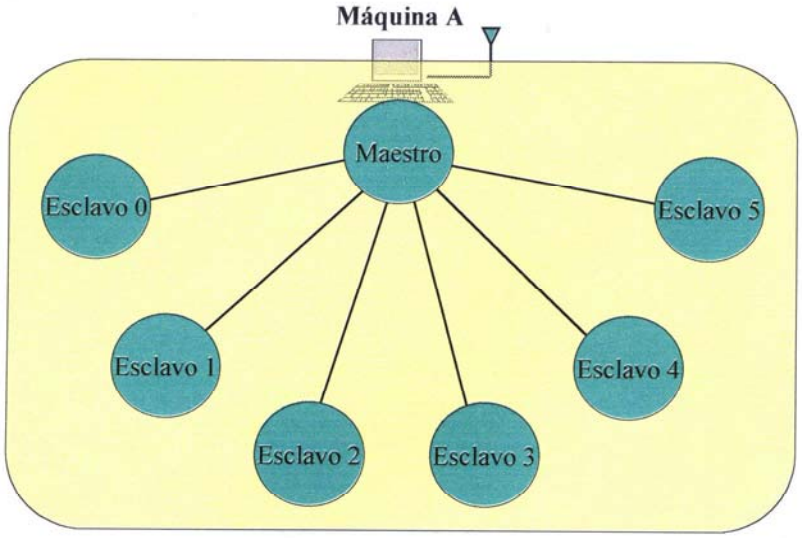


Figura 5.3: Procesos lanzados en la Configuración C del prototipo de Matriz por Matriz

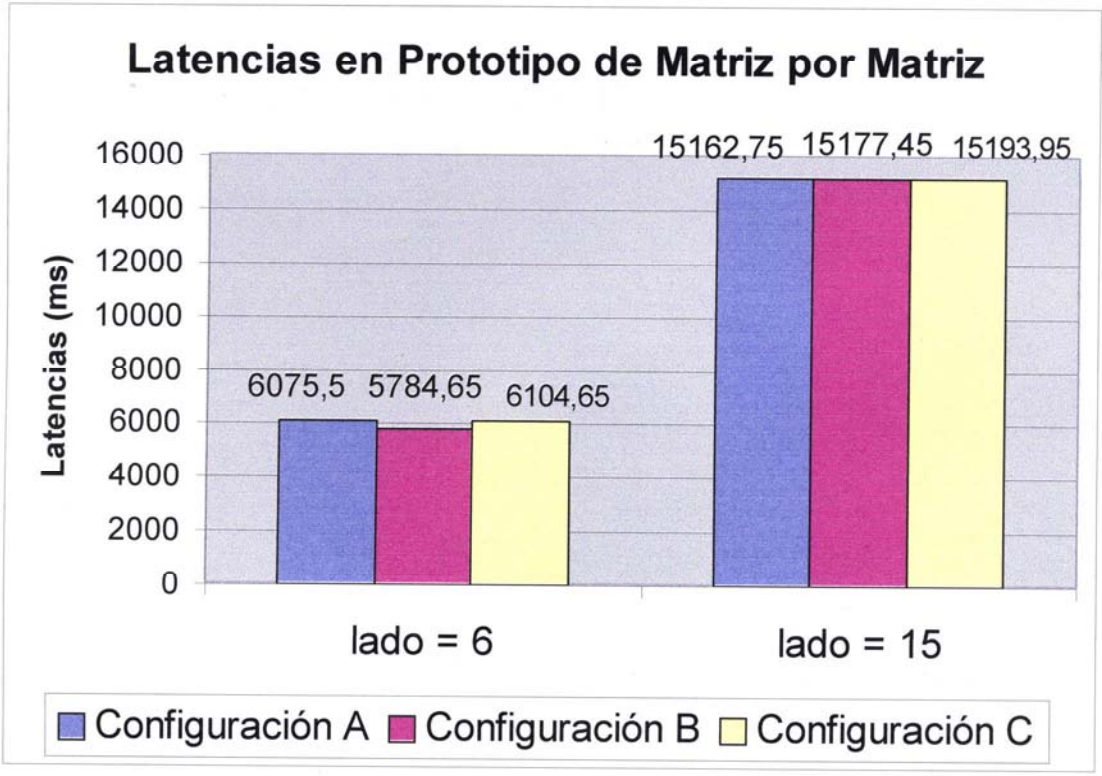


Figura 5.5: Latencias medias en las tres configuraciones

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

5.2.1 Resultados de las pruebas del prototipo de Matriz por Matriz

En la figura 5.5 mostramos la media de las latencias registradas en las dos pruebas. En ambas configuraciones hemos realizado pruebas para tamaño de las matrices de 6 y 15 filas. Debido a que la transmisión de la matriz B se realiza en una sola cápsula, el tamaño máximo de los paquetes UDP no permite realizar productos con este prototipo con matrices de tamaño superior a 18 filas.

5.2.2 Conclusiones del prototipo de Matriz por Matriz

En primer lugar, si comparamos la latencia media del producto distribuido de matrices de 6 filas con la de 15 filas podemos observar el lógico incremento en el tiempo de ejecución del algoritmo distribuido. Podemos apreciar como el incremento en tiempo de ejecución es lineal respecto al incremento en el tamaño de la matriz.

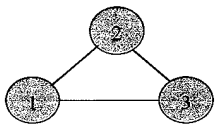
En segundo lugar, comparando para un mismo valor de filas de la matriz, podemos apreciar un ligero ahorro en tiempo de ejecución en las configuraciones A y B (Figuras 5.2 y 5.3 respectivamente) frente al de la configuración C (Figura 5.3). En concreto las configuraciones A y B suponen un ahorro de un 0,48% y un 5,53% respectivamente en tiempo de ejecución para matrices de 6 filas. Para matrices de 15 filas este ahorro es de un 0,21% y un 0,11% respectivamente.

Concluimos este apartado diciendo que el objetivo de este protocolo no era obtener un buen rendimiento en un cálculo distribuido. En ese caso no hubíeramos utilizado un software tan *"pesado"*. Por otra parte, respecto de la obtención de un primer conocimiento del uso del ANTS como herramienta de prototipado rápido de protocolos, si se han logrado satisfactoriamente los objetivos.

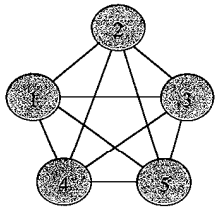
5.3 Protocolo de PingBroadcast versión 1

Para las pruebas realizadas con el prototipo de esta versión del protocolo de Ping con broadcast no se ha utilizado la infraestructura de red de la figura 5.1. Las pruebas realizadas con esta versión inicial han sido realizadas locales en una sola máquina.

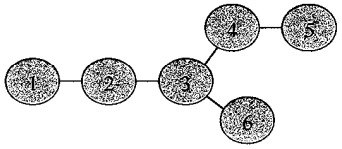
© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004



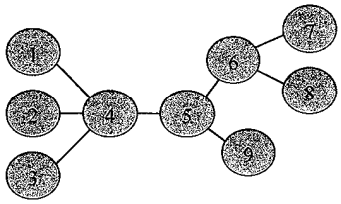
Configuración A



Configuración B



Configuración C



Configuración D

Figura 5.6: Configuraciones de las pruebas con el protocolo de PingBroadcast versión 1

```
connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
```

Configuración A

```
connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.1 0.0.0.4
connect 0.0.0.1 0.0.0.5
```

Configuración B

```
connect 0.0.0.1 0.0.0.2
connect 0.0.0.2 0.0.0.3
connect 0.0.0.3 0.0.0.4
connect 0.0.0.3 0.0.0.6
connect 0.0.0.4 0.0.0.5
```

Configuración C

```
connect 0.0.0.1 0.0.0.4
connect 0.0.0.2 0.0.0.4
connect 0.0.0.3 0.0.0.4
connect 0.0.0.4 0.0.0.5
connect 0.0.0.5 0.0.0.6
connect 0.0.0.5 0.0.0.9
connect 0.0.0.6 0.0.0.7
connect 0.0.0.6 0.0.0.8
```

Configuración D

Figura 5.7: Fragmentos de los ficheros ". config" de cada configuración

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

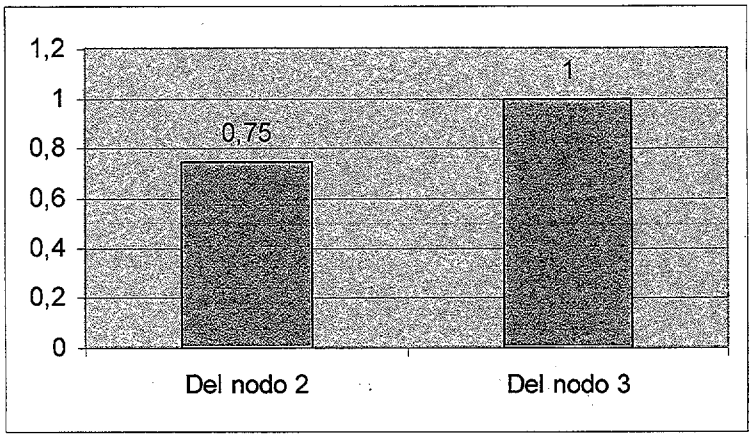


Figura 5.8: Latencias medias normalizadas en la configuración A

El entorno de prueba fue el siguiente: Un Pentium a 133MHz con 24Mb de memoria RAM. El sistema operativo es la versión 6.1 la distribución de Linux de Red Hat. La versión del JDK utilizada es la 1.2. Por último se ha utilizado la versión 5.0 de TCL/TK.

En la figura 5.6 mostramos las distintas configuraciones probadas. Además, en la figura 5.7 ponemos un fragmento del fichero ".config" utilizado con cada configuración. En el Anexo II y en el CD adjunto se muestran los ficheros de configuración de todas las pruebas de este Proyecto Fin de Carrera. En el siguiente apartado se muestran los resultados a cada una de estas pruebas.

5.3.1 Resultados de las pruebas del prototipo de *PingBroadcast* v1

En las figuras de la 5.8 a la 5.11 se muestran los valores de las medias de las latencias normalizadas. Entendemos por latencia entre un nodo *origen del ping* y otro al tiempo que pasa desde que el primero lanza un ping hasta que recibe la confirmación del segundo. Hay que destacar que como las configuraciones A y B son simétricas, se muestran sólo los retardos desde un nodo *origen del ping*.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

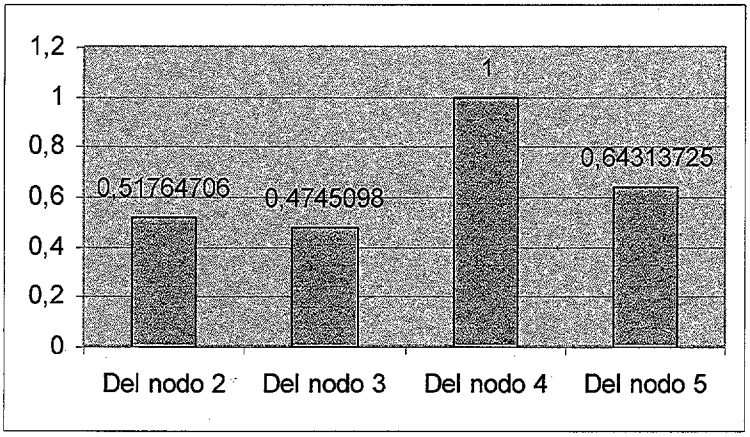


Figura 5.9: Latencias medias normalizadas en la configuración B

Hay que destacar algunos detalles respecto de las pruebas realizadas. Hay que tener en cuenta que todas las pruebas realizadas han sido ejecutadas localmente en la misma máquina. Esto tiene la consecuencia negativa de que cuantos más nodos tenga la configuración bajo prueba, peor se ejecutaría el sistema. Sería preferible realizar una batería de pruebas situando los nodos en diferentes máquinas, de manera que el resultado no se viera distorsionado por el tamaño de la red.

Teniendo en cuenta los pocos recursos de que se disponían a la hora de realizar estas cuatro pruebas hemos optado por no probar configuraciones mayores, que alejarían aún más el resultado de la realidad de una red con nodos en diferentes máquinas.

Como consecuencia de esto, cuanto mayor sea la configuración, mayores latencias presentan las cápsulas, tanto por el mayor número de nodos visitados como por la competencia por los escasos recursos disponibles.

El *periodoRefresco* es el margen de tiempo que configuramos para discriminar cuando se trata de cápsulas de inundación de un nuevo ping de cuando se trata de *ecos* de un ping ya recibido. Por lo dicho antes, el *periodoRefresco* debe ser, en nuestra ejecución local, configurado dependiendo del número de nodos de la prueba. Nosotros hemos puesto el valor de 500 mseg para las pruebas primera y segunda, 1000 mseg para la tercera y 2000 mseg para la cuarta prueba.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

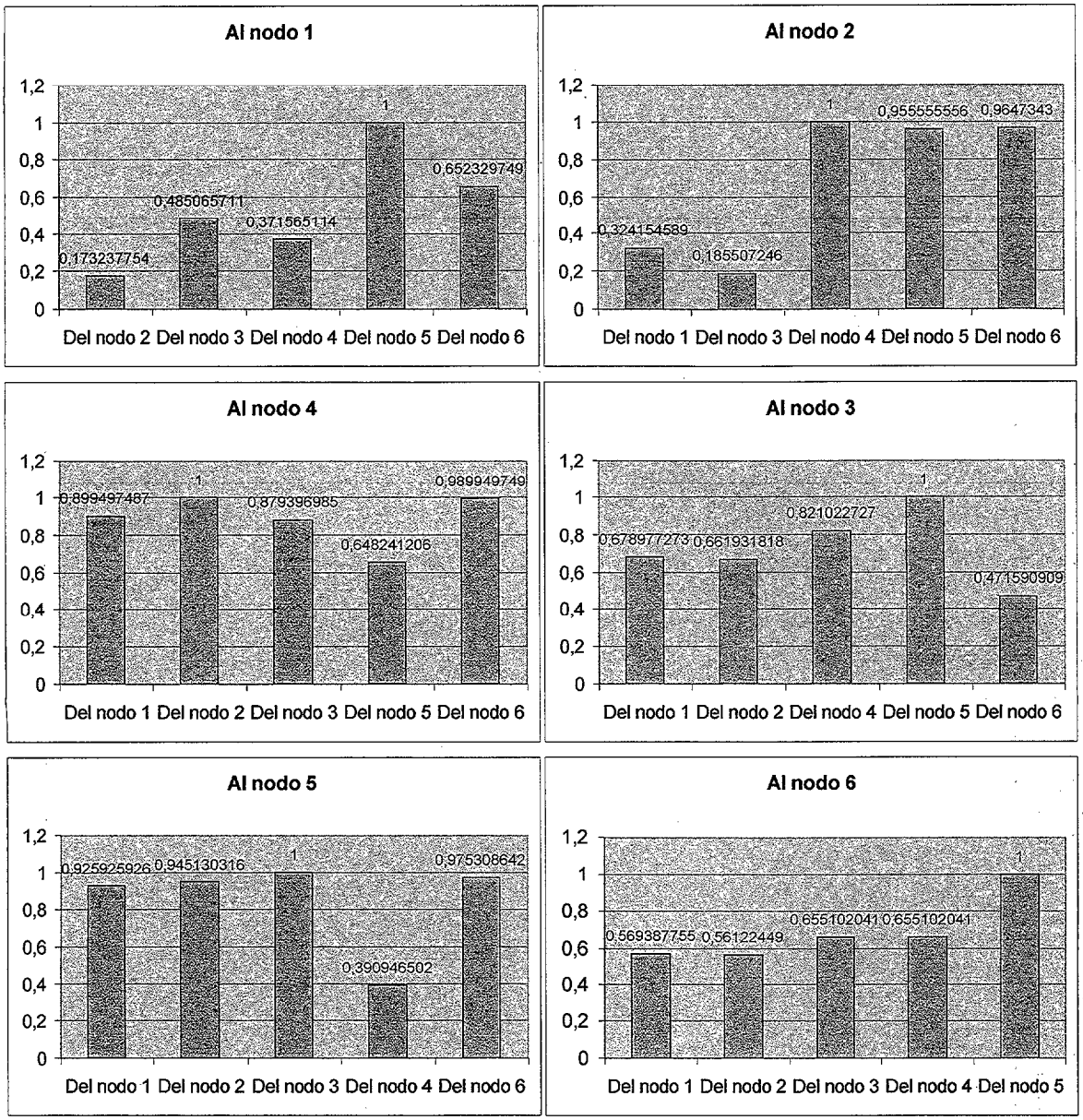


Figura 5.10: Latencias medias normalizadas en la configuración C

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

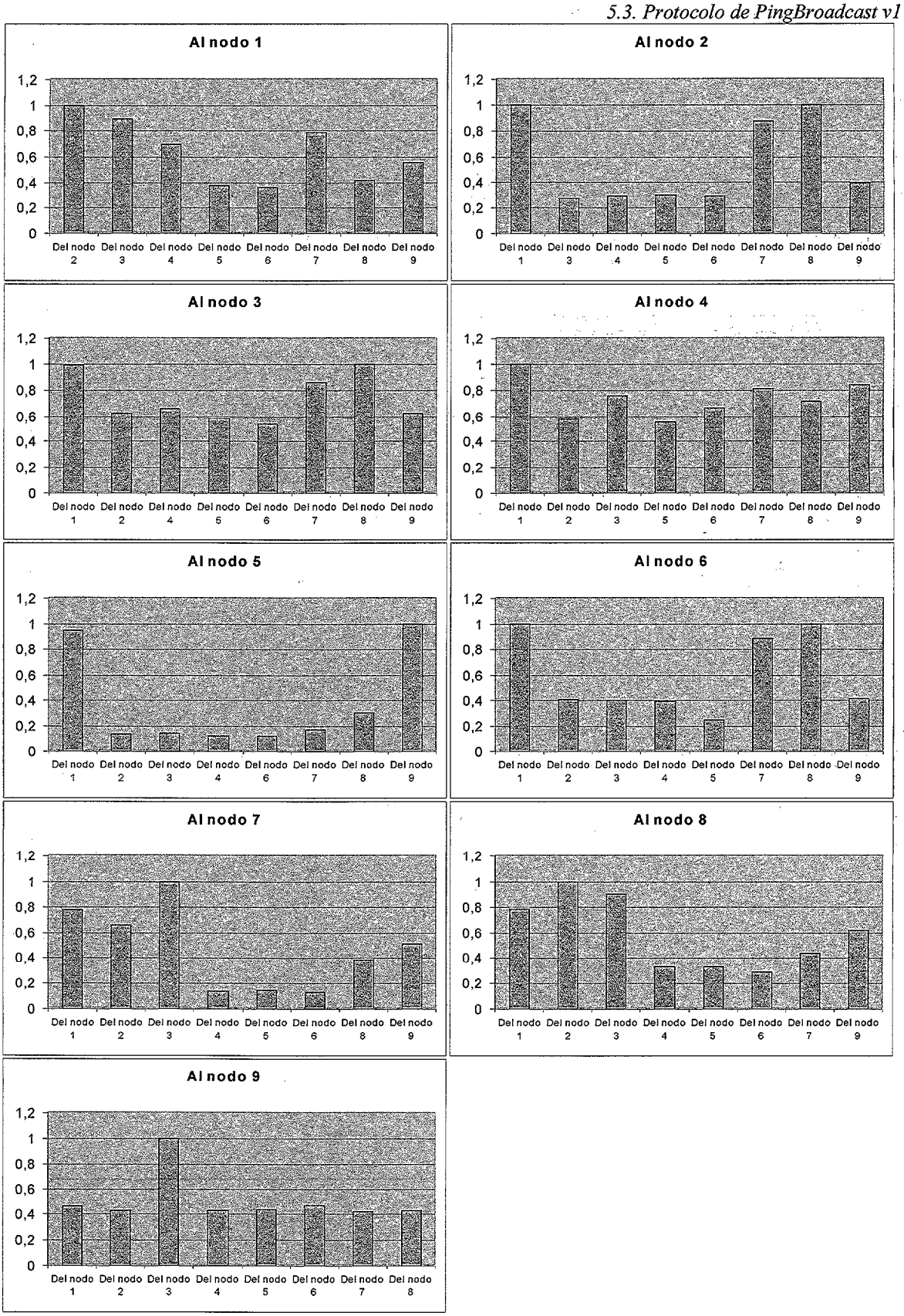


Figura 5.11: Latencias medias normalizadas en la configuración D

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

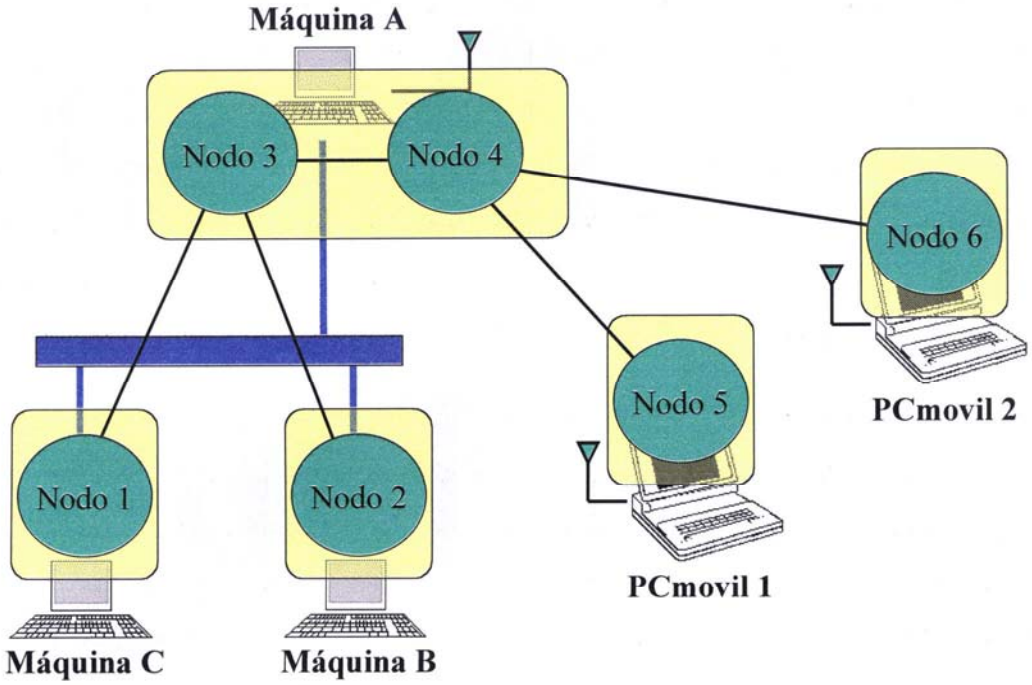


Figura 5.12: Mapeado de procesos a máquinas en el prototipo de PingBroadcast v2

Aún así, en la realización de las pruebas 3ª y 4ª en ocaciones sucedían periodos *transitorios* en los que *ecos* de un mismo ping se tomaban como nuevos pings. Los nodos a los que les afectaba más estos periodos transitorios son aquellos que eran *concentradores* de varias rutas, mientras que los nodos *finales* casi no sufrían este problema.

5.3.2 Conclusiones del prototipo de PingBroadcast v1

Las conclusiones que podemos extraer de las pruebas realizadas con este protocolo son, en líneas generales, satisfactorias. En primer lugar hemos podido comprobar la validez del protocolo diseñado con el uso de la plataforma. Una vez más se comprueba que esta plataforma nos permite el prototipado rápido de protocolos.

A pesar de los problemas con la escasez de recursos de los que hablábamos antes, podemos observar que las pruebas realizadas demuestran que una implementación distribuida de este protocolo podría ser de gran utilidad en redes en las cuales es necesario comprobar el estado de los enlaces periódicamente, por ejemplo redes WLAN's.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Queremos destacar la virtud de la *"ligereza"* de este protocolo a costa de una baja fiabilidad. Nos parece que es una buena aproximación al problema dado que, para aplicaciones que requieran una gran fiabilidad en el testeo de la red, siempre se puede hacer más robusto este protocolo desde el nivel de aplicación. Sin embargo, otras aplicaciones menos exigentes o aplicaciones paralelas, se pueden beneficiar de la rapidez y sencillez de este protocolo. Esto es, porque en este tipo de aplicaciones, generalmente se prima la velocidad de ejecución del protocolo frente a la fiabilidad del mismo.

Por último hay que destacar que una implementación de este protocolo a nivel de red mejoraría el rendimiento del mismo, porque se disminuirían los niveles de software necesarios al ejecutarlo directamente sobre IP. Sin embargo; téngase en cuenta que esto llevaría un coste elevado en la producción del software activo de red. Por eso investigamos otra mejora que se le puede realizar a este protocolo, que es la que presentamos en la siguiente versión trasladando el mecanismo de descarte de *pings* no válidos al nivel de red del ANTS.

5.4 Protocolo de PingBroadcast v2

Para realizar las pruebas con el prototipo de esta versión del protocolo de PingBroadcast se ha utilizado la infraestructura que se expone en la figura 5.1. Sobre esta infraestructura se lanzaron los procesos que se muestran en la figura 5.12.

Como puede verse, se ha lanzado en cada nodo una aplicación del prototipo, excepto en la máquina A, en la cual se han lanzado dos, una para la interfaz de red cableada y otra para la interfaz inalámbrica. En el Anexo II y en el CD adjunto se muestra el fichero de configuración de esta prueba.

Para la realización de las pruebas se ha procedido a disparar ráfagas de 20 peticiones de *ping* (1 *ping* cada 100 milisegundos) en todos los nodos de la topología. La dimensión de la cola FIFO que discrimina los ecos de *pings* ya recibidos (MAX_SELLOS) es de 10. Para cada petición se anotaba el tiempo en el que se lanzaba y se comparaba con el tiempo de respuesta de cada uno de los demás nodos. En la figura 5.13 se muestran las latencias medias de las pruebas realizadas.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

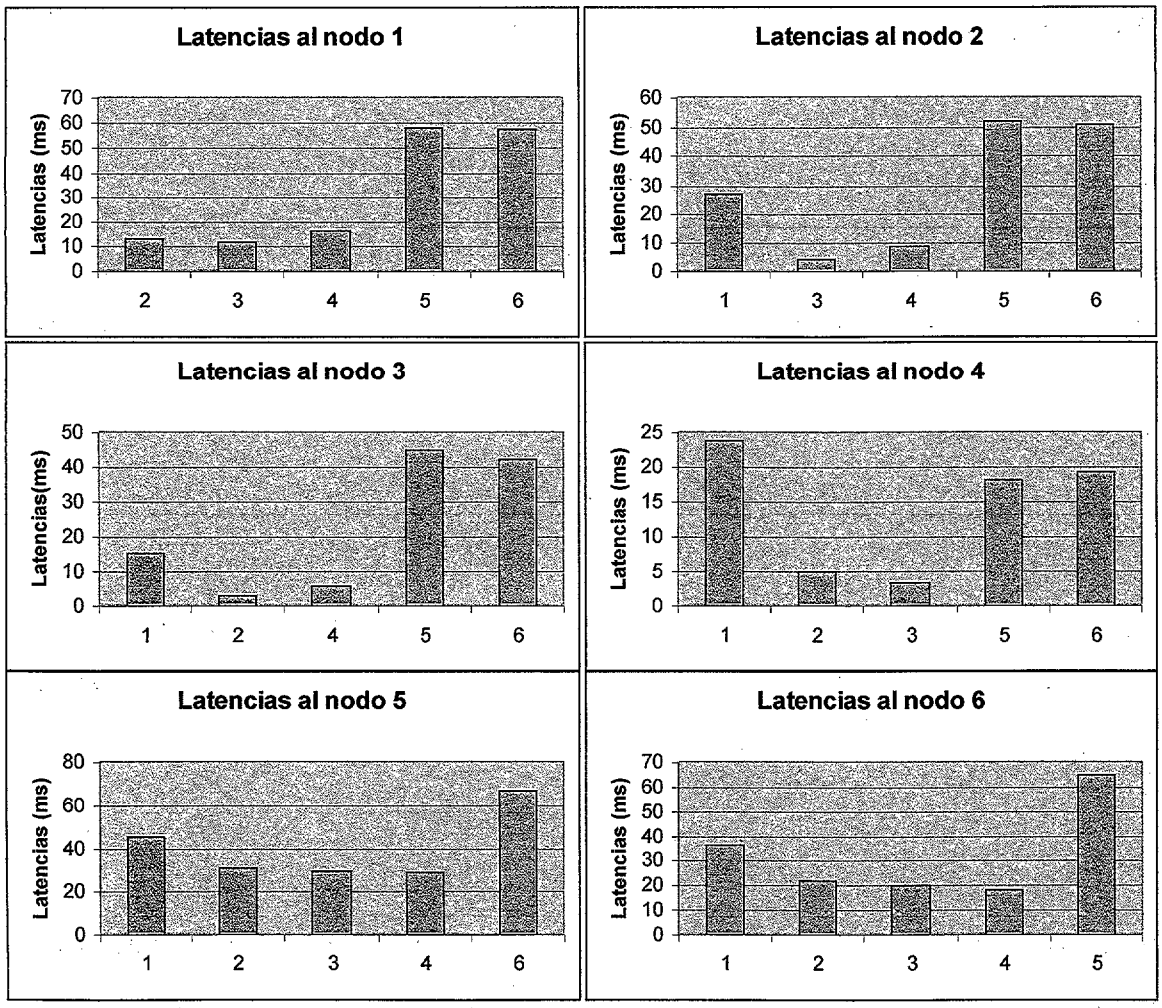


Figura 5.13: Latencias a cada nodo en el prototipo de PingBroadcastv2

5.4.1 Resultados de las pruebas del prototipo de PingBroadcast v2

En cada uno de los diagramas de barras de la figura 5.13 se muestra la latencia media desde que el nodo *origen del Ping* lanzaba una petición hasta que se recibía la respuesta de cada nodo.

En la tabla 5.1 se muestran la media total, el valor mínimo y el máximo de las latencias medias obtenidas durante las pruebas de este prototipo.

Latencia media total (ms)	28,1415349
Valor máximo de latencia (ms)	67,2780269
Valor mínimo de latencia (ms)	3,25555556

Tabla 5.1: Valores medios característicos en las pruebas del prototipo de PingBroadcast v2

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

5.4.2 Conclusiones del prototipo de PingBroadcast v2

A la vista de los resultados mostrados en la figura 5.13 y en la tabla 5.1, podemos decir qué el prototipo tiene un funcionamiento correcto.

Si nos fijamos en los diagramas de barras de la figura 5.13 se puede apreciar un efecto interesante. Según nos alejamos del nodo *origen del ping* la latencia de la respuesta a la petición de *ping* aumenta considerablemente. Esto se debe a que los paquetes de inundación, y las consiguientes confirmaciones deben atravesar un mayor número de nodos. Si a esto sumamos la existencia de segmentos de red inalámbrica, y la diversidad en la potencia de cálculo de las máquinas utilizadas, se entiende aún mejor este efecto.

Además hay que tener en cuenta que está ejecutándose en el nivel de aplicación y sobre una máquina virtual de Java. Sería de esperar que su implementación en un nivel inferior de la torre de protocolos mejorara las latencias.

Por otra parte, como puede apreciarse en la tabla 5.1, la latencia media mayor de las mostradas es de 67 milisegundos y la menor de 3 milisegundos. Este margen nos parece adecuado para la aplicación de este protocolo para la comprobación periódica expuesta en el capítulo 4. Aplicaciones de cálculo distribuido, de gestión de recursos de red, etc. en entornos inalámbricos o combinados cableados-inalámbricos, podrían requerir el uso de un protocolo similar.

Una característica importante de este protocolo es su *"ligereza"* manteniendo una medida aceptable de la latencia.

Por último hacer notar que este protocolo ha sido de gran utilidad como primera aproximación al diseño de un protocolo en una LAN-WLAN y su implementación y prueba con el ANTS. Nos ha permitido adquirir los conocimientos necesarios antes de acometer el diseño de protocolos multicast.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

5.5 Protocolo de Multicast en Tiempo Real v1

Para probar este protocolo hemos utilizado la Máquina A de la figura 5.1. Es decir, se ha probado localmente en la misma máquina por la necesidad de sincronización entre todos los nodos que tiene esta versión del protocolo. Se han realizado pruebas con las tres configuraciones que se muestran en la figura 5.14. En el Anexo II y en el CD adjunto se muestran los ficheros de configuración.

Para cada una de las configuraciones se ha realizando una batería de pruebas variando: el número de nodos suscritos al servicio y el *margenRT*. En la tabla 5.2 se presenta una lista de las 36 pruebas realizadas detallando configuración, *margenRT* y nodos suscritos al servicio. El tiempo de duración de cada prueba es aproximadamente de 300 segundos (5 minutos). En la última columna se expresa el número de la prueba entre paréntesis.

En la tabla 5.3 se muestran los parámetros que se han estudiado en cada prueba que dependen del tipo de aplicación que se trata.

5.5.1 Resultados de las pruebas del prototipo de Multicast en Tiempo Real v1

A continuación mostramos gráficamente los resultados relevantes que se han extraído del procesamiento de los datos capturados durante las pruebas. Hemos querido reflejar dos aspectos: el efecto de reducir el *margenRT* en el sistema y el impacto de la incorporación de un nuevo suscriptor. Para elloEn la figura 5.15 se reflejan las pérdidas en cada nodo cuando están todos los nodos suscritos variando el *margenRT*, para las tres configuraciones. La figura 5.16 nos muestra la latencia media en cada nodo, también con todos los nodos suscritos, variando el *margenRT*. La figura 5.17 muestra las pérdidas medias en cada nodo para diferente cantidad de nodos suscritos para las tres configuraciones. Por último, la figura 5.18 refleja las latencias medias en cada nodo también para diferente cantidad de nodos suscritos y todas las configuraciones.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Configuración	*MargenRT*	Nodos suscritos(# de la prueba)
Configuración A	10000	Todos suscritos(1); Nodo 3(2); Nodos 3 y 5(3)
	1000	Todos suscritos(4); Nodo 3(5); Nodos 3 y 5(6)
	300	Todos suscritos(7); Nodo 3(8); Nodos 3 y 5(9)
	100	Todos suscritos(10); Nodo 3(11); Nodos 3 y 5(12)
	30	Todos suscritos(13); Nodo 3(14); Nodos 3 y 5(15)
Configuración B	1000	Todos suscritos(16); Nodo 3(17); Nodos 3 y 5(18); Nodos 3, 5 y 7(19)
	100	Todos suscritos(20); Nodo 3(21); Nodos 3 y 5(22); Nodos 3, 5 y 7(23)
	30	Todos suscritos(24); Nodo 3(25); Nodos 3 y 5(26); Nodos 3, 5 y 7(27)
Configuración C	1000	Todos suscritos(28);Nodos 4 y 5(29);Nodos 4, 5, 10 y 12(30)
	100	Todos suscritos(31);Nodos 4 y 5(32);Nodos 4, 5, 10 y 12(33)
	30	Todos suscritos(34);Nodos 4 y 5(35);Nodos 4, 5, 10 y 12(36)

Tabla 5.2: Pruebas realizadas en el prototipo de MulticastRT v1

5.5.2 Conclusiones del prototipo de Multicast en Tiempo Real v1

Las conclusiones a las que hemos llegado las dividimos en dos aspectos. En primer lugar, respecto al efecto de disminuir el *margenRT* del sistema, si nos fijamos en las figuras 5.15 y 5.16 para las tres configuraciones podemos apreciar lo siguiente: A medida que reducimos el margen de tiempo real, se incrementa en las tres configuraciones las pérdidas de cápsulas. Esto sucede porque hay más cápsulas que se descartan por quedar fuera del margen de validez que hemos establecido globalmente.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Sender	Cápsulas multicast/segundo
Nodo Intermedio	Cápsulas multicast/segundo Suscripciones/segundo Cápsulas NACK/segundo N° de peticiones de retransmisión/segundo
Receiver	Cápsulas multicast/segundo Pérdidas de cápsulas (%) Latencia (mseg) Retransmisiones/seg

Tabla 5.3: Parámetros medidos en las pruebas

Con las latencias se produce el siguiente efecto. Si disminuimos el *margenRT* en general se produce una disminución notoria de la latencia media. Este comportamiento es lógico porque al disminuir el *margenRT*, las cápsulas con latencias mayores son descartadas. Por lo tanto, las cápsulas que llegan, a pesar de que son menos, llegan con una menor latencia.

En segundo lugar observamos el efecto que causa la incorporación de nuevos nodos al grupo multicast en el servicio recibido por los demás. En la figura 5.17 se representa el porcentaje medio de las pérdidas (de todos los márgenes de tiempo real) variando la cantidad de nodos suscritos al servicio. Como puede verse, en las tres configuraciones, las pérdidas se contienen a medida que se incorporan nuevos nodos al servicio. Esto es de vital importancia en un servicio multicast ya que no es justo que los nodos que están previamente suscritos vean empeorado el servicio que reciben ante una nueva incorporación.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

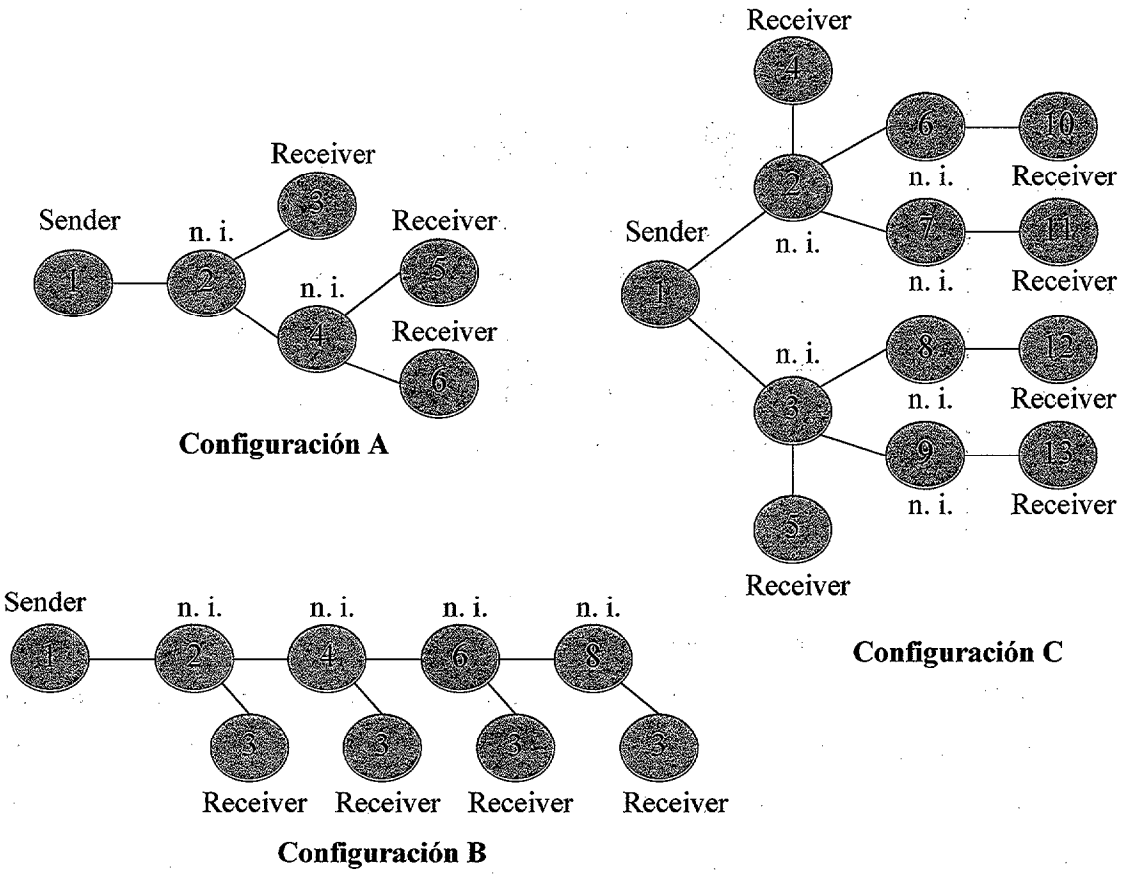


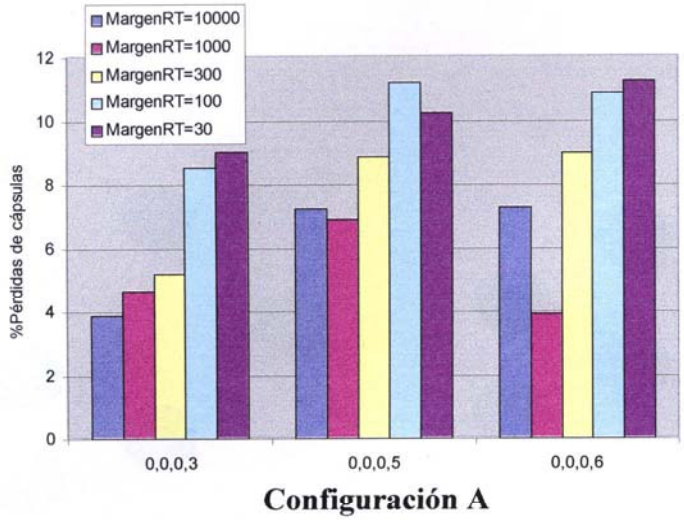
Figura 5.14: Configuraciones probadas en el prototipo de Multicast RT v1

En la figura 5.18 vemos este mismo efecto reflejado en la latencia media. Podemos ver la latencia media no aumenta con las nuevas incorporaciones. Es decir, si utilizamos la latencia y las pérdidas como índices indicativos del servicio que reciben los nodos debemos evitar que nuevas incorporaciones al servicio multicast provoquen que estos dos factores empeoren en los demás usuarios. En estas pruebas podemos ver como en la mayoría de los casos estos dos parámetros no sufren mermas significativas.

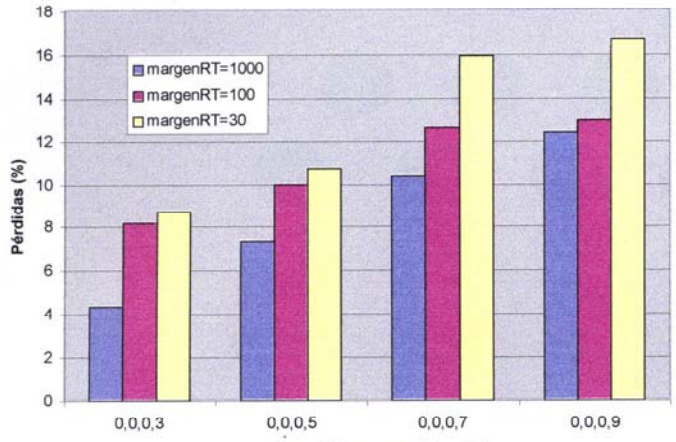
Por todo esto creemos que el protocolo diseñado y probado demuestra un comportamiento adecuado para grupos multicast que varíen dinámicamente. Esta flexibilidad y ampliabilidad dinámica lo hace óptimo para su uso en redes heterogéneas LAN-WLAN.

Además el diseño de este protocolo, y su implementación y prueba con el ANTS nos ha permitido adquirir un conocimiento añadido en los protocolos multicast.

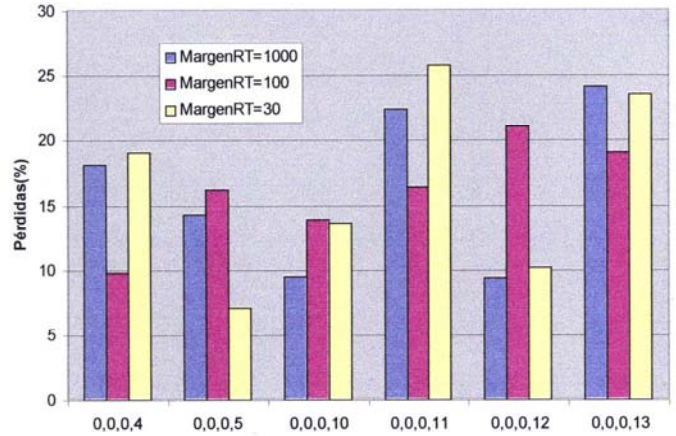
© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004



Configuración A



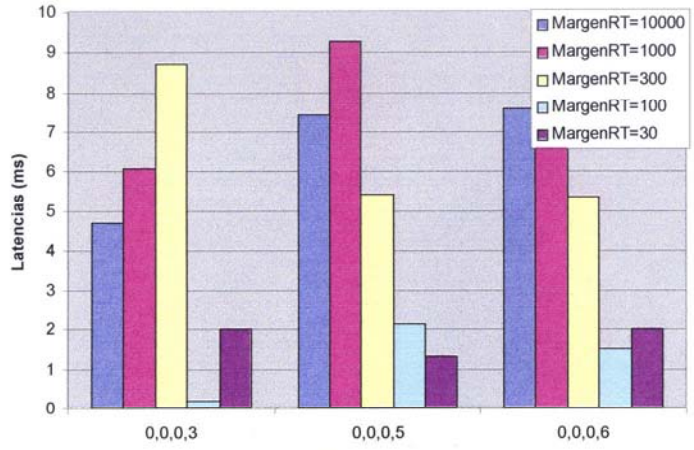
Configuración B



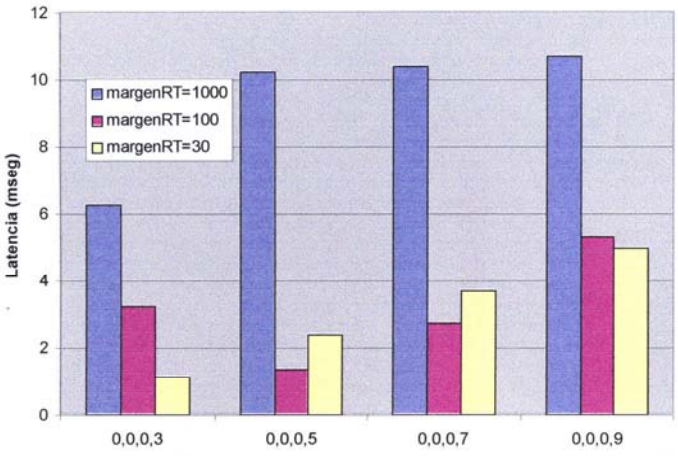
Configuración C

Figura 5.15: Pérdidas medias en el prototipo de Multicast RT v1

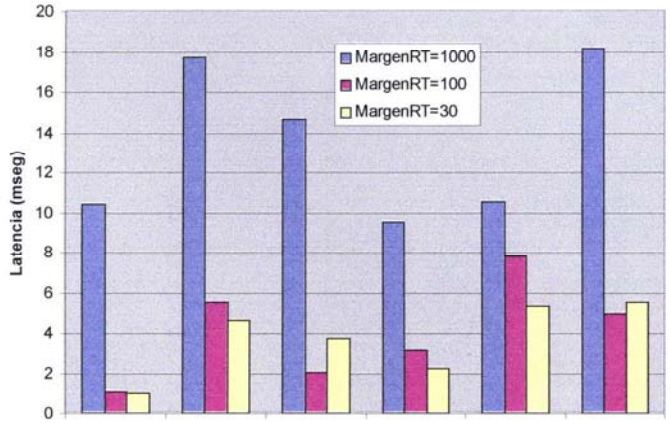
© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004



Configuración A



Configuración B



Configuración C

Figura 5.16: Latencias medias en el prototipo de Multicast RT v1

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

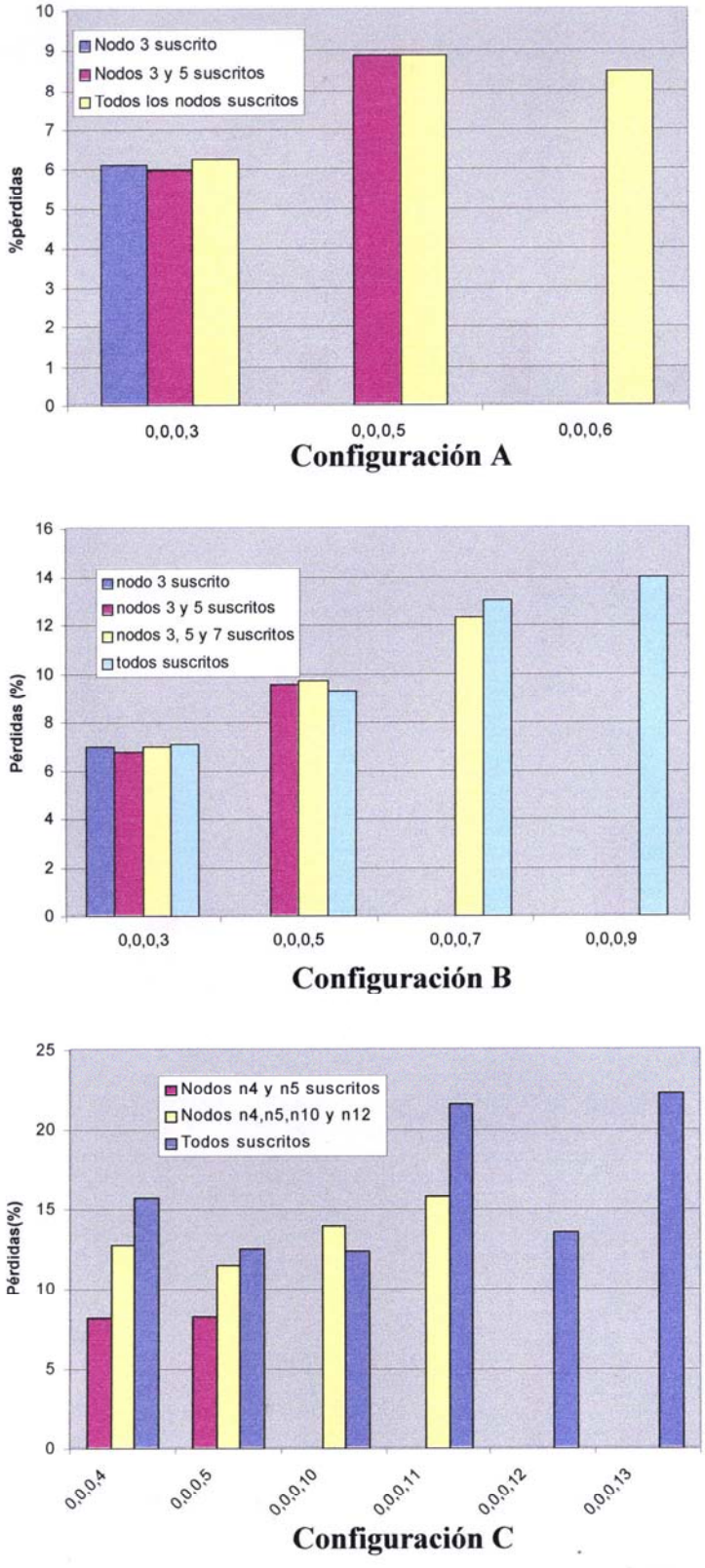
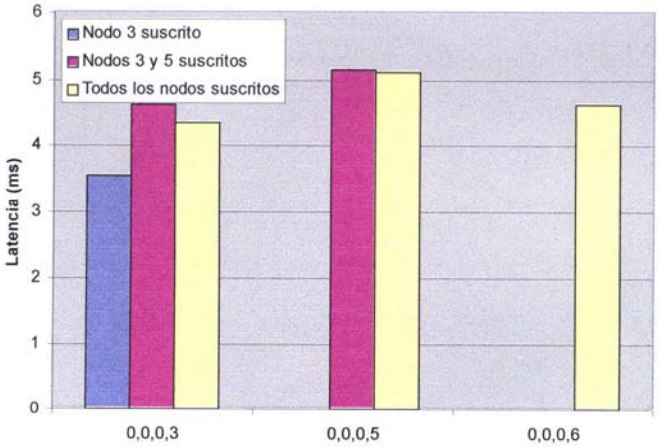
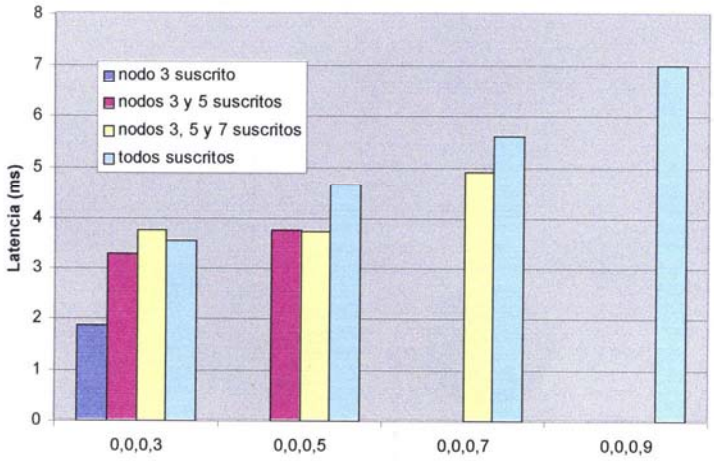


Figura 5.17: Efecto suscripción (Pérdidas)

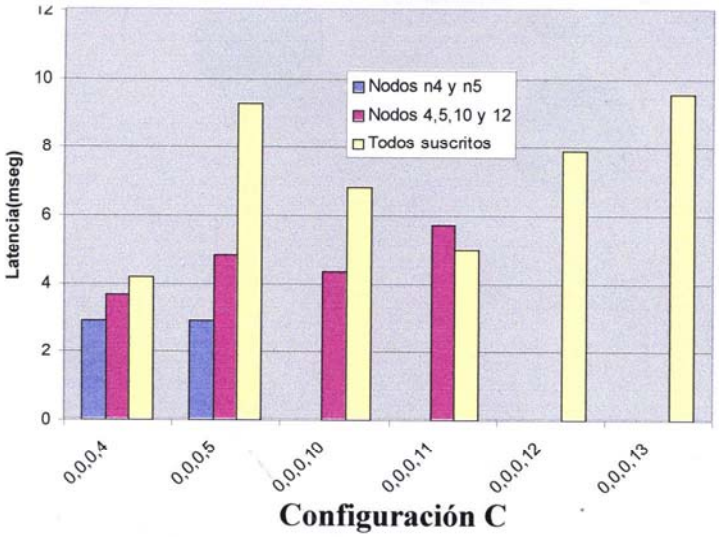
© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004



Configuración A



Configuración B



Configuración C

Figura 5.18: Efecto de la suscripción (Latencia)

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

5.6 Protocolo de Multicast en Tiempo Real v2

Basándonos en esa infraestructura de la figura 5.1 hemos diseñado cuatro configuraciones. En las figuras 5.19 (Configuración A), 5.20 (Configuración B), 5.21(Configuración C) y 5.22 (Configuración D) mostramos el mapeado de procesos a máquinas de las mismas. Entre paréntesis se indica el número de nodo (que posteriormente nombramos como *nodo i*).

Como puede apreciarse, en las configuraciones A, B y C se utiliza toda la infraestructura de la figura 5.1, es decir, tanto el segmento cableado como el cluster inalámbrico. Sin embargo, en la configuración D sólo se utiliza el cluster inalámbrico. En el Anexo II y en el CD adjunto se muestran los ficheros de configuración de las 4 topologías.

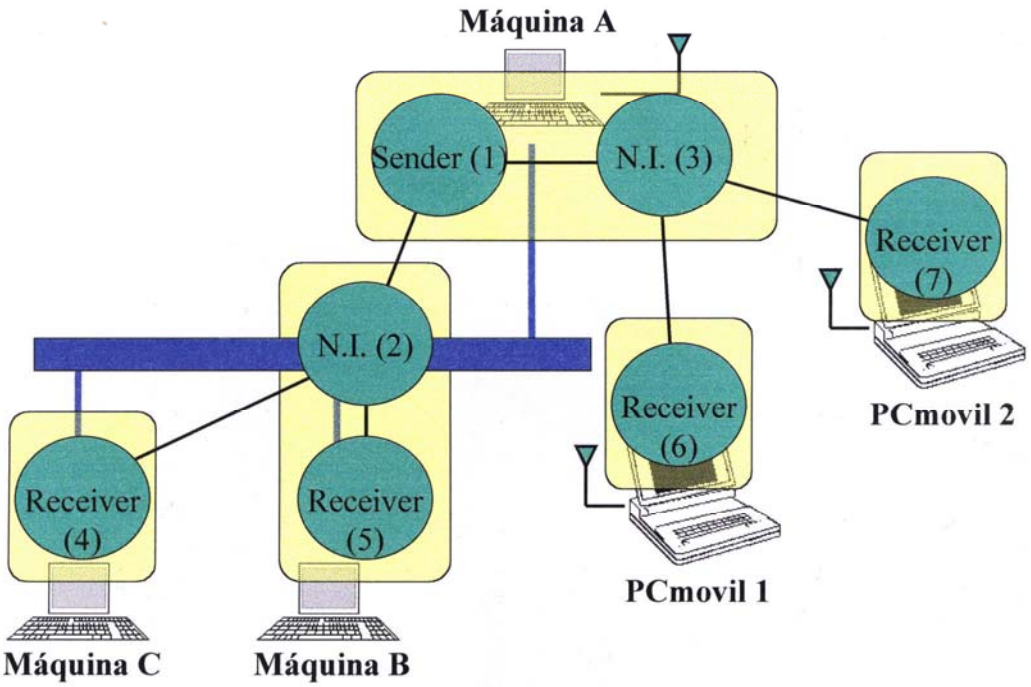


Figura 5.19: Mapeado de procesos en la configuración A para el prototipo de Multicast en RT v2

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

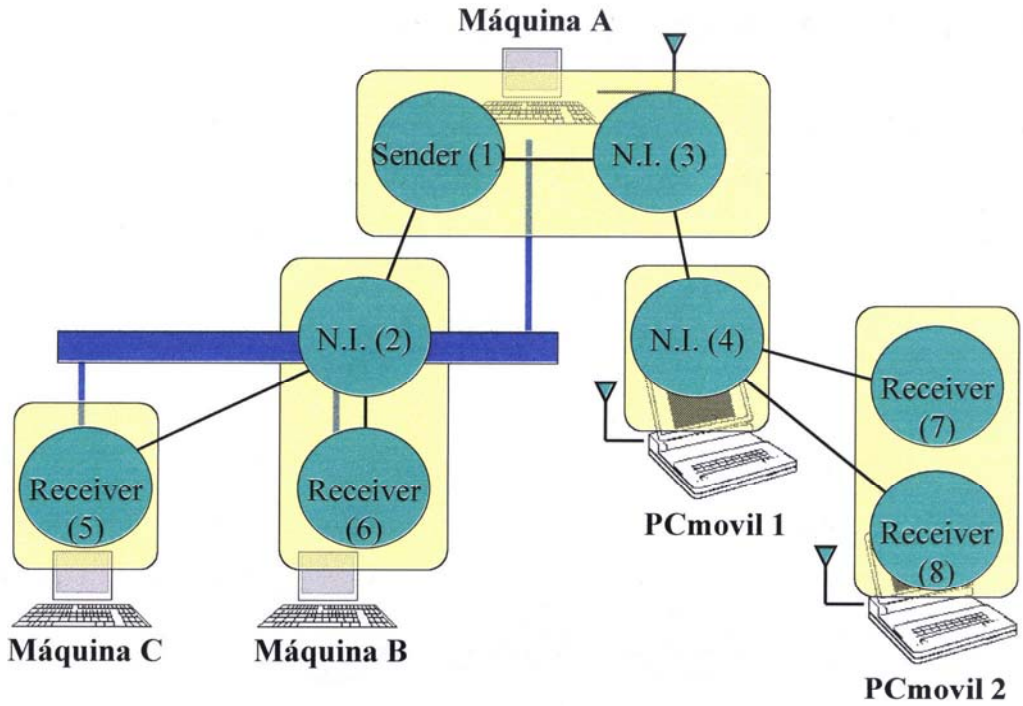


Figura 5.20: Mapeado de procesos en la configuración B para el prototipo de Multicast en RT v2

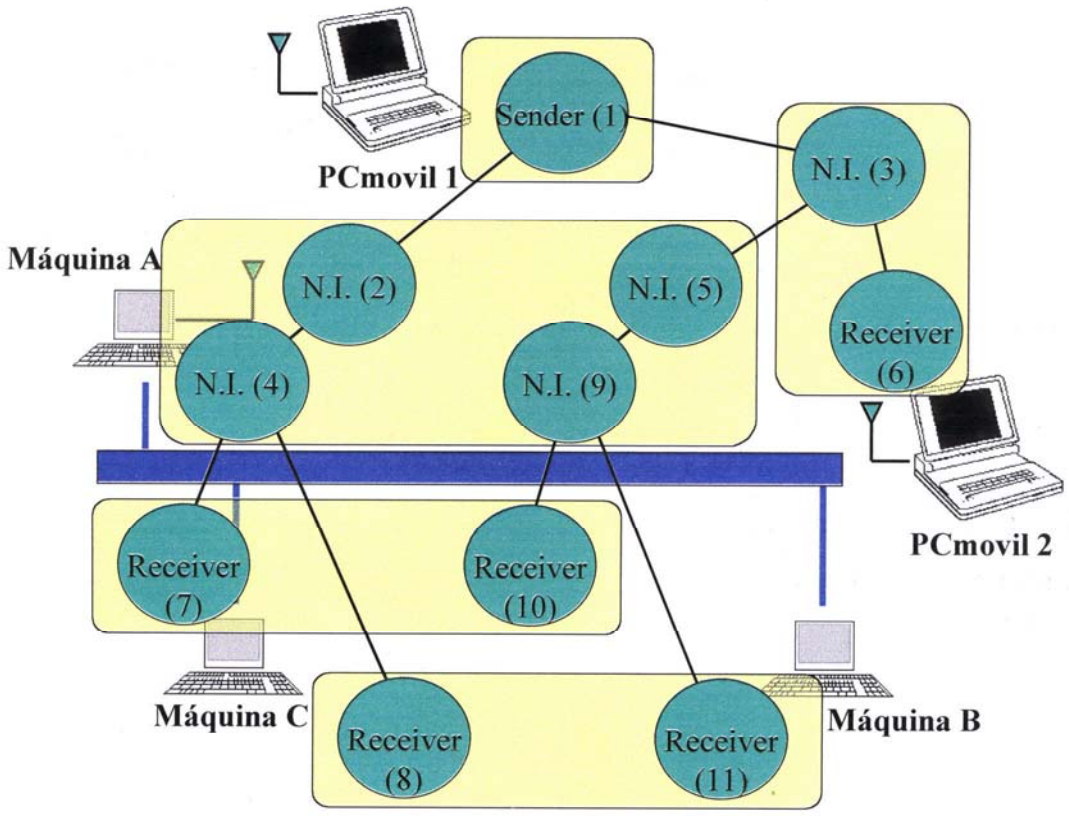


Figura 5.21: Mapeado de procesos en la configuración C para el prototipo de Multicast en RT v2

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

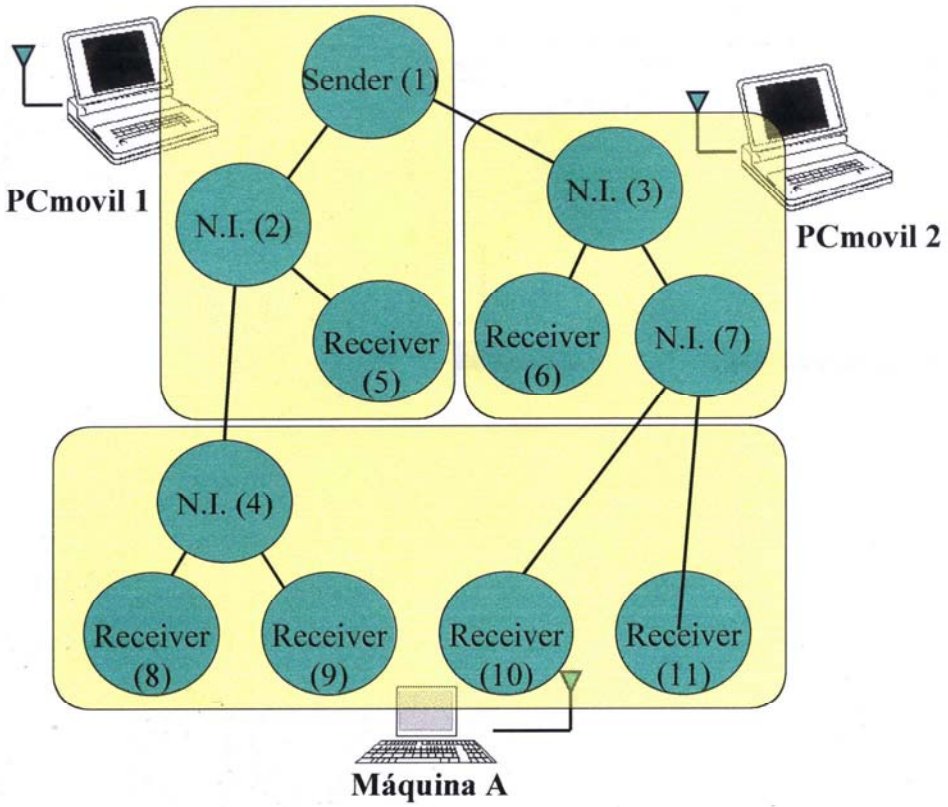


Figura 5.22: Mapeado de procesos en la configuración D para el prototipo de Multicast en RT v2

5.6.1 Resultados de las pruebas del prototipo de Multicast en Tiempo Real v2

Se han realizado cinco tipos de pruebas distintos. Hemos dispuesto la información en cinco tablas, una para cada tipo de prueba. Las cuatro primeras (tablas de la 5.4 a la 5.7) se corresponden con las configuraciones A, B, C y D respectivamente.

La tabla 5.8 se obtuvo con la configuración D, pero con la diferencia de que se introdujeron errores intencionadamente. En concreto, la quinta prueba se realizó haciendo que el Transmisor (Sender) transmitiera sólo el 80% de las cápsulas que producía. Sin embargo, se guardaban en la cache del nodo Transmisor todas las cápsulas. Esto se hizo para forzar a los nodos de la red a que elevaran peticiones de retransmisión, y comprobar así el funcionamiento del protocolo en el caso de una tasa de pérdidas considerable.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

	# Cápsulas multicast/s	# Cápsulas retransmitidas/s	# Cápsulas NACK/s	Peticiones retransmisión/s	Pérdidas(%)
Nodo 1	9,421708	0	0	0	
Nodo 2	9,0128555	0	0	0	
Nodo 3	9,3424565	0	0,0681945	0,0739849	
Nodo 4	9,1403325	0			0
Nodo 5	9,4968325	0			0
Nodo 6	20,17018	0,0093502			0,3916278
Nodo 7	9,625233	0,0607464			0,44130775

Tabla 5.4: Valores promedio con la configuración A en el prototipo de Multicast RTv2

	# Cápsulas Multicast/s	# Cápsulas retransmitidas/s	# Cápsulas de NACK/s	Peticiones de retransmisión/s	Pérdidas(%)
Nodo 1	9,462241	0	0	0	
Nodo 2	9,0736138	0	0	0	
Nodo 3	9,06960315	0	0,00478935	0,00478935	
Nodo 4	9,0929705	0,00430695	0,0086139	0,0086139	
Nodo 5	9,14359355	0			0
Nodo 6	9,10368715	0			0
Nodo 7	9,124756	0,00370541			0,0085225
Nodo 8	9,1382083	0,0007684			0,0393013

Tabla 5.5: Valores promedio con la configuración B en el prototipo de Multicast RTv2

	# Cápsulas multicast/s	# Cápsulas retransmitidas/s	# Cápsulas de NACK/s	Peticiones de retransmisión/s	Pérdidas(%)
Nodo 1	9,0745915	0	0	0	
Nodo 2	9,0827663	0	0	0	
Nodo 3	9,104488	0	0	0	
Nodo 4	9,0774885	0	0	0	
Nodo 5	9,0802898	0	0	0	
Nodo 6	9,108141	0			0
Nodo 7	9,153527	0			0
Nodo 8	9,09845	0			0
Nodo 9	9,0781405	0	0	0	
Nodo 10	9,155687	0			0
Nodo 11	9,09923695	0			0

Tabla 5.6: Valores promedio con la configuración C en el prototipo de Multicast RTv2

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

	# Cápsulas Multicast/s	# Cápsulas retransmitidas/s	# Cápsulas de NACK/s	Peticiones de retransmisión/s	Pérdidas(%)
Nodo 1	9,068915	0	0	0	
Nodo 2	9,0786931	0	0	0	
Nodo 3	9,0756972	0	0	0	
Nodo 4	9,089114	0	0	0	
Nodo 5	9,103172	0			0
Nodo 6	9,1049443	0			0
Nodo 7	9,089096	0	0	0	
Nodo 8	9,1074626	0			0
Nodo 9	9,1026385	0			0
Nodo 10	9,1124824	0			0
Nodo 11	9,1003455	0			0

Tabla 5.7: Valores promedio con la configuración D en el prototipo de Multicast RTv2

	# Cápsulas Multicast/s	# Cápsulas retransmitidas/s	# Cápsulas de NACK/s	Peticiones de retransmisión/s	Pérdidas(%)
Nodo 1	9,08123	1,7965135	1,7965155	1,7965155	
Nodo 2	7,2913555	1,03274	3,9997158	4,0022115	
Nodo 3	7,27939475	0,792535	4,7817275	4,7817275	
Nodo 4	7,254973	0,00834955	3,6134694	3,6203905	
Nodo 5	7,2995254	1,03604595			8,18399
Nodo 6	7,3080945	0,8014785			10,7242475
Nodo 7	7,27911335	0,001527	7,1556473	7,1556473	
Nodo 8	7,287963	0,0042485			19,5473748
Nodo 9	7,2876583	0,0043505			19,5401033
Nodo 10	7,303635	0			19,4915855
Nodo 11	7,29902965	0,0015421			19,4552845

Tabla 5.8: Valores promedio con la configuración D y un 20% de pérdidas provocadas en el nodo Sender en el prototipo de Multicast RTv2

Hay algunos detalles que deben ser explicados respecto de las tablas de la 5.4 a la 5.8. En primer lugar, los datos que se presentan en cada tabla para cada nodo dependen del tipo de nodo de que se trate. Para los nodos Transmisor (Sender) y Nodos Intermedios (NI) se representan la tasa de cápsulas multicast, la tasa de cápsulas retransmitidas, la tasa de cápsulas de NACK y la tasa total de peticiones de retransmisión. Hay que tener en cuenta que una cápsula de NACK puede contener varias peticiones de retransmisión. Por eso se separan los dos valores.

Para los nodos Receptor (Receiver) se representa la tasa de cápsulas multicast, tasa de cápsulas retransmitidas y pérdidas. A continuación pasamos a detallar la semántica de cada

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

una de estas variables. Nótese que la semántica de algunas de ellas varía dependiendo del tipo de nodo del que se trate.

- *Número de cápsulas multicast/s*: Tasa de cápsulas multicast. Se trata del número de cápsulas del servicio multicast por segundo transmitidas (en los nodos Transmisor) o recibidas (nodos intermedios y nodos Receptor).

- *Número de cápsulas retransmitidas/s*: Tasa de cápsulas retransmitidas. Se calcula como el número de cápsulas retransmitidas por segundo (en los nodos Transmisor) y recibidas (nodos intermedios y nodos Receptor).

- *Número de cápsulas de NACK/s*: Tasa de cápsulas de reconocimiento negativo (NACK) o lo que es lo mismo, de petición de retransmisión. Son el número de cápsulas de NACK por segundo recibidas. Solo tienen sentido en los nodos intermedios y Transmisor.

- *Peticiones de retransmisión*: Tasa de peticiones de retransmisión. Son el número total de peticiones de retransmisión por segundo. Es decir, el número de peticiones incluidas en las cápsulas de NACK. Dependiendo del tipo de pérdidas de cada entorno, puede coincidir con el valor anterior o no, según se pida mayormente la retransmisión de una sola cápsula o más. Sólo tiene sentido en los nodos intermedios y Transmisor.

- *Pérdidas*: Porcentaje de pérdidas de cápsulas de multicast. Tiene en cuenta las cápsulas que finalmente se retransmiten y las descuenta de las pérdidas.

5.6.2 Conclusiones del prototipo de Multicast en Tiempo Real v2

Como primera conclusión, a la vista de los resultados obtenidos podemos afirmar que el prototipo diseñado tiene un funcionamiento correcto.

En primer lugar consideremos en la configuración A (figura 5.19) y la tabla 5.4 que expone los resultados. Como puede verse en la tabla 5.4, sólo se producen pérdidas en los nodos inalámbricos. Por lo tanto, el Nodo Intermedio 3 es el único que recibe las peticiones de retransmisión. Las cápsulas de petición de transmisión y el número total de

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

peticiones de transmisión prácticamente coinciden, esto nos hace pensar que en la mayoría de los casos, el número de peticiones de retransmisión que llevan las cápsulas de NACK que recibe el Nodo Intermedio 3 es de una sola petición.

Como puede verse también en esta tabla 5.4, los nodos Receptores 5 y 6 reciben algunas cápsulas de retransmisión. Esto significa que algunas de las peticiones cursadas al Nodo Intermedio 3 han tenido como fruto la retransmisión solicitada.

Sin embargo las pérdidas en los nodos 5 y 6 no son 0. Esto se debe al efecto combinado de tres factores: *Filtrado activo* de las Cápsulas obsoletas a nivel de red, cápsulas que se solicitan pero no se reciben por ya no estar presentes en la cache del Nodo Intermedio 3, y cápsulas de NACK o retransmitidas que se pierden en el enlace entre el Nodo Intermedio 3 y los nodos Receptores.

Para los resultados de la configuración B (figura 5.20, tabla 5.5) nos parece necesario hacer las siguientes observaciones. Nuevamente se producen pérdidas solamente en el segmento inalámbrico de la topología. Al igual que en el caso anterior, las pérdidas son muy bajas pero existen. Como se puede apreciar las cápsulas de NACK que recibieron los nodos intermedios 3 y 4 contenían exactamente una sola petición de retransmisión, porque los valores de las dos columnas coinciden exactamente.

Si comparamos las peticiones de retransmisión que reciben el nodo 3 y el cuatro podemos ver como el nodo 3 recibe la mitad que el nodo 4. Esto demuestra que está funcionando la técnica de *fusing* de cápsulas con informaciones similares de la que hablamos cuando explicamos este protocolo. Es decir, que el nodo 4 está recibiendo dos peticiones de retransmisión de la misma cápsula multicast, una por cada uno de los nodos Receptor que cuelgan de éste, y el nodo 4 eleva una única petición al nodo 3. Esto se debe a que se produce la pérdida de una cápsula multicast entre el nodo 3 y el 4. Por eso el nodo 3 tampoco tiene la cápsula que solicitan los dos nodos Receptores y la tiene que pedir al nodo anterior.

Esta técnica de *fusing* ha permitido ahorrar el 50% del ancho de banda que se hubiera empleado si las dos peticiones de los dos Receptores hubieran sido enrutadas hacia el nodo intermedio anterior.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Ahora, si nos fijamos en las cápsulas retransmitidas que recibe el nodo 4 nos damos cuenta que obtiene casi todas las cápsulas que solicita, porque al nodo 3 le llegaron sin problema por el segmento cableado. No es exactamente la cantidad que solicita porque a nivel de red se han filtrado algunas cápsulas obsoletas.

Por esta misma razón (el *filtrado activo*), y por pérdidas en las propias cápsulas retransmitidas, el nodo Receptor 7 no recibe todas las cápsulas retransmitidas que le llegaron al nodo 4, y el Receptor 8 aún muchas menos.

Ahora nos fijamos en las configuraciones C y D, y en las tablas 5.6 y 5.7. En estas dos pruebas colocamos el nodo Transmisor (Sender) en el ordenador portátil 1 con la intención de provocar mayores pérdidas. Como puede verse en las tablas 5.6 y 5.7 las pérdidas son del 0% en todos los nodos Receptores. Solamente apreciamos dos diferencias entre estas configuraciones y el resto. La primera es que la media de la tasa de cápsulas por segundo que transmite el nodo Transmisor es de un 3,92% menos que cuando no es un ordenador portátil debido a la diferencia entre la potencia de las máquinas. La segunda es que la tarjeta inalámbrica del nodo que actúa como Transmisor es PCMCIA en lugar de la tarjeta ISA de la Maquina A en las otras configuraciones. Nótese que en la configuración D la máquina A realiza muchos cálculos por habersele mapeado muchos procesos.

Ya para terminar vamos a comentar las conclusiones extraídas de la última prueba, en la que utilizamos la configuración D con la salvedad de que provocamos que el nodo Transmisor no transmitiera sino el 80% de las cápsulas de datos multicast creadas. De esta manera pretendíamos forzar al prototipo a que funcionara en unas condiciones de pérdidas mucho más extremas. Hay que tener en cuenta que las pérdidas que introducimos se darán en toda la topología, por lo que las peticiones de retransmisión se cursarán hasta el mismo nodo Transmisor.

Si nos fijamos en la ubicación relativa de los nodos Receptores respecto de nodo Transmisor podemos ordenarlos de más cerca a más lejos de la siguiente manera:

- El nodo 5: Ubicación local, en la misma máquina que el Transmisor
- Los nodos 6, 8 y 9: Ubicación remota, separados por un enlace inalámbrico del Transmisor.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

- Los nodos 10 y 11: Ubicación remota, separados por dos enlaces inalámbricos del Transmisor.

Lógicamente, la misma ordenación según las pérdidas registradas es muy similar. En la tabla 5.8 podemos ver como el nodo que tiene menos pérdidas es el nodo 5 que sólo registra un 8,18 % de pérdidas. Luego el nodo 6 que registra un 10,72 % de pérdidas. Por último los nodos 8, 9, 10 y 11 que registran la práctica totalidad del 20 % de las pérdidas introducidas.

Las pérdidas del nodo Receptor 5 se atribuyen enteramente al filtrado activo a nivel de red que ha descartado las cápsulas obsoletas, ya que son muy poco probables pérdidas de otro tipo en el segmento cableado. Las pérdidas del resto de los nodos Receptor son provocadas por la combinación de los tres efectos que decíamos en las configuraciones A y B: Filtrado activo a nivel de red, eliminación de las cápsulas obsoletas de las caches correspondientes y pérdidas en los enlaces inalámbricos de cápsulas de NACK o de retransmisión.

Podemos ver también que la mayoría de las cápsulas de NACK portan una sola petición de retransmisión, porque ambas tasas prácticamente coinciden. Además es muy ilustrativo ver en la tabla 5 como las cápsulas retransmitidas recibidas van disminuyendo a medida que nos alejamos en la topología del nodo Transmisor. Cuanto mayor sea el trayecto que deben de hacer las cápsulas retransmitidas, es más probable que sean descartadas o perdidas. El nodo 10 no llega a recibir ninguna cápsula de retransmisión.

Gracias a esta última prueba podemos aseverar que el funcionamiento del prototipo del protocolo de multicast en tiempo real versión 2 es el esperado. Nuevamente tenemos que recordar que este prototipo está diseñado sobre una máquina virtual de Java. Sería deseable que se diseñara una aplicación a más bajo nivel, donde el rendimiento de la misma sería mucho mejor. Sin embargo, nuestro objetivo de probar la validez de este protocolo queda ampliamente satisfecho.

Además hemos profundizado en el conocimiento de los protocolos multicast, su diseño, implementación y prueba.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

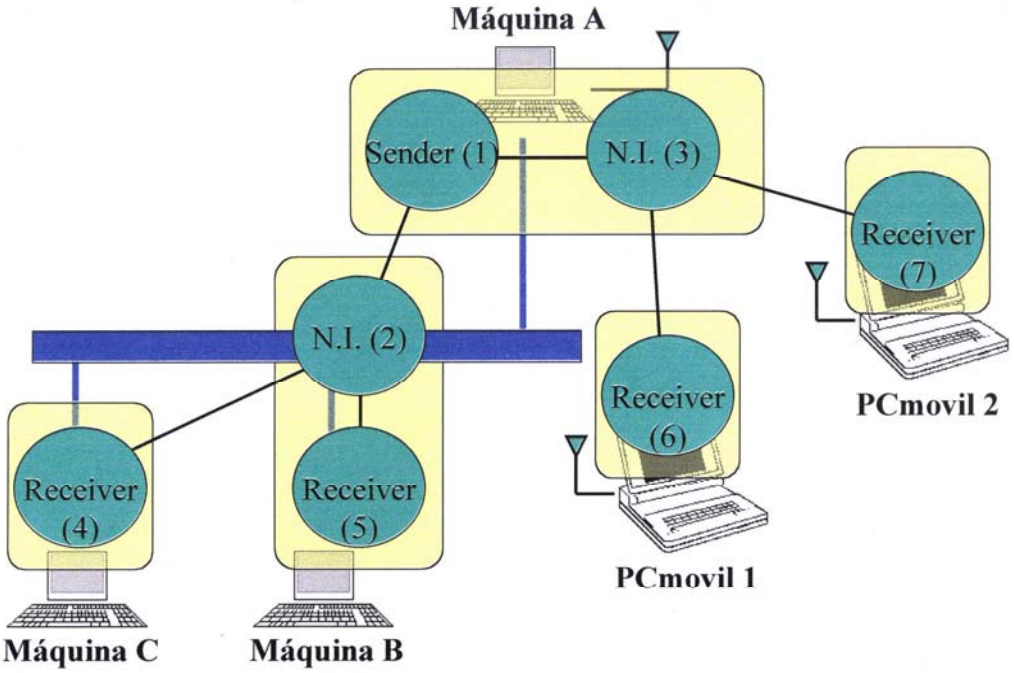


Figura 5.23: Mapeado de procesos en la configuración A para el prototipo de Multicast para Edición Cooperativa

5.7 Protocolo de Multicast para Edición Cooperativa v2

Nuevamente hemos utilizado la infraestructura de red de la figura 5.1. Sobre dicha arquitectura hemos creado tres configuraciones. En las figuras 5.23, 5.24 y 5.25 mostramos el mapeado de procesos a máquinas de dichas topologías.

En las tres configuraciones se utiliza toda la infraestructura de la figura 5.1, es decir, tanto el segmento cableado como el cluster inalámbrico. En el Anexo II y en el CD adjunto se muestran los ficheros de configuración de las 3 topologías.

Antes de pasar a mostrar los resultados de las pruebas realizadas debemos definir la latencia que es el parámetro que hemos medido en todas las pruebas. Es el intervalo de tiempo que transcurre desde que un cliente efectúa un cambio local hasta que recibe confirmación, por medio de una cápsula de cambios globales con su cambio incluido en ella, de que ese cambio ha sido recibido y actualizado en el nodo Transmisor (Sender).

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

5.7.1 Resultados de las pruebas del prototipo de Multicast para Edición cooperativa v2

En las figuras 5.26, 5.27 y 5.28 mostramos las latencias medias que se obtuvieron con las configuraciones A, B y C respectivamente. En la tabla 5.9 se muestran algunos valores medios característicos de cada configuración y totales.

5.7.2 Conclusiones del prototipo de Multicast para Edición Cooperativa v2

En primer lugar debemos expresar nuestra plena satisfacción con el funcionamiento del prototipo que estudiamos en este apartado. Este es el más sofisticado de todos los realizados en este proyecto. A pesar de la complejidad del protocolo, del número elevado de threads que se ejecutan y de la heterogeneidad intencionada de la infraestructura de red ha demostrado un funcionamiento correcto.

Si nos fijamos en la configuración A (figura 5.23) y las latencias medias obtenidas con ésta, es destacable como se produce un incremento en la latencia desde el nodo 4 hacia el nodo 7. Por una parte, dado que la máquina C está menos cargada que la máquina B, tiene mejores resultados a pesar de estar más lejos del nodo Transmisor (Sender). Tenemos que ser conscientes de lo pesadas que son ya las aplicaciones de este prototipo. Cada aplicación de los nodos Receptores (Receiver) lanza 4 threads, la de los Nodos Intermedios 1 thread y la aplicación del nodo Transmisor (Sender) 5 threads. Si a esto le sumamos la ejecución de los 13 tipos distintos de cápsula, las interfaces visuales, etc ... tenemos una carga considerable para las CPU's de las máquinas.

La latencia media mayor en los nodos Receptor 6 y 7 es perfectamente lógica, debido a la menor tasa de b.p.s. de los enlaces inalámbricos. Nótese que la latencia del nodo 7 es 16,45% mayor que la del nodo 6. Esto es debido a las complicaciones que introducen los canales inalámbricos, justificándose de esta manera los estudios que hemos realizado para prototipar los protocolos a implementar en este tipo de redes antes de la implementación definitiva.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Si nos fijamos ahora en los resultados mostrados en la figura 5.27 referentes a la configuración B (figura 5.24) podemos ver lo siguiente: Los nodos 9 y 11 son los que presentan una mayor latencia media. Los nodos 7 y 13 son los que registran menor latencia. La media de las latencias en los nodos del segmento cableado (del 7 al 10) es de 2615 ms. La media de las latencias en los nodos del segmento inalámbrico es de 2665 ms. Esto quiere decir que la diferencia entre la media de las latencias de los nodos del segmento cableado es solamente un 1,9% inferior a la del segmento inalámbrico. La intencionada heterogeneidad en ancho de banda, potencia de cálculo de las máquinas, sistemas operativos, etc ... de la configuración B, y lo pesadas que son las aplicaciones que están ejecutándose en cada prueba, son algunas de las causas de estos resultados. Esto justifica que hayamos prototipado este protocolo con ANTS para estudiar los efectos que producen esta heterogeneidad y las complicaciones de la mezcla de comunicaciones cableadas e inalámbricas.

Centrando nuestra atención en la configuración C (figura 5.25) y sus resultados (figura 5.28) exponemos lo siguiente: Aunque el nodo 9 está más alejado del Transmisor que el nodo 8, tiene una latencia media un 6% inferior. Esto se debe a que la máquina C está menos cargada que la máquina B.

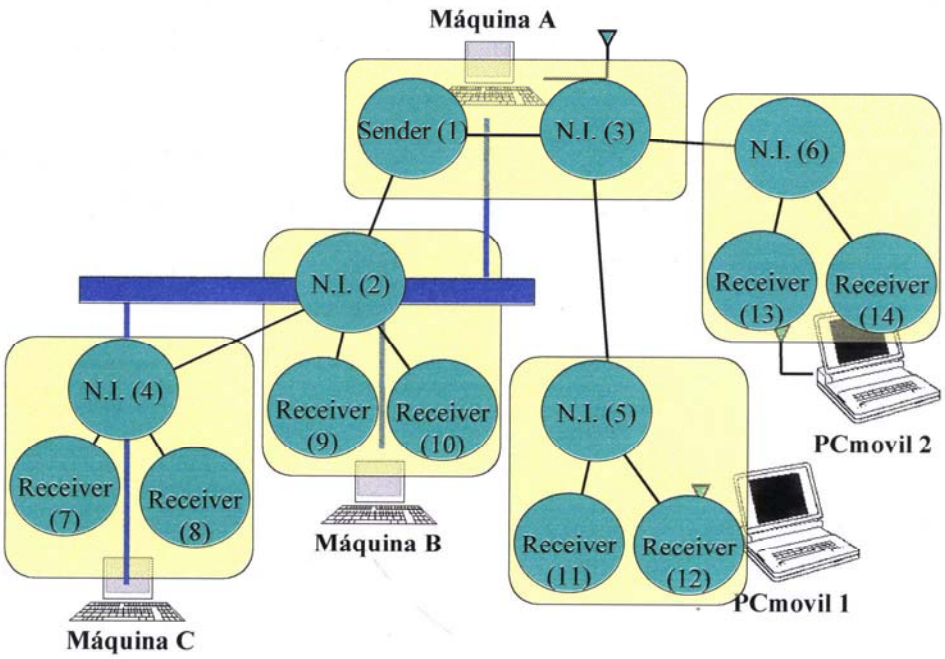


Figura 5.24: Mapeado de procesos en la configuración B
para el prototipo de Multicast para Edición Cooperativa

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

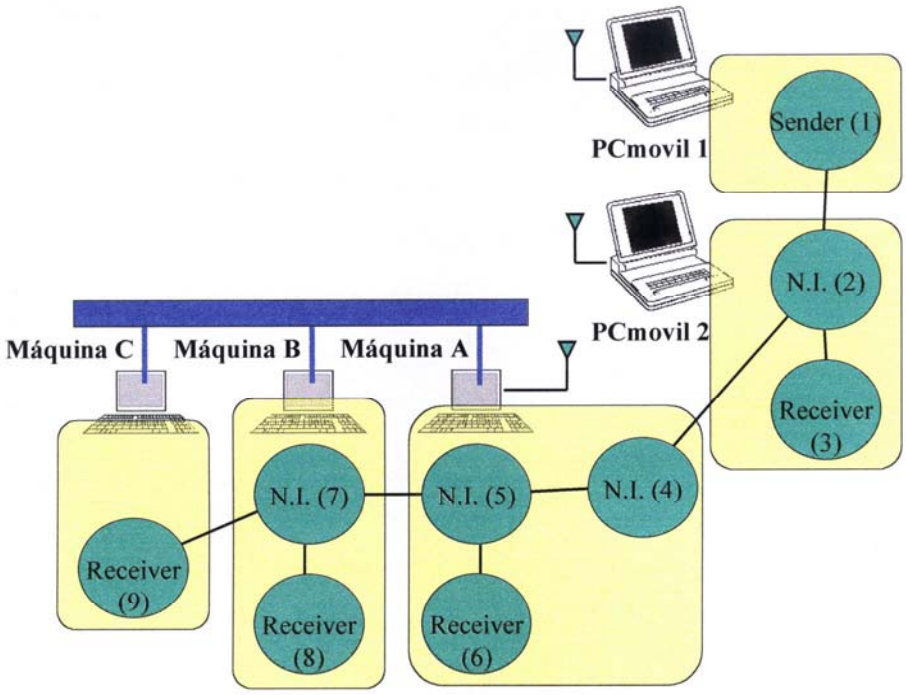


Figura 5.25: Mapeado de procesos en la configuración C para el prototipo de Multicast para Edición Cooperativa

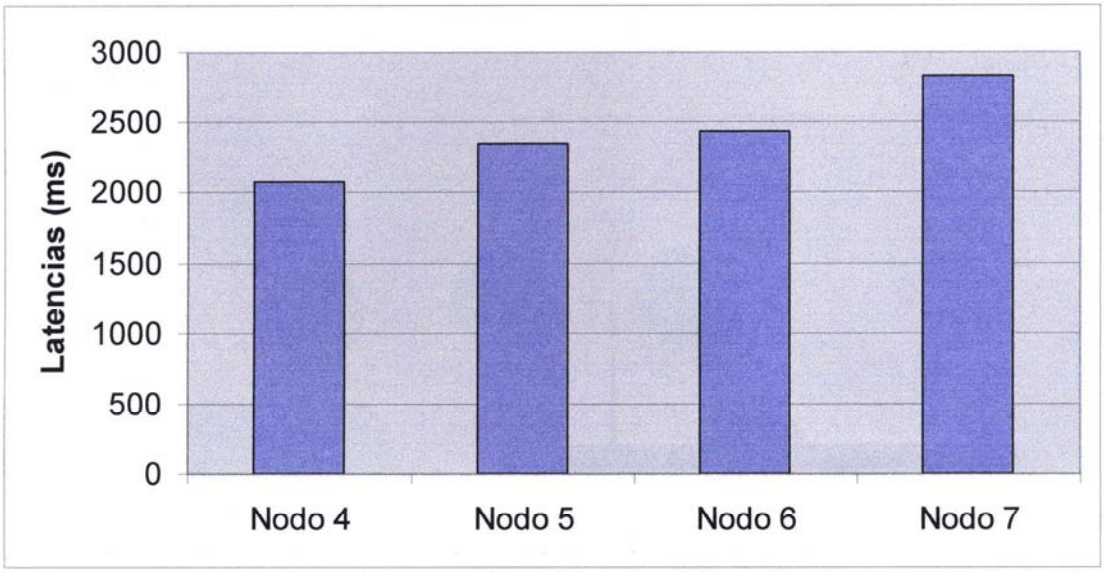


Figura 5.26: Latencias medias en la configuración A

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

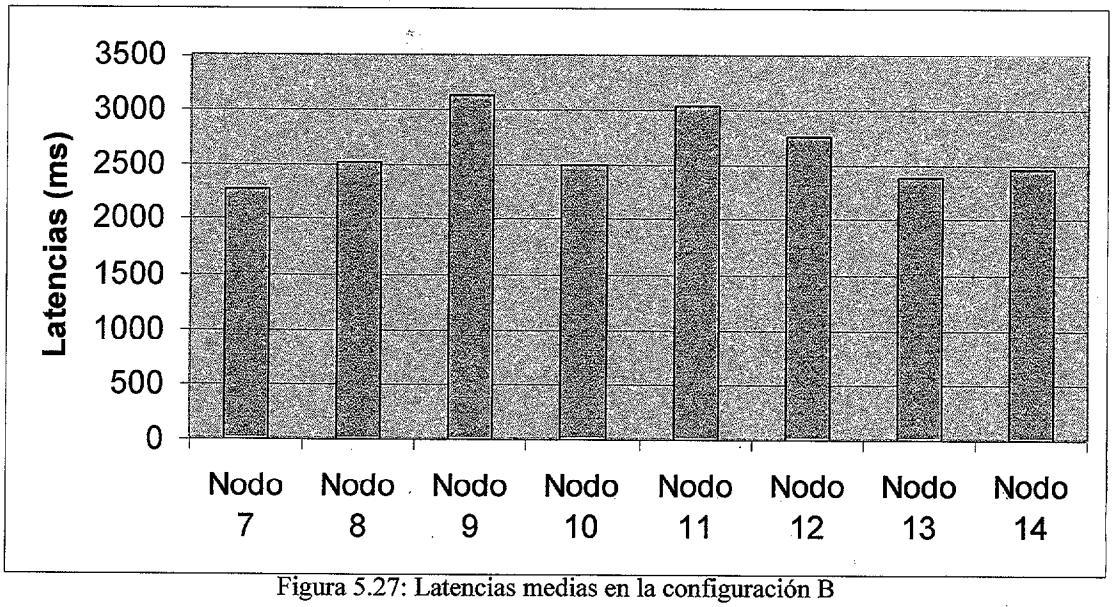


Figura 5.27: Latencias medias en la configuración B

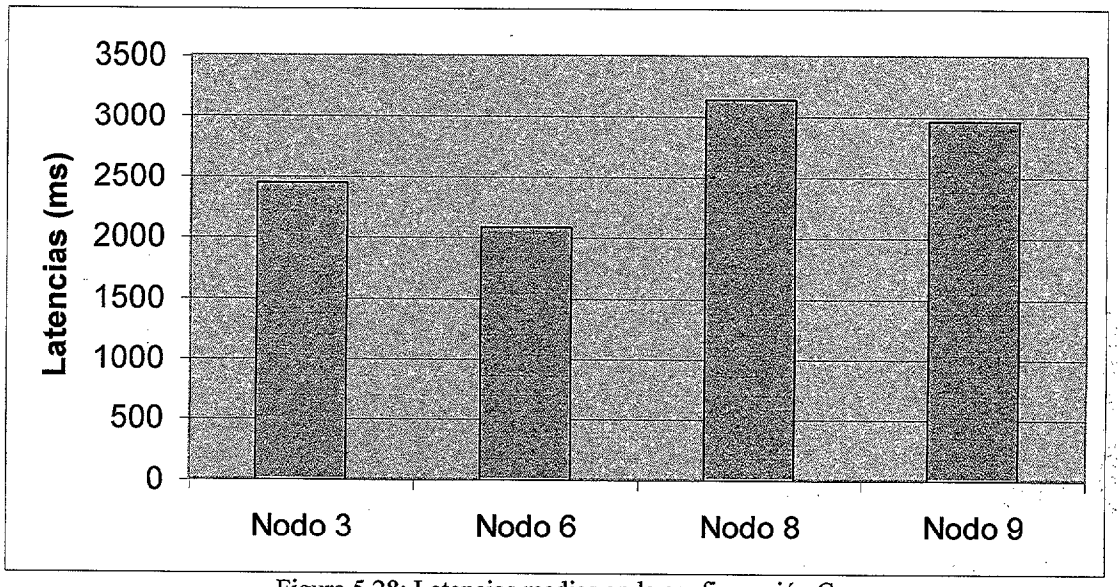


Figura 5.28: Latencias medias en la configuración C

	Promedio(ms)	Máximo(ms)	Mínimo(ms)
Configuración A	2422,17124	2829,46091	2082,65705
Configuración B	2640,02738	3129,6003	2288,28058
Configuración C	2662,85495	3147,35272	2081,89247
Total	2575,01785	3147,35272	2081,89247

Tabla 5.9: Valores característicos

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

La latencia media de los nodos 3 y 6 es inferior a la de los nodos 8 y 9. Esto es debido a que están más cerca del nodo Transmisor. Sin embargo la latencia media del nodo 3 es superior a la del nodo 6 a pesar de estar más cerca del Transmisor. Esto se debe a lo pesadas que son las aplicaciones y la diferencia de recursos entre la máquina A y el ordenador portátil 2.

Ya para terminar y observando la tabla 5.9 podemos decir que el promedio de las latencias se encuentra entre 2 y 3 segundos, lo cual es un valor bastante aceptable. Por otra parte nos parece significativo mencionar que las latencias medias máxima y mínima se corresponden ambas con la configuración C. Esto es debido a que esta topología ha sido diseñada pretendiendo que existiera una diferencia más dramática entre las trayectorias que unen cada uno de los nodos Receptores con el nodo Transmisor.

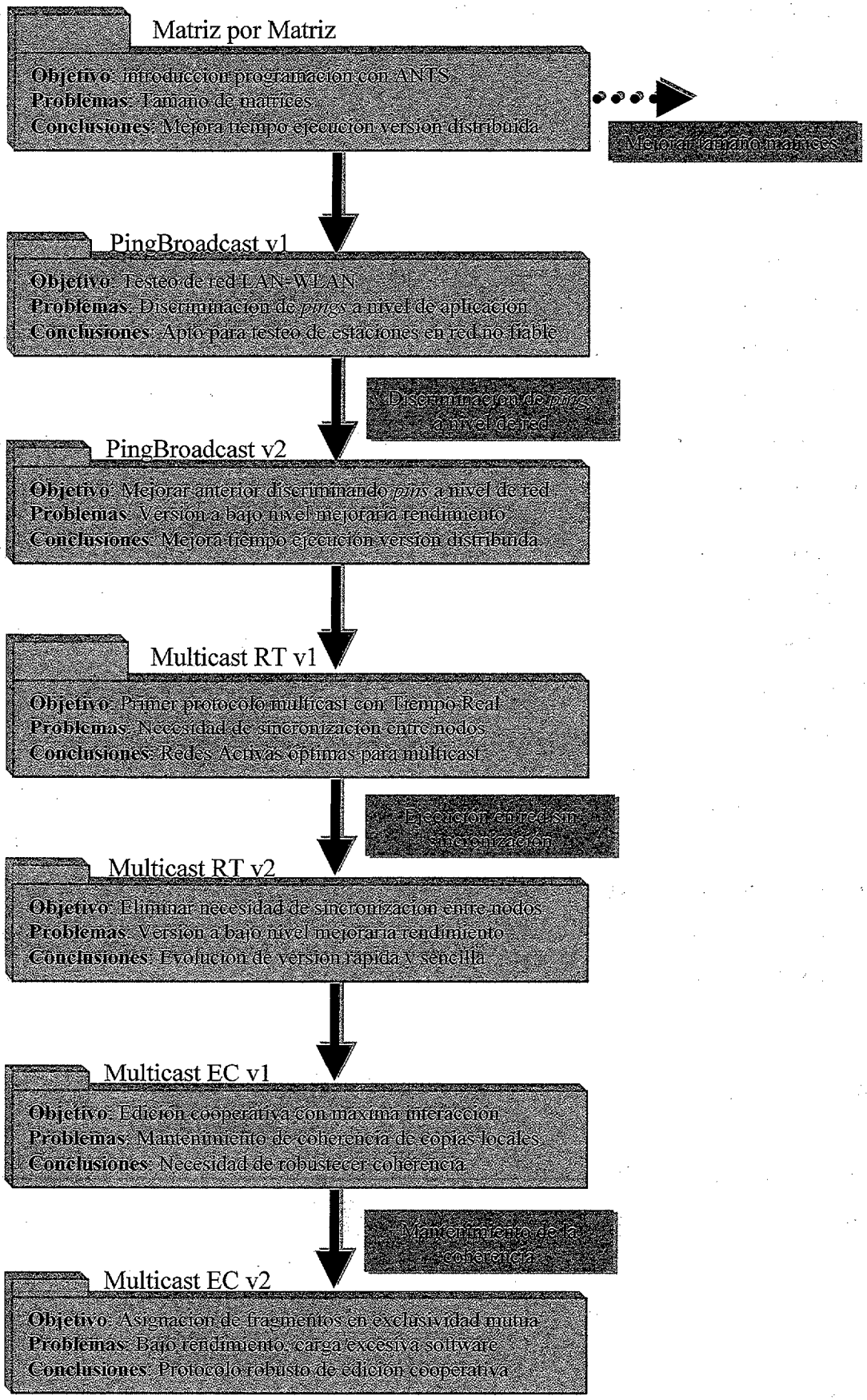
La conclusión final es que, aún habiendo demostrado un funcionamiento correcto, esta implementación de nuestro protocolo ha resultado tan pesada que la mayoría de las veces la latencia de los cambios es achacable más a lo cargado que esta el sistema que a las características de ancho de banda de los enlaces que unen el nodo en cuestión con el nodo Transmisor.

Sin embargo, una implementación de este protocolo en una red con latencias superiores, tales como Internet, podría ser ventajosa, dado que ahí las latencias de la red si que serían considerables en comparación con el tiempo de ejecución de nuestra aplicación. Si a esto le añadimos la posibilidad que siempre mencionamos de realizar una implementación del protocolo probado a un nivel no tan alto en la torre de protocolos, los resultados pueden ser mucho más esperanzadores.

5.8 Resumen de todos los protocolos explicados

El propósito de este último apartado del capítulo 5 es dar una visión clara de la evolución que ha seguido este Proyecto Fin de Carrera a través de los distintos protocolos diseñados hasta culminar con el Protocolo de Multicast para Edición Cooperativa, versión 2. En la figura 5.29 se muestra esquemáticamente cada protocolo, cuál fue el propósito de su diseño, que conclusiones se obtuvieron tras su prototipado y prueba, y que problemas presentaba.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Como puede verse en dicha figura, cada protocolo responde a una necesidad didáctica concreta, para la comprensión y aprendizaje necesarios antes de acometer el diseño del último protocolo. Sin embargo, no deja de ser cierto que para cada uno de estos protocolos se ha justificado su diseño también por la propia utilidad del mismo. De esta manera se ha logrado un doble objetivo: hemos diseñado seis protocolos útiles, que además suponen una evolución conceptual hasta el séptimo y más sofisticado protocolo.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Capítulo 6. Conclusiones y posibles ampliaciones

En este capítulo se presentan las principales conclusiones a las que hemos llegado en este proyecto así como algunas posibles ampliaciones del mismo.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital. 2004

6.1 Conclusiones

La elaboración de este Proyecto Fin de Carrera ha tenido como fruto las conclusiones que exponemos en este capítulo.

En primer lugar, hemos aprendido a utilizar el ANTS como herramienta rápida de prototipado de protocolos. Para ello fue necesario primero aprender a programar en Java. En concreto los aspectos de programación en red y programación de multithreads fueron de especial ayuda.

Una vez adquiridos los conocimientos básicos de este lenguaje de programación fue muy sencillo aprender a utilizar las primitivas del ANTS, a pesar de no disponer de mucha documentación al respecto. Para esta tarea nos sirvió el diseño del primer protocolo de producto de matriz por matriz. Este, a pesar de su simplicidad nos aportó un conocimiento inicial del ANTS muy valioso.

Lo siguiente fue el diseño sobre el papel de cada uno de los protocolos explicados. Esta parte fue de gran interés, dada la vital importancia que el diseño de protocolos tiene en cualquiera de las ramas de la ingeniería de telecomunicación. Aprendimos a diseñar protocolos siguiendo la clásica metodología *Top-Down*. Aprendimos a diseñar buscando ligereza en el protocolo, por ejemplo el protocolo de *PingBroadcast*. También aprendimos las técnicas adecuadas para diseñar buscando la robustez y fiabilidad del protocolo. Ejemplos de este caso son los protocolos de Multicast en Tiempo Real y Multicast para Edición Cooperativa.

Paralelamente a esta fase, fue necesario el aprendizaje de conceptos en dos áreas diferentes: Las Redes de Area Local Inalámbricas y los Protocolos Multicast. Fue preciso recolectar información científica en estas dos disciplinas. El método usual fue la búsqueda de artículos relacionados, la selección de la información necesaria, el estudio y la posterior aplicación de los conocimientos adquiridos. Cada uno de los protocolos diseñados son fruto de este fase previa de preparación.

Otra fase que nos reportó mucha información fue la fase de prueba de cada protocolo. Esta fase está diseminada en el tiempo. Todos los protocolos fueron probados localmente

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

justo después de su diseño, analizados los resultados y elaborada la correspondiente documentación. Luego, en la fase final de este proyecto los protocolos de Producto de Matriz por Matriz, la segunda versión de PingBroadcast, de Multicast en Tiempo Real y de Edición Cooperativa fueron probados en Red. Analizado el ingente volumen de resultados se elaboró la documentación correspondiente y se procedió a la escritura de esta memoria.

En el capítulo 5 se muestra un esquema y una ficha de cada protocolo. En cada ficha se explica brevemente la conclusión del diseño del protocolo que se trate. El propósito de ese anexo es aclarar la evolución conceptual que supuso el diseño de cada protocolo.

Las conclusiones globales respecto del diseño, prototipado y prueba de protocolos para redes WLAN por medio del uso del ANTS son las siguientes. El ANTS es una poderosa herramienta que permite el prototipado rápido de protocolos. El prototipo que más duró para implantarse fue la segunda versión del protocolo de Edición Cooperativa y solamente llevó un mes diseñarlo, prototiparlo y probarlo. Además también es enormemente fácil la evolución de prototipos ya realizados para implementar nuevas versiones de un protocolo, gracias a la intrínseca modularidad de la programación orientada a objeto.

El hecho de que esté construida en Java, un lenguaje extraordinariamente extendido, con poderosas facilidades para la programación en red, programación de threads, etc... es también una importante ventaja de esta herramienta. La documentación para el aprendizaje del Java es tan abundante que el único problema que uno puede tener es seleccionar la más adecuada para las necesidades propias.

Sin embargo, no todo son virtudes en el ANTS. El mismo hecho de que esté escrita en Java supone un importante lastre en lo que a rendimiento de las aplicaciones se refiere. Como expusimos en las conclusiones de las pruebas, el hecho de que nuestros prototipos se ejecuten sobre la máquina virtual de Java implica un importante coste en el tiempo de ejecución.

Otra conclusión a la que hemos llegado es lo adecuado que es el uso de tecnologías de Red Activa para las redes WLAN. La flexibilidad de la programación de las rutinas de encaminamiento, que pueden ser inyectadas por el usuario, supone un gran avance en la mejora de los protocolos que se diseñen para este tipo de redes. Estas redes, como otras,

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

tienen unas características de ancho de banda muy variables. Pero si utilizamos una red con la flexibilidad que tienen las redes activas podemos salvar los problemas de las redes WLAN.

La última conclusión a la que hemos llegado tiene que ver con la aplicabilidad de los protocolos diseñados para redes WLAN a aplicaciones que se ejecuten sobre Internet. Creemos que es necesaria realizar más trabajo en este área, pero con la experiencia que tenemos, fruto de las pruebas realizadas con los siete protocolos probados, creemos que es una opción válida.

6.2 Posibles ampliaciones

Creemos que sería interesante que futuros trabajos versaran sobre algunas de las líneas siguientes:

- Diseño de una plataforma de desarrollo de protocolos equivalente al ANTS a más bajo nivel en la torre de protocolos.
- Diseño de aplicaciones para redes WLAN que implementen los protocolos probados a más bajo nivel para mejorar el rendimiento de los mismos.
- Diseño de aplicaciones para Internet que implementen los protocolos probados para estudiar si es real la equivalencia entre el comportamiento de una red heterogénea LAN-WLAN y la Internet.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Bibliografía

En este apartado se expone la bibliografía utilizada en la realización de este proyecto. Cada referencia lleva un comentario de la misma.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Bibliografía

A continuación mostramos la bibliografía utilizada en la elaboración de este proyecto. Las referencias están ordenadas por orden de aparición en el texto de la memoria.

[1] B. Metcalfe, "The Next-Generation Internet", IEEE Internet Computing, January/February 2000.

Comentario: En este artículo se explica como las tecnologías relacionadas con Internet evolucionan rápidamente hacia lo que se ha dado llamar la Internet de Próxima Generación. Se hacen algunas reflexiones sobre los servicios que portará esta nueva Internet.

[2] D. Tennenhouse et al., "A Survey of Active Network Research", IEEE Communications Magazine, January 1997.

Comentario: En este artículo se da un resumen de las actividades de investigación en el área de las Redes Activas, incluido el trabajo en las tecnologías activas de bajo nivel. Se explica el impacto que las Redes Activas pueden tener en la aparición de nuevos servicios y se da un conjunto de patrones para la clasificación de las mismas.

[3] D. Wetherall, U. Legedza, and J. Guttag, "Introducing New Internet Services: Why and How," IEEE Network, May/June 1998.

Comentario: En este artículo se presentan algunas aplicaciones ejemplo que ilustran el tipo de servicios que pueden ofrecer las Redes Activas.

[4] M. Adeeb, "New Generation Internet and the Evolution Towards Active and Programmable Networks", 16th Radio Science Conference, Ain Shams University, Cairo, Egypt, Feb 1999.

Comentario: En este artículo se explica como la Internet ha sufrido una expansión considerable. Se habla de las tendencias hacia las que se cree que evoluciona y que nuevas tecnologías hay en marcha.

[5] S. Ohmori et al., "The Future Generations of Mobile Communications Based on Broadband Acces Technologies", IEEE Communications Magazine, December 2000.

Comentario: En este artículo se discuten las generaciones futuras en lo que a sistemas de comunicación móvil se refiere. Esto no sólo incluye las generaciones cuarta y quinta de comunicaciones celulares, sino también sistemas de acceso de banda ancha.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

[6] C. Elliott, "Building the Wireless Internet", IEEE Spectrum, January 2001.

Comentario: En este artículo se comentan aspectos relacionados con la Internet Inalámbrica. Se habla de lo que es, de los requerimientos técnicos que precisa y de cómo las redes actuales evolucionan hacia esta.

[7] R. O. LaMaire et al., "Wireless LANs and Mobile Networking: Standards and Future Directions", IEEE Communications Magazine, August 1996.

Comentario: En este artículo se explican dos estándares de comunicaciones inalámbricas: el IEEE 802.11 y el HIPERLAN. También se comentan algunos estándares de gestión del enlace inalámbrico.

[8] Amit B. Kulkarni and Gary J. Minden, "Active Networking Services for Wired/Wireless Networks," IEEE 1999.

Comentario: En este artículo se expone el concepto de clases de protocolo, para el diseño modular de nuevos protocolos. Se exponen algunos problemas de las redes inalámbricas y se proponen soluciones basadas en las clases de protocolo y por medio de las Redes Activas.

[9] D. L. Tennenhouse and D. Wetherall. Towards an active network architecture. In *Multimedia Computing and Networking 96*, San Jose, CA, Jan 1996.

Comentario: En este artículo se definen las Redes Activas. Se explican algunas aplicaciones que motivan la creación de facilidades de computación en la red. Se discuten aspectos de las Redes Activas, tales como la Seguridad, el control de los recursos, etc.

[10] D. Wetherall. Developing Network Protocols with the ANTS Toolkit. En la página web de David Wetheral en la universidad de Washington. http://www.cs.washington.edu.

Comentario: En este artículo se explica el modelo de programación de prototipos con el ANTS. Se comentan las clases más importantes de esta plataforma. Además se dan algunos ejemplos sencillos.

[11] D. Wetherall et al. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. En *1st Conf. On Open Architectures and Network Programming (OPENARCH'98),* páginas 117-129, San Francisco, CA, Abril 1998. IEEE.

Comentario: En este artículo se explica con detalle la arquitectura del ANTS, basada en técnicas de código móvil. Se comentan con detalle todas las abstracciones de esta plataforma.

[12] D. Wetherall, J. Guttag and D. L. Tennenhouse, "ANTS: Network Services Without the Red Tape," IEEE Computer, April 1999.

Comentario: En este artículo se explica someramenta la arquitectura del ANTS. También se dan algunas pinceladas de la programación con este prototipo y por último se muestran algunos resultados de rendimiento obtenidos.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

[13] E. L. Nygren, S. J. Garland, and F. Kaashoek, "PAN: A High-Performance Active Network Node Supporting Multiple Mobile Code Systems," IEEE, 1999.

Comentario: En este artículo se presenta el PAN, una arquitectura de Red Activa. Se comenta el diseño e implementación de esta arquitectura, y los resultados de rendimiento.

[14] D. S. Alexander et al., "The SwitchWare Active Network Architecture," IEEE Network, May/June 1998.

Comentario: En este artículo se expone la arquitectura de Red Activa SwitchWare. También se hace un listado del trabajo relacionado y de líneas futuras de investigación.

[15] D. S. Alexander et al., "A Secure Active Network Environment Architecture: Realization in SwitchWare," IEEE Network, May/June 1998.

Comentario: En este artículo se muestra la Arquitectura de Entorno Seguro de Red Activa, que actúa bajo la arquitectura Switchware.

[16] I. Hadzic and J. Smith, "On-the-fly Programmable Hardware for Networks," Proc. GLOBECOM 1998.

Comentario: En este artículo se explica la arquitectura de un proyecto en el cual se implementan los nodos activos en Hardware mediante el uso de FPGAs.

[17] M. Calderón, et al., "Active Network Support for Multicast Applications," IEEE Network, May/June 1998.

Comentario: En este artículo se explica el diseño, implementación y prueba de un protocolo de multicast con tres niveles de fiabilidad sobre el ANTS.

[18] D. S. Decasper, et al., "A Scalable High-Performance Active Network Node," IEEE Network, January/February 1999.

Comentario: En este artículo se expone una arquitectura Hardware que implementa una Red Activa para elevadas tasas de tráfico.

[19] S. Bhattacharjee et al. "An Architecture for Active Networking". Proc. INFOCOM'97, April 1997.

Comentario: En este artículo se expone una aproximación fuera de banda a las Redes Activas.

[20] S. Bhattacharjee, K. Calvert and E. Zegura, "Self-Organizing Wide-Area Network Caches", Proc. INFOCOM'98, April 1998.

Comentario: En este artículo se explica como las Redes Activas permiten realizar protocolos de cacheo distribuido.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

[21] E. Zegura, "CANEs: Composable Active Network Elements", Georgia Tech; http://www.cc.gatech.edu/projects/canes/

Comentario: En este artículo se expone una aproximación fuera de banda a las Redes Activas.

[22] S. Bhattacharjee, K. Calvert and E. Zegura, "Active Networking and the End-to-End Argument", Georgia Tech;

Comentario: En este artículo se comparan las redes de datos tradicionales en las que los protocolos se programan extremo a extremo con las Redes Activas.

[23] S. Bhattacharjee, K. Calvert and E. Zegura, "Directions in Active Networks", IEEE Communications Magazine, October 1998.

Comentario: En este artículo se apuntan posibles líneas futuras de evolución de las Redes Activas.

[24] B. Schwartz et al., "Smart Packets for Active Networks," Jan. 1998. Disponible en http://www.net-tech.bbn.com/smtpkts/smart.ps.gz

Comentario: En este artículo se explican los Smart Packets, la aproximación integrada a las Redes Activas que crearon en el BBN.

[25] Y. Yemini and S. DaSilva, "Towards programmable networks", IFIP/IEEE Int'l. Wksp. Dist. Syst.: Operation and Mgmt., October 1996.

Comentario: En este artículo se explica como las redes de datos actuales tienen a modelos de programación más flexibles que permitirán una más rápida innovación de los servicios de la red.

[26] C. Tshudin. The messenger environment M0 – a condensed description. En J. Vitek and C. Tschudin, editors, Mobile Object Systems – Towards the Programmable Internet, LNCS 1222, pages 149-156. Springer, Apr. 1997.

Comentario: En este artículo se explica el M0, una plataforma similar al ANTS.

[27] D. Masberger and D. Peterson. "Making paths explicit in the Scout operating system", 2nd Symp. Op. Syst. Design and Implementation, Oct 1996.

Comentario: En este artículo se expone el diseño de protocolos por medios de la creación de "paths" con el Scout, un sistema operativo diseñado para comunicaciones.

[28] A. B. Montz et al., "Scout: A Communications-Oriented Operating System," IEEE, 1995.

Comentario: En este artículo se explica con detalle el Scout, un sistema operativo con primitivas de comunicación a bajo nivel.

[29] J. Hartman et al., "Joust: A Platform for Liquid Software", IEEE Network, July 1998.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Comentario: En este artículo se explica el Joust, una implementación de la JVM de Java para correr sobre el Scout.

[30] C. Diot, W. Dabbous and Jon Crowcroft, "Multipoing Communication: A Survey of Protocols, Functions, and Mechanisms," IEEE Journal on Selected Areas in Communications, Vol. 15, No.3, April 1997.

Comentario: En este artículo se explican con gran nivel de detalle las comunicaciones multicast. Se exponen conceptos, protocolos y mecanismos.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Pliego de Condiciones

En este apartado se expone los pasos que han de seguirse para la instalación y uso del ANTS como herramienta de prototipado rápido de protocolos.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Procedimiento para la instalación del ANTS en una red de área local

Los requisitos para la instalación del ANTS en una red de área local son los siguientes. En primer lugar, los mensajes que intercambian las entidades del ANTS son paquetes UDP por lo que se necesita una infraestructura de red compatible con TCP/IP, por ejemplo, una red Ethernet.

Lo siguiente es instalar el jdk en todas las máquinas que van a ser utilizadas. Para el ANTS v1.0 la versión 1.2.2 del jdk es suficiente.

El ANTS también necesita el TCL/TK instalado para interpretar el script *makeroutes*.

El sistema operativo que debe estar instalado en las máquinas es recomendable que sea Linux, porque los paquetes de las distribuciones suelen instalar por defecto el TCL/TK, y porque tiene facilidades como el *make*. Sin embargo, en principio si se instalara TCL/TK sobre Windows no debería dar problema alguno.

A continuación se copian los ficheros del ANTS en todas las máquinas. Estos están organizados en los directorios ants\, ants\wrapper y utils\. Nuestras clases java se guardan en un directorio aparte, por limpieza, por ejemplo apps\.

Cuando hemos extendido en nuestro directorio las Clases hijas de las clases abstractas Capsule, Application y Protocol que necesitemos para nuestro protocolo, compilamos el ants y nuestro directorio. Una vez corregidos todos los errores de compilación se hace *make* de un fichero Makefile que lanze el script tcl *makeroutes* las veces necesarias para a partir de los ficheros ".config", generar los ficheros ".routes" que tiene las tablas de enrutamiento de los paquetes UDP. Los ficheros ".config" sirven para especificar los nombres de los nodos, que aplicaciones corren sobre cada uno de ellos.

Hecho esto, ya esta todo listo para lanzar nuestras aplicaciones. Para ello lanzamos un fichero script ".start" que arranque en cada máquina las aplicaciones que

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

corresponda. Lo que hará este script es ejecutar una instancia de la Clase del ANTS *ConfiguracionManager* por cada nodo que corra en esa máquina, a la que se le pasará el fichero ".config" con la configuración de la topología de la prueba y el nombre del nodo.

En el anexo II se exponen todos los ficheros ".config" de las pruebas realizadas en este proyecto, a modo de ejemplo. En el anexo III se muestra a modo de ejemplo un fichero Makefile y un fichero ".start".

Problemas que pueden aparecer

En primer lugar, es posible que no esté instalado el TCL/TK. A veces no se instala por defecto en todas las distribuciones de Linux. En este caso hay que instalar este paquete.

Otro problema con el que se puede encontrar uno es con la compilación de los ficheros java. Es recomendable no utilizar implementaciones dudosas del JDK como el kavac que trae algunas distribuciones de Linux. Lo más seguro es desinstalar todo lo que intente emular al Java e instalar un JDK estándar.

A la hora de realizar pruebas suele suceder que hay que modificar los ficheros de configuración. Es un error común modificarlos sólo en una máquina y no en el resto de la red.

Otro error a la hora de la ejecución es no generar un nuevo fichero de rutas cada vez que se modifique el fichero de configuración, ejecutando el *script makeroutes*.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Presupuesto

En esta parte del proyecto se expone el presupuesto del mismo. Se desglosa en fases y se calcula el coste material y la mano de obra.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

1. Introducción

El propósito de este apartado es presentar el presupuesto de este proyecto fin de carrera. Este consta del presupuesto de la mano de obra más el presupuesto de material.

Para calcular el coste de la mano de obra hemos subdividido la elaboración del proyecto en 8 fases. El coste de cada fase se calcula con el número de horas empleado en la misma utilizando los baremos del COIT.

Durante toda la elaboración de este proyecto se ha llevado un diario en el que se anotaban las horas empleadas en cada actividad. De esta manera, hemos podido realizar un conteo tan preciso de las horas trabajadas.

Las fases de que ha constado este proyecto son las siguientes:

1. Aprendizaje del Java
2. Aprendizaje del ANTS
3. Instalación de software necesario
4. Documentación sobre los diferentes protocolos
5. Diseño de los protocolos
6. Programación de los prototipos
7. Diseño y realización de las pruebas
8. Elaboración de la documentación de este PFC

2. Descripción de cada fase

El propósito de este apartado es comentar el contenido de cada una de las fases antes mencionadas.

2.1 Aprendizaje del Java

Se adquirieron los conocimientos necesarios en este lenguaje de programación para poder afrontar el diseño de los prototipos con la herramienta ANTS. Los aspectos sobre los que más se incidió son la programación en red y la programación con múltiples threads.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

2.2 Aprendizaje del ANTS

Comprensión y adquisición de los conocimientos necesarios para la implementación de prototipos. Se utilizó el código fuente del ANTS y algunos pequeños ejemplos facilitados por los autores.

2.3 Instalación del software necesario

Se buscó el software necesario y se procedió a su instalación en todas las máquinas utilizadas. En el pliego de condiciones se explica como es este procedimiento.

2.4 Documentación sobre los diferentes protocolos

Se buscó información científica para la adquisición de los conocimientos necesarios de cada tipo de protocolos. El tipo de documentación utilizada fueron publicaciones del IEEE (*Instituto de Ingenieros Eléctricos y Electrónicos*).

2.5 Diseño de protocolos

Se diseño siguiendo la metodología *Top-Down* cada protocolo sobre el papel, partiendo de unos objetivos y unas restricciones y acotaciones.

2.6 Programación de los prototipos

Fase en la que se programaron los prototipos en Java siguiendo la metodología explicada en este proyecto.

2.7 Diseño y realización de las pruebas

Previo a la realización de cada prueba se estudio la topología necesaria para la obtención de la máxima información sobre cada prototipo.

2.8 Elaboración de la documentación de este PFC

Elaboración de la memoria, pliego de condiciones y presupuesto de este proyecto fin de carrera. También incluye la elaboración del CD adjunto a este proyecto fin de carrera.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

3. Presupuesto

3.1 Mano de obra

Seguidamente exponemos el presupuesto de la mano de obra empleada en cada fase. El número de horas trabajadas en cada fase se indica en la columna de la izquierda. El tiempo equivalente en meses, con una media de 8 horas al día, 5 días a la semana y 4,3 semanas al mes es de 8 meses.

Fase	Horas
Aprendizaje del Java	43
Aprendizaje del ANTS	75,41
Instalación del software necesario	36
Documentación sobre los diferentes protocolos	49,3
Diseño de los protocolos	137,5
Programación de los prototipos	198,96
Diseño y realización de las pruebas	91,7
Elaboración de la documentación del PFC	262,33
Total horas	884,2

Tabla P.1: Fases del proyecto

Los honorarios debidos a esta mano de obra se calculan siguiendo la fórmula del COIT siguiente:

H = 10.500*He + 9.700 * Hn

He = Horas especiales * C

Hn = Horas normales * C

C = Coeficiente reductor

Utilizando un coeficiente reductor del 0,4 los honorarios por mano de obra son los siguientes:

H = 9.700 * 884,2 * 0,4 = 3.430.696 pesetas = 20618,90 euros

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

3.2 Coste material

En este apartado se explica el coste de los materiales utilizados en este proyecto fin de carrera.

Concepto	Importe Total (pesetas/euros)	Tiempo de uso (meses)	Amortización (pesetas/euros)
6 Ordenadores Personales de sobremesa	1.200.000/7212,15	8	1.200.000/7212,15
2 Ordenadores portátiles	600.000/3606,07	8	600.000/3606,07
1 Impresora *chorro de tinta*	40.000/240,40	8	40.000/240,40
SO Windows 95	16.000/96,16	8	16.000/96,16
SO Linux	0/0	8	0/0
Office 97	30.000/180,30	8	30.000/180,30
Encuadernación	10.000/60,10	8	10.000/60,10
Fungibles	25.000/150,25	8	25.000/150,25
Total coste material			1.921.000/11545,44

Tabla P.2: Coste material del proyecto

Todo el coste material tiene como plazo de amortización los 8 meses de elaboración del proyecto.

3.3 Coste total del proyecto

En la tabla P.3 presentamos el coste total de este proyecto

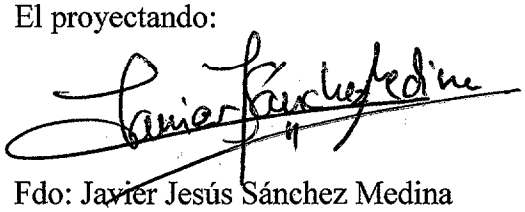
Concepto	Coste (pesetas/euros)
Mano de obra	3.430.696/20618,90
Coste material	1.921.000/11545,44
TOTAL	5.351.696/32164,34

Tabla P.3: Presupuesto total

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

El presupuesto total de este proyecto es de CINCO MILLONES TRESCIENTAS CINCUENTA Y UNA MIL SEISCIENTAS NOVENTA Y SEIS PESETAS.

El proyectando:



Fdo: Javier Jesús Sánchez Medina

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Anexos

En esta parte del proyecto se exponen los tres anexos a la memoria del mismo. Un Anexo es para el código fuente, otro para los ficheros de configuración de las pruebas y el último, para ficheros necesarios para la ejecución de las pruebas.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

ANEXO I. Código Fuente

En este anexo se presenta el código fuente de todas los prototipos diseñados en la realización de este proyecto fin de carrera.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

MatrizXMatrizv1

BCapsule.java

```
/*                                BCapsule.java
 *
 *              Prototipo para el producto de matriz por matriz
 *                    Autor: Javier Jesús Sánchez Medina
 *                            Comenzado: 19/09/00
 */


package MatrizXMatrizv1.apps;

import ants.*;
import java.util.*;

/** Cápsula de broadcast con la matriz B.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class BCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.BCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.BCapsule");
  protected byte[] pid() { return PID; }

  public int length() {
    return super.length();
  }

  public Xdr encode() {
    Xdr xdr = super.encode();
    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();
    return xdr;
  }


  public boolean evaluate(Node n) {
      if (n.getAddress()==getSrc()) //si estamos en el nodo origen (el
master)
            return n.sendToNeighbors(this);
      else return n.deliverToApp(this, dpt); //Si estamos en el destino
se pasa a la aplicación
  }

  public BCapsule() { }

  public BCapsule(short sa, short da, int na, ByteArray d) {
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      super(sa, da, na, d);
  }
}
```

UniCapsule.java

```
/*                              UniCapsule.java
 *              Prototipo para el producto de matriz por matriz
 *                    Autor: Javier Jesús Sánchez Medina
 *                          Comenzado: 19/09/00
 */


package MatrizXMatrizv1.apps;

import ants.*;
import java.util.*;

/** Cápsula de unicast con una fila de la matriz A.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class UniCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.UniCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.UniCapsule");
  protected byte[] pid() { return PID; }

  public int length() {
    return super.length();
  }

  public Xdr encode() {
    Xdr xdr = super.encode();
    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();
    return xdr;
  }

  public boolean evaluate(Node n) {
    if ( n.getAddress() == getDst())
      return n.deliverToApp(this, dpt);
    else
      return n.routeForNode(this, getDst());
  }

  public UniCapsule() { }
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public UniCapsule(short sa, short da, int na, ByteArray d) {
      super(sa, da, na, d);

  }
}
```

MxmProtocol.java

```
/*                              MxmProtocol.java
 *
 *              Prototipo para el producto de matriz por matriz
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 19/09/00
 */

package MatrizXMatrizv1.apps;

import ants.*;
/** Protocolo del producto de matriz por matriz
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class MxmProtocol extends Protocol {

  public MxmProtocol() throws Exception {
    startProtocolDefn();

    startGroupDefn();
    addCapsule("apps.BCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.UniCapsule");
    endGroupDefn();

    endProtocolDefn();
  }
}
```

MApplication.java

```
/*                              MApplication.java
 *
 *          Ejemplo para el aprendizaje de la utilización del ANTS del
MIT
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 19/09/00
 */


package apps;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;


public class MApplication
  extends Application
  implements Runnable
{

  final public static String[] defaults = {};

  int lado;
  int interv=1000,ncapreci=0;
  int[] target;
  Button pinger,FIN;
  Label resultadoLabel, estado, ncapreciDisplay;
  long tiempo_inicio_calculo;

  int cuentRecibi=0;
  int[][] R;
  int[][] A;
  int[][] B;
  int[][] C;
  boolean[] recibido;

  synchronized public void receive(Capsule cap) {
    super.receive(cap);

    if (cap instanceof UniCapsule) {
      UniCapsule resCap = (UniCapsule) cap;
      Xdr buf = new Xdr(resCap.getData(),0);
      int indice = buf.INT();
      if(recibido[indice]==false){
        recibido[indice] = true;
        ncapreci++;
        ncapreciDisplay.setText("  N° de cápsulas Unicast recibidas:
"+ncapreci);
        for (int j=0;j<lado;j++) //colocamos los resultados en la pos
correcta
          R[indice][j] = buf.INT();
        estado.setText("Recibido resultado n° "+(cuentRecibi++)+" de la
fila "+indice);

        if (ncapreci>=(lado)){
            ncapreci=0;
            estado.setText("Recibidos todos los resultados");
            resultadoLabel.setText(new String("R[0][0]="+R[0][0])+"
C[0][0]="+C[0][0]);
            boolean iguales=true;
            for (int i=0;i<lado;i++)
                  for (int j=0;j<(lado);j++)
                        if (R[i][j]!=C[i][j])  iguales = false;
            long latencia = thisNode().time() - tiempo_inicio_calculo;
            if (iguales){
              resultadoLabel.setText("Resultado recibido
correcto!!!:o)");
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
              thisNode().log("Resultado recibido correcto!!!:o)");
              thisNode().log(">>Latencia del calculo = "+ latencia);
            }
            else resultadoLabel.setText("Resultado recibido
INcorrecto!!!:o(");
        }//if
      }//if
    }//if
  }//receive

  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == pinger) {
      estado.setText("       ");
      new Thread(this).start();
      return true;
    } else if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.exit(1);
      return true;
    } else
      return super.handleEvent(evt);
  }

  void inicializa_matrices(){
    R = new int[lado][lado];
    A = new int[lado][lado];
    B = new int[lado][lado];
    C = new int[lado][lado];

    Random r = new Random();
    for (int i=0;i<(lado);i++){
       for (int j=0;j<lado;j++){
          A[i][j]=r.nextInt();
          B[i][j]=r.nextInt();
       }//for j
    }//for i

    for (int i=0;i<(lado);i++){
       for (int j=0;j<lado;j++){
          int aux = 0;
          for (int k=0;k<lado;k++)
               aux += A[i][k]*B[k][j];
          C[i][j] = aux;
       }//for j
    }//for i
  }

  public void run()
  {

   inicializa_matrices();

   //primero broadcast de la matriz B
   tiempo_inicio_calculo = thisNode().time();

   ByteArray buf = new ByteArray(Xdr.INT*lado*lado);
   Xdr xdr = new Xdr(buf,0);
   for (int i=0; i < lado; i++) {
      for (int j=0; j < lado; j++) {
           xdr.PUT(B[i][j]);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      }//for j
    }//for i

    BCapsule bcap = new BCapsule(port,port,0,buf);
    send(bcap);
    estado.setText("Broadcast de B realizado...");
    thisNode().sleep(interv);

    for(int i=0; i<lado; i++) //inicializamos el vector de índices
recibidos
      recibido[i] = false;

    //segundo unicast de la matriz A y el índice
    for (int i=0; i < lado; i++) {
       ByteArray buf2 = new ByteArray(Xdr.INT*(lado+1));
         Xdr xdr2 = new Xdr(buf2,0);
         int indice = i;
         xdr2.PUT(indice); //ponemos el índice
       for (int j=0; j < lado; j++) {
            xdr2.PUT(A[i][j]);
         }//for j

       UniCapsule ucap = new
UniCapsule(port,port,target[i%target.length],buf2);
       send(ucap);
         estado.setText("Unicast de la fila de A no "+i+" realizado...");

         thisNode().sleep(interv);

    }//for
  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    target = new int[0];
    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-lado")) {
      lado = (int)(Integer.parseInt(k.arg(i)));
      k.strike(i);
      }
      //le introducimos las direcciones de los currantes
      //formato: -target numero_currantes currante_1 currante_2 ...
      if (k.key(i).equals("-target")) {
      int[] aux_target = new int[target.length+1];
      System.arraycopy(target,0,aux_target,0,target.length);
      aux_target[target.length] =
(int)(NodeAddress.fromString(k.arg(i)));
      target = aux_target;
      k.strike(i);
      }
    }

    super.setArgs(k);
  }

  public void start()
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    throws Exception
  {

    thisNode().register(new MxmProtocol());

    resize(800, 800);

    recibido = new boolean[lado];

    // 2 rows by 2 columns with 2 pixel point spaces in between them
    setLayout(new BorderLayout());

    pinger = new Button("Inicio Del Cálculo");
    estado= new Label("lado = "+lado,Label.CENTER);
    FIN = new Button("FIN");
    ncapreciDisplay=new Label("  N° de cápsulas recibidas: 0
",Label.CENTER);
    resultadoLabel= new Label("",Label.CENTER);

    Panel stateDisplay = new Panel();
    stateDisplay.setLayout(new GridLayout(2, 1, 1, 1));
    stateDisplay.add(new Label("Estado", Label.CENTER));
    stateDisplay.add(estado, Label.CENTER);

    Panel topPanel = new Panel(), cenPanel = new Panel(), botPanel = new
Panel();
    topPanel.setLayout(new GridLayout(1, 2, 1, 1));
    topPanel.setLayout(new GridLayout(1, 2, 1, 1));
    cenPanel.setLayout(new GridLayout(1, 2, 1, 1));
    cenPanel.setLayout(new GridLayout(1, 2, 1, 1));
    botPanel.setLayout(new GridLayout(3, 1, 1, 1));

    topPanel.add(pinger);
    topPanel.add(stateDisplay);
    topPanel.add(FIN);
    cenPanel.add(new Label("Resultados",Label.CENTER));
    cenPanel.add(new Label("A  x  B  =  C",Label.CENTER));
    botPanel.add(ncapreciDisplay);
    botPanel.add(cenPanel);
    botPanel.add(resultadoLabel,Label.CENTER);

    add("North", topPanel);
    add("Center", botPanel);
    pack();
    show();

  }

  public MApplication()
    throws Exception
  {
    super();
  }
}
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

CApplication.java

```
/*                              CApplication.java
 *
 *          Ejemplo para el aprendizaje de la utilización del ANTS del
MIT
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 19/09/00
 */


package apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;


public class CApplication
  extends Application
  implements Runnable
{

  final public static String[] defaults = {};

  int lado;
  int interv=500;
  Label estado;
  Button inicio,FIN;

  int cuentRecibi=0,indice;
  int[] A;//fila de unicast
  int[][] B;//matriz de broadcast
  boolean tengoA=false;
  boolean tengoB=false;
  int ultimo_indice;

  public void calculaYEnvia(int[] A,int[][] B,int destino){

      ByteArray buf = new ByteArray(Xdr.INT*(lado+1));
        Xdr xdr = new Xdr(buf,0);
        xdr.PUT(indice);
      for (int j=0;j<lado;j++){
          int aux = 0;
          for (int k=0;k<lado;k++){
              aux += A[k]*B[k][j];
          }//for k
          xdr.PUT(aux);
      }//for j

      estado.setText("Cálculo realizado...enviando resultados");
      UniCapsule resulCap = new UniCapsule(port,port,destino,buf);//la
enviamos al master
      send(resulCap);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      estado.setText("Resultados enviados...");

  }//calculaYEnvia

  synchronized public void receive(Capsule cap) {
    super.receive(cap);
    if (cap instanceof BCapsule){
      B = new int[lado][lado];      //matriz de ladoXlado
      estado.setText("Cápsula de Broadcast recibida...");
      BCapsule bCap = (BCapsule) cap;
      Xdr buf = new Xdr(bCap.getData(),0);
      for (int i=0;i<lado;i++)
           for(int j=0;j<lado;j++){
            B[i][j]=buf.INT();
           }//for j
      tengoB=true;
      ultimo_indice = -1;
//          if (tengoA) {
//                calculaYEnvia(A,B,bCap.getSrc());
//                tengoA=false;
//                tengoB=false;
//          }//if tengoA
    }else if (cap instanceof UniCapsule){
      A = new int[lado];      //vector de 1xlado
        UniCapsule uCap = (UniCapsule) cap;
      Xdr buf = new Xdr(uCap.getData(),0);
      indice = buf.INT();
      if(indice != ultimo_indice){
        ultimo_indice = indice;
        estado.setText("Fila "+indice+" recibida...");
        for (int i=0;i<lado;i++)
           A[i]= buf.INT();
        tengoA=true;
        if (tengoB) {
            calculaYEnvia(A,B,uCap.getSrc());
            tengoA=false;
//                tengoB=false;
        }//if tengoB
       }//if//


    }//if
  }//receive

  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == inicio) {
      new Thread(this).start();
      return true;
    } else if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.exit(1);
      return true;
    } else
      return super.handleEvent(evt);
  }

  public void run()
  {
      estado.setText("Vamos a ver si nos llega algo...");
      tengoB=false;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {

    k.merge(defaults);

    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-lado")) {
      lado = (int)(Integer.parseInt(k.arg(i)));
      k.strike(i);
      }
    }

    super.setArgs(k);

  }

  public void start()
    throws Exception
  {

    thisNode().register(new MxmProtocol());

    resize(1000, 800);


    // 2 rows by 2 columns with 2 pixel point spaces in between them
    setLayout(new BorderLayout());

    inicio = new Button("INICIO");
    estado = new Label("                                          ",
Label.CENTER);
    FIN = new Button("FIN");

    Panel principal = new Panel();
    principal.setLayout(new GridLayout(4, 1, 1, 1));
    principal.add(inicio);
    principal.add(new Label("Estado del currante", Label.CENTER));
    principal.add(estado);
    principal.add(FIN);

    add("Center", principal);
    pack();
    show();
  }

  public CApplication()
    throws Exception
  {
    super();
  }
}
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

PingBroadcastv1

FloodCapsule.java

```
/*                              FloodCapsule.java
 *
 *                      Protocolo de pingbroadcast versión 1
 *                       Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 28/08/00
 */


package PingBroadcastv1.apps;

import ants.*;
/** Cápsula de ping de inundación
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class FloodCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.FloodCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.FloodCapsule");
  protected byte[] pid() { return PID; }

  int origen; //dirección del origen del Ping

  public int length() {
    return super.length()+Xdr.INT;
  }

  public Xdr encode() {
    Xdr xdr = super.encode();
    xdr.PUT(origen);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();
    origen = xdr.INT();

    return xdr;
  }


  public boolean evaluate(Node n) {
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      if (n.getAddress()==getSrc()) //si estamos en el nodo origen
            return n.sendToNeighbors(this);
      else return n.deliverToApp(this, dpt); //Si estamos en el destino
se pasa a la aplicación
  }

  public FloodCapsule() { }

  public FloodCapsule(short sa, short da, int na, int o, ByteArray d) {
    super(sa, da, na, d);
    origen = o;
  }

}
```

ACKCapsule.java

```
/*                              ACKCapsule.java
 *
 *                     Protocolo de pingbroadcast versión 1
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 28/08/00
 */


package PingBroadcastv1.apps;

import ants.*;
import java.util.*;
/** Cápsula de reconocimiento de ping hacia el origen del mismo
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */
public class ACKCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.ACKCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.ACKCapsule");
  protected byte[] pid() { return PID; }

  public int length() {
    return super.length();
  }

  public Xdr encode() {
    Xdr xdr = super.encode();
    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();
    return xdr;
  }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public boolean evaluate(Node n) {
    if ( n.getAddress() == getDst())
      return n.deliverToApp(this, dpt);
    else
      return n.routeForNode(this, getDst());
  }

  public ACKCapsule() { }

  public ACKCapsule(short sa, short da, int na, ByteArray d) {
      super(sa, da, na, d);

  }
}
```

PingBApplication.java

```
/*                              PingBApplication.java
 *                       Protocolo de pingbroadcast versión 1
 *                        Autor: Javier Jesús Sánchez Medina
 *                                Comenzado: 28/08/00
 */


package PingBroadcastv1.apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;
/** Tiempo
*/
class Tiempo{
   private long tiempo;
   long get (){
     return tiempo;
   }
   void put (long t){
     tiempo = t;
   }
   Tiempo(long t){
     tiempo = t;
   }
}


/** Aplicación que corre en cada nodo del prototipo
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class PingBApplication
  extends Application
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  implements Runnable
{
  int NNodos=6;

  Hashtable TablaConocida;
  Integer IndiceTabla;
  long dif = 0, periodoRefresco=1000;

  final public static String[] defaults = {};

  int capRecibidas = 0, capFloodRecibidas=0, capACKRecibidas=0, delay;
  long beginTime;
  Button pinger,inicio,fin;
  Label cReci, cFReci, cAReci, control;
  Label[] resultadosPing = new Label[NNodos-1];

  synchronized public void receive(Capsule cap) {
    super.receive(cap);
    capRecibidas++;
    cReci.setText(""+capRecibidas);

    if (cap instanceof FloodCapsule){
      FloodCapsule fcap = (FloodCapsule) cap;
      capFloodRecibidas++;//contamos incluso los ecos del propio ping
      cFReci.setText(""+capFloodRecibidas);
        //Ahora con el origen observamos en la tabla de
        //Cápsulas conocidas si ya hemos recibido esta
      int origen = fcap.origen;
      if (thisNode().getAddress() != origen){ //si no estoy en el origen
del ping

        IndiceTabla = new Integer(origen);
        dif = 0;
        if (TablaConocida.containsKey(IndiceTabla)) {  //Si está presente
el origen
            dif = System.currentTimeMillis()-
((Tiempo)TablaConocida.get(IndiceTabla)).get();
            control.setText("dif="+dif);
          }
          if
(!(TablaConocida.containsKey(IndiceTabla))||(dif>periodoRefresco)){
          TablaConocida.put(IndiceTabla,new Tiempo(thisNode().time()));
//la metemos en la tabla
          ByteArray buf = new ByteArray(Xdr.LONG);
          Xdr xdr = new Xdr(buf,0);
          xdr.PUT(thisNode().time());
          ACKCapsule aC = new ACKCapsule(port,port,origen,buf);
          send(aC);
          send(fcap);   //la enviamos a todos los vecinos
        }//if
      }//if
    }else if (cap instanceof ACKCapsule){
        control.setText("Recibi una ACKCapsule");
      ACKCapsule acap = (ACKCapsule) cap;
      capACKRecibidas++;
      cAReci.setText(""+capACKRecibidas);
      int Origen = acap.getSrc();
      Xdr buf = new Xdr(acap.getData(), 0);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      long lat = thisNode().time() - buf.LONG();  //Latencia del ACK
desde el nodo destino
      resultadosPing[capACKRecibidas%(NNodos-1)].setText
           ("Nodo n° " + NodeAddress.toString(Origen) + ". Latencia: " +
Long.toString(lat) + " ms");
    }
  }

  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == pinger) {
        new Thread(this).start();
        return true;
    } else if (evt.id == Event.ACTION_EVENT && evt.target == inicio){
      TablaConocida = new Hashtable();
      return true;
    } else if (evt.id == Event.ACTION_EVENT && evt.target == fin){
      System.exit(0);
      return true;
    } else
      return super.handleEvent(evt);
  }

  public void run()
  {
      FloodCapsule c = new
FloodCapsule(port,port,0,thisNode().getAddress(), new ByteArray(0));
      send(c);
  }

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {

    thisNode().register(new PingBProtocol());

    resize(600, 200);


    // 2 rows by 2 columns with 2 pixel point spaces in between them
    setLayout(new BorderLayout());

    pinger = new Button("Ping Broadcast " +
NodeAddress.toString(thisNode().getAddress()));
    inicio = new Button("Inicio");
    fin = new Button("FIN");

    Panel outputDisplay = new Panel();
    outputDisplay.setLayout(new GridLayout(8, 1, 1, 1));
    outputDisplay.add(inicio);
    outputDisplay.add(fin);
    outputDisplay.add(new Label("Cápsulas recibidas", Label.CENTER));
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    cReci = new Label("0", Label.CENTER);
    outputDisplay.add(cReci);
    outputDisplay.add(new Label("Cápsulas de inundación", Label.CENTER));
    cFReci = new Label("0", Label.CENTER);
    outputDisplay.add(cFReci);
    outputDisplay.add(new Label("Cápsulas de ACK", Label.CENTER));
    cAReci = new Label("0", Label.CENTER);
    outputDisplay.add(cAReci);

    Panel leftPanel = new Panel();
    leftPanel.setLayout(new GridLayout(3,1,1,1));

    Panel resultados = new Panel();
    resultados.setLayout(new GridLayout((NNodos-1),1,1,1));
    for (int i=0;i<NNodos-1;i++){
      resultadosPing[i] = new Label("
",Label.CENTER);
      resultados.add(resultadosPing[i]);
    }

    control = new Label ("controlando...",Label.CENTER);

    leftPanel.add(pinger);
    leftPanel.add(resultados);
    leftPanel.add(control);

    Panel topPanel = new Panel();
    topPanel.setLayout(new GridLayout(1, 2, 1, 1));

    topPanel.add(leftPanel);
    topPanel.add(outputDisplay);

    add("Center", topPanel);
    pack();
    show();
  }

  public PingBApplication()
    throws Exception
  {
    super();
  }
}
```

PingBProtocol.java

```
/*                          PingBProtocol.java
 *
 *                  Protocolo de pingbroadcast versión 1
 *                   Autor: Javier Jesús Sánchez Medina
 *                          Comenzado: 28/08/00
 */
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
package PingBroadcastv1.apps;

import ants.*;
/** Protocolo de pingbroadcast versión 1
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class PingBProtocol extends Protocol {

  public PingBProtocol() throws Exception {
    startProtocolDefn();

    startGroupDefn();
    addCapsule("apps.FloodCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.ACKCapsule");
    endGroupDefn();

    endProtocolDefn();
  }
}
```

PingBroadcastv2

FloodCapsule.java

```
/*                              FloodCapsule.java
 *
 *                    Protocolo de pingbroadcast versión 2
 *                     Autor: Javier Jesús Sánchez Medina
 *                             Comenzado: 4/4/01
 */


package PingBroadcastv2.apps;

import ants.*;
import ants.wrapper.*;

/** Cápsula de inundación para la optención de información de ping de
toda la red
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class FloodCapsule
  extends DataCapsule
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
{
  final private static byte[] MID = findMID("apps.FloodCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.FloodCapsule");
  protected byte[] pid() { return PID; }

  int MAX_SELLOS = 10; //número máximo de sellos que almancenas los nodos
en el nivel de red.
  int TIEMPO_MAX_NODE_CACHE = 100000;  //tiempo máximo que la cache del
nodo almacena la info de los sellos (segs)

  int group,sello;

  public int length() {
    return super.length() + Xdr.INT + Xdr.INT;
  }

  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(group);
    xdr.PUT(sello);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    group = xdr.INT();
    sello = xdr.INT();

    return xdr;
  }


  public boolean evaluate(Node n) {
      W_JAI m = (W_JAI)n.getCache().get(group);
      boolean encontrado = false;
      if (m!=null){
      for (int i = 0; m.ai1 != null && i < m.ai1.length; i++)
        if (m.ai1[i] == sello) { encontrado = true; break; }
      }else{
      m = new W_JAI();
      n.getCache().put(group, m, TIEMPO_MAX_NODE_CACHE);
      int[] aux = new int[MAX_SELLOS];
      for (int j=0; j<MAX_SELLOS; j++)
        aux[j] = 0;
      m.ai1 = aux;
      }
      if (!encontrado){ //si no se tenía ese sello desplazamos los
existentes
      int[] aux = new int[MAX_SELLOS];
      System.arraycopy(m.ai1,0,aux,1,MAX_SELLOS-1);
      aux[0] = sello;
      m.ai1 = aux;
      n.deliverToApp(this,dpt); //y lo propagamos
      n.sendToNeighbors(this);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      }
    return true;
  }//evaluate

  public FloodCapsule() { }

  public FloodCapsule(short sa, short da, int group, int sello) {
    super(sa, da, 0, new ByteArray(0));
    this.group = group;
    this.sello = sello;
  }

}
```

ACKCapsule.java

```
/*                              ACKCapsule.java
 *
 *                     Protocolo de pingbroadcast versión 2
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 4/4/01
 */


package PingBroadcastv2.apps;

import ants.*;
import java.util.*;
/** Cápsula de reconocimiento de ping hacia el origen del mismo
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */


public class ACKCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.ACKCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.ACKCapsule");
  protected byte[] pid() { return PID; }

  int sello;

  public int length() {
    return super.length() + Xdr.INT;
  }

  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(sello);

    return xdr;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    sello = xdr.INT();

    return xdr;
  }

  public boolean evaluate(Node n) {
    if ( n.getAddress() == getDst())
      return n.deliverToApp(this, dpt);
    else
      return n.routeForNode(this, getDst());
  }

  public ACKCapsule() { }

  public ACKCapsule(short sa, short da, int na, int sello) {
    super(sa, da, na, new ByteArray(0));

    this.sello = sello;
  }
}
```

PingBApplication.java

```
/*                              PingBApplication.java
 *                      Protocolo de pingbroadcast versión 2
 *                       Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 4/4/01
 */


package PingBroadcastv2.apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;
/** Tiempo
 */
class Tiempo{
   private long tiempo;
   long get (){
     return tiempo;
   }
   void put (long t){
     tiempo = t;
   }
   Tiempo(long t){
     tiempo = t;
   }
}
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
/**   Thread que se encarga de limpiar la tabla de sellos de la
aplicación
 */
class LimpiaTablaSellos
  implements Runnable
{
  long periodoRefresco = 3000; //cada 3 segundos...
  PingBApplication pingBApp;

  public void run(){
    try {
      Thread.sleep(periodoRefresco);
    } catch (InterruptedException e) { }
    Enumeration e = pingBApp.TablaSellos.keys();
    while(e.hasMoreElements()){
      Integer auxKey = (Integer)e.nextElement();
      long auxlong=
((Long)(pingBApp.TablaSellos.get(auxKey))).longValue();
      if (pingBApp.thisNode().time() - auxlong > periodoRefresco)
      pingBApp.TablaSellos.remove(auxKey);
    }
  }//run

  LimpiaTablaSellos(PingBApplication pingBApp){
    this.pingBApp = pingBApp;
  }
}//LimpiaTablaSellos

/** Aplicación que corre en cada nodo del prototipo
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class PingBApplication
  extends Application
  implements Runnable
{
  int NNodos=6;
  int NPings = 20;
  int periodicidadPings = 100;

  Hashtable TablaSellos;

  final public static String[] defaults = {};

  int capRecibidas = 0, capFloodRecibidas=0, capACKRecibidas=0, delay;
  long[][] latencias;
  Button pinger,fin;
  Label cReci, cFReci, cAReci, control;
  Label[] resultadosPing = new Label[NNodos-1];

  synchronized public void receive(Capsule cap) {
    super.receive(cap);
    capRecibidas++;
    cReci.setText(""+capRecibidas);

    if (cap instanceof FloodCapsule){
      FloodCapsule fcap = (FloodCapsule) cap;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      capFloodRecibidas++;//contamos incluso los ecos del propio ping
      cFReci.setText(""+capFloodRecibidas);
        //Ahora con el origen observamos en la tabla de
        //Cápsulas conocidas si ya hemos recibido esta
      int origen = fcap.getSrc();
      int sello = fcap.sello;
      if (thisNode().getAddress() != origen){ //si no estoy en el origen
del ping
        ACKCapsule aC = new ACKCapsule(port,port,origen,sello);
        send(aC);
      }
    }else if (cap instanceof ACKCapsule){
      ACKCapsule acap = (ACKCapsule) cap;
      capACKRecibidas++;
      cAReci.setText(""+capACKRecibidas);
      int origen = acap.getSrc();
      int sello = acap.sello;
      if (TablaSellos.containsKey(new Integer(sello))){
        long lat = thisNode().time() - ((Long)(TablaSellos.get(new
Integer(sello)))).longValue();
        resultadosPing[capACKRecibidas%(NNodos-1)].setText
                ("Nodo n° " + NodeAddress.toString(origen) + ".
Latencia: " + Long.toString(lat) + " ms");

        int len = latencias[origen-1].length;
        long[] auxlat = new long[len+1];
        System.arraycopy(latencias[origen-1],0,auxlat,0,len); //OJO:el
nodo 1 en la posición 0
        auxlat[len] = lat;   //añadimos la nueva latencia a los
resultados
        latencias[origen-1]= auxlat;
      }
    }
  }

  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == pinger) {
        new Thread(this).start();
        return true;
    } else if (evt.id == Event.ACTION_EVENT && evt.target == fin){
      for(int i=0; i<latencias.length; i++){
        System.out.println("latencias del nodo " +
NodeAddress.toString(i+1) + ":");
        for (int j = 0; j<latencias[i].length; j++){
          System.out.println(">"+latencias[i][j]);
        }
      }
      System.exit(0);
      return true;
    }else  return super.handleEvent(evt);
  }

  public void run()
  {
    for (int i=0; i<NPings; i++){
      int sello = (new Random()).nextInt();
      FloodCapsule c = new
FloodCapsule(port,port,thisNode().getAddress(), sello);
      TablaSellos.put(new Integer(sello),new Long(thisNode().time()));
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      send(c);
      try {
        Thread.sleep(periodicidadPings);
      } catch (InterruptedException e) { }
      control.setText("ping n° "+i);
    }
  }

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {

    thisNode().register(new PingBProtocol());

    resize(600, 200);


    // 2 rows by 2 columns with 2 pixel point spaces in between them
    setLayout(new BorderLayout());

    pinger = new Button("Ping Broadcast " +
NodeAddress.toString(thisNode().getAddress()));
    fin = new Button("FIN");

    Panel outputDisplay = new Panel();
    outputDisplay.setLayout(new GridLayout(7, 1, 1, 1));
    outputDisplay.add(fin);
    outputDisplay.add(new Label("Cápsulas recibidas", Label.CENTER));
    cReci = new Label("0", Label.CENTER);
    outputDisplay.add(cReci);
    outputDisplay.add(new Label("Cápsulas de inundación", Label.CENTER));
    cFReci = new Label("0", Label.CENTER);
    outputDisplay.add(cFReci);
    outputDisplay.add(new Label("Cápsulas de ACK", Label.CENTER));
    cAReci = new Label("0", Label.CENTER);
    outputDisplay.add(cAReci);

    Panel leftPanel = new Panel();
    leftPanel.setLayout(new GridLayout(3,1,1,1));

    Panel resultados = new Panel();
    resultados.setLayout(new GridLayout((NNodos-1),1,1,1));
    for (int i=0;i<NNodos-1;i++){
      resultadosPing[i] = new Label("
",Label.CENTER);
      resultados.add(resultadosPing[i]);
    }

    control = new Label ("controlando...",Label.CENTER);

    leftPanel.add(pinger);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    leftPanel.add(resultados);
    leftPanel.add(control);

    Panel topPanel = new Panel();
    topPanel.setLayout(new GridLayout(1, 2, 1, 1));

    topPanel.add(leftPanel);
    topPanel.add(outputDisplay);

    add("Center", topPanel);
    pack();
    show();

    TablaSellos = new Hashtable();
    new Thread(new LimpiaTablaSellos(this)).start();
    latencias = new long[NNodos][];
    for (int i=0; i<NNodos; i++){   //inicializamos los resultados
      latencias[i] = new long[0];
    }
  }

  public PingBApplication()
    throws Exception
  {
    super();
  }
}
```

PingBProtocol.java

```
/*                              PingBProtocol.java
 *
 *                     Protocolo de pingbroadcast versión 2
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 4/4/01
 */

package PingBroadcastv2.apps;

import ants.*;
/**
 *                     Protocolo de pingbroadcast versión 2
 *                     @author Javier Jesús Sánchez Medina
 */

public class PingBProtocol extends Protocol {

  public PingBProtocol() throws Exception {
    startProtocolDefn();

    startGroupDefn();
    addCapsule("apps.FloodCapsule");
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.ACKCapsule");
    endGroupDefn();

    endProtocolDefn();
  }
}
```

MCastRTv1

LimpiaTablaEnviadas.java

```
/*                              LimpiaTablaEnviadas.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 19/10/00
 */
package MCastRTv1.apps;

import java.util.*;

import ants.*;
import MCastRTv1.apps.variablesglobales;

/** Thread que limpia la tabla de capsulas multicast enviadas.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

class LimpiaTablaEnviadas
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  MCastApplication app;

  public void run() {

    while (true) {
      try {
      Thread.sleep(vg.tiempoLimpieza+10);
      } catch (InterruptedException e) {
      continue;
      }
      Enumeration claves = app.enviadas.keys();
      Enumeration sellos = app.enviadas.elements();
      while (claves.hasMoreElements()){
        Integer auxkey = (Integer)claves.nextElement();
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        if((System.currentTimeMillis()-
(long)((Tiempo)sellos.nextElement()).get())>vg.margenRT)
          app.enviadas.remove(auxkey); //quitamos la info obsoleta
      }//while
    }//while
  }

  LimpiaTablaEnviadas(MCastApplication app) {
    this.app = app;
  }
}
```

MCastRTProtocol.java

```
/*                              MCastRTProtocol.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                      Autor: Javier Jesús Sánchez Medina
 *                             Comenzado: 19/10/00
 */

package MCastRTv1.apps;

import ants.*;
/** Protocolo multicast orientado a tiempo real.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class MCastRTProtocol extends Protocol {

  public MCastRTProtocol() throws Exception {
    startProtocolDefn();

    startGroupDefn();
    addCapsule("apps.MDataCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.RDataCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.NACKCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.SCapsule");
    endGroupDefn();

    endProtocolDefn();
  }
}
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

UIT-ETSIT BIBLIOTECA DE TELECOMUNICACIONES

SenderApplication.java

```
/*                              SenderApplication.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                    Autor: Javier Jesús Sánchez Medina
 *                            Comenzado: 19/10/00
 */


package MCastRTv1.apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;
import MCastRTv1.apps.Tiempo;
import MCastRTv1.apps.LimpiaTablaEnviadas;
import MCastRTv1.apps.variablesglobales;
/** Aplicación del nodo maestro (Sender)
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class SenderApplication
  extends MCastApplication
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  int target;
  int group;
  boolean enviando = false;

  Button FIN;
  Button send;
  Label petlabel;
  Label pettotallabel;
  Label mdatalabel;
  Label rdatalabel;
  Label nombre;
  int petcount=0;
  int pettotalcount=0;
  int rdatacount=0;
  int interv = 100;  //tiempo entre cápsula y cápsula de la transmisión
  int indice = 0;    //indice del paquete multicast actual

  long mdatabeginTime;
  long rdatabeginTime;
  long petbeginTime;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  final public static String[] defaults = {};  //periodo de limpieza
tabla

  synchronized public void receive(Capsule cap) {
    super.receive(cap);

    if (cap instanceof NACKCapsule) {
      NACKCapsule nackCap = (NACKCapsule) cap;
      petcount++;
      pettotalcount += nackCap.npeticiones;
      switch ((petcount % 10)) {
        case 0:
        petbeginTime = thisNode().time();
        break;
        case 9:
        double diff = (((double)thisNode().time() - (double)
petbeginTime) / 1000.0);
        thisNode().log("Capsulas de petición por segundo:" + 10.0/diff);
        petlabel.setText(String.valueOf(10.0/diff));
        thisNode().log("Total peticiones por segundo:" +
pettotalcount/diff);
        pettotallabel.setText(String.valueOf(pettotalcount/diff));
          pettotalcount = 0;
        break;
      }//switch
      for (int i = 0; i<nackCap.npeticiones; i++){
        Integer aux = new Integer(nackCap.indice[i]);
        if (enviadas.containsKey(aux)){
          send(new
RDataCapsule(port,port,nackCap.getSrc(),aux.intValue(),(long)((Tiempo)env
iadas.get(aux)).get(),null));
        rdatacount++;
          switch ((rdatacount % 10)) {
          case 0:
          rdatabeginTime = thisNode().time();
          break;
          case 9:
          double diff2 = (((double)thisNode().time() - (double)
rdatabeginTime) / 1000.0);
          thisNode().log("Retransmisiones por segundo:" + 10.0/diff2);
          rdatalabel.setText(String.valueOf(10.0/diff2));
          break;
        }//switch
      }//if
      }//for
    }
  }


  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == send) {
      enviando = !enviando;
        return true;
    }else if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.exit(0);
      return true;
    }else return super.handleEvent(evt);
  }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public void run(){
    while (true){
      if (enviando){
        int auxindice = indice++;
        long auxsello = System.currentTimeMillis();
        enviadas.put(new Integer((int)(auxindice)),new
Tiempo(auxsello));//guardamos la cap enviada
        MDataCapsule mdcap = new
MDataCapsule(group,target,port,port,auxindice,auxsello,null);
      send(mdcap);

        switch ((indice % 10)) {
        case 0:
        mdatabeginTime = thisNode().time();
        break;
        case 9:
        double diff = (((double)thisNode().time() - (double)
mdatabeginTime) / 1000.0);
        thisNode().log("Capsulas multicast transmitidas por segundo:" +
10.0/diff);
        mdatalabel.setText(String.valueOf(10.0/diff));
        break;
        }//switch
      }
      try {
      Thread.sleep(interv);
      } catch (InterruptedException e) {
      continue;
      }
    }//while
  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-target")) {
        target = NodeAddress.fromString(k.arg(i));
        k.strike(i);
      } else if (k.key(i).equals("-group")) {
      group = NodeAddress.fromString(k.arg(i));
      k.strike(i);
      }
    }

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {
    thisNode().register(new MCastRTProtocol());

    enviadas = new Hashtable();

    resize(300, 200);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    Panel resultados = new Panel ();
    resultados.setLayout(new GridLayout(11,1,1,1));
    FIN = new Button ("FIN");
    resultados.add(FIN);
    send = new Button("Enviar a " + NodeAddress.toString(group));
    resultados.add(send);
    resultados.add(new Label("No de cápsulas de datos(caps/seg)",
Label.CENTER));
    mdatalabel = new Label("0", Label.CENTER);
    resultados.add(mdatalabel);
    resultados.add(new Label("No de cápsulas retransmitidas(caps/seg)",
Label.CENTER));
    rdatalabel = new Label("0", Label.CENTER);
    resultados.add(rdatalabel);
    resultados.add(new Label("No de cápsulas de petición de
retransmitidas(caps/seg)", Label.CENTER));
    petlabel = new Label("0", Label.CENTER);
    resultados.add(petlabel);
    resultados.add(new Label("No de peticiones de
retransmitidas(pets/seg)", Label.CENTER));
    pettotallabel = new Label("0", Label.CENTER);
    resultados.add(pettotallabel);
    nombre = new Label ("Nombre:" +
NodeAddress.toString(thisNode().getAddress()),Label.CENTER);
    resultados.add(nombre);

    add("Center", resultados);

    new Thread(new LimpiaTablaEnviadas(this)).start();
    new Thread(this).start();
    pack();
    show();

  }

  public SenderApplication() {
    super();
  }
}
```

NACKCapsule.java

```
/*                              NACKCapsule.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 19/10/00
 */


package MCastRTv1.apps;

import ants.*;
import ants.wrapper.*;
import MCastRTv1.apps.variablesglobales;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
/** Cápsula de datos reconocimiento negativo(petición de retransmisión).
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class NACKCapsule
   extends DataCapsule
{
  final private static byte[] MID = findMID("apps.NACKCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.NACKCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int npeticiones; //número de peticiones que se hacen
  int[] indice;    //paquetes que se piden por su índice
  long[] sellotiempo;   //sellos de tiempo del siguiente recibido de cada
uno

  public int length() {
    int s = Xdr.INT + (Xdr.INT + Xdr.LONG) * npeticiones;
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(npeticiones);
    for (int i = 0; i<npeticiones; i++)
      xdr.PUT(indice[i]);
    for (int i = 0; i<npeticiones; i++)
      xdr.PUT(sellotiempo[i]);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    npeticiones = xdr.INT();
    indice = new int[npeticiones];
    for (int i = 0; i<npeticiones; i++)
      indice[i] = xdr.INT();
    sellotiempo = new long[npeticiones];
    for (int i = 0; i<npeticiones; i++)
      sellotiempo[i] = xdr.LONG();
    return xdr;
  }

  public boolean evaluate(Node n) {
    boolean obsoletos = true;
    for (int i = 0; i<npeticiones; i++){
      if (System.currentTimeMillis()-sellotiempo[i]<vg.margenRT){
        obsoletos = false;
        break;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      }//if
    }//for
    if (!obsoletos){
      if (n.getAddress() == getSrc()) //si estamos en el nodo origen
      return n.routeForNode(this, getDst());
      else return n.deliverToApp(this, dpt);
    }
    else {
      return true; //desechamos la cápsula
    }
  }

  public NACKCapsule() { }

  public NACKCapsule(short sa, short da, int na, int[] ind, long[] sello)
{
    super (sa,da,na,new ByteArray(0));

    npeticiones = ind.length;
    indice = ind;
    sellotiempo = sello;
  }
}
```

Tiempo.java

```
/*                              Tiempo.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 19/10/00
 */
package MCastRTv1.apps;
/** Tiempo
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class Tiempo {
   private long t;
   long get (){
      return t;
   }
   Tiempo (long time){
      this.t = time;
   }//constructor
}//Tiempo
```

NodeApplication.java

```
/*                              NodeApplication.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 19/10/00
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
 */


package MCastRTv1.apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;
import MCastRTv1.apps.Tiempo;
import MCastRTv1.apps.LimpiaTablaEnviadas;
import MCastRTv1.apps.variablesglobales;

/** Thread que se encarga de mantener al nodo intermedio suscrito
mientras proceda
*/

class Suscribe
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  NodeApplication napp;
  int nsuscripciones;

  public void run() {

    while (true) {
      try {
      Thread.sleep(vg.tiempoSuscripcion+10);
      } catch (InterruptedException e) {
      continue;
      }
      if (napp.suscount>nsuscripciones){ //se sigue suscribiendo si hay
hijos que quieren
        nsuscripciones = napp.suscount;
        napp.send(new
SCapsule(napp.group,napp.target,napp.getPort(),napp.getPort()));
      }
    }
  }

  Suscribe(NodeApplication napp) {
    this.napp = napp;
    nsuscripciones = 0;
  }
}

/** Aplicación que corre en los nodos intermedios.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class NodeApplication
  extends MCastApplication
  implements Runnable
{
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  variablesglobales vg = new variablesglobales();

  int target;
  int group;
  int nodoAnterior;

  public short getPort(){
    return port;
  }

  Button FIN;

  Label petlabel;
  Label pettotallabel;
  Label mDatalabel;
  Label rDatalabel;
  Label suslabel;
  Label nombre;

  int petcount = 0;
  int pettotalcount = 0;
  int mdatacount = 0;
  int rdatacount = 0;
  int suscount = 0;
  long petbeginTime;
  long mdatabeginTime;
  long susbeginTime;
  long rdatabeginTime;

  final public static String[] defaults = { };
  //periodo de limpieza tabla enviadas y de petición de NACKs
  //ojo:tiempoPeticionNACKs debe ser << margenRT para que sirva de algo

  synchronized public void receive(Capsule cap) {
    super.receive(cap);

    if (cap instanceof NACKCapsule) {
      NACKCapsule nackCap = (NACKCapsule) cap;
      petcount++;
      pettotalcount += nackCap.npeticiones;
      switch ((petcount % 10)) {
      case 0:
      petbeginTime = thisNode().time();
      break;
      case 9:
      double diff = (((double)thisNode().time() - (double) petbeginTime)
/ 1000.0);
        thisNode().log("Capsulas de petición por segundo:" + 10.0/diff);
      petlabel.setText(String.valueOf(10.0/diff));
        thisNode().log("Total peticiones por segundo:" +
pettotalcount/diff);
      pettotallabel.setText(String.valueOf(pettotalcount/diff));
        pettotalcount = 0;
      break;
      }//switch
      for (int i = 0; i<nackCap.npeticiones; i++){
        Integer aux = new Integer(nackCap.indice[i]);
        if (enviadas.containsKey(aux))
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
          send(new
RDataCapsule(port,port,nackCap.getSrc(),aux.intValue(),(long)((Tiempo)env
iadas.get(aux)).get(),null));
        else {
          pedidas.put(aux,new
Tiempo((long)nackCap.sellotiempo[i]));//ponemos la petición en la lista
        }//if
      }//for

    }else if (cap instanceof MDataCapsule) {
      mdatacount++;
      switch ((mdatacount % 10)) {
      case 0:
      mdatabeginTime = thisNode().time();
      break;
      case 9:
      double diff = (((double)thisNode().time() - (double)
mdatabeginTime) / 1000.0);
      thisNode().log("Capsulas multicast transmitidas por segundo:" +
10.0/diff);
      mDatalabel.setText(String.valueOf(10.0/diff));
      break;
      }//switch
      MDataCapsule mdCap = (MDataCapsule) cap;
      nodoAnterior = mdCap.getPrevious();
      enviadas.put(new Integer(mdCap.indice),new
Tiempo(mdCap.sellotiempo));     //solo guardamos el sello de tiempo
      //en realidad deberíamos guardar también los datos multicast

    }else if (cap instanceof RDataCapsule) {
      rdatacount++;
      switch ((rdatacount % 10)) {
      case 0:
      rdatabeginTime = thisNode().time();
      break;
      case 9:
      double diff = (((double)thisNode().time() - (double)
rdatabeginTime) / 1000.0);
      thisNode().log("Retransmisiones recibidas por segundo:" +
10.0/diff);
      rDatalabel.setText(String.valueOf(10.0/diff));
      break;
      }//switch
      RDataCapsule rdCap = (RDataCapsule) cap;
      enviadas.put(new Integer(rdCap.indice),new
Tiempo(rdCap.sellotiempo));

    }else if (cap instanceof SCapsule){
      suscount++; //solamenta vamos contando las suscripciones
      switch ((suscount % 10)) {
      case 0:
      susbeginTime = thisNode().time();
      break;
      case 9:
      double diff = (((double)thisNode().time() - (double) susbeginTime)
/ 1000.0);
      thisNode().log("Suscripciones por segundo:" + 10.0/diff);
      suslabel.setText(String.valueOf(10.0/diff));
      break;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      }//switch
    }
  }


  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.exit(0);
        return true;
    } else return super.handleEvent(evt);
  }

  public void run(){
    while (true) {
      try {
      Thread.sleep(vg.tiempoPeticionNACKs+10);
      } catch (InterruptedException e) {
      continue;
      }
      Enumeration claves = pedidas.keys();
      Enumeration sellos = pedidas.elements();

      int[] indi = new int[0];
      long[] sells = new long[0];
      while (claves.hasMoreElements()){
        Integer auxkey = (Integer)claves.nextElement();
        if((System.currentTimeMillis()-
(long)((Tiempo)sellos.nextElement()).get())>vg.margenRT)
          pedidas.remove(auxkey); //quitamos la info obsoleta
        else {
        int len = (indi != null) ? indi.length : 0;
        int[] auxindices = new int[len+1];
        long[] auxsellos = new long[len+1];
        if (len > 0) {
          System.arraycopy(indi, 0, auxindices, 0, len);
          System.arraycopy(sells, 0, auxsellos, 0, len);
        }
        auxindices[len] = auxkey.intValue();
        auxsellos[len] = (long)((Tiempo)pedidas.get(auxkey)).get();
        indi = auxindices;
        sells = auxsellos;
        }//if
      }//while
      if (indi.length>0) //si hay alguna petición útil
        send(new NACKCapsule(port, port, nodoAnterior, indi, sells));
    }//while

  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-target")) {
        target = NodeAddress.fromString(k.arg(i));
        k.strike(i);
      } else if (k.key(i).equals("-group")) {
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      group = NodeAddress.fromString(k.arg(i));
      k.strike(i);
      }
    }

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {
    thisNode().register(new MCastRTProtocol());

    enviadas = new Hashtable();
    pedidas = new Hashtable();

    resize(300, 200);

    Panel resultados = new Panel ();
    resultados.setLayout(new GridLayout(12,1,1,1));

    FIN = new Button("FIN");
    resultados.add(FIN);
    resultados.add(new Label("No de Capsulas de Retransmision(caps/seg)",
Label.CENTER));
    petlabel = new Label("0", Label.CENTER);
    resultados.add(petlabel);
    resultados.add(new Label("No Peticiones de Retransmision(pets/seg)",
Label.CENTER));
    pettotallabel = new Label("0", Label.CENTER);
    resultados.add(pettotallabel);
    resultados.add(new Label("No Cápsulas de Multicast(caps/seg)",
Label.CENTER));
    mDatalabel = new Label("0", Label.CENTER);
    resultados.add(mDatalabel);
    resultados.add(new Label("No Cápsulas Retransmitidas(caps/seg)",
Label.CENTER));
    rDatalabel = new Label("0", Label.CENTER);
    resultados.add(rDatalabel);
    resultados.add(new Label("No Cápsulas de suscripción(caps/seg)",
Label.CENTER));
    suslabel = new Label("0", Label.CENTER);
    resultados.add(suslabel);
    nombre = new Label
("Nombre:"+NodeAddress.toString(thisNode().getAddress()),Label.CENTER);

    resultados.add(nombre);

    add("Center", resultados);

    new Thread(new LimpiaTablaEnviadas(this)).start();
    new Thread(new Suscribe(this)).start();
    new Thread(this).start();
    pack();
    show();

  }

  public NodeApplication() {
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    super();
  }
}
```

ReceiverApplication.java

```
/*                              ReceiverApplication.java
 *
 *             Ejemplo de protocolo multicast orientado a tiempo real
 *                     Autor: Javier Jesús Sánchez Medina
 *                             Comenzado: 19/10/00
 */


package MCastRTv1.apps;

import java.awt.*;

import ants.*;
import utils.*;
import MCastRTv1.apps.variablesglobales;
/** Aplicación que corre en los nodos Receiver
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class ReceiverApplication
  extends Application
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  int target;   //target se refiere al nodo sender
  int group;
  boolean suscrito = false;  // esta suscrito?
  boolean primeraCapDatos = true;

  Button suscribirse;
  Button FIN;
  Label mDatalabel;
  Label rDatalabel;
  Label perdidaslabel;
  Label latencialabel;
  Label nombre;
  int mdatacount;
  int rdatacount;
  int mdata_rdatacount;
  int percount;
  long mdatabeginTime;
  long rdatabeginTime;
  long latenciaMedia;

  int ultimoindice;   //debe inicializarse al valor de ínidice de la
primera cap de datos recibida
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  final public static String[] defaults = {};  //periodo de emisión cap
subcripcion

  synchronized public void receive(Capsule cap) {
    super.receive(cap);

    if (cap instanceof MDataCapsule) {
      MDataCapsule mCap = (MDataCapsule) cap;
      if (primeraCapDatos){  //en el caso de que se trate de la primera
cap de datos
      ultimoindice = mCap.indice;
      primeraCapDatos = false;
      latenciaMedia = 0;
      mdatacount = rdatacount = mdata_rdatacount = 0;
      }
      latenciaMedia += thisNode().time() - mCap.sellotiempo;
      mdatacount++;
      mdata_rdatacount++;

      switch (mdatacount%10){
      case 0:
      mdatabeginTime = thisNode().time();
      break;
      case 9:
      double diff = (((double)thisNode().time() - (double)
mdatabeginTime) / 1000.0);
      thisNode().log("Capsulas multicast transmitidas por segundo:" +
10.0/diff);
      mDatalabel.setText(String.valueOf(10.0/diff));
      thisNode().log("Perdidas de capsulas (%):" +
percount*100.0/(percount+mdatacount+rdatacount));
      perdidaslabel.setText(String.valueOf(percount*100.0/(percount+mdata
count+rdatacount)));
      thisNode().log("Latencia media del servicio (ms):"
+latenciaMedia/mdata_rdatacount);
      latencialabel.setText(String.valueOf(latenciaMedia/mdata_rdatacount
));
      mdata_rdatacount = 0;
      latenciaMedia = 0;
      break;
      }//switch

      if (mCap.indice>ultimoindice+1){  //si nos faltan cápsulas hay que
pedirlas al nodo anterior
      int nodoAnterior = mCap.getPrevious();
        int[] indices = new int[0];
        long[] sellos = new long[0];
        for (int i = ultimoindice+1; i<mCap.indice; i++){
          percount++;
        int len = (indices != null) ? indices.length : 0;
        int[] auxindices = new int[len+1];
        long[] auxsellos = new long[len+1];
        if (len > 0) {
          System.arraycopy(indices, 0, auxindices, 0, len);
          System.arraycopy(sellos, 0, auxsellos, 0, len);
        }
        auxindices[len] = i;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        auxsellos[len] = mCap.sellotiempo; //en todas Las NACKs ponemos
el sello de la sig cap
        indices = auxindices;              //cuando la sig esté obsoleta,
las anteriores también
        sellos = auxsellos;
      }//for
      send(new NACKCapsule(port, port, nodoAnterior, indices, sellos));
      }//if
      ultimoindice = mCap.indice;

    }else if (cap instanceof RDataCapsule) {
      RDataCapsule rdcap = (RDataCapsule) cap;
      rdatacount++;
      mdata_rdatacount++;
      latenciaMedia += thisNode().time() - rdcap.sellotiempo;
      switch (rdatacount%10){
      case 0:
      rdatabeginTime = thisNode().time();
      break;
      case 9:
      double diff = (((double)thisNode().time() - (double)
rdatabeginTime) / 1000.0);
      thisNode().log("Retransmisiones recibidas por segundo:" +
10.0/diff);
      rDatalabel.setText(String.valueOf(10.0/diff));
      break;
      }//switch
      percount--;

    }//if
  }


  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == suscribirse) {
      suscrito = !suscrito;
      //en el caso en que se suscriba ponemos inicialmente el flag
primera cap datos a '1'
      if (suscrito) {
        primeraCapDatos = true;
      }
        return true;
    }else if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.exit(0);
        return true;
    } else return super.handleEvent(evt);
  }

  public void run(){
    while (true) {
      try {
      Thread.sleep(vg.tiempoSuscripcion+10);
      } catch (InterruptedException e) {
      continue;
      }
      if (suscrito){
        send(new SCapsule(group,target,port,port));
      }
    }//while
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-target")) {
        target = NodeAddress.fromString(k.arg(i));
        k.strike(i);
      } else if (k.key(i).equals("-group")) {
      group = NodeAddress.fromString(k.arg(i));
      k.strike(i);
      }
    }

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {
    thisNode().register(new MCastRTProtocol());

    resize(300, 200);

    Panel resultados = new Panel ();
    resultados.setLayout(new GridLayout(11,1,1,1));
    FIN = new Button ("FIN");
    resultados.add(FIN);
    suscribirse = new Button("Suscribirse a " +
NodeAddress.toString(group));
    resultados.add(suscribirse);
    resultados.add(new Label("Cápsulas de datos multicast
(Caps/segundo)",Label.CENTER));
    mDatalabel = new Label("0", Label.CENTER);
    resultados.add(mDatalabel);
    resultados.add(new Label("Cápsulas de datos retransmitidos
(Caps/segundo)",Label.CENTER));
    rDatalabel = new Label("0", Label.CENTER);
    resultados.add(rDatalabel);
    resultados.add(new Label("Capsulas perdidas (%)",Label.CENTER));
    perdidaslabel = new Label("0", Label.CENTER);
    resultados.add(perdidaslabel);
    resultados.add(new Label("Latencia media de las cápsulas
(ms)",Label.CENTER));
    latencialabel = new Label("0", Label.CENTER);
    resultados.add(latencialabel);
    nombre = new Label
("Node:"+NodeAddress.toString(thisNode().getAddress()),Label.CENTER);
    resultados.add(nombre);

    add("Center", resultados);

    new Thread(this).start();
    pack();
    show();
  }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public ReceiverApplication() {
    super();
  }
}
```

MCastApplication.java

```
/*                              MCastApplication.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 19/10/00
 */
package MCastRTv1.apps;

import java.util.*;
import ants.*;
/** Clase madre de las aplicaciones Sender y Node del prototipo.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class MCastApplication
  extends Application
{
  Hashtable enviadas;
  Hashtable pedidas;
}
```

MDataCapsule.java

```
/*                              MDataCapsule.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 19/10/00
 */


package MCastRTv1.apps;

import ants.*;
import ants.wrapper.*;
import MCastRTv1.apps.variablesglobales;
/** Cápsula de datos multicast.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class MDataCapsule
   extends DataCapsule
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
{
  final private static byte[] MID = findMID("apps.MDataCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.MDataCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int group;
  int target;
  int indice;
  long sellotiempo;

  public int length() {
    int s = Xdr.INT + Xdr.INT + Xdr.INT + Xdr.LONG;
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(group);
    xdr.PUT(target);
    xdr.PUT(indice);
    xdr.PUT(sellotiempo);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    group = xdr.INT();
    target = xdr.INT();
    indice = xdr.INT();
    sellotiempo = xdr.LONG();

    return xdr;
  }

  public boolean evaluate(Node n) {
    if ((System.currentTimeMillis()-sellotiempo)<vg.margenRT) {//si la
cápsula es válida para RT
      W_JAI m = (W_JAI)n.getCache().get(group, target);
      if (m != null) {
        if (m.ai1 != null)
        for (int i = 0; i < m.ai1.length; i++)
          n.routeForNode(this, m.ai1[i]);
        return n.deliverToApp(this, dpt); //pasamos la cáp a la
aplicación siempre
      }
      return true;       //no hay suscriptores
    }else return true;  //si la cápsula es obsoleta se desecha
  }

  public MDataCapsule() { }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public MDataCapsule(int ga, int ta, short sa, short da, int ind, long
sello, ByteArray d) {
    super (sa,da,0,d);

    group = ga;
    target = ta;
    indice = ind;
    sellotiempo = sello;
  }

}
```

RDataCapsule.java

```
/*                              RDataCapsule.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                     Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 19/10/00
 */


package MCastRTv1.apps;

import ants.*;
import ants.wrapper.*;
import MCastRTv1.apps.variablesglobales;
/** Cápsula de datos retransmitidos.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class RDataCapsule
   extends DataCapsule
{
  final private static byte[] MID = findMID("apps.RDataCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.RDataCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int indice;
  long sellotiempo;

  public int length() {
    int s = Xdr.INT + Xdr.LONG;
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(indice);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    xdr.PUT(sellotiempo);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    indice = xdr.INT();
    sellotiempo = xdr.LONG();

    return xdr;
  }

  public boolean evaluate(Node n) {
    if ((System.currentTimeMillis()-sellotiempo)<vg.margenRT) {//si la
cápsula es válida para RT
      if (n.getAddress()==getDst())
        return n.deliverToApp(this, dpt);
      else return n.routeForNode(this, getDst());
    }else return true;//si la cápsula es obsoleta se desecha
  }

  public RDataCapsule() { }

  public RDataCapsule(short sa, short da, int na, int ind, long sello,
ByteArray d) {
    super (sa,da,na,d);

    indice = ind;
    sellotiempo = sello;
  }

}
```

SCapsule.java

```
/*                              SCapsule.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                     Autor: Javier Jesús Sánchez Medina
 *                             Comenzado: 19/10/00
 */


package MCastRTv1.apps;

import ants.*;
import ants.wrapper.*;
import MCastRTv1.apps.variablesglobales;
/** Cápsula de suscripción al servicio
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */
public class SCapsule
  extends Capsule
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
{
  final private static byte[] MID = findMID("apps.SCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.SCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int group;
  int target;
  short spt;
  short dpt;
  int reverse;

  public int length() {
    return super.length() + Xdr.INT + Xdr.INT + Xdr.SHORT + Xdr.SHORT +
Xdr.INT;
  }

  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(group);
    xdr.PUT(target);
    xdr.PUT(spt);
    xdr.PUT(dpt);
    xdr.PUT(reverse);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    group = xdr.INT();
    target = xdr.INT();
    spt = xdr.SHORT();
    dpt = xdr.SHORT();
    reverse = xdr.INT();

    return xdr;
  }


  public boolean evaluate(Node n) {
    W_JAI m = (W_JAI)n.getCache().get(group, target);
    if (m == null) {
      m = new W_JAI();
      n.getCache().put(group, target, m, 5);
    } else {
    }
    if (reverse != 0) {//no estoy en el nodo origen
      boolean found = false;
      for (int i = 0; m.ai1 != null && i < m.ai1.length; i++) {
      if (m.ai1[i] == reverse) { found = true; break; }
      }
      if (!found) {
      int len = (m.ai1 != null) ? m.ai1.length : 0;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      int[] nn = new int[len+1];
      if (len > 0) System.arraycopy(m.ai1, 0, nn, 0, len);
      nn[len] = reverse;
      m.ai1 = nn;
      }
      if (n.getAddress() != target) {
        return n.deliverToApp(this,dpt);
      }else return true;
    }else {//estoy en el nodo origen
      if (System.currentTimeMillis() - m.j1 < vg.tiempoSuscripcion)
return true;
      else m.j1 = System.currentTimeMillis();
      reverse = n.getAddress();
      return n.routeForNode(this, target);
    }
  }

  public SCapsule() { }

  public SCapsule(int ga, int ta, short sa, short da) {
    group = ga;
    target = ta;
    spt = sa;
    dpt = da;
    this.reverse = 0;
  }
}
```

variablesglobales.java

```
/*                              variablesglobales.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                     Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 19/10/00
 */
package MCastRTv1.apps;
/** Variables globales del prototipo
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class variablesglobales {
public long margenRT = 30;
public int tiempoLimpieza = 1000;
public int tiempoPeticionNACKs = 200;
public int tiempoSuscripcion = 300;
}
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

MCastRTv2

AuxTipo.java

```
/*                                  AuxTipo.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                                  versión en red (v2)
 *                        Autor: Javier Jesús Sánchez Medina
 *                                Comenzado: 13/12/00
 */


package MCastRTv2.apps;
/**
 *
 *      Tipo auxiliar para el filtrado activo
 *
 *      @author: Javier Jesús Sánchez Medina
 */

public class AuxTipo{
     public int[] indices;
     public long[] sellos;
     public AuxTipo(int[] i, long[] j){
        indices = i;
        sellos = j;
     }
  }
```

LimpiaTablaEnviadas.java

```
/**                               LimpiaTablaEnviadas.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                                  versión en red (v2)
 *                        Autor: Javier Jesús Sánchez Medina
 *                                Comenzado: 13/12/00
 */

package MCastRTv2.apps;

import java.util.*;

import ants.*;
import MCastRTv2.apps.variablesglobales;
/** Thread que limpia la tabla de capsulas multicast enviadas.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
class LimpiaTablaEnviadas
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  MCastApplication app;

  public void run() {

    while (true) {
      try {
      Thread.sleep(vg.tiempoLimpieza+10);
      } catch (InterruptedException e) {
      continue;
      }
      Enumeration claves = app.enviadas.keys();
      Enumeration sellos = app.enviadas.elements();
      while (claves.hasMoreElements()){
        Integer auxkey = (Integer)claves.nextElement();
        if((System.currentTimeMillis()-
(long)((Tiempo)sellos.nextElement()).get())>vg.margenRT)
          app.enviadas.remove(auxkey); //quitamos la info obsoleta
      }//while
    }//while
  }

  LimpiaTablaEnviadas(MCastApplication app) {
    this.app = app;
  }
}
```

MCastApplication.java

```
/*                              MCastApplication.java
 *
 *             Ejemplo de protocolo multicast orientado a tiempo real
 *                                  versión en red (v3)
 *                        Autor: Javier Jesús Sánchez Medina
 *                                Comenzado: 18/04/00
 */


package MCastRTv2.apps;

import java.util.*;
import ants.*;

/** Madre de todas las aplicaciones de la topología multicast.
 *
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class MCastApplication
  extends Application
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
{
  Hashtable enviadas;
  Hashtable pedidas;
}
```

MCastCapsule.java

```
/*                              MCastCapsule.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                                versión en red (v2)
 *                        Autor: Javier Jesús Sánchez Medina
 *                                Comenzado: 13/12/00
 */


package MCastRTv2.apps;

import ants.*;
/**   Cápsula madre de las cápsulas de datos multicast, las
confirmaciones negativas y las de
 *    datos retransmitidos
 */
public class MCastCapsule
   extends DataCapsule
{
  int group;
  int target;

  public int length() {
    int s = Xdr.INT + Xdr.INT;
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(group);
    xdr.PUT(target);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    group = xdr.INT();
    target = xdr.INT();

    return xdr;
  }

  static AuxTipo eliminarEntrada(AuxTipo a, int i){
     int len = a.indices.length;
     if (len>1) {
        int[] aux1 = new int[len-1];
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        long[] aux2 = new long[len-1];
        System.arraycopy(a.indices,0,aux1,0,i);
        System.arraycopy(a.sellos,0,aux2,0,i);
        System.arraycopy(a.indices,i+1,aux1,i,len-i-1);
        System.arraycopy(a.sellos,i+1,aux2,i,len-i-1);
        return new AuxTipo(aux1,aux2);
      }else {
        return new AuxTipo(new int[0],new long[0]);
      }
  }//eliminarEntrada

  static AuxTipo nuevaEntrada(AuxTipo a, int indice, long sello){ //la
añade siempre al final
      int len = a.indices.length;
      int[] aux1 = new int[len+1];
      long[] aux2 = new long[len+1];
      System.arraycopy(a.indices,0,aux1,0,len);
      System.arraycopy(a.sellos,0,aux2,0,len);
      aux1[len] = indice;
      aux2[len] = sello;
      return new AuxTipo(aux1,aux2);
  }//nuevaEntrada

  public MCastCapsule() { }

  public MCastCapsule(int ga, int ta, short sa, short da, int na,
ByteArray d) {
     super (sa,da,na,d);

     group = ga;
     target = ta;
  }

}
```

MCastRTProtocol.java

```
/**                              MCastRTProtocol.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                                  versión en red (v2)
 *                        Autor: Javier Jesús Sánchez Medina
 *                                Comenzado: 13/12/00
 */

package MCastRTv2.apps;

import ants.*;
/**
 * Protocolo multicast orientado a tiempo real v2
 *
 * @author: Javier Jesús Sánchez Medina
 *
 */

public class MCastRTProtocol extends Protocol {
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public MCastRTProtocol() throws Exception {
    startProtocolDefn();

    startGroupDefn();
    addCapsule("apps.MDataCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.RDataCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.NACKCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.SCapsule");
    endGroupDefn();

    endProtocolDefn();
  }
}
```

MDataCapsule.java

```
/*                              MDataCapsule.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                                versión en red (v2)
 *                        Autor: Javier Jesús Sánchez Medina
 *                                Comenzado: 13/12/00
 */


package MCastRTv2.apps;

import ants.*;
import ants.wrapper.*;
import MCastRTv2.apps.variablesglobales;
/** Cápsula que distribuye los datos multicast
 *
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class MDataCapsule
   extends MCastCapsule
{
  final private static byte[] MID = findMID("apps.MDataCapsule");
  protected byte[] mid() { return MID; }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  final private static byte[] PID = findPID("apps.MDataCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int indice;
  int nodoPrevio; //nos hace falta para pasarle a la aplicacion el nodo
anterior verdadero

  public int length() {
    int s = Xdr.INT + Xdr.INT;
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(indice);
    xdr.PUT(nodoPrevio);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    indice = xdr.INT();
    nodoPrevio = xdr.INT();

    return xdr;
  }

  public boolean evaluate(Node n) {

    W_JAIAIAJ m = (W_JAIAIAJ)n.getCache().get(group, target);
    if (m != null) {
      //FILTRADO ACTIVO
      //limpieza de lo obsoleto
      int i = 0;
      while ((m.ai2.length>0)&&((n.time() - m.aj1[0])>vg.margenRT)){
          AuxTipo aux = (AuxTipo) eliminarEntrada(new AuxTipo(m.ai2,
m.aj1), 0);
        m.ai2 = aux.indices;
        m.aj1 = aux.sellos;
        }
      //limpieza de lo obsoleto

      if((m.ai2.length == 0)||(m.ai2[m.ai2.length-1]<indice+1)){
        AuxTipo aux = (AuxTipo) nuevaEntrada(new AuxTipo(m.ai2,
m.aj1),indice +1, n.time() + vg.margenRT);
        //programamos la llegada del proximo indice
        m.ai2 = aux.indices;
        m.aj1 = aux.sellos;
        //seguimos para la distribución multicast
      }else{
        int j;
        boolean encontrado = false;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        for (j = 0; j<m.ai2.length; j++)
          if (indice == m.ai2[j]){
            encontrado = true;
            break;
          }
        if (!encontrado){
          return true; //se desecha el paquete obsoleto
        }else{
        }
      }
      //FILTRADO ACTIVO
      if (n.getAddress()!=getSrc()) //si no estamos en el nodo origen le
pasamos a la aplicación el previo
         nodoPrevio = getPrevious();
      if (m.ai1 != null){
        for (int k = 0; k < m.ai1.length; k++)
          n.routeForNode(this, m.ai1[k]);
      }
    } return n.deliverToApp(this, dpt); //pasamos la cáp a la aplicación
  }//evaluate()

  public MDataCapsule() { }

  public MDataCapsule(int ga, int ta, short sa, short da, int ind,
ByteArray d) {
    super (ga,ta,sa,da,0,d);

    indice = ind;
  }

}
```

NACKCapsule.java

```
/*                              NACKCapsule.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                                  versión en red (v2)
 *                        Autor: Javier Jesús Sánchez Medina
 *                                  Comenzado: 13/12/00
 */


package MCastRTv2.apps;

import ants.*;
import ants.wrapper.*;
import MCastRTv2.apps.variablesglobales;

/** Cápsula que solicita la retransmisión de los datos multicast
 *
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */
public class NACKCapsule
   extends MCastCapsule
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
{
  final private static byte[] MID = findMID("apps.NACKCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.NACKCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int npeticiones; //número de peticiones que se hacen
  int[] indice;    //paquetes que se piden por su índice

  public int length() {
    int s = Xdr.INT + (Xdr.INT ) * npeticiones;
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(npeticiones);
    for (int i = 0; i<npeticiones; i++)
      xdr.PUT(indice[i]);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    npeticiones = xdr.INT();
    indice = new int[npeticiones];
    for (int i = 0; i<npeticiones; i++)
      indice[i] = xdr.INT();

    return xdr;
  }

  public boolean evaluate(Node n) {

    W_JAIAIAJ m = (W_JAIAIAJ)n.getCache().get(group, target);
    if (m == null){ //si estamos en un Receiver
      return n.routeForNode(this, getDst());
    }else{
      // FILTRADO ACTIVO
      if (m.ai2.length==0){
        return true; //todas las peticiones están obsoletas o ya no se
está suscrito
      }else{
        boolean encontrado = false;
        int posicion = -1;
        for (int j = 0; j < npeticiones; j++)
        for (int i = 0; i<m.ai2.length; i++)
            if (m.ai2[i] == indice[j]) {
              encontrado = true; break;
            }//if
        if (!encontrado){
        //si no hay registros no obsoletos desechamos la cápsula
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
          return true;
        }else {
        }
      }//if
      if (n.getAddress() == getSrc())
      return n.routeForNode(this, getDst());
      else
        return n.deliverToApp(this, dpt); //si estamos en el destino
    }
  }

  public NACKCapsule() { }

  public NACKCapsule(int ga, int ta, short sa, short da, int na, int[]
ind) {
    super (ga,ta,sa,da,na,new ByteArray(0));

    npeticiones = ind.length;
    indice = ind;

  }
}
```

NodeApplication.java

```
/*                              NodeApplication.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                                  versión en red (v3)
 *                        Autor: Javier Jesús Sánchez Medina
 *                                Comenzado: 18/04/00
 */


package MCastRTv2.apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;
import MCastRTv2.apps.Tiempo;
import MCastRTv2.apps.LimpiaTablaEnviadas;
import MCastRTv2.apps.variablesglobales;

/** Thread que se encarga de mantener al nodo intermedio suscrito
mientras proceda
*/

class Suscribe
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  NodeApplication napp;
  int nsuscripciones;
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public void run() {

    while (true) {
      try {
      Thread.sleep(vg.tiempoSuscripcion+10);
      } catch (InterruptedException e) {
      continue;
      }
      if (napp.suscount>nsuscripciones){ //se sigue suscribiendo si hay
hijos que quieren
        nsuscripciones = napp.suscount;
        napp.send(new
SCapsule(napp.group,napp.target,napp.getPort(),napp.getPort(),true));
      }
    }
  }

  Suscribe(NodeApplication napp) {
    this.napp = napp;
    nsuscripciones = 0;
  }
}

/**
*     Nodos que solicitan ser admitidos en el servicio multicast
*/

class Peticionarios{

  int[] nodos;
  long tiempo;

  Peticionarios(){ //constructor
    nodos = new int[0];
  }

}//Peticionarios

class LimpiaListaPeticionarios
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  NodeApplication napp;

  public void run() {
    while (true) {
      try {
      Thread.sleep(vg.tiempoLimpieza);
      } catch (InterruptedException e) {
      continue;
      }
      Enumeration e = napp.peticionarios.keys();
      while (e.hasMoreElements()){
      Integer auxIndice = (Integer)e.nextElement();
      Peticionarios auxPet = (Peticionarios)
napp.peticionarios.get(auxIndice);
      if ((napp.thisNode().time() - auxPet.tiempo)>vg.margenRT)
        napp.peticionarios.remove(auxIndice);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      }
    }
  }//run

  LimpiaListaPeticionarios(NodeApplication napp){
    this.napp = napp;
  }//constructor
}

/** Aplicación que corre en los nodos intermedios de la topología
multicast.
 *
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class NodeApplication
  extends MCastApplication
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  int target;
  int group;
  int nodoAnterior;

  Hashtable peticionarios;

  public short getPort(){
    return port;
  }

  Button FIN;

  Label petlabel;
  Label pettotallabel;
  Label mDatalabel;
  Label rDatalabel;
  Label suslabel;
  Label estado;
  Label nombre;

  int petcount;
  int pettotalcount;
  int mdatacount;
  int rdatacount;
  int suscount;
  long mdatabeginTime;

  double[] estadisticas_caps_mcast_seg;
  double[] estadisticas_caps_retransmitidas_seg;
  double[] estadisticas_caps_peticiones_retransmision_seg;
  double[] estadisticas_peticiones_retransmision_seg;
  double[] estadisticas_caps_suscripcion_seg;

  final public static String[] defaults = { };
  //periodo de limpieza tabla enviadas y de petición de NACKs
  //ojo:tiempoPeticionNACKs debe ser << margenRT para que sirva de algo
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  synchronized public void receive(Capsule cap) {
    super.receive(cap);

    if (cap instanceof NACKCapsule) {
      NACKCapsule nackCap = (NACKCapsule) cap;
      petcount++;
      pettotalcount += nackCap.npeticiones;
      for (int i = 0; i<nackCap.npeticiones; i++){
        Integer aux = new Integer(nackCap.indice[i]);
        if (enviadas.containsKey(aux)){
          thisNode().log("SI encontrado indice en enviadas");
          send(new
RDataCapsule(group,target,port,port,nackCap.getSrc(),aux.intValue(),null)
);
        }else {
          thisNode().log("NO encontrado indice en enviadas");
          pedidas.put(aux,new Tiempo(thisNode().time()));//ponemos la
petición en la lista
        Peticionarios auxPet;
        if(peticionarios.containsKey(aux)){
          auxPet = (Peticionarios)peticionarios.get(aux);
        }else auxPet = new Peticionarios();
        int[] auxnodos = new int[auxPet.nodos.length+1];
        System.arraycopy(auxPet.nodos,0,auxnodos,0,auxPet.nodos.length);
        auxnodos[auxPet.nodos.length] = nackCap.getSrc(); //ponemos el
nuevo peticionario
        auxPet.nodos = auxnodos;
        auxPet.tiempo = thisNode().time();
        peticionarios.put(aux,auxPet); //actualizamos la lista de
peticionarios de ese indice
        }//if
      }//for

    }else if (cap instanceof MDataCapsule) { //Las cápsulas multicast son
las que disparan la toma de medidas
      mdatacount++;
      MDataCapsule mdCap = (MDataCapsule) cap;
      nodoAnterior = mdCap.nodoPrevio;
      enviadas.put(new Integer(mdCap.indice),new
Tiempo(thisNode().time()));
      //solo guardamos el tiempo actual
      //en una aplicación real deberíamos guardar también los datos
multicast
      if(mdatacount>=vg.NCapsulasPeriodo){ //toca hacer estadísticas
      double periodo = thisNode().time() - mdatabeginTime;
      estado.setText("mdCap.indice = "+mdCap.indice);
      double media_caps_mcast_seg = 1000 * mdatacount / periodo;
      double media_caps_retransmitidas_seg = 1000 * rdatacount / periodo;
      double media_caps_peticiones_retransmi_seg = 1000 * petcount /
periodo;
      double media_peticiones_retransmi_seg = 1000 * pettotalcount /
periodo;
      double media_suscripciones_seg = 1000 * suscount / periodo;

      mDatalabel.setText(String.valueOf(media_caps_mcast_seg));
      rDatalabel.setText(String.valueOf(media_caps_retransmitidas_seg));
      petlabel.setText(String.valueOf(media_caps_peticiones_retransmi_seg
));
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      pettotallabel.setText(String.valueOf(media_peticiones_retransmi_seg
));
      suslabel.setText(String.valueOf(media_suscripciones_seg));

      int len1 = estadisticas_caps_mcast_seg.length;
      int len2 = estadisticas_caps_retransmitidas_seg.length;
      int len3 = estadisticas_caps_peticiones_retransmision_seg.length;
      int len4 = estadisticas_peticiones_retransmision_seg.length;
      int len5 = estadisticas_caps_suscripcion_seg.length;
      double[] aux1 = new double[len1+1];
      double[] aux2 = new double[len2+1];
      double[] aux3 = new double[len3+1];
      double[] aux4 = new double[len4+1];
      double[] aux5 = new double[len5+1];
      System.arraycopy(estadisticas_caps_mcast_seg,0,aux1,0,len1);
      System.arraycopy(estadisticas_caps_retransmitidas_seg,0,aux2,0,len2
);
      System.arraycopy(estadisticas_caps_peticiones_retransmision_seg,0,a
ux3,0,len3);
      System.arraycopy(estadisticas_peticiones_retransmision_seg,0,aux4,0
,len4);
      System.arraycopy(estadisticas_caps_suscripcion_seg,0,aux5,0,len5);
      aux1[len1] = media_caps_mcast_seg;
      aux2[len2] = media_caps_retransmitidas_seg;
      aux3[len3] = media_caps_peticiones_retransmi_seg;
      aux4[len4] = media_peticiones_retransmi_seg;
      aux5[len5] = media_suscripciones_seg;

      estadisticas_caps_mcast_seg = aux1;
      estadisticas_caps_retransmitidas_seg = aux2;
      estadisticas_caps_peticiones_retransmision_seg = aux3;
      estadisticas_peticiones_retransmision_seg = aux4;
      estadisticas_caps_suscripcion_seg = aux5;

      petcount = pettotalcount = mdatacount = rdatacount = suscount = 0;
      mdatabeginTime = thisNode().time();
      }


    }else if (cap instanceof RDataCapsule) {
      rdatacount++;
      RDataCapsule rdCap = (RDataCapsule) cap;
      enviadas.put(new Integer(rdCap.indice),new
Tiempo(thisNode().time()));
      Integer auxIndice = new Integer(rdCap.indice);
      if (peticionarios.containsKey(auxIndice)){  //multicast del grupo
de peticionarios de ese índice
      int[] auxnodos =
((Peticionarios)(peticionarios.get(auxIndice))).nodos;
      for(int i = 0; i< auxnodos.length; i++){
        send(new
RDataCapsule(group,target,port,port,auxnodos[i],auxIndice.intValue(),null
));
      }
      }

    }else if (cap instanceof SCapsule){
      SCapsule sCap = (SCapsule) cap;
      if(sCap.suscripcion)
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        suscount++;
    }
  }


  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.out.println("Cápsulas multicast por segundo: ");
      double media = 0;
      for (int i = 0; i<estadisticas_caps_mcast_seg.length; i++){
      System.out.println("==>"+estadisticas_caps_mcast_seg[i]);
      media += estadisticas_caps_mcast_seg[i];
      }
      System.out.println("Cápsulas multicast por segundo, MEDIA => "+
media/estadisticas_caps_mcast_seg.length);
      System.out.println("==========================");

      System.out.println("Cápsulas retransmitidas por segundo(Recibidas):
");
      media = 0;
      for (int i = 0; i<estadisticas_caps_retransmitidas_seg.length;
i++){
      System.out.println("==>"+estadisticas_caps_retransmitidas_seg[i]);
      media += estadisticas_caps_retransmitidas_seg[i];
      }
      System.out.println("Cápsulas retransmitidas por segundo(Recibidas),
MEDIA => "+ media/estadisticas_caps_retransmitidas_seg.length);
      System.out.println("==========================");

      System.out.println("Cápsulas de petición de retransmisión por
segundo: ");
      media = 0;
      for (int i = 0;
i<estadisticas_caps_peticiones_retransmision_seg.length; i++){
      System.out.println("==>"+estadisticas_caps_peticiones_retransmision
_seg[i]);
      media += estadisticas_caps_peticiones_retransmision_seg[i];
      }
      System.out.println("Cápsulas de petición de retransmisión por
segundo, MEDIA =>
"+media/estadisticas_caps_peticiones_retransmision_seg.length);
      System.out.println("==========================");

      System.out.println("Peticiones de retransmisión por segundo: ");
      media = 0;
      for (int i = 0; i<estadisticas_peticiones_retransmision_seg.length;
i++){
      System.out.println("==>"+estadisticas_peticiones_retransmision_seg[
i]);
      media += estadisticas_peticiones_retransmision_seg[i];
      }
      System.out.println("Peticiones de retransmisión por segundo, MEDIA
=> "+ media/estadisticas_caps_retransmitidas_seg.length);
      System.out.println("==========================");

      System.out.println("Cápsulas de suscripción por segundo: ");
      media = 0;
      for (int i = 0; i<estadisticas_caps_suscripcion_seg.length; i++){
      System.out.println("==>"+estadisticas_caps_suscripcion_seg[i]);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      media += estadisticas_caps_suscripcion_seg[i];
      }
      System.out.println("Cápsulas de suscripción por segundo, MEDIA =>
"+ media/estadisticas_caps_retransmitidas_seg.length);
      System.out.println("==========================");
      System.exit(0);
      return true;
    } else return super.handleEvent(evt);
  }

  public void run(){
    while (true) {
      try {
      Thread.sleep(vg.tiempoPeticionNACKs+10);
      } catch (InterruptedException e) {
      continue;
      }
      Enumeration claves = pedidas.keys();
      Enumeration sellos = pedidas.elements();

      int[] indi = new int[0];
      long[] sells = new long[0];
      while (claves.hasMoreElements()){
        Integer auxkey = (Integer)claves.nextElement();
        if((System.currentTimeMillis()-
(long)((Tiempo)sellos.nextElement()).get())>vg.margenRT)
          pedidas.remove(auxkey); //quitamos la info obsoleta
        else {
        int len = (indi != null) ? indi.length : 0;
        int[] auxindices = new int[len+1];
        long[] auxsellos = new long[len+1];
        if (len > 0) {
          System.arraycopy(indi, 0, auxindices, 0, len);
          System.arraycopy(sells, 0, auxsellos, 0, len);
        }
        auxindices[len] = auxkey.intValue();
        auxsellos[len] = (long)((Tiempo)pedidas.get(auxkey)).get();
        indi = auxindices;
        sells = auxsellos;
        }//if
      }//while
      if (indi.length>0){ //si hay alguna petición útil
        thisNode().log("Enviando petición a instancias
superiores("+NodeAddress.toString(nodoAnterior)+")");
        send(new NACKCapsule(group,target,port, port, nodoAnterior,
indi));
      }
    }//while

  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-target")) {
        target = NodeAddress.fromString(k.arg(i));
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        k.strike(i);
      } else if (k.key(i).equals("-group")) {
      group = NodeAddress.fromString(k.arg(i));
      k.strike(i);
      }
    }

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {
    thisNode().register(new MCastRTProtocol());

    resize(300, 200);

    Panel resultados = new Panel ();
    resultados.setLayout(new GridLayout(13,1,1,1));

    FIN = new Button("FIN");
    resultados.add(FIN);
    resultados.add(new Label("No de Capsulas de Retransmision(caps/seg)",
Label.CENTER));
    petlabel = new Label("0", Label.CENTER);
    resultados.add(petlabel);
    resultados.add(new Label("No Peticiones de Retransmision(pets/seg)",
Label.CENTER));
    pettotallabel = new Label("0", Label.CENTER);
    resultados.add(pettotallabel);
    resultados.add(new Label("No Cápsulas de Multicast(caps/seg)",
Label.CENTER));
    mDatalabel = new Label("0", Label.CENTER);
    resultados.add(mDatalabel);
    resultados.add(new Label("No Cápsulas Retransmitidas(caps/seg)",
Label.CENTER));
    rDatalabel = new Label("0", Label.CENTER);
    resultados.add(rDatalabel);
    resultados.add(new Label("No Cápsulas de suscripción(caps/seg)",
Label.CENTER));
    suslabel = new Label("0", Label.CENTER);
    resultados.add(suslabel);
    estado = new Label ("What's Up?...",Label.CENTER);
    resultados.add(estado);
    nombre = new Label
("Nombre:"+NodeAddress.toString(thisNode().getAddress()),Label.CENTER);

    resultados.add(nombre);

    add("Center", resultados);

    enviadas = new Hashtable();
    pedidas = new Hashtable();
    peticionarios = new Hashtable();
    petcount = pettotalcount = mdatacount = rdatacount = suscount = 0;
    long mdatabeginTime = thisNode().time();

    estadisticas_caps_mcast_seg = new double[0];
    estadisticas_caps_retransmitidas_seg = new double[0];
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    estadisticas_caps_peticiones_retransmision_seg = new double[0];
    estadisticas_peticiones_retransmision_seg = new double[0];
    estadisticas_caps_suscripcion_seg = new double[0];

    new Thread(new LimpiaTablaEnviadas(this)).start();
    new Thread(new LimpiaListaPeticionarios(this)).start();
    new Thread(new Suscribe(this)).start();
    new Thread(this).start();
    pack();
    show();

  }

  public NodeApplication() {
    super();
  }
}
```

RDataCapsule.java

```
/*                              RDataCapsule.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                                versión en red (v2)
 *                        Autor: Javier Jesús Sánchez Medina
 *                                Comenzado: 13/12/00
 */


package MCastRTv2.apps;

import ants.*;
import ants.wrapper.*;
import MCastRTv2.apps.variablesglobales;
/** Cápsula datos retransmitidos
 *
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */
public class RDataCapsule
   extends MCastCapsule
{
  final private static byte[] MID = findMID("apps.RDataCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.RDataCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int indice;

  public int length() {
    int s = Xdr.INT;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(indice);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    indice = xdr.INT();

    return xdr;
  }

  public boolean evaluate(Node n) {
    n.log("estoy aqui y porto el indice "+ indice);
    if(n.getAddress() == getDst())
      return n.deliverToApp(this, dpt);
    else{
      W_JAIAIAJ m = (W_JAIAIAJ)n.getCache().get(group, target);
      if ((m==null)||(m.ai2.length==0)){
        return true; //todas las peticiones están obsoletas o ya no se
está suscrito
      }else{
        boolean encontrado = false;
      int posicion = 0;
        if (m.ai2.length>0)
          for (int i = 0; (i < m.ai2.length)&&(m.ai2[i]<=indice); i++)
            if (m.ai2[i] == indice) {
              encontrado = true; posicion = i; break;
            }//if
        if (encontrado){ // si encontramos registro para ese índice
            AuxTipo aux = (AuxTipo) eliminarEntrada(new AuxTipo(m.ai2,
m.aj1), posicion);
          m.ai2 = aux.indices;
          m.aj1 = aux.sellos;
            if (n.getAddress() == getSrc())
            return n.routeForNode(this, getDst());
          else return n.deliverToApp(this, dpt);
        }else{
          return true; //si no hay peticiones no obsoletas desechamos la
cápsula
        }
      } //if
    }
  }//evaluate()

  public RDataCapsule() { }

  public RDataCapsule(int ga, int ta, short sa, short da, int na, int
ind, ByteArray d) {
    super (ga,ta,sa,da,na,d);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    indice = ind;
  }

}
```

ReceiverApplication.java

```
/*                              ReceiverApplication.java
 *
 *             Ejemplo de protocolo multicast orientado a tiempo real
 *                                 versión en red (v2)
 *                         Autor: Javier Jesús Sánchez Medina
 *                                 Comenzado: 18/04/00
 */


package MCastRTv2.apps;

import java.awt.*;

import ants.*;
import utils.*;
import MCastRTv2.apps.variablesglobales;


/** Aplicación que corre en los nodos Receiver de la topología multicast.
 *
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */


public class ReceiverApplication
  extends Application
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  int target;   //target se refiere al nodo sender
  int group;
  boolean suscrito = false;  // esta suscrito?
  boolean primeraCapDatos = true;

  Button suscribirse;
  Button FIN;
  Label mDatalabel;
  Label rDatalabel;
  Label perdidaslabel;
  Label nombre;
  Label estado;
  int mdatacount;
  int rdatacount;
  int percount;
  long mdatabeginTime;

  double[] estadisticas_caps_mcast_seg;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  double[] estadisticas_caps_retransm_seg;
  double[] estadisticas_caps_perdidas;

  int ultimoindice;   //debe inicializarse al valor de ínidice de la
primera cap de datos recibida

  final public static String[] defaults = {};  //periodo de emisión cap
subcripcion

  synchronized public void receive(Capsule cap) {
    super.receive(cap);

    if (cap instanceof MDataCapsule) {
      if(!suscrito)
      send(new SCapsule(group,target,port,port,suscrito));
      else{
        MDataCapsule mCap = (MDataCapsule) cap;
        if (primeraCapDatos){  //en el caso de que se trate de la primera
cap de datos
        ultimoindice = mCap.indice;
        primeraCapDatos = false;
        mdatacount = rdatacount = percount = 0;
        mdatabeginTime = thisNode().time();
        }

//        estado.setText("mCap.indice = "+ mCap.indice);
        mdatacount++;

        if (mCap.indice>ultimoindice+1){  //si nos faltan cápsulas hay
que pedirlas al nodo anterior
          int[] indices = new int[0];
          for (int i = ultimoindice+1; i<mCap.indice; i++){
            percount++;
          int len = (indices != null) ? indices.length : 0;
          int[] auxindices = new int[len+1];
          if (len > 0) {
            System.arraycopy(indices, 0, auxindices, 0, len);
          }
          auxindices[len] = i;
          indices = auxindices;              //cuando la sig esté
obsoleta, las anteriores también
        }//for
        send(new
NACKCapsule(group,target,port,port,mCap.nodoPrevio,indices));
        }//if

        if(mdatacount>=vg.NCapsulasPeriodo){ //toca hacer estadísticas
        double periodo = thisNode().time() - mdatabeginTime;
        double media_caps_mcast_seg = 1000 * mdatacount / periodo;
        double media_caps_retransm_seg = 1000 * rdatacount / periodo;
        double perdidas = (percount - rdatacount)*100/(percount +
mdatacount);

        mDatalabel.setText(String.valueOf(media_caps_mcast_seg));
        rDatalabel.setText(String.valueOf(media_caps_retransm_seg));
        perdidaslabel.setText(String.valueOf(perdidas));

        int len1 = estadisticas_caps_mcast_seg.length;
        int len2 = estadisticas_caps_retransm_seg.length;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        int len3 = estadisticas_caps_perdidas.length;
        double[] aux1 = new double[len1+1];
        double[] aux2 = new double[len2+1];
        double[] aux3 = new double[len3+1];
        System.arraycopy(estadisticas_caps_mcast_seg,0,aux1,0,len1);
        System.arraycopy(estadisticas_caps_retransm_seg,0,aux2,0,len2);
        System.arraycopy(estadisticas_caps_perdidas,0,aux3,0,len3);
        aux1[len1] = media_caps_mcast_seg;
        aux2[len2] = media_caps_retransm_seg;
        aux3[len3] = perdidas;
        estadisticas_caps_mcast_seg = aux1;
        estadisticas_caps_retransm_seg = aux2;
        estadisticas_caps_perdidas = aux3;

          percount = 0;
        mdatacount = 0;
        rdatacount = 0;
        mdatabeginTime = thisNode().time();
        }
        ultimoindice = mCap.indice;
      }
    }else if (cap instanceof RDataCapsule) {
      RDataCapsule rdcap = (RDataCapsule) cap;
      rdatacount++;
      estado.setText("rdatacount = "+ rdatacount);
    }//if
  }

  public void run(){
    while (true) {
      try {
      Thread.sleep(vg.tiempoSuscripcion+10);
      } catch (InterruptedException e) {
      continue;
      }
      if (primeraCapDatos){
        send(new SCapsule(group,target,port,port,true));
      }
    }//while
  }//run

  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == suscribirse) {
      suscrito = !suscrito;
      //en el caso en que se suscriba ponemos inicialmente el flag
primeraCapDatos a '1'
      if (suscrito) {
        primeraCapDatos = true;
      }
      send(new SCapsule(group,target,port,port,suscrito));
        return true;
    }else if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.out.println("Cápsulas multicast por segundo: ");
      double media = 0;
      for (int i = 0; i<estadisticas_caps_mcast_seg.length; i++){
      System.out.println("==>"+estadisticas_caps_mcast_seg[i]);
      media += estadisticas_caps_mcast_seg[i];
      }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      System.out.println("Cápsulas multicast por segundo, MEDIA => "+
media/estadisticas_caps_mcast_seg.length);
      System.out.println("=========================");

      System.out.println("Cápsulas retransmitidas por segundo(Recibidas):
");
      media = 0;
      for (int i = 0; i<estadisticas_caps_retransm_seg.length; i++){
      System.out.println("==>"+estadisticas_caps_retransm_seg[i]);
      media += estadisticas_caps_retransm_seg[i];
      }
      System.out.println("Cápsulas retransmitidas por segundo(Recibidas),
MEDIA => "+ media/estadisticas_caps_retransm_seg.length);
      System.out.println("=========================");

      System.out.println("Pérdidas(%): ");
      media = 0;
      for (int i = 0; i<estadisticas_caps_perdidas.length; i++){
      System.out.println("==>"+estadisticas_caps_perdidas[i]);
      media += estadisticas_caps_perdidas[i];
      }
      System.out.println("Pérdidas(%), MEDIA => "+
media/estadisticas_caps_perdidas.length);
      System.out.println("=========================");
      System.exit(0);
      return true;
    } else return super.handleEvent(evt);
  }

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-target")) {
        target = NodeAddress.fromString(k.arg(i));
        k.strike(i);
      } else if (k.key(i).equals("-group")) {
      group = NodeAddress.fromString(k.arg(i));
      k.strike(i);
      }
    }

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {
    thisNode().register(new MCastRTProtocol());

    resize(300, 200);

    Panel resultados = new Panel ();
    resultados.setLayout(new GridLayout(11,1,1,1));
    FIN = new Button ("FIN");
    resultados.add(FIN);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    suscribirse = new Button("Suscribirse a " +
NodeAddress.toString(group));
    resultados.add(suscribirse);
    resultados.add(new Label("Cápsulas de datos multicast
(Caps/segundo)",Label.CENTER));
    mDatalabel = new Label("0", Label.CENTER);
    resultados.add(mDatalabel);
    resultados.add(new Label("Cápsulas de datos retransmitidos
(Caps/segundo)",Label.CENTER));
    rDatalabel = new Label("0", Label.CENTER);
    resultados.add(rDatalabel);
    resultados.add(new Label("Capsulas perdidas (%)",Label.CENTER));
    perdidaslabel = new Label("0", Label.CENTER);
    resultados.add(perdidaslabel);
    estado = new Label ("What's Up?...",Label.CENTER);
    resultados.add(estado);
    nombre = new Label
("Node:"+NodeAddress.toString(thisNode().getAddress()),Label.CENTER);
    resultados.add(nombre);

    add("Center", resultados);

    estadisticas_caps_mcast_seg = new double[0];
    estadisticas_caps_retransm_seg = new double[0];
    estadisticas_caps_perdidas = new double[0];
    mdatacount = 0;
    rdatacount = 0;
    percount = 0;
    primeraCapDatos = false;
    new Thread(this).start();
    pack();
    show();
  }

  public ReceiverApplication() {
    super();
  }
}
```

SCapsule.java

```
/*                              SCapsule.java
 *
 *           Ejemplo de protocolo multicast orientado a tiempo real
 *                                versión en red (v2)
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 13/12/00
 */

package MCastRTv2.apps;

import ants.*;
import ants.wrapper.*;
import MCastRTv2.apps.variablesglobales;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
/** Cápsula que suscribe a un Receiver
 *
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */
public class SCapsule
  extends Capsule
{
  final private static byte[] MID = findMID("apps.SCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.SCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int group;
  int target;
  short spt;
  short dpt;
  int reverse;
  boolean suscripcion; //true: suscripción, false: desuscripcion

  public int length() {
    return super.length() + Xdr.INT + Xdr.INT + Xdr.SHORT + Xdr.SHORT +
Xdr.INT + Xdr.BOOLEAN;
  }

  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(group);
    xdr.PUT(target);
    xdr.PUT(spt);
    xdr.PUT(dpt);
    xdr.PUT(reverse);
    xdr.PUT(suscripcion);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    group = xdr.INT();
    target = xdr.INT();
    spt = xdr.SHORT();
    dpt = xdr.SHORT();
    reverse = xdr.INT();
    suscripcion = xdr.BOOLEAN();

    return xdr;
  }

  public boolean evaluate(Node n) {
    if(n.getAddress()==getSrc()){ //si estoy en un nodo Receiver
      reverse = n.getAddress();
      return n.routeForNode(this,target);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    }else { //si no estoy en un nodo Receiver
      W_JAIAIAJ m = (W_JAIAIAJ)n.getCache().get(group, target);
      boolean found = false;
      int indice=0;
      if (m!=null) //buscamos una entrada para el nodo anterior
      for (int i = 0; m.ai1 != null && i < m.ai1.length; i++) {
        if (m.ai1[i] == reverse) { found = true; indice = i; break; }
      }
      if(suscripcion){ //si me estoy suscribiendo
      if (m == null) { //no habia registro de ninguna clase
        m = new W_JAIAIAJ();
        n.getCache().put(group, target, m,
vg.TIEMPO_DURACION_SUSCRIPCION);
      }
      if(!found){//no habia entrada para este nodo anterior
        int len = (m.ai1 != null) ? m.ai1.length : 0;
        int[] nn = new int[len+1];
        if (len > 0)
          System.arraycopy(m.ai1, 0, nn, 0, len);
        nn[len] = reverse;
        m.ai1 = nn;
      }//if !found
      //ahora que ya hay una entrada para multicast hasta este nodo
anterior
      //Señalizamos a la aplicacion
      n.deliverToApp(this,dpt);
      }else{//si me estoy desuscribiendo
      if (m==null){
        return n.deliverToApp(this,dpt); //le comunicamos a la aplicación
que ya no tiene nodos suscritos
      }
      if (found){ //si había entrada para este nodo anterior
        int len = m.ai1.length; //la quitamos
        int[] nn = new int[len-1];
        if (len > 1) {
          if (indice>0)
            System.arraycopy(m.ai1, 0, nn, 0, indice);
          if (indice<m.ai1.length-1)
            System.arraycopy(m.ai1, indice+1, nn, indice, m.ai1.length-
(indice+1));
        }
        m.ai1 = nn;

        if (m.ai1.length <1){ //si ya nadie está suscrito se envía al
nodo siguiente y a la aplicación para señalizarlo
          m = null;
          reverse = n.getAddress();
          if (n.getAddress()!=target)
            n.routeForNode(this,target);
          return n.deliverToApp(this,dpt);
        }
        return true;
      }//if found
      return true;
      }//if(suscripcion)
      return true;
    }//si no estoy en un nodo Receiver
  }//evaluate
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public SCapsule() { }

  public SCapsule(int ga, int ta, short sa, short da, boolean
suscripcion) {
    group = ga;
    target = ta;
    spt = sa;
    dpt = da;
    this.reverse = 0;
    this.suscripcion = suscripcion;
  }
}
```

SenderApplication.java

```
/*                              SenderApplication.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                                versión en red (v3)
 *                      Autor: Javier Jesús Sánchez Medina
 *                              Comenzado: 18/04/00
 */


package MCastRTv2.apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;
import MCastRTv2.apps.Tiempo;
import MCastRTv2.apps.LimpiaTablaEnviadas;
import MCastRTv2.apps.variablesglobales;

/** Aplicación que corre en el nodo Sender de la topología multicast.
 *
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */


public class SenderApplication
  extends MCastApplication
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  int target;
  int group;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  boolean enviando = false;

  Button FIN;
  Button send;
  Label petLabel;
  Label petTotalLabel;
  Label mDataLabel;
  Label rDataLabel;
  Label estado;
  Label nombre;
  int petcount;
  int pettotalcount;
  int rdatacount;
  int interv = 100;  //tiempo entre cápsula y cápsula de la transmisión
  int indice;    //indice del paquete multicast actual

  long mdatabeginTime;

  double[] estadisticas_caps_mcast_seg;
  double[] estadisticas_caps_retransmitidas_seg;
  double[] estadisticas_caps_peticiones_retransmision_seg;
  double[] estadisticas_peticiones_retransmision_seg;

  final public static String[] defaults = {};  //periodo de limpieza
tabla

  synchronized public void receive(Capsule cap) {
    super.receive(cap);

    if (cap instanceof NACKCapsule) {
      NACKCapsule nackCap = (NACKCapsule) cap;
      petcount++;
      pettotalcount += nackCap.npeticiones;
      estado.setText("nackCap.npeticiones = "+nackCap.npeticiones);
      for (int i = 0; i<nackCap.npeticiones; i++){
        Integer aux = new Integer(nackCap.indice[i]);
        if (enviadas.containsKey(aux)){
          send(new
RDataCapsule(group,target,port,port,nackCap.getSrc(),aux.intValue(),null)
);
        rdatacount++;
      }//if
      }//for
    }
  }


  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == send) {
      enviando = !enviando;
        return true;
    }else if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.out.println("Cápsulas multicast por segundo: ");
      double media = 0;
      for (int i = 0; i<estadisticas_caps_mcast_seg.length; i++){
      System.out.println("==>"+estadisticas_caps_mcast_seg[i]);
      media += estadisticas_caps_mcast_seg[i];
      }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      System.out.println("Cápsulas multicast por segundo, MEDIA => "+
media/estadisticas_caps_mcast_seg.length);
      System.out.println("==========================");

      System.out.println("Cápsulas retransmitidas por
segundo(Transmitidas): ");
      media = 0;
      for (int i = 0; i<estadisticas_caps_retransmitidas_seg.length;
i++){
      System.out.println("==>"+estadisticas_caps_retransmitidas_seg[i]);
      media += estadisticas_caps_retransmitidas_seg[i];
      }
      System.out.println("Cápsulas retransmitidas por
segundo(Transmitidas), MEDIA =>
"+media/estadisticas_caps_retransmitidas_seg.length);
      System.out.println("==========================");

      System.out.println("Cápsulas de petición de retransmisión por
segundo: ");
      media = 0;
      for (int i = 0;
i<estadisticas_caps_peticiones_retransmision_seg.length; i++){
      System.out.println("==>"+estadisticas_caps_peticiones_retransmision
_seg[i]);
      media += estadisticas_caps_peticiones_retransmision_seg[i];
      }
      System.out.println("Cápsulas de petición de retransmisión por
segundo, MEDIA =>
"+media/estadisticas_caps_peticiones_retransmision_seg.length);
      System.out.println("==========================");

      System.out.println("Peticiones de retransmisión por segundo: ");
      media = 0;
      for (int i = 0; i<estadisticas_peticiones_retransmision_seg.length;
i++){
      System.out.println("==>"+estadisticas_peticiones_retransmision_seg[
i]);
      media += estadisticas_peticiones_retransmision_seg[i];
      }
      System.out.println("Peticiones de retransmisión por segundo, MEDIA
=> "+ media/estadisticas_caps_retransmitidas_seg.length);
      System.out.println("==========================");

      System.exit(0);
      return true;
    }else return super.handleEvent(evt);
  }

  public void run(){
    while (true){
      if (enviando){
        int auxindice = indice++;
        //guardamos la cap enviada
//    if ((indice%5)!=0)    //provocamos perdidas voluntarias
          enviadas.put(new Integer((int)(auxindice)),new
Tiempo(thisNode().time()));
        MDataCapsule mdcap = new
MDataCapsule(group,target,port,port,auxindice,null);
//    if ((indice%2)==0)    //provocamos perdidas voluntarias
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
            send(mdcap);
      if ((indice % vg.NCapsulasPeriodo)==0){ //toca hacer estadísticas

        double periodo = thisNode().time() - mdatabeginTime;
        double media_caps_mcast_seg = 1000 * vg.NCapsulasPeriodo /
periodo;
        double media_caps_retransmitidas_seg = 1000 * rdatacount /
periodo;
        double media_caps_peticiones_retransmi_seg = 1000 * petcount /
periodo;
        double media_peticiones_retransmi_seg = 1000 * pettotalcount /
periodo;

        mDataLabel.setText(String.valueOf(media_caps_mcast_seg));

rDataLabel.setText(String.valueOf(media_caps_retransmitidas_seg));

petLabel.setText(String.valueOf(media_caps_peticiones_retransmi_seg));

petTotalLabel.setText(String.valueOf(media_peticiones_retransmi_seg));

        int len1 = estadisticas_caps_mcast_seg.length;
        int len2 = estadisticas_caps_retransmitidas_seg.length;
        int len3 = estadisticas_caps_peticiones_retransmision_seg.length;
        int len4 = estadisticas_peticiones_retransmision_seg.length;
        double[] aux1 = new double[len1+1];
        double[] aux2 = new double[len2+1];
        double[] aux3 = new double[len3+1];
        double[] aux4 = new double[len4+1];
        System.arraycopy(estadisticas_caps_mcast_seg,0,aux1,0,len1);

System.arraycopy(estadisticas_caps_retransmitidas_seg,0,aux2,0,len2);

System.arraycopy(estadisticas_caps_peticiones_retransmision_seg,0,aux3,0,
len3);

System.arraycopy(estadisticas_peticiones_retransmision_seg,0,aux4,0,len4)
;
        aux1[len1] = media_caps_mcast_seg;
        aux2[len2] = media_caps_retransmitidas_seg;
        aux3[len3] = media_caps_peticiones_retransmi_seg;
        aux4[len4] = media_peticiones_retransmi_seg;

        estadisticas_caps_mcast_seg = aux1;
        estadisticas_caps_retransmitidas_seg = aux2;
        estadisticas_caps_peticiones_retransmision_seg = aux3;
        estadisticas_peticiones_retransmision_seg = aux4;

        petcount = pettotalcount = rdatacount = 0;
        mdatabeginTime = thisNode().time();

      }
      }
      try {
      Thread.sleep(interv);
      } catch (InterruptedException e) {
      continue;
      }
    }//while
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-target")) {
        target = NodeAddress.fromString(k.arg(i));
        k.strike(i);
      } else if (k.key(i).equals("-group")) {
      group = NodeAddress.fromString(k.arg(i));
      k.strike(i);
      }
    }

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {
    thisNode().register(new MCastRTProtocol());

    enviadas = new Hashtable();

    resize(300, 200);

    Panel resultados = new Panel ();
    resultados.setLayout(new GridLayout(12,1,1,1));
    FIN = new Button ("FIN");
    resultados.add(FIN);
    send = new Button("Enviar a " + NodeAddress.toString(group));
    resultados.add(send);
    resultados.add(new Label("No de cápsulas de datos(caps/seg)",
Label.CENTER));
    mDataLabel = new Label("0", Label.CENTER);
    resultados.add(mDataLabel);
    resultados.add(new Label("No de cápsulas retransmitidas(caps/seg)",
Label.CENTER));
    rDataLabel = new Label("0", Label.CENTER);
    resultados.add(rDataLabel);
    resultados.add(new Label("No de cápsulas de petición de
retransmitidas(caps/seg)", Label.CENTER));
    petLabel = new Label("0", Label.CENTER);
    resultados.add(petLabel);
    resultados.add(new Label("No de peticiones de
retransmitidas(pets/seg)", Label.CENTER));
    petTotalLabel = new Label("0", Label.CENTER);
    resultados.add(petTotalLabel);
    estado = new Label("Vamo a Ver...",Label.CENTER);
    resultados.add(estado);
    nombre = new Label ("Nombre:" +
NodeAddress.toString(thisNode().getAddress()),Label.CENTER);
    resultados.add(nombre);

    add("Center", resultados);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    new Thread(new LimpiaTablaEnviadas(this)).start();
    new Thread(this).start();
    pack();
    show();

    petcount = pettotalcount = rdatacount = indice = 0;
    long mdatabeginTime = thisNode().time();

    estadisticas_caps_mcast_seg = new double[0];
    estadisticas_caps_retransmitidas_seg = new double[0];
    estadisticas_caps_peticiones_retransmision_seg = new double[0];
    estadisticas_peticiones_retransmision_seg = new double[0];
  }

  public SenderApplication() {
    super();
  }
}
```

Tiempo.java

```
/*                                  Tiempo.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                                  versión en red (v2)
 *                        Autor: Javier Jesús Sánchez Medina
 *                                Comenzado: 13/12/00
 */


package MCastRTv2.apps;
/**
 *   Tiempo
 */

public class Tiempo {
   private long t;
   long get (){
      return t;
   }
   Tiempo (long time){
      this.t = time;
   }//constructor
}//Tiempo
```

variablesglobales.java

```
/*                              variablesglobales.java
 *
 *            Ejemplo de protocolo multicast orientado a tiempo real
 *                                  versión en red (v2)
 *                        Autor: Javier Jesús Sánchez Medina
 *                                Comenzado: 13/12/00
 */
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
package MCastRTv2.apps;
/** Variables globales del prototipo
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class variablesglobales {
public long margenRT = 200;
public int tiempoLimpieza = 3000;
public int tiempoPeticionNACKs = 50;
public int tiempoSuscripcion = 1000;
public int NCapsulasPeriodo = 10;
public int TIEMPO_DURACION_SUSCRIPCION = 100000;
}
```

MCastECv1

DocCapsule.java

```
/**                                        DocCapsule.java
 *
 *                  Ejemplo de protocolo multicast de edición cooperativa
 *                                  Comenzado: 16/01/01
 */

package MCastECv1.apps;

import ants.*;
import MCastECv1.apps.variablesglobales;

/** Cápsula con fragmento de documento actualizado. Un índice indica la
posición del primer byte en el documento
 *  y se utiliza para confirmar la recepción.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class DocCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.DocCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.DocCapsule");
  protected byte[] pid() { return PID; }

  int indice; //indice del primer short del fragmento
  int nfrag;  //número de fragmentos total
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  short[] fragmento;
  variablesglobales vg = new variablesglobales();

  public int length() {
    int s = Xdr.INT + Xdr.INT + Xdr.SHORT * vg.frag_doc;
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(indice);
    xdr.PUT(nfrag);
    for (int i=0; i<vg.frag_doc; i++){
      xdr.PUT(fragmento[i]);
    }

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    indice = xdr.INT();
    nfrag = xdr.INT();
    fragmento = new short[vg.frag_doc];
    for (int i=0; i<vg.frag_doc; i++){
      fragmento[i]=xdr.SHORT();
    }

    return xdr;
  }

  public boolean evaluate(Node n) {  //enrutamiento unicast hacia el
Writer que solicitó el doc actualizado
    n.log("estoy aki");
    if (n.getAddress() != getDst())
      return n.routeForNode(this, getDst());
    else return n.deliverToApp(this, dpt);
  }//evaluate

  public DocCapsule() { }

  public DocCapsule(short sa, short da, int na, int ind, int nf, short[]
frag) {
    super (sa,da,na,new ByteArray(0));

    indice = ind;
    nfrag = nf;
    fragmento = frag;
  }
}
```

ChangesCapsule.java

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
/**                              ChangesCapsule.java
 *
 *              Ejemplo de protocolo multicast de edición cooperativa
 *                                  Comenzado: 16/01/01
 */

package MCastECv1.apps;

import ants.*;

/** Cápsula con cambios locales no actualizados.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class ChangesCapsule
extends DataCapsule
{
  final private static byte[] MID = findMID("apps.ChangesCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.ChangesCapsule");
  protected byte[] pid() { return PID; }

  byte ncambios;     //número de cambios que se envían (como máximo 256)
  int[] cambios;     //cambios que se envían
  int indiceprimero;

  public int length() {
    int s = Xdr.BYTE + (Xdr.INT) * ncambios + Xdr.INT;
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(ncambios);
    for (int i = 0; i<ncambios; i++)
      xdr.PUT(cambios[i]);
    xdr.PUT(indiceprimero);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    ncambios = xdr.BYTE();
    cambios = new int[ncambios];
    for (int i = 0; i<ncambios; i++)
      cambios[i] = xdr.INT();
    indiceprimero = xdr.INT();

    return xdr;
  }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public boolean evaluate(Node n) {  //enrutamiento unicast hacia el
Sender
    if ( n.getAddress() == getDst()){
      return n.deliverToApp(this, dpt);
    }else{
      n.log("ncambios="+ncambios);
      return n.routeForNode(this, getDst());
    }
  }

  public ChangesCapsule() { }

  public ChangesCapsule(short sa, short da, int na, int[] cam, int pi) {
    super (sa,da,na,new ByteArray(0));

    ncambios =(byte) cam.length;
    cambios = cam;
    indiceprimero = pi;
  }
}
```

DocACKCapsule.java

```
/**                                        DocACKCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa
 *                                    Comenzado: 16/01/01
 */

package MCastECv1.apps;

import ants.*;
import ants.wrapper.*;

/** Cápsula de confirmación de la recepción de un fragmento del documento
global
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class DocACKCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.DocACKCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.DocACKCapsule");
  protected byte[] pid() { return PID; }

  int indice; //indice del fragmento del documento recibido

  public int length() {
    int s = Xdr.INT;
    return super.length() + s;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  }

  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(indice);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    indice = xdr.INT();

    return xdr;
  }

  public boolean evaluate(Node n) {  //enrutamiento unicast hacia el
Sender
    n.log("estoy aki");
    if ( n.getAddress() == getDst())
      return n.deliverToApp(this, dpt);
    else
      return n.routeForNode(this, getDst());

  }

  public DocACKCapsule() { }

  public DocACKCapsule(short sa, short da, int na, int i) {
    super (sa,da,na,new ByteArray(0));
    indice = i;
  }
}
```

DocRequestCapsule.java

```
/**                          DocRequestCapsule.java
 *
 *              Ejemplo de protocolo multicast de edición cooperativa
 *                                  Comenzado: 16/01/01
 */

package MCastECv1.apps;

import ants.*;
import ants.wrapper.*;

/** Cápsula con petición de documento actualizado o petición se
desuscripción
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
public class DocRequestCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.DocRequestCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.DocRequestCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int group;
  int target;
  short spt;
  short dpt;
  int reverse;
  boolean suscripcion; // '1' cuando se está suscribiendo y '0' cuando se
está desuscribiendo

  public int length() {
    return super.length() + Xdr.INT + Xdr.INT + Xdr.SHORT + Xdr.SHORT +
Xdr.INT + Xdr.BOOLEAN;
  }

  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(group);
    xdr.PUT(target);
    xdr.PUT(spt);
    xdr.PUT(dpt);
    xdr.PUT(reverse);
    xdr.PUT(suscripcion);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    group = xdr.INT();
    target = xdr.INT();
    spt = xdr.SHORT();
    dpt = xdr.SHORT();
    reverse = xdr.INT();
    suscripcion = xdr.BOOLEAN();

    return xdr;
  }

  public boolean evaluate(Node n) {  //solicitando el doc actualizado o
la desuscripcion
    if(n.getAddress()==getSrc()){ //si estoy en un nodo Writer
      reverse = n.getAddress();
      return n.routeForNode(this,target);
    }else { //si no estoy en un nodo Writer
      W_JAIIAI m = (W_JAIIAI)n.getCache().get(group, target);
      boolean found = false;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      int indice=0;
      if (m!=null) //buscamos una entrada para el nodo anterior
      for (int i = 0; m.ai1 != null && i < m.ai1.length; i++) {
        if (m.ai1[i] == reverse) { found = true; indice = i; break; }
      }

      if(suscripcion){ //si me estoy suscribiendo
      if (m == null) { //no habia registro de ninguna clase
        m = new W_JAIIAI();
        n.getCache().put(group, target, m, vg.TIEMPO_MAXIMO_SUSCRIPCION);
        m.i1 = 0;  //esperando_confirmacion = 0
      }
      if(!found){//no habia entrada para este nodo anterior
        int len = (m.ai1 != null) ? m.ai1.length : 0;
        int[] nn = new int[len+1];
        int[] confirmaciones = new int[len+1];
        if (len > 0) {
          System.arraycopy(m.ai1, 0, nn, 0, len);
          System.arraycopy(m.ai2, 0, confirmaciones, 0, len);
        }
        nn[len] = reverse;
        confirmaciones[len] = 0;// se pone a uno cuando se recibe una
confirmacion de este nodo
        m.ai1 = nn;
        m.ai2 = confirmaciones;
      }//if !found
      //ahora que ya hay una entrada para multicast hasta este nodo
anterior
      //cursamos esta petición de documento hasta el Sender y señalizamos
a la aplicacion
      n.deliverToApp(this,dpt);
      reverse = n.getAddress();
      return n.routeForNode(this,target);

      }else{//si me estoy desuscribiendo
      if (m==null){
        return n.deliverToApp(this,dpt); //le comunicamos a la aplicación
que ya no tiene nodos suscritos
      }
      if (found){ //si había entrada para este nodo anterior
        int len = m.ai1.length; //la quitamos
        int[] nn = new int[len-1];
        int[] confirmaciones = new int[len-1];
        if (len > 1) {
          if (indice>0){
            System.arraycopy(m.ai1, 0, nn, 0, indice);
            System.arraycopy(m.ai2, 0, confirmaciones, 0, indice);
          }
          if (indice<m.ai1.length-1){
            System.arraycopy(m.ai1, indice+1, nn, indice, m.ai1.length-
(indice+1));
            System.arraycopy(m.ai2, indice+1, confirmaciones, indice,
m.ai2.length-(indice+1));
          }
        }
        m.ai1 = nn;
        m.ai2 = confirmaciones;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        if (m.ai1.length <1){ //si ya nadie está suscrito se envía al
 nodo siguiente y a la aplicación para señalizarlo
          m = null;
          reverse = n.getAddress();
          if (n.getAddress()!=target)
            n.routeForNode(this,target);
          return n.deliverToApp(this,dpt);
        }
        return true;
      }//if found
      return true;
      }//if(suscripcion)
    }//if (si estoy en un nodo Writer)
  }//evaluate

  public DocRequestCapsule() { }

  public DocRequestCapsule(int ga, int ta, short sa, short da, int na,
boolean s) {
    super (sa,da,na,new ByteArray(0));
    group = ga;
    target = ta;
    reverse = 0;
    suscripcion = s;
  }
}
```

GChangesCapsule.java

```
/**                                          GChangesCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa
 *                                    Comenzado: 16/01/01
 */

package MCastECv1.apps;

import ants.*;
import ants.wrapper.*;
import MCastECv1.apps.variablesglobales;

/** Cápsula con los cambios globales que se envía haciendo multicast a
todos los nodos suscritos
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */


public class GChangesCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.GChangesCapsule");
  protected byte[] mid() { return MID; }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  final private static byte[] PID = findPID("apps.GChangesCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int group;
  int target;

  byte nnodos;    //número de nodos cuyos cambios se envían (como máximo
256)
  int[] nodos;
  byte[] ncambios; //número de cambios por nodo
  int[][] cambios;    //cambios que se envían por cada nodo cambios[nodo
i][cambio j]
  int[] indiceprimero; //indice del primer cambio en cada nodo
  int sello; //sello para la identificación de esta cápsula de cambios
global.

  public int length() {
    int s = Xdr.INT + Xdr.INT + Xdr.BYTE + (Xdr.INT + Xdr.BYTE + Xdr.INT)
* nnodos + Xdr.INT;
    for (int j = 0; j<nnodos; j++)
       s +=(int)(Xdr.INT * ncambios[j]); //número de cambios que se
envían por nodo
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(group);
    xdr.PUT(target);
    xdr.PUT(nnodos);
    for (int j = 0; j<nnodos; j++){
       xdr.PUT(nodos[j]);
       xdr.PUT(ncambios[j]);
       for (int k = 0; k<ncambios[j]; k++)
          xdr.PUT(cambios[j][k]);
       xdr.PUT(indiceprimero[j]);
    }
    xdr.PUT(sello);
    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    group = xdr.INT();
    target = xdr.INT();
    nnodos = xdr.BYTE();
    nodos = new int[nnodos];
    cambios = new int[nnodos][];
    ncambios = new byte[nnodos];
    indiceprimero = new int[nnodos];
    for (short j = 0; j<nnodos; j++){
       nodos[j] = xdr.INT();
       ncambios[j] = xdr.BYTE();
       cambios[j] = new int[ncambios[j]];
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        for (short k = 0; k<ncambios[j]; k++){
           System.out.println("k="+k);
           cambios[j][k] = xdr.INT();
        }
        indiceprimero[j] = xdr.INT();
     }
     sello = xdr.INT();
     return xdr;
   }

  public boolean evaluate(Node n) {  //enrutamiento multicast desde el
Sender
     n.log("yo toy aki");
     if(n.getAddress() == getDst())//si estoy en un nodo Writer
       return n.deliverToApp(this,dpt);
     else {
       W_JAIIAI m = (W_JAIIAI)n.getCache().get(group, target);
       if (n.getAddress() != getSrc()){
           if (m != null) {
           if (m.i1 == 0){//si no está esperando confirmación
             //se confirma inmediatamente
             m.i1 = sello;
             //ponemos esperando_confirmacion = sello;
             //se distribuye por multicast
             for (int i = 0; i < m.ai1.length; i++)
               n.routeForNode(this, m.ai1[i]);
             return n.deliverToApp(this,dpt);//Se pasa a la aplicación
para confirmarla
           }else{//si está esperando confirmación
             //se descarta esta nueva GCCap o se confirma en el caso de
que sea una GCCap ya recibida
             if (m.i1 == sello) //si es una GCCap ya recibida se pasa a la
aplicacion para volver a confirmar
             n.deliverToApp(this,dpt);
             return true;
           }
           }
         //si m==null o no hay ningun suscriptor se pasa al nivel de
aplicacion para señalizarlo
         return n.deliverToApp(this,dpt);
       }else{ //si estamos en el nodo origen simplemente hacemos multicast
de la GCCap
         if (m != null)
           if (m.ai1 != null)
           for (int i = 0; i < m.ai1.length; i++)
             return n.routeForNode(this, m.ai1[i]);
         return true;
       }
     }
   }

  public GChangesCapsule() { }

  public GChangesCapsule(int ga,int ta,short sa,short da,int na,int[]
nod,int[][] cam,int[] indp, int s) {
     super (sa,da,na,new ByteArray(0));

     group = ga;
     target = ta;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    nnodos = (byte) nod.length;
    nodos = nod;
    ncambios = new byte[nnodos];
    for (byte i=0; i<nnodos; i++)
      ncambios[i] = (byte)cam[i].length;
    cambios = cam;
    indiceprimero = indp;
    sello = s;
  }
}
```

NodeApplication.java

```
/*                              NodeApplication.java
 *
 *              Ejemplo de protocolo multicast de edición cooperativa
 *                                  Comenzado: 16/01/01
 */

package MCastECv1.apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;
import MCastECv1.apps.*;

/** Aplicación que corre en los nodos intermedios de la topología
multicast.
 *
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class NodeApplication
  extends MotherApplication
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  boolean retransmitiendo_gccap;      // está retransmitiendo una cápsula
de cámbios globales
  boolean suscrito;

  Label estado;

  Button FIN;
  Label nombre;

  GChangesCapsule lCap;

  final public static String[] defaults = {};

  synchronized public void receive(Capsule cap) {
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    super.receive(cap);

    if (cap instanceof GChangesCapsule) {
      lCap = (GChangesCapsule) cap;
      if (suscrito){
      retransmitiendo_gccap = true;
      send(new
GCACKCapsule(group,target,port,port,lCap.getSrc(),lCap.sello));
      }else{
      send(new DocRequestCapsule(group,target,port,port,lCap.getSrc(),
suscrito));
      }
    }else if (cap instanceof GCACKCapsule) {
      thisNode().log("Recibida GCACKCapsule");
      retransmitiendo_gccap = false;
    }else if (cap instanceof DocRequestCapsule){
      DocRequestCapsule drcap = (DocRequestCapsule)cap;
      if (drcap.suscripcion)
      suscrito = true;
      else
      suscrito = false;
    }//if
  }

  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.exit(0);
        return true;
    }else return super.handleEvent(evt);
  }

  public void run(){
    while (true) {
      try {
      Thread.sleep(vg.tiempoRetransmision_gccap);
      } catch (InterruptedException e) {
      continue;
      }
      if (retransmitiendo_gccap){ //periódicamente retransmitimos la
cápsula de cámbios globales
        send(new
GCACKCapsule(group,target,port,port,lCap.getSrc(),lCap.sello));
       //hasta recibir confirmación de todos los suscriptores
      }
    }//while
  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-target")) {
        target = NodeAddress.fromString(k.arg(i));
        k.strike(i);
      } else if (k.key(i).equals("-group")) {
      group = NodeAddress.fromString(k.arg(i));
      k.strike(i);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      }
    }

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {
    thisNode().register(new MCastECProtocol());

    retransmitiendo_gccap = false;
    suscrito = false;

    resize(300, 200);

    Panel unico = new Panel();
    unico.setLayout(new GridLayout(3,1,1,1));
    FIN = new Button ("FIN");
    unico.add(FIN);
    estado = new Label(" nada de momento                ",Label.CENTER);
    unico.add(estado);
    nombre = new Label
("Node:"+NodeAddress.toString(thisNode().getAddress()),Label.CENTER);
    unico.add(nombre);

    add("Center", unico);

    new Thread(this).start();
    pack();
    show();
  }

  public NodeApplication() {
    super();
  }
}
```

SenderApplication.java

```
/*                              SenderApplication.java
 *
 *             Ejemplo de protocolo multicast de edición cooperativa
 *
 *                              Comenzado: 16/01/01
 */

package MCastECv1.apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;
import MCastECv1.apps.*;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
/**   thread que envía los fragmentos del documento global a los
peticionarios del mismo
*/
class EnviaFragmentosDoc
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  SenderApplication sapp;
  int indice_frag=0;

  public void run(){
    while (true) {
      try {
      Thread.sleep(vg.tiempoNoTransmisionFragsDoc);
      } catch (InterruptedException e) {
      continue;
      }
      if (sapp.enviando_doc_completo>0){
      Enumeration e = sapp.peticionarios_doc_completo.keys();
      FragDoc d = (FragDoc)sapp.fragmentos_doc.get(new
Integer(indice_frag));
      while (e.hasMoreElements()){
        Integer auxKey = (Integer)e.nextElement();
        Integer[] aux =
(Integer[])sapp.peticionarios_doc_completo.get(auxKey);
        if (aux[indice_frag].intValue()<1)
          sapp.send(new
DocCapsule(sapp.getPort(),sapp.getPort(),auxKey.intValue(),indice_frag,sa
pp.getNFrag(),d.get()));
        else {
          boolean entrada_obsoleta = true;
          for(int i = 0; i<aux.length; i++)
            if(aux[i].intValue()==0) {
            entrada_obsoleta = false;
            break;
            }
          if (entrada_obsoleta){ //si ese nodo ya ha confirmado la
recepción de todo el doc global
            sapp.peticionarios_doc_completo.remove(auxKey);
            sapp.enviando_doc_completo--; //decrementamos el semáforo
          }
        }
      }//while (e.hasMoreElements())
      indice_frag = (indice_frag + 1) % sapp.nfrag;

      try {
        Thread.sleep(vg.tiempoTransmisionFragsDoc);
      } catch (InterruptedException ie) {
        continue;
      }
      }//if (sapp.enviando_doc_completo>0)
    }//while (true)
  }//run
  EnviaFragmentosDoc(SenderApplication sapp) {
    this.sapp = sapp;
  }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
}//EnviaFragmentosDoc

/**
 *    Thread que se encarga de fabricar las cápsulas de cambios globales
 */
class FabricaGCCaps
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  SenderApplication sapp;

  public void run() {
    while (true) {
      try {
      Thread.sleep(vg.tiempoFabricacionGCCaps);
      } catch (InterruptedException e) {
      continue;
      }
      while (!sapp.cambios_recibidos.esta_vacio){ //voy por aki
        int[] nodos = new int[0];
      byte nnodos = 0;    //número de nodos cuyos cambios se envían (como
máximo 256)
      byte[] ncambios = new byte[0]; //número de cambios por nodo
      int[][] cambios = new int[0][0];  //cambios que se envían x cada
nodo cambios[nodo i][cambio j]
      int[] indiceprimero = new int[0]; //indice del primer cambio en
cada nodo
      int i = 0;
      while
((i<vg.n_nodos_gccap_max)&&(!sapp.cambios_recibidos.esta_vacio)){
        ChangesCapsule auxcap = (ChangesCapsule)
sapp.cambios_recibidos.sacarCap();
        int aux_uia; //el último índice aplicado de ese nodo
        int aux_indice;//el primer índice a aplicar
        if (sapp.ultimo_indice_aplicado.containsKey(new
Integer(auxcap.getSrc()))){
          aux_uia = ((Integer)sapp.ultimo_indice_aplicado.get(new
Integer(auxcap.getSrc()))).intValue();
          if (aux_uia<auxcap.indiceprimero)
            aux_indice = 0;
          else
            aux_indice = aux_uia - auxcap.indiceprimero +1;
        }else {
          aux_indice = 0;
        }
        int dif = (int)auxcap.ncambios-aux_indice; //cuantos cambios
nuevos tenemos
        sapp.estado.setText("dif="+dif);
        if(dif>0){ //si hay cambios nuevos que actualizar
          nnodos++;
          int len = nodos.length;
          int[] aux = new int[len+1];
          System.arraycopy(nodos,0,aux,0,len);
          aux[len] = auxcap.getSrc();
          nodos = aux;
          byte[] aux2 = new byte[len+1];
          System.arraycopy(ncambios,0,aux2,0,len);
          aux2[len] = auxcap.ncambios;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
            ncambios = aux2;
            int[][] aux3 = new int[len+1][];
            System.arraycopy(cambios,0,aux3,0,len);
            cambios = aux3;
            cambios[i] = new int[dif];
            System.arraycopy(auxcap.cambios,aux_indice,cambios[i],0,dif);
            int[] aux4 = new int [len + 1];
            System.arraycopy(indiceprimero,0,aux4,0,len);
            aux4[len] = auxcap.indiceprimero+aux_indice;
            indiceprimero = aux4;
            sapp.documento_global =
sapp.aplicar_cambios(sapp.documento_global,auxcap.cambios,aux_indice);
            sapp.ultimo_indice_aplicado.put(new
Integer(auxcap.getSrc()),new
Integer(auxcap.indiceprimero+(int)auxcap.ncambios-1));
            i++;
          }//if
        }//While --- GChangesCap nueva
        if (nnodos>0){
          int sello = (new Random()).nextInt();
          GChangesCapsule gccap = new
GChangesCapsule(sapp.getGroup(),sapp.getTarget(),sapp.getPort(),sapp.getP
ort(),

        0,nodos,cambios,indiceprimero,sello);

          if(!sapp.HayGCCapParaTx)
            sapp.actualizaGCCapLocal(gccap);
          else
            sapp.gccaps_a_tx.meterCap(gccap);
        }
        }
      }
    }

    FabricaGCCaps(SenderApplication sapp) {
      this.sapp = sapp;
    }
}//FabricaGCCaps

/** Aplicación que corre en el nodo Sender. Recibe los cambios de todos
los Writer suscritos y
 *  retransmite los cambios globales a todos.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class SenderApplication
  extends MotherApplication
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

//  int target;   //target se refiere al nodo sender
//  int group;
  int enviando_doc_completo; //número de Writers que están recibiendo el
documento completo.
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  int nfrag; //número fragmentos en que se divide el documentos global
actualmente

  Hashtable  peticionarios_doc_completo;
  Hashtable  fragmentos_doc;
  Hashtable ultimo_indice_aplicado;
  PilaCapsFIFO  gccaps_a_tx;
  PilaCapsFIFO  cambios_recibidos;
  GChangesCapsule GCCapLocal;
  boolean  HayGCCapParaTx;
  boolean HayAlguienSuscrito;

  short[] documento_global;

  Button FIN;
  public Label estado;

  public int getNFrag(){
    return nfrag;
  }

  private short[] relleno_ceros (short[] doc, int n){
    short[] aux = new short[documento_global.length + n];
    for (int i=0; i<documento_global.length+n; i++)
      aux[i] = 0;
    System.arraycopy(documento_global,0,aux,0,documento_global.length);
    return aux;
  }//relleno_ceros

  public void actualizaGCCapLocal(GChangesCapsule gccap){
    GCCapLocal = new GChangesCapsule();
    GCCapLocal = gccap;
    HayGCCapParaTx = true;
  }

  final public static String[] defaults = {};  //periodo de emisión cap
subcripcion

  synchronized public void receive(Capsule cap) {
    super.receive(cap);

    if (cap instanceof ChangesCapsule) {
      thisNode().log("Recibida ChangesCapsule");
      ChangesCapsule CCap = (ChangesCapsule) cap;
      if((!ultimo_indice_aplicado.containsKey(new
Integer(CCap.getSrc())))||
      (((Integer)(ultimo_indice_aplicado.get(new
Integer(CCap.getSrc())))).intValue()<CCap.indiceprimero+CCap.ncambios-1))
        cambios_recibidos.meterCap(CCap);

    }else if (cap instanceof GCACKCapsule) {
      thisNode().log("Recibida GChangesACKCapsule");
      if (!gccaps_a_tx.esta_vacio) {
      GCCapLocal = new GChangesCapsule();
      GCCapLocal = (GChangesCapsule) gccaps_a_tx.sacarCap(); //saca GCCap
en el caso de que haya
      HayGCCapParaTx = true;
      }else{
      HayGCCapParaTx = false;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      }
    }else if(cap instanceof DocRequestCapsule){
      DocRequestCapsule drcap = (DocRequestCapsule) cap;
      if(drcap.suscripcion){ //si se está suscribiendo
      HayAlguienSuscrito = true;
        if (!peticionarios_doc_completo.containsKey(new
Integer(drcap.getSrc()))){
          if (enviando_doc_completo<1){ //si es la primera petición
fraccionamos el documento
          short[] aux_documento_global;
          aux_documento_global = documento_global;
          if(aux_documento_global.length>1){
              int len = aux_documento_global.length;
              int lz = ((len % vg.frag_doc)>0)? vg.frag_doc - (len %
vg.frag_doc):0;
              if (lz>0)
              aux_documento_global =
relleno_ceros(aux_documento_global,lz);
            nfrag = (int)aux_documento_global.length/vg.frag_doc;
            for (int i=0; i<nfrag; i++){
              short[] aux = new short[vg.frag_doc];
              System.arraycopy(aux_documento_global,0,aux,0,vg.frag_doc);
              fragmentos_doc.put(new Integer(i),new FragDoc(aux));
            }
          }else{//si el doc global está vacío ponemos un fragmento con
ceros
            nfrag = 1;
            short[] aux =
relleno_ceros(aux_documento_global,vg.frag_doc);
            fragmentos_doc.put(new Integer(0),new FragDoc(aux));
          }//(documento_global.length>1)
          }
          enviando_doc_completo++;
          Integer[] aux = new Integer[nfrag];
          for (int i = 0; i<nfrag; i++)
          aux[i] = new Integer(0); //no se ha recibido confirmación de
ningún fragmento
          //ponemos una entrada en la tabla por el nuevo nodo
          peticionarios_doc_completo.put(new
Integer(drcap.getSrc()),aux);
        }//if !peticionarios_doc_completo.containsKey
      }else{
      HayAlguienSuscrito = false; //recibe está cápsula cuando ya no hay
nadie suscrito
      }

    }else if(cap instanceof DocACKCapsule){
      thisNode().log("Recibida DocACKCapsule");
      DocACKCapsule dackcap = (DocACKCapsule) cap;
      if (fragmentos_doc.containsKey(new Integer(dackcap.indice))){
//existe entrada para el indice?
      Integer[] aux = (Integer[])peticionarios_doc_completo.get(new
Integer(dackcap.getSrc()));
      aux[dackcap.indice] = new Integer(1);//ponemos el flag
correspondiente a 1
      peticionarios_doc_completo.put(new Integer(dackcap.getSrc()),aux);
      }
    }//if
  }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.exit(0);
        return true;
    } else return super.handleEvent(evt);
  }

  public void run(){
    while (true) {
      try {
      Thread.sleep(vg.tiempoRetransmision_gccap);
      } catch (InterruptedException e) {
      continue;
      }
      if
((enviando_doc_completo<1)&&(HayGCCapParaTx)&&(HayAlguienSuscrito)){ //si
no se está enviando el documento completo
          send(GCCapLocal);
      }//if
    }//while
  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-target")) {
        target = NodeAddress.fromString(k.arg(i));
        k.strike(i);
      } else if (k.key(i).equals("-group")) {
      group = NodeAddress.fromString(k.arg(i));
      k.strike(i);
      }
    }

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {
    thisNode().register(new MCastECProtocol());

    enviando_doc_completo = 0;
    peticionarios_doc_completo = new Hashtable();
    gccaps_a_tx = new PilaCapsFIFO();
    cambios_recibidos = new PilaCapsFIFO();
    fragmentos_doc = new Hashtable();
    documento_global = new short[0];
    ultimo_indice_aplicado = new Hashtable();
    HayGCCapParaTx = false;
    HayAlguienSuscrito = false;

    resize(300, 200);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    Panel unico = new Panel ();
    unico.setLayout(new GridLayout(2,1,1,1));
    FIN = new Button ("FIN");
    unico.add(FIN);
    estado = new Label("De momento naita                    ",
Label.CENTER);
    unico.add(estado);


    add("Center", unico);

    new Thread(this).start();
    new Thread(new FabricaGCCaps(this)).start();
    new Thread(new EnviaFragmentosDoc(this)).start();
    pack();
    show();
  }

  public SenderApplication() {
    super();
  }
}
```

WriterApplication.java

```
/*                              WriterApplication.java
 *
 *             Ejemplo de protocolo multicast de edición cooperativa
 *                                  Comenzado: 16/01/01
 */

package MCastECv1.apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;
import MCastECv1.apps.*;

/**
 *    Thread que se encarga de enviar al Sender los cambios locales
 *
 */

class EnviaCambios
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  WriterApplication wapp;

  public void run() {
    while (true) {
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      try {
      Thread.sleep(vg.tiempoEnvioCambios);
      } catch (InterruptedException e) {
      continue;
      }
      if((!wapp.recibiendo_doc)&&(wapp.get_indice_cambio_antiguo
()<wapp.get_indice_cambio())&&(wapp.suscrito)){
      //si no estoy recibiendo el documento total,si hay cambios sin
confirmar y estoy suscrito
        for (int
i=wapp.get_indice_cambio_antiguo();i<wapp.get_indice_cambio();i = i +
vg.n_cambios_max){
        int L;
        if (wapp.get_indice_cambio()-i >= vg.n_cambios_max)
          L = vg.n_cambios_max;
        else L = wapp.get_indice_cambio()-i;
        int[] cam = new int[0];
        for (int j = 0; j<L; j++){
          int [] aux = new int[cam.length +1];
          System.arraycopy(cam, 0, aux, 0, cam.length);
          aux[cam.length] = (int)((Integer)wapp.tabla_cambios.get(new
Integer(i+j))).intValue();
          cam = aux;
        }
        wapp.send(new
ChangesCapsule(wapp.getPort(),wapp.getPort(),wapp.getTarget(), cam, i));
        }//for
      }//if
    }//while
  }//run
  EnviaCambios(WriterApplication wapp) {
    this.wapp = wapp;
  }//constructor
}//EnviaCambios

/** Aplicación que corre en los nodos Writer y que modifica su copia
local del documento
 *  transmitiendo los cambios al Sender
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */
public class WriterApplication
  extends MotherApplication
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  public boolean suscrito = false;      // está suscrito?

  short[] documento_local;
  Hashtable tabla_cambios;
  Hashtable ultimos_cambios_global;
  boolean recibiendo_doc = false;
  boolean esperando_primer_frag = false;
  Hashtable fragmentos_recibidos;
  int nfrag_recibidos = 0;
  int ultimo_sello_recibido = 0;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
Button suprimir;
Button insertar;
Button sustituir;
TextField indice,caracter_nuevo;
TextArea doc_local;
Label estado;

Button suscribirse;
Button FIN;
Label nombre;

int indice_cambio = 0; //indice del próximo cambio a registrar
int indice_cambio_antiguo = 0;  //indice del próximo cambio a confirmar

public int get_indice_cambio (){
  return indice_cambio;
}

public int get_indice_cambio_antiguo (){
  return indice_cambio_antiguo;
}


public void refresca_pantalla(){
  String cadena = new String("");
  for (int i = 0; i<documento_local.length; i++){
    cadena = cadena + (char) documento_local[i];
  }
  doc_local.setText(cadena);
}

private short[] elimina_zero_padding(short[] doc){
  short[] aux1 = new short[doc.length];
  aux1 = doc;
  while ((aux1.length>1)&&(aux1[aux1.length-1]==0)){
    int len = aux1.length;
    short[] aux2 = new short[len-1];
    System.arraycopy(aux1,0,aux2,0,len-1);
    aux1 = aux2;
  }
  return aux1;
}//elimina_zero_padding

final public static String[] defaults = {};

synchronized public void receive(Capsule cap) {
  super.receive(cap);
  if (cap instanceof DocCapsule) {
    thisNode().log("Recibida DocCapsule");
    DocCapsule dcap = (DocCapsule) cap;
    esperando_primer_frag = false;
    if (!fragmentos_recibidos.containsKey(new Integer(dcap.indice))){
    fragmentos_recibidos.put(new Integer(dcap.indice),new Integer(1));
    nfrag_recibidos++;
    short[] fragmento_nuevo = new short[vg.frag_doc];
    fragmento_nuevo = dcap.fragmento;
      int L = vg.frag_doc*(dcap.indice+1);
    if (documento_local.length <= vg.frag_doc*dcap.indice){
        int len = documento_local.length;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
          short[] aux2 = new short[L];
          System.arraycopy(documento_local, 0, aux2, 0, len);
          documento_local = aux2;
        }

System.arraycopy(fragmento_nuevo,0,documento_local,vg.frag_doc*dcap.indic
e,vg.frag_doc);
      if(nfrag_recibidos >= dcap.nfrag){
        documento_local = elimina_zero_padding(documento_local);
        recibiendo_doc = false; //finaliza la recepción de los fragmentos
del doc;
      }
      }
      send(new DocACKCapsule(port,port,target, dcap.indice)); //siempre
enviamos ACK del fragmento
      refresca_pantalla();
    }else if (cap instanceof GChangesCapsule) {
      GChangesCapsule gccap = (GChangesCapsule) cap;
      if (suscrito){
      send(new
GCACKCapsule(group,target,port,port,gccap.getSrc(),gccap.sello));
      if(!(gccap.sello == ultimo_sello_recibido)){
        ultimo_sello_recibido = gccap.sello;
          for (int i = 0; i<gccap.nnodos; i++){
          if (gccap.nodos[i] == thisNode().getAddress()){//son los
cambios propios confirmados
            indice_cambio_antiguo =
gccap.indiceprimero[i]+gccap.ncambios[i];//actualizamos la info del
último cambio confirmado
          }else{ //son cambios introducidos por otro nodo
            if (ultimos_cambios_global.containsKey(new
Integer(gccap.nodos[i]))){
              int auxindice =
(int)(((Integer)ultimos_cambios_global.get(new
Integer(gccap.nodos[i]))).intValue());
              int ind_primero;
              if (auxindice<gccap.indiceprimero[i])
              ind_primero = 0;
              else
              ind_primero = auxindice - gccap.indiceprimero[i] + 1;
              documento_local = aplicar_cambios(documento_local,
gccap.cambios[i], ind_primero);
              ultimos_cambios_global.remove(new Integer(gccap.nodos[i]));
            }else{
              documento_local = aplicar_cambios(documento_local,
gccap.cambios[i], 0);
            }
            ultimos_cambios_global.put(new Integer(gccap.nodos[i]),new
Integer(gccap.indiceprimero[i]+gccap.cambios[i].length-1));
          }
        }
        refresca_pantalla();
      }//if
      }else{ //si no estoy suscrito enviamos una cápsula de desuscricion
      send(new DocRequestCapsule(group,target,port,port,gccap.getSrc(),
suscrito));
      }
    }//if
  }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public boolean handleEvent(Event evt) {
    if ((!recibiendo_doc)&&(suscrito)&&(evt.id == Event.ACTION_EVENT &&
        ((evt.target == suprimir)||(evt.target == insertar)||(evt.target
== sustituir)))) {
      int auxindice = Integer.parseInt(indice.getText());
        int len = documento_local.length;
      if ((len > auxindice)||(len == 0)){
        int op=1;
        if(len == 0){//si el documento es vacio, aceptamos la inserción
de un caracter en la pos 0
          auxindice = 0;
        }else if(evt.target == suprimir)
          op = 0;
        else if(evt.target == sustituir)
          op = 2;
        char auxchar = (char)(new
String(caracter_nuevo.getText())).charAt(0);;
        short auxshort = (short)(auxchar);
        int[] cambio = new int[1];
        cambio[0] = ((int)(((int)(auxshort<<2)+op)<<14)) + auxindice;
        documento_local = aplicar_cambios(documento_local, cambio, 0);
        tabla_cambios.put(new Integer(indice_cambio),new
Integer(cambio[0]));
          indice_cambio++;    //lo actualizamos al valor del próximo
cambio
        refresca_pantalla();
        return true;
      }else{
        estado.setText("error en la introducción del índice");
        refresca_pantalla();
        return true;
      }
    }else if (evt.id == Event.ACTION_EVENT && evt.target == suscribirse)
{
      suscrito = !suscrito;
      send(new DocRequestCapsule(group,target,port,port,target,
suscrito));
      //si suscrito = true:suscripción , si suscrito=false:desuscripción
      if (suscrito) { //cuando nos suscribimos, una copia del doc se pide
        esperando_primer_frag = true;
        recibiendo_doc = true;
        fragmentos_recibidos = new Hashtable();
        nfrag_recibidos = 0;
        documento_local = new short[0];
      }
        return true;
    }else if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.exit(0);
        return true;
    } else return super.handleEvent(evt);
  }

  public void run(){ //el thread se encarga de suscribirnos cuando sea
preciso
    while (true) {
      try {
      Thread.sleep(vg.tiempoPeticionDoc);
      } catch (InterruptedException e) {
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      continue;
      }
      if ((esperando_primer_frag)&&(suscrito)) // si no hemos recibido el
primer fragmento
      send(new DocRequestCapsule(group,target,port,port,target,
suscrito));
    }
  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-target")) {
        target = NodeAddress.fromString(k.arg(i));
        k.strike(i);
      } else if (k.key(i).equals("-group")) {
      group = NodeAddress.fromString(k.arg(i));
      k.strike(i);
      }
    }

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {
    thisNode().register(new MCastECProtocol());

    tabla_cambios = new Hashtable();
    ultimos_cambios_global = new Hashtable();
    recibiendo_doc = false;
    documento_local = new short[0];

    resize(300, 200);

    Panel cambios = new Panel();
    cambios.setLayout(new GridLayout(4,1,1,1));
    Panel basicas = new Panel();
    basicas.setLayout(new GridLayout(1,2,1,1));
    suscribirse = new Button("Suscribirse a " +
NodeAddress.toString(group));
    FIN = new Button ("FIN");
    basicas.add(suscribirse);
    basicas.add(FIN);
    cambios.add(basicas);
    Panel operaciones = new Panel();
    operaciones.setLayout(new GridLayout(1,3,1,1));
    suprimir = new Button("Suprimir");
    insertar = new Button("Insertar");
    sustituir = new Button("Sustituir");
    operaciones.add(suprimir);
    operaciones.add(insertar);
    operaciones.add(sustituir);
    cambios.add(operaciones);
    Panel operandos = new Panel();
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    operandos.setLayout(new GridLayout(2,2,1,1));
    operandos.add(new Label("Indice",Label.CENTER));
    operandos.add(new Label("Nuevo Caracter",Label.CENTER));
    indice = new TextField("0000");
    operandos.add(indice);
    caracter_nuevo = new TextField("0000");
    operandos.add(caracter_nuevo);
    cambios.add(operandos);
    estado = new Label("De momento naita"+target,Label.CENTER);
    cambios.add(estado);

    Panel resultados = new Panel ();
    resultados.setLayout(new GridLayout(2,1,1,1));
    doc_local = new TextArea(" ",5
,20,TextArea.SCROLLBARS_VERTICAL_ONLY);
    doc_local.setEditable(false);
    resultados.add(doc_local);
    nombre = new Label
("Node:"+NodeAddress.toString(thisNode().getAddress()),Label.CENTER);
    resultados.add(nombre);

    add("West", cambios);
    add("Center", resultados);

    new Thread(this).start();
    new Thread(new EnviaCambios(this)).start();
    pack();
    show();
  }

  public WriterApplication() {
    super();
  }
}
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

variablesglobales.java

```
/**                              variablesglobales.java
 *
 *             Ejemplo de protocolo multicast de edición cooperativa
 *                                  Comenzado: 16/01/01
 */


package MCastECv1.apps;

/** Variables globales del prototipo
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */


public class variablesglobales {
```

```
public int tiempoSuscripcion = 2000; //como se realiza una sola
suscripción y desuscripción,ha de ser grande para ahorar CPU
public int ncapsDesuscripcion =5; //número de cápsulas de desuscripción
que se envían
public int frag_doc = 50; //tamaño en shorts de cada fragmento de
documento enviado
public int n_cambios_max = 20; //número máximo de van en cada cápsula
public int n_nodos_gccap_max = 5; //número máximo de nodos caben en cada
cápsula gccap
public int tiempoEnvioCambios = 2000; //tiempo de envío de los cambios
locales al sender
public int tiempoRetransmision_gccap = 2000; //tiempo de retransmisión de
los cambios globales
public int tiempoConfirmacionGCCaps = 2000; //tiempo de retransmisión de
confirmación de los cambios globales
public int tiempoFabricacionGCCaps = 1000; //tiempo de envío de los
cambios locales al sender
public int tiempoTransmisionFragsDoc = 50; //tiempo de transmisión de los
fragmentos del doc global
public int tiempoNoTransmisionFragsDoc = 2000; //tiempo de duerme cuando
no transmite fragmentos del doc global
public int tiempoPeticionDoc = 1000; //tiempo pasado el cual reenvia
petición de fragmentos de doc (si no se ha recibido ninguno)
public int longitud_MAX_DOC = 16384; //14 bits en el índice
public int TIEMPO_MAXIMO_SUSCRIPCION = 100000; //tiempo máximo que un
nodo writer estará suscrito para mantener su info en cache del nodo
}
```

PilaCapsFIFO.java

```
/**                              PilaCapsFIFO.java
 *
 *            Ejemplo de protocolo multicast de edición cooperativa
 *                                  Comenzado: 16/01/01
 */

package MCastECv1.apps;

import ants.*;

/** Pila FIFO para el almacenamiento ordenado de cápsulas de cualquier
tipo
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class PilaCapsFIFO{
   private DataCapsule[] vector;
   public boolean esta_vacio;
   DataCapsule sacarCap(){
      if (!esta_vacio){
      int len = vector.length-1;
      if (len < 1){
        DataCapsule aux = new DataCapsule();
        aux = vector[0];
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        esta_vacio = true;
        vector = new DataCapsule[0];
        return aux;
      }else{
        DataCapsule[] aux = new DataCapsule[len];
          DataCapsule aux2 = new DataCapsule();
        System.arraycopy(vector, 1, aux, 0, len);
        aux2 = vector[0];
        vector = aux;
        esta_vacio = false;
        return aux2;
      }
      }else return null;
   }
   void meterCap(DataCapsule cap){
      if (!esta_vacio){
      int len = vector.length;
        DataCapsule[] aux = new DataCapsule[len + 1];
        System.arraycopy(vector, 0, aux, 0, vector.length);
        aux[vector.length] = cap;
      vector = aux;
      }else{
      esta_vacio = false;
        vector = new DataCapsule[1];
      vector[0] = cap;
      }
   }

   PilaCapsFIFO (){
      this.esta_vacio = true;
      this.vector = new DataCapsule[0];

   }//constructor
}//PilaCapsFIFO
```

MotherApplication.java

```
/**                          MotherApplication.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa
 *                                Comenzado: 16/01/01
 */

package MCastECv1.apps;
import ants.*;

/** Aplicación que Madre de las Writer y Sender
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class MotherApplication
  extends Application
{
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  int target,group;
  short port;


  public short getPort(){
    return port;
  }

  public int getTarget(){
    return target;
  }

  public int getGroup(){
    return group;
  }

  public short[] aplicar_cambios(short[] doc_1, int[] cambs, int
ind_primero){
    short[] aux_doc = doc_1;
    for (int i = ind_primero; i < cambs.length; i++){
      short[] aux;
      int auxcamb = cambs[i];
      int indice = (int)(cambs[i]&16383);
      byte op = (byte)((cambs[i]>>14)&3);
      short c = (short)((cambs[i]>>16)&65535);
      int len = aux_doc.length;
      switch (op){
        case 0://suprimir
        aux = new short[len-1];
        System.arraycopy(aux_doc,0,aux,0,indice);
          if (indice<len-1)
             System.arraycopy(aux_doc,indice+1,aux,indice,len-indice-1);
        break;
        case 1://insertar
        aux = new short[len+1];
        System.arraycopy(aux_doc,0,aux,0,indice);
        aux[indice] = c;
          if (indice<len)
             System.arraycopy(aux_doc,indice,aux,indice+1,len-indice);
        break;
        case 2://sustituir
        aux = new short[len];
        System.arraycopy(aux_doc,0,aux,0,indice);
        aux[indice] = c;
          if (indice<len-1)
             System.arraycopy(aux_doc,indice+1,aux,indice+1,len-indice-
1);
        break;
      default:
        aux = aux_doc;
      }//switch
      aux_doc = aux;
    }
    return aux_doc;
  }//aplicar_cambios
}
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

FragDoc.java

```
/**                                    FragDoc.java
 *
 *                Ejemplo de protocolo multicast de edición cooperativa
 *                                    Comenzado: 16/01/01
 */

package MCastECv1.apps;
import MCastECv1.apps.*;

/** Clase que porta un fragmento del documento en shorts
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class FragDoc {
   private short[] f;
   variablesglobales vg = new variablesglobales();
   short[] get (){
      return f;
   }
   FragDoc (short[] frag){
      f = new short[vg.frag_doc];
      f = frag;
   }//constructor
}//FragDoc
```

GCACKCapsule.java

```
/**                                GCACKCapsule.java
 *
 *            Ejemplo de protocolo multicast de edición cooperativa
 *                                    Comenzado: 16/01/01
 */

package MCastECv1.apps;

import ants.*;
import ants.wrapper.*;
import MCastECv1.apps.variablesglobales;

/** Confirmación de la recepción de cápsula de cambios globales. Se
realiza fussing de estas cápsulas
 *  en los nodos intermedios para ahorrar ancho de banda
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class GCACKCapsule
  extends DataCapsule
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
{
  final private static byte[] MID = findMID("apps.GCACKCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.GCACKCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int sello; //entero enviado por el Sender como sello de fragmento de
doc

  int group;
  int target;
  int reverse;

  public int length() {
    int s = Xdr.INT + Xdr.INT + Xdr.INT + Xdr.INT;
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(group);
    xdr.PUT(target);
    xdr.PUT(reverse);

    xdr.PUT(sello);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    group = xdr.INT();
    target = xdr.INT();
    reverse = xdr.INT();

    sello = xdr.INT();

    return xdr;
  }

  public boolean evaluate(Node n) {
    if (n.getAddress() == getSrc()){ //si estoy en el nodo origen
      reverse = n.getAddress();
      n.log("estoy aki, en el origen y target="+target);
      return n.routeForNode(this, target);
    }else{ //si no estoy en el nodo origen
      n.log("estoy aki, en otro sitio");
      W_JAIIAI m = (W_JAIIAI)n.getCache().get(group, target);
      if (m == null)
        return true; //no hay nodos suscritos
      else {
      int i;
      for(i=0;i<m.ai1.length;i++){ //buscamos la entrada de este nodo
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
System.out.println("m.ai1["+i+"]="+NodeAddress.toString(m.ai1[i]));
        if (m.ai1[i]==reverse) break;
      }
      m.ai2[i] = 1; //ponemos confirmación del nodo origen
        boolean recibido_de_todos = true;
        for (int j = 0; j<m.ai1.length; j++){
        if(m.ai2[j]==0){ //si hay alguno del que no se haya recibido
confirmación
          recibido_de_todos = false;
          break;
        }//if
        }//for
        if (recibido_de_todos) {
        m.i1 = 0; //esperando_confirmacion = 0
          for (int j = 0; j<m.ai1.length; j++)
          m.ai2[j] = 0; //ponemos a cero todas las confirmaciones
        return n.deliverToApp(this,dpt); //y le pasamos a la aplicación
la confirmación definitiva
      }
      return true;
      }
    }//
  }

  public GCACKCapsule() { }

  public GCACKCapsule(int ga, int ta, short sa, short da, int na, int s)
{
    super (sa,da,na,new ByteArray(0));
    group = ga;
    target = ta;
    reverse = 0;
    sello = s;
  }
}
```

MCastECProtocol.java

```
/**                              MCastECProtocol.java
 *
 *              Ejemplo de protocolo multicast de edición cooperativa
 *                      @author Javier Jesús Sánchez Medina
 *                              Comenzado: 16/01/01
 */

package MCastECv1.apps;

import ants.*;

/**
 *    protocolo multicast de procesamiento de texto distribuido
 *
 *
 *
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
 */

public class MCastECProtocol extends Protocol {

  public MCastECProtocol() throws Exception {
    startProtocolDefn();

    startGroupDefn();
    addCapsule("apps.ChangesCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.GChangesCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.GCACKCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.DocRequestCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.DocCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.DocACKCapsule");
    endGroupDefn();

    endProtocolDefn();
  }
}
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

MCastECv2

DocCapsule.java

```
/*                                        DocCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativav2
 *                                        Comenzado: 19/02/01
 */

package MCastECv2.apps;

import ants.*;
import MCastECv2.apps.variablesglobales;

/** Cápsula con fragmento de documento actualizado. Un índice indica la
posición del primer byte en el documento
 *  y se utiliza para confirmar la recepción.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class DocCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.DocCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.DocCapsule");
  protected byte[] pid() { return PID; }

  int indice; //indice del primer short del fragmento
  int nfrag;  //número de fragmentos total
  short[] fragmento;
  variablesglobales vg = new variablesglobales();

  public int length() {
    int s = Xdr.INT + Xdr.INT + Xdr.SHORT * vg.frag_doc;
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(indice);
    xdr.PUT(nfrag);
    for (int i=0; i<vg.frag_doc; i++){
      xdr.PUT(fragmento[i]);
    }

    return xdr;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    indice = xdr.INT();
    nfrag = xdr.INT();
    fragmento = new short[vg.frag_doc];
    for (int i=0; i<vg.frag_doc; i++){
      fragmento[i]=xdr.SHORT();
    }

    return xdr;
  }

  public boolean evaluate(Node n) {  //enrutamiento unicast hacia el
Writer que solicitó el doc actualizado
    if (n.getAddress() != getDst())
      return n.routeForNode(this, getDst());
    else return n.deliverToApp(this, dpt);
  }//evaluate

  public DocCapsule() { }

  public DocCapsule(short sa, short da, int na, int ind, int nf, short[]
frag) {
    super (sa,da,na,new ByteArray(0));

    indice = ind;
    nfrag = nf;
    fragmento = frag;
  }
}
```

ChangesCapsule.java

```
/**                                        GChangesCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                  Comenzado: 19/02/01
 */

package MCastECv2.apps;

import ants.*;
import ants.wrapper.*;
import MCastECv2.apps.variablesglobales;

/** Cápsula con los cambios globales que se envía haciendo multicast a
todos los nodos suscritos
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
public class GChangesCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.GChangesCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.GChangesCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int Rgroup;
  int target;

  byte nnodos;    //número de nodos cuyos cambios se envían (como máximo
256)
  int[] nodos;
  byte[] ncambios; //número de cambios por nodo
  int[][] cambios;    //cambios que se envían por cada nodo cambios[nodo
i][cambio j]
  int[] indiceprimero; //indice del primer cambio en cada nodo
  int sello; //sello para la identificación de esta cápsula de cambios
global.

  public int length() {
    int s = Xdr.INT + Xdr.INT + Xdr.BYTE + (Xdr.INT + Xdr.BYTE + Xdr.INT)
* nnodos + Xdr.INT;
    for (int j = 0; j<nnodos; j++)
       s +=(int)(Xdr.INT * ncambios[j]); //número de cambios que se
envían por nodo
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(Rgroup);
    xdr.PUT(target);
    xdr.PUT(nnodos);
    for (int j = 0; j<nnodos; j++){
       xdr.PUT(nodos[j]);
       xdr.PUT(ncambios[j]);
       for (int k = 0; k<ncambios[j]; k++)
          xdr.PUT(cambios[j][k]);
       xdr.PUT(indiceprimero[j]);
    }
    xdr.PUT(sello);
    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    Rgroup = xdr.INT();
    target = xdr.INT();
    nnodos = xdr.BYTE();
    nodos = new int[nnodos];
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    cambios = new int[nnodos][];
    ncambios = new byte[nnodos];
    indiceprimero = new int[nnodos];
    for (short j = 0; j<nnodos; j++){
       nodos[j] = xdr.INT();
       ncambios[j] = xdr.BYTE();
       cambios[j] = new int[ncambios[j]];
       for (short k = 0; k<ncambios[j]; k++){
          cambios[j][k] = xdr.INT();
       }
       indiceprimero[j] = xdr.INT();
    }
    sello = xdr.INT();
    return xdr;
  }

  public boolean evaluate(Node n) {  //enrutamiento multicast desde el
Sender
    W_JAIIAI m = (W_JAIIAI)n.getCache().get(Rgroup, target);
    if (m != null) {// si estamos en un nodo intermedio o en el Sender
      if (m.i1 == 0){//si no está esperando confirmación
      //se confirma inmediatamente
      m.i1 = sello;
      //ponemos esperando_confirmacion = sello;
      //se distribuye por multicast
      for (int i = 0; i < m.ai1.length; i++)
        n.routeForNode(this, m.ai1[i]);
      return n.deliverToApp(this,dpt);//Se pasa a la aplicación para
confirmarla
      }else//si está esperando confirmación
      //se descarta esta nueva GCCap o se confirma en el caso de que sea
una GCCap ya recibida
      if (m.i1 == sello) //si es una GCCap ya recibida se pasa a la
aplicacion para volver a confirmar
        n.deliverToApp(this,dpt);
    }else n.deliverToApp(this,dpt);//si m==null estamos en un nodo Reader

    return true;
  }

  public GChangesCapsule() { }

  public GChangesCapsule(int ga,int ta,short sa,short da,int na,int[]
nod,int[][] cam,int[] indp, int s) {
    super (sa,da,na,new ByteArray(0));

    Rgroup = ga;
    target = ta;
    nnodos = (byte) nod.length;
    nodos = nod;
    ncambios = new byte[nnodos];
    for (byte i=0; i<nnodos; i++)
      ncambios[i] = (byte)cam[i].length;
    cambios = cam;
    indiceprimero = indp;
    sello = s;
  }
}
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

DocACKCapsule.java

```
/*                                  DocACKCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                  Comenzado: 19/02/01
 */

package MCastECv2.apps;

import ants.*;
import ants.wrapper.*;

/** Cápsula de confirmación de la recepción de un fragmento del documento
global
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class DocACKCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.DocACKCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.DocACKCapsule");
  protected byte[] pid() { return PID; }

  int indice; //indice del fragmento del documento recibido

  public int length() {
    int s = Xdr.INT;
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(indice);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    indice = xdr.INT();

    return xdr;
  }

  public boolean evaluate(Node n) {  //enrutamiento unicast hacia el
Sender
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    if ( n.getAddress() == getDst())
      return n.deliverToApp(this, dpt);
    else
      return n.routeForNode(this, getDst());

  }

  public DocACKCapsule() { }

  public DocACKCapsule(short sa, short da, int na, int i) {
    super (sa,da,na,new ByteArray(0));
    indice = i;
  }
}
```

WriterSCapsule.java

```
/**                              WriterSCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                  Comenzado: 19/02/01
 */

package MCastECv2.apps;

import ants.*;
import ants.wrapper.*;

/** Cápsula con petición de suscripción o petición se desuscripción como
escritor
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class WriterSCapsule
  extends Capsule
{
  final private static byte[] MID = findMID("apps.WriterSCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.WriterSCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int Wgroup;
  int target;
  short spt;
  short dpt;
  int reverse;
  boolean suscripcion; // '1' cuando se está suscribiendo y '0' cuando se
está desuscribiendo

  public int length() {
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    return super.length() + Xdr.INT + Xdr.INT + Xdr.SHORT + Xdr.SHORT +
Xdr.INT + Xdr.BOOLEAN;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(Wgroup);
    xdr.PUT(target);
    xdr.PUT(spt);
    xdr.PUT(dpt);
    xdr.PUT(reverse);
    xdr.PUT(suscripcion);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    Wgroup = xdr.INT();
    target = xdr.INT();
    spt = xdr.SHORT();
    dpt = xdr.SHORT();
    reverse = xdr.INT();
    suscripcion = xdr.BOOLEAN();

    return xdr;
  }

  public boolean evaluate(Node n) {  //enrutamiento unicast hacia el
Sender
    if (!suscripcion){
      W_JAI m = (W_JAI)n.getCache().get(Wgroup, target);
      if ((m != null)&&(reverse != 0)) {
        boolean found = false;
      int i = 0;
        for (i = 0; m.ai1 != null && i < m.ai1.length; i++) {
        if (m.ai1[i] == reverse) { found = true; break; }
        }
        if (found) { //quitamos esta entrada
        int len = m.ai1.length;
        int[] nn = new int[len-1];
        System.arraycopy(m.ai1, 0, nn, 0, i);
        if (len > i+1) System.arraycopy(m.ai1, i+1, nn, i, len-i-1);
        m.ai1 = nn;
        }
      }
    }
    if ( n.getAddress() == target)
      return n.deliverToApp(this, dpt);
    else {
      reverse = n.getAddress();
      return n.routeForNode(this, target);
    }
  }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public WriterSCapsule() { }

  public WriterSCapsule(int Wga, int ta, short sa, short da, boolean
suscrip) {

    Wgroup = Wga;
    target = ta;
    spt = sa;
    dpt = da;
    reverse = 0;
    suscripcion = suscrip;
  }
}
```

Tabla_indices.java

```
/**                                Tabla_indices.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                  Comenzado: 19/02/01
 */

package MCastECv2.apps;
import MCastECv2.apps.*;

/** Clase que almacena la información de los índices inicial y final de
 *  cada segmento del documento global que está siendo modificado por
algún Writer.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class Tabla_indices {
   private int[] writers;
   private int[] indice_inicial;
   private int[] indice_final;
   variablesglobales vg = new variablesglobales();

   //busqueda dicotómica
   int busca_posicion(int i_inicial){//devuelve el índice de dónde
deberia ir el writer
      int indice_a=0;
      int indice_b=writers.length-1;
      int punto_medio = (int)((indice_b+indice_a)/2);
      while (indice_a<indice_b){
      if (indice_inicial[punto_medio]>i_inicial){
            indice_b = punto_medio;
      }else{
            indice_a = punto_medio+1;
      }
      punto_medio = (int)((indice_b+indice_a)/2);
      }
      return punto_medio;
   }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    int existe_writer(int writer){ //devuelve la posición del writer en el
 caso de que exista, si no -1
       int encontrado = -1;
       for (int i=0;i<writers.length;i++)
       if (writers[i] == writer){
         encontrado = i;
         break;
       }
       return encontrado;
    }

    void inserta_nuevo_writer (int writer, int i_inicial, int i_final){
     if(existe_writer(writer)<0){
       int len = writers.length;
       int[] aux_w = new int[len+1];
       int[] aux_i_i = new int[len+1];
       int[] aux_i_f = new int[len+1];
       int punto_insercion=0;
       if (len>0){ //si la lista no está vacía
       if (i_inicial>indice_inicial[len-1]){//si va el último
         punto_insercion = len;
       }else{
         punto_insercion = busca_posicion(i_inicial);
       }
       System.arraycopy(writers,0,aux_w,0,punto_insercion);
       System.arraycopy(indice_inicial,0,aux_i_i,0,punto_insercion);
       System.arraycopy(indice_final,0,aux_i_f,0,punto_insercion);
       aux_w[punto_insercion]=writer;
       aux_i_i[punto_insercion]=i_inicial;
       aux_i_f[punto_insercion]=i_final;
       if (i_inicial<indice_inicial[len-1]){//si no va el último

System.arraycopy(writers,punto_insercion,aux_w,punto_insercion+1,len -
punto_insercion);

System.arraycopy(indice_inicial,punto_insercion,aux_i_i,punto_insercion+1
,len - punto_insercion);

System.arraycopy(indice_final,punto_insercion,aux_i_f,punto_insercion+1,l
en - punto_insercion);
       }
       }else{ //si la lista está vacía
       aux_w[punto_insercion]=writer;
       aux_i_i[punto_insercion]=i_inicial;
       aux_i_f[punto_insercion]=i_final;
       }
       writers = aux_w;
       indice_inicial = aux_i_i;
       indice_final = aux_i_f;
     }//si existe el writer, desechamos la petición porque debe de ser un
error (debe solicitar primero la salida)
    }

    void saca_writer (int writer){
     int posicion = existe_writer(writer);
     int len = writers.length;
     if(posicion>=0){ //si existe el writer
       int[] aux_w = new int[len-1];
       int[] aux_i_i = new int[len-1];
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      int[] aux_i_f = new int[len-1];
      System.arraycopy(writers,0,aux_w,0,posicion);
      System.arraycopy(indice_inicial,0,aux_i_i,0,posicion);
      System.arraycopy(indice_final,0,aux_i_f,0,posicion);
      if (len>1){
      System.arraycopy(writers,posicion+1,aux_w,posicion,len-
(posicion+1));
      System.arraycopy(indice_inicial,posicion+1,aux_i_i,posicion,len-
(posicion+1));
      System.arraycopy(indice_final,posicion+1,aux_i_f,posicion,len-
(posicion+1));
      }
      writers = aux_w;
      indice_inicial = aux_i_i;
      indice_final = aux_i_f;
    }//si no existe el writer, desechamos la petición porque debe de ser
un error
   }

   int getIndiceInicial(int indice_writer){
     return indice_inicial[indice_writer];
   }

   int getIndiceFinal(int indice_writer){
     return indice_final[indice_writer];
   }

   void modifica_indices(int writer, boolean insercion){//insercion =
false: supresion
     int posicion = existe_writer(writer);
     if (posicion>-1){
      if(insercion)
      indice_final[posicion]++;
      else indice_final[posicion]--;
      for(int i= posicion+1; i<writers.length; i++){
      if (insercion){
        indice_final[i]++;
        indice_inicial[i]++;
      }else{
        indice_final[i]--;
        indice_inicial[i]--;
      }
      }
     }
   }

   Tabla_indices (){
     writers = new int[0];
     indice_inicial = new int[0];
     indice_final = new int[0];
   }//constructor
}//Tabla_indices
```

GChangesCapsule.java

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
/**                                          GChangesCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                   Comenzado: 19/02/01
 */

package MCastECv2.apps;

import ants.*;
import ants.wrapper.*;
import MCastECv2.apps.variablesglobales;

/** Cápsula con los cambios globales que se envía haciendo multicast a
todos los nodos suscritos
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */


public class GChangesCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.GChangesCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.GChangesCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int Rgroup;
  int target;

  byte nnodos;     //número de nodos cuyos cambios se envían (como máximo
256)
  int[] nodos;
  byte[] ncambios; //número de cambios por nodo
  int[][] cambios;     //cambios que se envían por cada nodo cambios[nodo
i][cambio j]
  int[] indiceprimero; //indice del primer cambio en cada nodo
  int sello; //sello para la identificación de esta cápsula de cambios
global.

  public int length() {
    int s = Xdr.INT + Xdr.INT + Xdr.BYTE + (Xdr.INT + Xdr.BYTE + Xdr.INT)
* nnodos + Xdr.INT;
    for (int j = 0; j<nnodos; j++)
       s +=(int)(Xdr.INT * ncambios[j]); //número de cambios que se
envían por nodo
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(Rgroup);
    xdr.PUT(target);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    xdr.PUT(nnodos);
    for (int j = 0; j<nnodos; j++){
       xdr.PUT(nodos[j]);
       xdr.PUT(ncambios[j]);
       for (int k = 0; k<ncambios[j]; k++)
          xdr.PUT(cambios[j][k]);
       xdr.PUT(indiceprimero[j]);
    }
    xdr.PUT(sello);
    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    Rgroup = xdr.INT();
    target = xdr.INT();
    nnodos = xdr.BYTE();
    nodos = new int[nnodos];
    cambios = new int[nnodos][];
    ncambios = new byte[nnodos];
    indiceprimero = new int[nnodos];
    for (short j = 0; j<nnodos; j++){
       nodos[j] = xdr.INT();
       ncambios[j] = xdr.BYTE();
       cambios[j] = new int[ncambios[j]];
       for (short k = 0; k<ncambios[j]; k++){
          cambios[j][k] = xdr.INT();
       }
       indiceprimero[j] = xdr.INT();
    }
    sello = xdr.INT();
    return xdr;
  }

  public boolean evaluate(Node n) {  //enrutamiento multicast desde el
Sender
    W_JAIIAI m = (W_JAIIAI)n.getCache().get(Rgroup, target);
    if (m != null) {// si estamos en un nodo intermedio o en el Sender
      if (m.i1 == 0){//si no está esperando confirmación
      //se confirma inmediatamente
      m.i1 = sello;
      //ponemos esperando_confirmacion = sello;
      //se distribuye por multicast
      for (int i = 0; i < m.ai1.length; i++)
        n.routeForNode(this, m.ai1[i]);
      return n.deliverToApp(this,dpt);//Se pasa a la aplicación para
confirmarla
      }else//si está esperando confirmación
      //se descarta esta nueva GCCap o se confirma en el caso de que sea
una GCCap ya recibida
      if (m.i1 == sello) //si es una GCCap ya recibida se pasa a la
aplicacion para volver a confirmar
        n.deliverToApp(this,dpt);
    }else n.deliverToApp(this,dpt);//si m==null estamos en un nodo Reader

    return true;
  }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public GChangesCapsule() { }

  public GChangesCapsule(int ga,int ta,short sa,short da,int na,int[]
nod,int[][] cam,int[] indp, int s) {
    super (sa,da,na,new ByteArray(0));

    Rgroup = ga;
    target = ta;
    nnodos = (byte) nod.length;
    nodos = nod;
    ncambios = new byte[nnodos];
    for (byte i=0; i<nnodos; i++)
      ncambios[i] = (byte)cam[i].length;
    cambios = cam;
    indiceprimero = indp;
    sello = s;
  }
}
```

NodeApplication.java

```
/**                              NodeApplication.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                  Comenzado: 19/02/01
 */

package MCastECv2.apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;
import MCastECv2.apps.*;

/** Aplicación que corre en los nodos intermedios de la topología
multicast.
 *
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class NodeApplication
  extends MotherApplication
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  boolean retransmitiendo_gccap;      // está retransmitiendo una cápsula
de cámbios globales
  boolean suscrito_lector;

  Label estado;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  Button FIN;
  Label nombre;

  GChangesCapsule lCap;

  final public static String[] defaults = {};

  synchronized public void receive(Capsule cap) {
    super.receive(cap);

    if (cap instanceof GChangesCapsule) {
      lCap = (GChangesCapsule) cap;
      if (suscrito_lector){
      retransmitiendo_gccap = true;
      send(new
GCACKCapsule(Rgroup,target,port,port,lCap.getSrc(),lCap.sello));
      }else{
      send(new ReaderSCapsule(Rgroup,target,port,port,lCap.getSrc(),
suscrito_lector));
      }
    }else if (cap instanceof GCACKCapsule) {
      thisNode().log("Recibida GCACKCapsule");
      retransmitiendo_gccap = false;
    }else if (cap instanceof ReaderSCapsule){
      ReaderSCapsule rscap = (ReaderSCapsule)cap;
      if (rscap.suscripcion)
      suscrito_lector = true;
      else
      suscrito_lector = false;
    }//if
  }

  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.exit(0);
        return true;
    }else return super.handleEvent(evt);
  }

  public void run(){
    while (true) {
      try {
      Thread.sleep(vg.tiempoRetransmision_gccap);
      } catch (InterruptedException e) {
      continue;
      }
      if (retransmitiendo_gccap){ //periódicamente retransmitimos la
cápsula de cámbios globales
        send(new
GCACKCapsule(Rgroup,target,port,port,lCap.getSrc(),lCap.sello));
       //hasta recibir confirmación de todos los suscriptores
      }
    }//while
  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      for (int i = 0; i < k.length(); i++) {
        if (k.key(i).equals("-target")) {
          target = NodeAddress.fromString(k.arg(i));
          k.strike(i);
        } else if (k.key(i).equals("-Rgroup")) {
        Rgroup = NodeAddress.fromString(k.arg(i));
        k.strike(i);
        } else if (k.key(i).equals("-Wgroup")) {
        Wgroup = NodeAddress.fromString(k.arg(i));
        k.strike(i);
        }
      }

      super.setArgs(k);
    }

    public void start()
      throws Exception
    {
      thisNode().register(new MCastECProtocol());

      retransmitiendo_gccap = false;
      suscrito_lector = false;

      resize(300, 200);

      Panel unico = new Panel();
      unico.setLayout(new GridLayout(3,1,1,1));
      FIN = new Button ("FIN");
      unico.add(FIN);
      estado = new Label(" nada de momento                ",Label.CENTER);
      unico.add(estado);
      nombre = new Label
("Node:"+NodeAddress.toString(thisNode().getAddress()),Label.CENTER);
      unico.add(nombre);

      add("Center", unico);

      new Thread(this).start();
      pack();
      show();
    }

    public NodeApplication() {
      super();
    }
}
```

FragDoc.java

```
/*                                FragDoc.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
 *                                        Comenzado: 19/02/01
 */

package MCastECv2.apps;
import MCastECv2.apps.*;

/** Clase que porta un fragmento del documento en shorts
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class FragDoc {
   private short[] f;
   variablesglobales vg = new variablesglobales();
   short[] get (){
      return f;
   }
   FragDoc (short[] frag){
      f = new short[vg.frag_doc];
      f = frag;
   }//constructor
}//FragDoc
```

SenderApplication.java

```
/**                                SenderApplication.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativav2.2
 *                                    Comenzado: 19/02/01
 */



package MCastECv2.apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;
import MCastECv2.apps.*;


/**   thread que envía los fragmentos del documento global a los
peticionarios del mismo
*/
class EnviaFragmentosDoc
implements Runnable
{
  variablesglobales vg = new variablesglobales();

  SenderApplication sapp;
  int indice_frag=0;

  public void run(){
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    while (true) {
      try {
      Thread.sleep(vg.tiempoNoTransmisionFragsDoc);
      } catch (InterruptedException e) {
      continue;
      }
      while
((!sapp.HayGCCapParaTx)&&(sapp.enviando_doc_completo>0)){//espera a
recibir confirmación de todos los cambios realizados
      sapp.send(new
StopSendingChangesCapsule(sapp.getWGroup(),sapp.getTarget(),sapp.getPort(
),sapp.getPort(), true));
      Enumeration e = sapp.peticionarios_doc_completo.keys();
      FragDoc d = (FragDoc)sapp.fragmentos_doc.get(new
Integer(indice_frag));
      while (e.hasMoreElements()){
        Integer auxKey = (Integer)e.nextElement();
        Integer[] aux =
(Integer[])sapp.peticionarios_doc_completo.get(auxKey);
        if (aux[indice_frag].intValue()<1)
          sapp.send(new
DocCapsule(sapp.getPort(),sapp.getPort(),auxKey.intValue(),indice_frag,sa
pp.getNFrag(),d.get()));
        else {
          boolean entrada_obsoleta = true;
          for(int i = 0; i<aux.length; i++)
            if(aux[i].intValue()==0) {
            entrada_obsoleta = false;
            break;
            }
          if (entrada_obsoleta){ //si ese nodo ya ha confirmado la
recepción de todo el doc global
            sapp.peticionarios_doc_completo.remove(auxKey);
            sapp.enviando_doc_completo--; //decrementamos el semáforo
          }
        }
      }//while (e.hasMoreElements())
      indice_frag = (indice_frag + 1) % sapp.nfrag;

      try {
        Thread.sleep(vg.tiempoTransmisionFragsDoc);
      } catch (InterruptedException ie) {
        continue;
      }
      }//while(sapp.enviando_doc_completo>0)
    }//while (true)
  }//run
  EnviaFragmentosDoc(SenderApplication sapp) {
    this.sapp = sapp;
  }

}//EnviaFragmentosDoc

/**
 *    Thread que se encarga de fabricar las cápsulas de cambios globales
 */
class FabricaGCCaps
  implements Runnable
{
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  variablesglobales vg = new variablesglobales();

  SenderApplication sapp;

  public void run() {
    while (true) {
      try {
      Thread.sleep(vg.tiempoFabricacionGCCaps);
      } catch (InterruptedException e) {
      continue;
      }

      //si hay alguna petición de nuevo lector no fabricamos GCCaps
      while ((sapp.enviando_doc_completo
<1)&&(!sapp.cambios_recibidos.esta_vacio)&&(sapp.peticiones_esperadas<1))
{
        int[] nodos = new int[0];
      byte nnodos = 0;    //número de nodos cuyos cambios se envían (como
máximo 256)
      byte[] ncambios = new byte[0]; //número de cambios por nodo
      int[][] cambios = new int[0][0];  //cambios que se envían x cada
nodo cambios[nodo i][cambio j]
      int[] indiceprimero = new int[0]; //indice del primer cambio en
cada nodo
      int i = 0;
      while
((i<vg.n_nodos_gccap_max)&&(!sapp.cambios_recibidos.esta_vacio)){
        ChangesCapsule auxcap = (ChangesCapsule)
sapp.cambios_recibidos.sacarCap();
        int aux_uia; //el último índice aplicado de ese nodo
        int aux_indice;//el primer índice a aplicar
        if (sapp.ultimo_indice_aplicado.containsKey(new
Integer(auxcap.getSrc()))){
          aux_uia = ((Integer)sapp.ultimo_indice_aplicado.get(new
Integer(auxcap.getSrc()))).intValue();
          if (aux_uia<auxcap.indiceprimero)
            aux_indice = 0;
          else
            aux_indice = aux_uia - auxcap.indiceprimero +1;
        }else {
          aux_indice = 0;
        }
        int dif = (int)auxcap.ncambios-aux_indice; //cuantos cambios
nuevos tenemos
        int posicion_writer =
sapp.tabla_indices.existe_writer(auxcap.getSrc());  //si existe el writer
en la tabla de indices
        if((dif>0)&&(posicion_writer>-1)){ //si hay cambios nuevos que
actualizar
          nnodos++;
          int len = nodos.length;
          int[] aux = new int[len+1];
          System.arraycopy(nodos,0,aux,0,len);
          aux[len] = auxcap.getSrc();
          nodos = aux;
          byte[] aux2 = new byte[len+1];
          System.arraycopy(ncambios,0,aux2,0,len);
          aux2[len] = auxcap.ncambios;
          ncambios = aux2;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
            int[][] aux3 = new int[len+1][];
            System.arraycopy(cambios,0,aux3,0,len);
            cambios = aux3;
            int[] auxcambios = new int[dif];
            System.arraycopy(auxcap.cambios,aux_indice,auxcambios,0,dif);
            //aqui modificar el valor del índice local para que sea global
            int indice_principio_fragmento =
sapp.tabla_indices.getIndiceInicial(posicion_writer);

            for (int k= 0; k<dif; k++)
              auxcambios[k] += indice_principio_fragmento;

            cambios[i] = new int[dif];
            System.arraycopy(auxcambios,0,cambios[i],0,dif);
            int[] aux4 = new int [len + 1];
            System.arraycopy(indiceprimero,0,aux4,0,len);
            aux4[len] = auxcap.indiceprimero+aux_indice;
            indiceprimero = aux4;
            sapp.documento_global =
sapp.aplicar_cambios(sapp.documento_global,auxcambios,0,auxcap.getSrc());
            sapp.ultimo_indice_aplicado.put(new
Integer(auxcap.getSrc()),new
Integer(auxcap.indiceprimero+(int)auxcap.ncambios-1));
            i++;
          }//if
        }//While --- GChangesCap nueva
        if (nnodos>0){
          int sello = (new Random()).nextInt();
          GChangesCapsule gccap = new
GChangesCapsule(sapp.getRGroup(),sapp.getTarget(),sapp.getPort(),sapp.get
Port(),

        0,nodos,cambios,indiceprimero,sello);

          if(!sapp.HayGCCapParaTx)
            sapp.actualizaGCCapLocal(gccap);
          else
            sapp.gccaps_a_tx.meterCap(gccap);
        }
        }
      }
    }

    FabricaGCCaps(SenderApplication sapp) {
      this.sapp = sapp;
    }
}//FabricaGCCaps

/**   Thread que gestiona las peticiones de incorporación al grupo de
Escritores
 */
class GestionAspirantesWriter
  implements Runnable{
  variablesglobales vg = new variablesglobales();

  SenderApplication sapp;

  public void run() {
    while(true){
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      try {
        Thread.sleep(vg.tiempoGestionAspirantesWriter);
      } catch (InterruptedException ie) {
        continue;
      }
      if ((sapp.peticiones_esperadas>0)&&(!sapp.HayGCCapParaTx)){
        sapp.send(new
StopSendingChangesCapsule(sapp.getWGroup(),sapp.getTarget(),sapp.getPort(
),sapp.getPort(), true));
        while(sapp.HayGCCapParaTx){
          try {
            Thread.sleep(vg.tiempoEsperaConfirmacionEnGestion);
          } catch (InterruptedException ie) {
            continue;
          }
        }
        int notificacion_no = 0;
        sapp.peticiones_recibidas = new Hashtable();

while((notificacion_no<vg.no_max_notificaciones)&&(sapp.peticiones_espera
das>0)){
          Enumeration e = sapp.aspirantes_writer.keys();
          while (e.hasMoreElements()){
            Integer aspirante = (Integer)e.nextElement();
            Integer aux = (Integer)sapp.aspirantes_writer.get(aspirante);
            if (aux.intValue()<0) //si no se ha recibido la petición de
este aspirante
              sapp.send(new
SendYourRequestCapsule(sapp.getPort(),sapp.getPort(),aspirante.intValue()
));
          }
          notificacion_no++;
          try {
            Thread.sleep(vg.tiempoEsperaPeticionAspirantes);
          } catch (InterruptedException ie) {
            continue;
          }
        }
        sapp.send(new
StopSendingChangesCapsule(sapp.getWGroup(),sapp.getTarget(),sapp.getPort(
),sapp.getPort(), false));
        sapp.peticiones_esperadas = 0;

        Enumeration e = sapp.aspirantes_writer.keys();
        Hashtable tabla_auxiliar = new Hashtable();
        sapp.resolucion_aspirantes = new Hashtable();
        sapp.sello_resolucion = (new Random()).nextInt();
        sapp.confirmaciones_resultado_esperadas = 0;

sapp.estado.setText("confirmaciones_resultado_esperadas="+sapp.confirmaci
ones_resultado_esperadas);

        while (e.hasMoreElements()){
          Integer aspirante = (Integer)e.nextElement();
          Integer indice_aspirante =
(Integer)sapp.aspirantes_writer.get(aspirante);
          if(indice_aspirante.intValue()>-1){
            Integer[] aux_vector_aspirantes = new Integer[1];
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
            if (tabla_auxiliar.containsKey(indice_aspirante)){//ya hay
aspirantes a ese índice
              Integer[] vector_aspirantes =
(Integer[])tabla_auxiliar.get(indice_aspirante);
              aux_vector_aspirantes = new
Integer[vector_aspirantes.length+1];

System.arraycopy(vector_aspirantes,0,aux_vector_aspirantes,0,vector_aspir
antes.length);
              aux_vector_aspirantes[vector_aspirantes.length] =
aspirante;
            }else

if(indice_aspirante.intValue()>sapp.vector_ocupacion.length-1){
              //hay que aumentar el tamaño del doc
                    int aux  = sapp.vector_ocupacion.length;
                    do{
                      int len = sapp.vector_ocupacion.length;
                      short[] aux_doc_global = new short[len+1];
                      boolean[] aux_vector_ocupacion = new
boolean[len+1];

System.arraycopy(sapp.documento_global,0,aux_doc_global,0,len);

System.arraycopy(sapp.vector_ocupacion,0,aux_vector_ocupacion,0,len);
                      aux_doc_global[len]= (short)((char)(new String("
")).charAt(0));
                      aux_vector_ocupacion[len] = false;
                      sapp.documento_global = aux_doc_global;
                      sapp.vector_ocupacion = aux_vector_ocupacion;
                      aux++;
                    }while(aux <=
indice_aspirante.intValue()+vg.no_max_indices);
              aux_vector_aspirantes[0] = aspirante;
              tabla_auxiliar.put(indice_aspirante,aux_vector_aspirantes);
              }else
if(sapp.vector_ocupacion[indice_aspirante.intValue()]==false){
              aux_vector_aspirantes[0] = aspirante;
              tabla_auxiliar.put(indice_aspirante,aux_vector_aspirantes);
            }else{  //denegaciones automáticas
              sapp.resolucion_aspirantes.put(aspirante,new
Resultado_peticion(sapp.sello_resolucion));
                sapp.confirmaciones_resultado_esperadas++;

sapp.estado.setText("confirmaciones_resultado_esperadas="+sapp.confirmaci
ones_resultado_esperadas);
            }
          }
        }//while

        Enumeration e2 = tabla_auxiliar.keys();

        while (e2.hasMoreElements()) { //decisión sobre los aspirantes
          Integer indice = (Integer)e2.nextElement();
          Integer[] aspirantes = (Integer[]) tabla_auxiliar.get(indice);
          int aspirante_ganador;
          if (aspirantes.length>1){
            aspirante_ganador = ((new
Random()).nextInt())%(aspirantes.length);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
            for (int i = 0; i<aspirantes.length; i++)
            if (i!=aspirante_ganador){
              sapp.resolucion_aspirantes.put(aspirantes[i],new
Resultado_peticion(sapp.sello_resolucion));//ponemos las denegadas
                sapp.confirmaciones_resultado_esperadas++;

sapp.estado.setText("confirmaciones_resultado_esperadas="+sapp.confirmaci
ones_resultado_esperadas);
            }
          }else{
            aspirante_ganador = 0;
          }
          int indice_inicial = indice.intValue();
          int indice_final = indice.intValue();
          int i = 0;
          do {
            sapp.vector_ocupacion[indice_final]=true;
            i++;
            indice_final++;
          }
while((indice_final<sapp.vector_ocupacion.length)&&(sapp.vector_ocupacion
[indice_final]==false)&&(i<vg.no_max_indices));
          if ((indice_final>sapp.vector_ocupacion.length-
1)||(sapp.vector_ocupacion[indice_final]==true))
            indice_final--;

sapp.tabla_indices.inserta_nuevo_writer(aspirantes[aspirante_ganador].int
Value(),indice_inicial,indice_final);

sapp.resolucion_aspirantes.put(aspirantes[aspirante_ganador],new
Resultado_peticion(indice_inicial,i,sapp.sello_resolucion));
          sapp.confirmaciones_resultado_esperadas++;

sapp.estado.setText("confirmaciones_resultado_esperadas="+sapp.confirmaci
ones_resultado_esperadas);
        }//while
        sapp.aspirantes_writer = new Hashtable(); //reiniciamos la lista
de aspirantes
      }
    }
  }//run()
  public GestionAspirantesWriter(SenderApplication sender)  {
    sapp = sender;
  }
}//GestionAspirantesWriter

/** Thread que se encarga de notificar a los aspirantes a escritor
  * la resolución de sus peticiones
  */
class Notificador_resultados_peticiones
  implements Runnable{

  variablesglobales vg = new variablesglobales();
  SenderApplication sapp;

  public void run() {
    while(true){
      try {
        Thread.sleep(vg.tiempoNotificacionAspirantes);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      } catch (InterruptedException ie) {
        continue;
      }
      if (sapp.confirmaciones_resultado_esperadas>0) { //mientras no se
le confirme la recepción de no
        Enumeration e = sapp.resolucion_aspirantes.keys();
        while (e.hasMoreElements()){
          Integer aspirante = (Integer)e.nextElement();
          Resultado_peticion resul =
(Resultado_peticion)sapp.resolucion_aspirantes.get(aspirante);
          sapp.send(new
ResolutionCapsule(sapp.getPort(),sapp.getPort(),aspirante.intValue(),resu
l.aceptacion,resul.indice_inicial,resul.no_indices,resul.sello));
        }
      }
    }//while(true)
  }//run

  public Notificador_resultados_peticiones(SenderApplication sender)  {
    sapp = sender;
  }
}//Notificador_resultados_peticiones

/** Aplicación que corre en el nodo Sender. Recibe los cambios de todos
los escritorse suscritos y
 *  retransmite los cambios globales a todos los lectores.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class SenderApplication
  extends MotherApplication
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  int enviando_doc_completo; //número de Readers que están recibiendo el
documento completo.
  int nfrag; //número fragmentos en que se divide el documentos global
actualmente

  Hashtable  peticionarios_doc_completo;
  Hashtable aspirantes_writer;
  Hashtable resolucion_aspirantes;
  Hashtable peticiones_recibidas;
  Hashtable  fragmentos_doc;
  Hashtable ultimo_indice_aplicado;
  PilaCapsFIFO  gccaps_a_tx;
  PilaCapsFIFO  cambios_recibidos;
  GChangesCapsule GCCapLocal;
  boolean  HayGCCapParaTx;
  boolean HayAlguienSuscrito_lector;
  int peticiones_esperadas = 0;
  int confirmaciones_resultado_esperadas = 0;
  int sello_resolucion;
  boolean[] vector_ocupacion;
  Tabla_indices tabla_indices;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  short[] documento_global;

  Button FIN;
  public Label estado;

  public int getNFrag(){
    return nfrag;
  }

  private short[] relleno_ceros (short[] doc, int n){
    short[] aux = new short[documento_global.length + n];
    for (int i=0; i<documento_global.length+n; i++)
      aux[i] = 0;
    System.arraycopy(documento_global,0,aux,0,documento_global.length);
    return aux;
  }//relleno_ceros

  public void actualizaGCCapLocal(GChangesCapsule gccap){
    GCCapLocal = new GChangesCapsule();
    GCCapLocal = gccap;
    HayGCCapParaTx = true;
  }

  public short[] aplicar_cambios(short[] doc_1, int[] cambs, int
ind_primero, int writer){
    short[] aux_doc = new short[doc_1.length];
    aux_doc = doc_1;
    for (int i = ind_primero; i < cambs.length; i++){
      short[] aux;
      int auxcamb = cambs[i];
      byte op = (byte)((cambs[i]>>14)&3);
      short c = (short)((cambs[i]>>16)&65535);
      int indice = (int)(cambs[i]&16383);
      int indice_inicio_fragmento =
tabla_indices.getIndiceInicial(tabla_indices.existe_writer(writer));
      boolean[] aux_vector_ocupacion;

      int len = aux_doc.length;
      switch (op){
        case 0://suprimir
        if (len>0){
          aux = new short[len-1];
          System.arraycopy(aux_doc,0,aux,0,indice);
            if (indice<len-1){
               System.arraycopy(aux_doc,indice+1,aux,indice,len-indice-
1);
          }
          tabla_indices.modifica_indices(writer, false);
          aux_vector_ocupacion = new boolean[vector_ocupacion.length-1];

System.arraycopy(vector_ocupacion,0,aux_vector_ocupacion,0,indice_inicio_
fragmento);
          if(indice_inicio_fragmento<vector_ocupacion.length-1)
            System.arraycopy(vector_ocupacion,indice_inicio_fragmento+1,

      aux_vector_ocupacion,indice_inicio_fragmento,vector_ocupacion.lengt
h-indice_inicio_fragmento-1);
          vector_ocupacion = aux_vector_ocupacion;
        }else     aux = new short[0];
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        break;
        case 1://insertar
        aux = new short[len+1];
        System.arraycopy(aux_doc,0,aux,0,indice);
        aux[indice] = c;
          if (indice<len){
             System.arraycopy(aux_doc,indice,aux,indice+1,len-indice);
        }
        tabla_indices.modifica_indices(writer, true);
        aux_vector_ocupacion = new boolean[vector_ocupacion.length+1];

System.arraycopy(vector_ocupacion,0,aux_vector_ocupacion,0,indice_inicio_
fragmento);
        aux_vector_ocupacion[indice_inicio_fragmento]=true;
        System.arraycopy(vector_ocupacion,indice_inicio_fragmento,

      aux_vector_ocupacion,indice_inicio_fragmento+1,vector_ocupacion.len
gth-indice_inicio_fragmento);
        vector_ocupacion = aux_vector_ocupacion;
        break;
        case 2://sustituir
        if (len>0){
          aux = new short[len];
          System.arraycopy(aux_doc,0,aux,0,len);
          aux[indice] = c;
        }else     aux = new short[0];
        break;
      default:
        aux = aux_doc;
      }//switch
      aux_doc = aux;
    }
    return aux_doc;
  }//aplicar_cambios


  void inicializar_todas_las_listas(){
    HayAlguienSuscrito_lector = false;
    enviando_doc_completo = 0;
    confirmaciones_resultado_esperadas = 0;
    peticiones_esperadas = 0;
    peticionarios_doc_completo = new Hashtable();
    aspirantes_writer = new Hashtable();
    resolucion_aspirantes = new Hashtable();
    peticiones_recibidas = new Hashtable();
    vector_ocupacion = new boolean[documento_global.length];
    for (int i=0; i<vector_ocupacion.length; i++)
      vector_ocupacion[i] = false;
    tabla_indices = new Tabla_indices();
  }//inicializar_todas_las_listas()



  final public static String[] defaults = {};  //periodo de emisión cap
subcripcion

  synchronized public void receive(Capsule cap) {
    super.receive(cap);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    if (cap instanceof ChangesCapsule) {
      thisNode().log("Recibida ChangesCapsule");
      if ((peticiones_esperadas>0)||(enviando_doc_completo>0)){//si se
está atendiendo a un nuevo lector o escritor
      send(new
StopSendingChangesCapsule(getWGroup(),getTarget(),getPort(),getPort(),
true));
      }
      ChangesCapsule CCap = (ChangesCapsule) cap;
      if((!ultimo_indice_aplicado.containsKey(new
Integer(CCap.getSrc())))||
      (((Integer)(ultimo_indice_aplicado.get(new
Integer(CCap.getSrc())))).intValue()<CCap.indiceprimero+CCap.ncambios-1))
        cambios_recibidos.meterCap(CCap);

    }else if (cap instanceof GCACKCapsule) {
      thisNode().log("Recibida GChangesACKCapsule");
      if (!gccaps_a_tx.esta_vacio) {
      GCCapLocal = new GChangesCapsule();
      GCCapLocal = (GChangesCapsule) gccaps_a_tx.sacarCap(); //saca GCCap
en el caso de que haya
      HayGCCapParaTx = true;
      }else{//deja de transmitir cambios globales solo cuando se vacia la
cola
      HayGCCapParaTx = false;
      }
    }else if(cap instanceof ReaderSCapsule){ //peticion para suscribirse
como lector
      ReaderSCapsule rscap = (ReaderSCapsule) cap;
      if(rscap.suscripcion){ //si se está suscribiendo
      HayAlguienSuscrito_lector = true;
        if (!peticionarios_doc_completo.containsKey(new
Integer(rscap.getSrc()))){
          if (enviando_doc_completo<1){ //si es la primera petición
fraccionamos el documento
          fragmentos_doc = new Hashtable();
          short[] aux_documento_global;
          aux_documento_global = documento_global;
          if(aux_documento_global.length>1){
              int len = aux_documento_global.length;
              int lz = ((len % vg.frag_doc)>0)? vg.frag_doc - (len %
vg.frag_doc):0;
              if (lz>0)
              aux_documento_global =
relleno_ceros(aux_documento_global,lz);
            nfrag = (int)aux_documento_global.length/vg.frag_doc;
            for (int i=0; i<nfrag; i++){
              short[] aux = new short[vg.frag_doc];

System.arraycopy(aux_documento_global,i*vg.frag_doc,aux,0,vg.frag_doc);
              fragmentos_doc.put(new Integer(i),new FragDoc(aux));
            }
          }else{//si el doc global está vacío ponemos un fragmento con
ceros
            nfrag = 1;
            short[] aux =
relleno_ceros(aux_documento_global,vg.frag_doc);
            fragmentos_doc.put(new Integer(0),new FragDoc(aux));
          }//(documento_global.length>1)
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
          }
          enviando_doc_completo++;
          Integer[] aux = new Integer[nfrag];
          for (int i = 0; i<nfrag; i++)
          aux[i] = new Integer(0); //no se ha recibido confirmación de
ningún fragmento
          //ponemos una entrada en la tabla por el nuevo nodo
          peticionarios_doc_completo.put(new
Integer(rscap.getSrc()),aux);
        }//if !peticionarios_doc_completo.containsKey
      }else{
      inicializar_todas_las_listas(); //recibe está cápsula cuando ya no
hay nadie suscrito
      }
    }else if(cap instanceof WriterSCapsule){ //petición de suscribirse
como escritor
      WriterSCapsule wscap = (WriterSCapsule) cap;
      if(wscap.suscripcion){ //si se está suscribiendo
        if (!aspirantes_writer.containsKey(new Integer(wscap.getSrc()))){
//si no tenemos una peticion de este
        aspirantes_writer.put(new Integer(wscap.getSrc()),new Integer(-
1)); //ponemos una entrada para el nuevo aspirante a writer
        peticiones_esperadas++;
      }
      }else{//si se está desuscribiendo
        int desuscriptor = (new Integer(wscap.getSrc())).intValue();
        int posicion = tabla_indices.existe_writer(desuscriptor);
      if (posicion>-1){
          int indice_inicial = tabla_indices.getIndiceInicial(posicion);
          int indice_final = tabla_indices.getIndiceFinal(posicion);
          do{
            vector_ocupacion[indice_inicial] = false;
            indice_inicial++;
          }while(indice_inicial<=indice_final);
          tabla_indices.saca_writer(desuscriptor);
      }
      send(new WriterSACKCapsule(wscap.getSrc(),getPort(),getPort()));
      }
    }else if(cap instanceof DocACKCapsule){
      thisNode().log("Recibida DocACKCapsule");
      DocACKCapsule dackcap = (DocACKCapsule) cap;
      if (fragmentos_doc.containsKey(new Integer(dackcap.indice)))
//existe entrada para el indice?
      if(peticionarios_doc_completo.containsKey(new
Integer(dackcap.getSrc()))){
        Integer[] aux = (Integer[])peticionarios_doc_completo.get(new
Integer(dackcap.getSrc()));
        aux[dackcap.indice] = new Integer(1);//ponemos el flag
correspondiente a 1
        peticionarios_doc_completo.put(new
Integer(dackcap.getSrc()),aux);
      }
    }else if(cap instanceof RequestCapsule){
      RequestCapsule rcap = (RequestCapsule) cap;
      if (aspirantes_writer.containsKey(new Integer(rcap.getSrc()))){
//si tenemos una peticion de este
      aspirantes_writer.put(new Integer(rcap.getSrc()),new
Integer(rcap.indice)); //modificamos la entrada para el aspirante a
writer
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      if (!peticiones_recibidas.containsKey(new Integer(rcap.getSrc()))){
        peticiones_recibidas.put(new Integer(rcap.getSrc()),new
Integer(1));
        peticiones_esperadas--;
      }
      }
    }else if(cap instanceof ResolutionACKCapsule){
      ResolutionACKCapsule rsackcap = (ResolutionACKCapsule) cap;
      if ((rsackcap.sello ==
sello_resolucion)&&(resolucion_aspirantes.containsKey(new
Integer(rsackcap.getSrc())))){
      //si es una confirmación no obsoleta y no se ha recibido ya
      resolucion_aspirantes.remove(new Integer(rsackcap.getSrc()));
      confirmaciones_resultado_esperadas--;
      estado.setText("confirmaciones_resultado_esperadas="+confirmaciones
_resultado_esperadas);
      }
    }//if
  }


  public boolean handleEvent(Event evt) {
    if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.exit(0);
        return true;
    } else return super.handleEvent(evt);
  }

  public void run(){
    while (true) {
      try {
      Thread.sleep(vg.tiempoRetransmision_gccap);
      } catch (InterruptedException e) {
      continue;
      }
      if
((enviando_doc_completo<1)&&(HayGCCapParaTx)&&(HayAlguienSuscrito_lector)
){ //si no se está enviando el documento completo
          send(GCCapLocal);
      }//if
    }//while
  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-target")) {
        target = NodeAddress.fromString(k.arg(i));
        k.strike(i);
      } else if (k.key(i).equals("-Rgroup")) {
      Rgroup = NodeAddress.fromString(k.arg(i));
      k.strike(i);
      } else if (k.key(i).equals("-Wgroup")) {
      Wgroup = NodeAddress.fromString(k.arg(i));
      k.strike(i);
      }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    }

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {
    thisNode().register(new MCastECProtocol());

    documento_global = new short[0];
    inicializar_todas_las_listas();
    gccaps_a_tx = new PilaCapsFIFO();
    cambios_recibidos = new PilaCapsFIFO();
    ultimo_indice_aplicado = new Hashtable();
    HayGCCapParaTx = false;

    resize(300, 200);

    Panel unico = new Panel ();
    unico.setLayout(new GridLayout(2,1,1,1));
    FIN = new Button ("FIN");
    unico.add(FIN);
    estado = new Label("De momento naita                    ",
Label.CENTER);
    unico.add(estado);


    add("Center", unico);

    new Thread(this).start();
    new Thread(new FabricaGCCaps(this)).start();
    new Thread(new EnviaFragmentosDoc(this)).start();
    new Thread(new GestionAspirantesWriter(this)).start();
    new Thread(new Notificador_resultados_peticiones(this)).start();
    pack();
    show();
  }

  public SenderApplication() {
    super();
  }
}
```

ReaderSCapsule.java

```
/**                          ReaderSCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                              Comenzado: 19/02/01
 */

package MCastECv2.apps;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
import ants.*;
import ants.wrapper.*;

/** Cápsula con petición de suscripción o petición se desuscripción como
lector
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class ReaderSCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.ReaderSCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.ReaderSCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int Rgroup;
  int target;
  short spt;
  short dpt;
  int reverse;
  boolean suscripcion; // '1' cuando se está suscribiendo y '0' cuando se
está desuscribiendo

  public int length() {
    return super.length() + Xdr.INT + Xdr.INT + Xdr.SHORT + Xdr.SHORT +
Xdr.INT + Xdr.BOOLEAN;
  }

  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(Rgroup);
    xdr.PUT(target);
    xdr.PUT(spt);
    xdr.PUT(dpt);
    xdr.PUT(reverse);
    xdr.PUT(suscripcion);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    Rgroup = xdr.INT();
    target = xdr.INT();
    spt = xdr.SHORT();
    dpt = xdr.SHORT();
    reverse = xdr.INT();
    suscripcion = xdr.BOOLEAN();

    return xdr;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  }

  public boolean evaluate(Node n) {  //solicitando el doc actualizado o
la desuscripcion
    if(n.getAddress()==getSrc()){ //si estoy en un nodo Writer
      reverse = n.getAddress();
      return n.routeForNode(this,target);
    }else { //si no estoy en un nodo Writer
      W_JAIIAI m = (W_JAIIAI)n.getCache().get(Rgroup, target);
      boolean found = false;
      int indice=0;
      if (m!=null) //buscamos una entrada para el nodo anterior
      for (int i = 0; m.ai1 != null && i < m.ai1.length; i++) {
        if (m.ai1[i] == reverse) { found = true; indice = i; break; }
      }

      if(suscripcion){ //si me estoy suscribiendo
      if (m == null) { //no habia registro de ninguna clase
        m = new W_JAIIAI();
        n.getCache().put(Rgroup, target, m,
vg.TIEMPO_MAXIMO_SUSCRIPCION);
        m.i1 = 0;  //esperando_confirmacion = 0
      }
      if(!found){//no habia entrada para este nodo anterior
        int len = (m.ai1 != null) ? m.ai1.length : 0;
        int[] nn = new int[len+1];
        int[] confirmaciones = new int[len+1];
        if (len > 0) {
          System.arraycopy(m.ai1, 0, nn, 0, len);
          System.arraycopy(m.ai2, 0, confirmaciones, 0, len);
        }
        nn[len] = reverse;
        confirmaciones[len] = 0;// se pone a uno cuando se recibe una
confirmacion de este nodo
        m.ai1 = nn;
        m.ai2 = confirmaciones;
      }//if !found
      //ahora que ya hay una entrada para multicast hasta este nodo
anterior
      //cursamos esta petición de documento hasta el Sender y señalizamos
a la aplicacion
      n.deliverToApp(this,dpt);
      reverse = n.getAddress();
      return n.routeForNode(this,target);

      }else{//si me estoy desuscribiendo
      if (m==null){
        return n.deliverToApp(this,dpt); //le comunicamos a la aplicación
que ya no tiene nodos suscritos
      }
      if (found){ //si había entrada para este nodo anterior
        int len = m.ai1.length; //la quitamos
        int[] nn = new int[len-1];
        int[] confirmaciones = new int[len-1];
        if (len > 1) {
          if (indice>0){
            System.arraycopy(m.ai1, 0, nn, 0, indice);
            System.arraycopy(m.ai2, 0, confirmaciones, 0, indice);
          }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
          if (indice<m.ai1.length-1){
            System.arraycopy(m.ai1, indice+1, nn, indice, m.ai1.length-
(indice+1));
            System.arraycopy(m.ai2, indice+1, confirmaciones, indice,
m.ai2.length-(indice+1));
          }
        }
        m.ai1 = nn;
        m.ai2 = confirmaciones;

        if (m.ai1.length <1){ //si ya nadie está suscrito se envía al
nodo siguiente y a la aplicación para señalizarlo
          m = null;
          reverse = n.getAddress();
          if (n.getAddress()!=target)
            n.routeForNode(this,target);
          return n.deliverToApp(this,dpt);
        }
        return true;
      }//if found
      return true;
      }//if(suscripcion)
    }//if (si estoy en un nodo Writer)
  }//evaluate

  public ReaderSCapsule() { }

  public ReaderSCapsule(int ga, int ta, short sa, short da, int na,
boolean s) {
    super (sa,da,na,new ByteArray(0));
    Rgroup = ga;
    target = ta;
    reverse = 0;
    suscripcion = s;
  }
}
```

variablesglobales.java

```
/**                              variablesglobales.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                  Comenzado: 19/02/01
 */


package MCastECv2.apps;

/** Variables globales del prototipo
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */


public class variablesglobales {
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
public int tiempoSuscripcion = 2000; //como se realiza una sola
suscripción y desuscripción,ha de ser grande para ahorar CPU
public int ncapsDesuscripcion =5; //número de cápsulas de desuscripción
que se envían
public int frag_doc = 500; //tamaño en shorts de cada fragmento de
documento enviado
public int n_cambios_max = 20; //número máximo de van en cada cápsula
public int n_nodos_gccap_max = 5; //número máximo de nodos caben en cada
cápsula gccap
public int tiempoEnvioCambios = 2000; //tiempo de envío de los cambios
locales al sender
public int tiempoRetransmision_gccap = 2000; //tiempo de retransmisión de
los cambios globales
public int tiempoConfirmacionGCCaps = 2000; //tiempo de retransmisión de
confirmación de los cambios globales
public int tiempoFabricacionGCCaps = 1000; //tiempo de envío de los
cambios locales al sender
public int tiempoTransmisionFragsDoc = 50; //tiempo de transmisión de los
fragmentos del doc global
public int tiempoNoTransmisionFragsDoc = 2000; //tiempo de duerme cuando
no transmite fragmentos del doc global
public int tiempoPeticionDoc = 1000; //tiempo pasado el cual reenvia
petición de fragmentos de doc (si no se ha recibido ninguno)
public int longitud_MAX_DOC = 16384; //14 bits en el índice
public int TIEMPO_MAXIMO_SUSCRIPCION = 100000; //tiempo máximo que un
nodo writer estará suscrito para mantener su info en cache del nodo
public int tiempoGestionAspirantesWriter = 2000;//tiempo pasado el cual
el Sender observa si hay nuevas peticiones de ser writer
public int tiempoEsperaConfirmacionEnGestion = 100;//tiempo que el Sender
espera confirmacion de cambios para decidir nuevas peticiones
public int no_max_notificaciones = 3; //número máximo de notificaciones
public int tiempoEsperaPeticionAspirantes = 1000; // tiempo que espera
por la peticion de un fragmento concreto por parte de los aspirantes
public int no_max_indices = 20; //número máximo de indices del documento
global que se le asignan a un nuevo Writer
public int tiempoNotificacionAspirantes = 500; //Tiempo entre
notificacion y notificacion de los resultados de peticiones a Writer
public int tiempoEsperaParaEnvioCambios = 3000; //tiempo que el writer
espera antes de volver a enviar los cambios locales
public int tiempoLiberaFragmento = 3000; //tiempo que el writer espera
antes de volver a intentar liberar su fragmento
}
```

PilaCapsFIFO.java

```
/**                              PilaCapsFIFO.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa
 *                                  Comenzado: 16/01/01
 */

package MCastECv2.apps;

import ants.*;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
/** Pila FIFO para el almacenamiento ordenado de cápsulas de cualquier
tipo
 *
 *. @author Javier Jesús Sánchez Medina
 *
 */

public class PilaCapsFIFO{
   private DataCapsule[] vector;
   public boolean esta_vacio;
   DataCapsule sacarCap(){
      if (!esta_vacio){
      int len = vector.length-1;
      if (len < 1){
        DataCapsule aux = new DataCapsule();
        aux = vector[0];
        esta_vacio = true;
        vector = new DataCapsule[0];
        return aux;
      }else{
        DataCapsule[] aux = new DataCapsule[len];
          DataCapsule aux2 = new DataCapsule();
        System.arraycopy(vector, 1, aux, 0, len);
        aux2 = vector[0];
        vector = aux;
        esta_vacio = false;
        return aux2;
      }
      }else return null;
   }
   void meterCap(DataCapsule cap){
      if (!esta_vacio){
      int len = vector.length;
        DataCapsule[] aux = new DataCapsule[len + 1];
        System.arraycopy(vector, 0, aux, 0, vector.length);
        aux[vector.length] = cap;
      vector = aux;
      }else{
      esta_vacio = false;
        vector = new DataCapsule[1];
      vector[0] = cap;
      }
   }

   PilaCapsFIFO (){
      this.esta_vacio = true;
      this.vector = new DataCapsule[0];

   }//constructor
}//PilaCapsFIFO
```

MotherApplication.java

```
/**                              MotherApplication.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
 *                                        Comenzado: 19/02/01
 */


package MCastECv2.apps;
import ants.*;


/** Aplicación que Madre de las Receiver y Sender
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class MotherApplication
  extends Application
{

  int target,Rgroup,Wgroup;
  short port;

  public short getPort(){
    return port;
  }

  public int getTarget(){
    return target;
  }

  public int getRGroup(){
    return Rgroup;
  }

  public int getWGroup(){
    return Wgroup;
  }
}
```

ReceiverApplication.java

```
/**                              ReceiverApplication.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                  Comenzado: 19/02/01
 */

package MCastECv2.apps;

import java.awt.*;
import java.util.*;

import ants.*;
import utils.*;
import MCastECv2.apps.*;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
/**
 *    Thread que se encarga de enviar al Sender los cambios locales
 *
 */
class EnviaCambios
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  ReceiverApplication rapp;

  public void run() {
    while (true) {
      try {
      Thread.sleep(vg.tiempoEnvioCambios);
      } catch (InterruptedException e) {
      continue;
      }
      if((!rapp.recibiendo_doc)&&(rapp.get_indice_cambio_antiguo
()<rapp.get_indice_cambio())&&(rapp.suscrito_escritor)&&(rapp.fragmento_c
oncedido)&&(!rapp.parar_envio_cambios)){
      //si no estoy recibiendo el documento total,si hay cambios sin
confirmar y estoy suscrito
        for (int
i=rapp.get_indice_cambio_antiguo();i<rapp.get_indice_cambio();i = i +
vg.n_cambios_max){
        int L;
        if (rapp.get_indice_cambio()-i >= vg.n_cambios_max)
          L = vg.n_cambios_max;
        else L = rapp.get_indice_cambio()-i;
        int[] cam = new int[0];
        for (int j = 0; j<L; j++){
          int [] aux = new int[cam.length +1];
          System.arraycopy(cam, 0, aux, 0, cam.length);
          if(rapp.tabla_cambios.containsKey(new Integer(i+j)))
            aux[cam.length] = (int)((Integer)rapp.tabla_cambios.get(new
Integer(i+j))).intValue();
          else break;
          cam = aux;
        }
        rapp.send(new
ChangesCapsule(rapp.getPort(),rapp.getPort(),rapp.getTarget(), cam, i));
        }//for
      }//if
    }//while
  }//run
  EnviaCambios(ReceiverApplication rapp) {
    this.rapp = rapp;
  }//constructor
}//EnviaCambios

/** Thread que para momentaneamente la trasnmisión de los cambios locales
 */
class EsperaParaEnviarCambios
  implements Runnable
{
  variablesglobales vg = new variablesglobales();
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
   ReceiverApplication rapp;
   public void run(){
     try {
       Thread.sleep(vg.tiempoEsperaParaEnvioCambios);
     } catch (InterruptedException e) { }
     rapp.parar_envio_cambios = false;
   }//run

   EsperaParaEnviarCambios(ReceiverApplication rapp){
     this.rapp = rapp;
   }
}//EsperaParaEnviarCambios

/** Thread que se encarta de liberar un fragmento previamente reservado
 */
class Liberar_fragmento
  implements Runnable
{
  variablesglobales vg = new variablesglobales();

  ReceiverApplication rapp;

  public void run() {
    while (true) {
      try {
      Thread.sleep(vg.tiempoLiberaFragmento);
      } catch (InterruptedException e) {
      continue;
      }
      if(rapp.liberando_fragmento){
        rapp.send(new
WriterSCapsule(rapp.getWGroup(),rapp.getTarget(),rapp.getPort(),rapp.getP
ort(),false));
//
      rapp.estado.setText("liberando_fragmento="+rapp.liberando_fragmento
);
      }
    }//while
  }//run
  Liberar_fragmento(ReceiverApplication rapp) {
    this.rapp = rapp;
  }//constructor
}//Liberar_fragmento

/** Aplicación que corre en los nodos Receiver y que modifica su copia
local del documento
 *  transmitiendo los cambios al Sender
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class ReceiverApplication
  extends MotherApplication
  implements Runnable
{
  variablesglobales vg = new variablesglobales();
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public boolean suscrito_lector = false;      // está suscrito como
lector?
  public boolean suscrito_escritor = false;      // está suscrito como
escritor?
  public boolean fragmento_concedido = false;  //se le ha concedido un
fragmento para editar?
  public boolean liberando_fragmento = false;
  public int indice_inicio_fragmento,longitud_fragmento;

  short[] documento_local,fragmento_local;
  Hashtable tabla_cambios;
  Hashtable tabla_tiempos_cambios;
  Hashtable ultimos_cambios_global;
  boolean recibiendo_doc = false;
  boolean esperando_primer_frag = false;
  boolean esperando_para_pedir_frag = false;
  Hashtable fragmentos_recibidos;
  int nfrag_recibidos = 0;
  int ultimo_sello_recibido = 0;
  boolean parar_envio_cambios = false;

  Button suprimir;
  Button insertar;
  Button sustituir;
  TextField indice_local,caracter_nuevo,indice_global;
  TextArea doc_local,frag_local;
  Label estado,longitud_documento_label,longitud_fragmento_label;

  Button Refrescar_global;
  Button suscribirse_lector;
  Button suscribirse_escritor;
  Button FIN;
  Label nombre;

  int indice_cambio = 0; //indice del próximo cambio a registrar
  int indice_cambio_antiguo = 0;  //indice del próximo cambio a confirmar

  public int get_indice_cambio (){
    return indice_cambio;
  }

  public int get_indice_cambio_antiguo (){
    return indice_cambio_antiguo;
  }


  public void refresca_pantalla_fragmento(){
    String cadena = new String("");
    for (int i = 0; i<fragmento_local.length; i++){
      cadena = cadena + (char) fragmento_local[i];
    }
    frag_local.setText(cadena);
    longitud_fragmento_label.setText("longitud fragmento =
"+fragmento_local.length);

estado.setText("antiguo="+indice_cambio_antiguo+".proximo="+indice_cambio
);
  }
  public void refresca_pantalla_global(){
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    String cadena = new String("");
    for (int i = 0; i<documento_local.length; i++){
      cadena = cadena + (char) documento_local[i];
    }
    doc_local.setText(cadena);
    longitud_documento_label.setText("longitud documento =
"+documento_local.length);
  }

  private short[] elimina_zero_padding(short[] doc){
    short[] aux1 = new short[doc.length];
    aux1 = doc;
    while ((aux1.length>0)&&(aux1[aux1.length-1]==0)){
      int len = aux1.length;
      short[] aux2 = new short[len-1];
      System.arraycopy(aux1,0,aux2,0,len-1);
      aux1 = aux2;
    }
    return aux1;
  }//elimina_zero_padding

  public short[] aplicar_cambios(short[] frag_1, int[] cambs, int
ind_primero){
    short[] aux_doc = new short[frag_1.length];
    aux_doc = frag_1;
    for (int i = ind_primero; i < cambs.length; i++){
      short[] aux;
      int auxcamb = cambs[i];
      int indice = (int)(cambs[i]&16383);
      byte op = (byte)((cambs[i]>>14)&3);
      short c = (short)((cambs[i]>>16)&65535);

      while(aux_doc.length<indice+1){ //si el índice está fuera del
tamaño del doc, aumentamos este con espacios
      short[] aux_documento_local = new short[(aux_doc.length+1)];
      System.arraycopy(aux_doc,0,aux_documento_local,0,aux_doc.length);
      aux_documento_local[aux_doc.length] = (short)((char)(new String("
")).charAt(0));
      aux_doc = aux_documento_local;
      }

      int len = aux_doc.length;
      switch (op){
        case 0://suprimir
        if(len>0){
          aux = new short[len-1];
          System.arraycopy(aux_doc,0,aux,0,indice);
            if (indice<len-1){
               System.arraycopy(aux_doc,indice+1,aux,indice,len-indice-
1);
          }
        }else aux = new short[0];
        break;
        case 1://insertar
        aux = new short[len+1];
        System.arraycopy(aux_doc,0,aux,0,indice);
        aux[indice] = c;
          if (indice<len){
             System.arraycopy(aux_doc,indice,aux,indice+1,len-indice);
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        }
        break;
        case 2://sustituir
        if(len>0){
          aux = new short[len];
          System.arraycopy(aux_doc,0,aux,0,len);
          aux[indice] = c;
        }else aux = new short[0];
        break;
      default:
        aux = aux_doc;
      }//switch
      aux_doc = aux;
    }
    return aux_doc;
  }//aplicar_cambios
  final public static String[] defaults = {};

  synchronized public void receive(Capsule cap) {
    super.receive(cap);
    if (cap instanceof DocCapsule) {
      thisNode().log("Recibida DocCapsule");
      DocCapsule dcap = (DocCapsule) cap;
      esperando_primer_frag = false;
      if (!fragmentos_recibidos.containsKey(new Integer(dcap.indice))){
      fragmentos_recibidos.put(new Integer(dcap.indice),new Integer(1));
      nfrag_recibidos++;
      short[] fragmento_nuevo = new short[vg.frag_doc];
      fragmento_nuevo = dcap.fragmento;

        int L = vg.frag_doc*(dcap.indice+1);
      if (documento_local.length <= vg.frag_doc*dcap.indice){
          int len = documento_local.length;
          short[] aux2 = new short[L];
          System.arraycopy(documento_local, 0, aux2, 0, len);
          documento_local = aux2;
        }

System.arraycopy(fragmento_nuevo,0,documento_local,vg.frag_doc*dcap.indic
e,vg.frag_doc);
      if(nfrag_recibidos >= dcap.nfrag){
        documento_local = elimina_zero_padding(documento_local);
        recibiendo_doc = false; //finaliza la recepción de los fragmentos
del doc;
      }
//    estado.setText("documento_local.length = "+documento_local.length);
      refresca_pantalla_global();
      }
      send(new DocACKCapsule(port,port,target, dcap.indice)); //siempre
enviamos ACK del fragmento
    }else if (cap instanceof GChangesCapsule) {
      GChangesCapsule gccap = (GChangesCapsule) cap;
      if (suscrito_lector){
      send(new
GCACKCapsule(Rgroup,target,port,port,gccap.getSrc(),gccap.sello));
      if(!(gccap.sello == ultimo_sello_recibido)){ //si es una GCCap
Nueva
        ultimo_sello_recibido = gccap.sello;
        estado.setText("recibida capsula de cambios globales nueva");
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
          for (int i = 0; i<gccap.nnodos; i++){
          if (gccap.nodos[i] == thisNode().getAddress()){//son los
cambios propios confirmados
            for (int j = indice_cambio_antiguo;
j<(gccap.indiceprimero[i]+gccap.ncambios[i]);j++){
            if (tabla_cambios.containsKey(new Integer(j))){
                estado.setText("tabla_cambios.containsKey del indice
"+j);
              tabla_cambios.remove(new Integer(j)); //quitamos los
cambios ya confirmados de la tabla
              long latencia = thisNode().time()-
((Tiempo)(tabla_tiempos_cambios.get(new Integer(j)))).get();
              //medimos la latencia del cambio
              thisNode().log("latencia del cambio "+j+":"+latencia);
              tabla_tiempos_cambios.remove(new Integer(j)); //quitamos el
tiempo de los cambios ya confirmados de la tabla
              indice_cambio_antiguo = j+1; //actualizamos la info del
último cambio confirmado
            }else break;
              }
          }
          if (ultimos_cambios_global.containsKey(new
Integer(gccap.nodos[i]))){
            int auxindice =
(int)(((Integer)ultimos_cambios_global.get(new
Integer(gccap.nodos[i]))).intValue());
            int ind_primero;
            if (auxindice<gccap.indiceprimero[i])
            ind_primero = 0;
            else
            ind_primero = auxindice - gccap.indiceprimero[i] + 1;
            if(documento_local==null)
            documento_local = new short[0];
            documento_local = aplicar_cambios(documento_local,
gccap.cambios[i], ind_primero);
            ultimos_cambios_global.remove(new Integer(gccap.nodos[i]));
          }else{
            documento_local = aplicar_cambios(documento_local,
gccap.cambios[i], 0);
          }
          ultimos_cambios_global.put(new Integer(gccap.nodos[i]),new
Integer(gccap.indiceprimero[i]+gccap.cambios[i].length-1));
        }
//      estado.setText("aplicados GChanges");
//      refresca_pantalla(); //cada cambio global no refresca la
pantalla
      }//if
      }else{ //si no estoy suscrito enviamos una cápsula de desuscricion
      send(new
ReaderSCapsule(Rgroup,target,port,port,target,suscrito_lector));
      }
    }else if (cap instanceof StopSendingChangesCapsule) {
      StopSendingChangesCapsule ssccap = (StopSendingChangesCapsule) cap;
      if (ssccap.parando){
      parar_envio_cambios = true;
      new Thread(new EsperaParaEnviarCambios(this)).start();
      }else parar_envio_cambios = false;
    }else if (cap instanceof SendYourRequestCapsule) {
      int auxindice = Integer.parseInt(indice_global.getText());
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      send(new RequestCapsule(getPort(),getPort(),target,auxindice));
      esperando_para_pedir_frag = false;
    }else if (cap instanceof ResolutionCapsule){
      ResolutionCapsule rsolcap = (ResolutionCapsule) cap;
      if((suscrito_escritor)&&(!fragmento_concedido)){ //si aún no se
habia recibido respuesta
      if (rsolcap.aceptacion){
        fragmento_concedido = true;
        indice_inicio_fragmento = rsolcap.indice_inicial;
//
estado.setText("indice_inicio_fragmento="+indice_inicio_fragmento);
        longitud_fragmento = rsolcap.no_indices;
        fragmento_local = new short[longitud_fragmento];
        if (documento_local == null)
          documento_local = new short[0];

while(documento_local.length<indice_inicio_fragmento+longitud_fragmento+1
){
            short[] aux_documento_local = new
short[(documento_local.length+1)];

System.arraycopy(documento_local,0,aux_documento_local,0,documento_local.
length);
            aux_documento_local[documento_local.length] =
(short)((char)(new String(" ")).charAt(0));
            documento_local = aux_documento_local;
        }

System.arraycopy(documento_local,indice_inicio_fragmento,fragmento_local,
0,longitud_fragmento);
        //a partir de ahora el fragmento local y el documento local están
separados
          tabla_cambios = new Hashtable();
          tabla_tiempos_cambios = new Hashtable();
        indice_cambio_antiguo = indice_cambio;

        refresca_pantalla_fragmento();
        estado.setText("petición de fragmento aceptada");
        }else{
        fragmento_concedido = false;
        estado.setText("petición de fragmento denegada");
        suscribirse_escritor.setLabel("Sus. Escritor");
        suscrito_escritor = false;
        }
      }
      send(new
ResolutionACKCapsule(getWGroup(),target,getPort(),getPort(),rsolcap.sello
,rsolcap.aceptacion));
    }else if (cap instanceof WriterSACKCapsule) {//if
      liberando_fragmento = false;
//    estado.setText("liberando_fragmento="+liberando_fragmento);
    }
  }

  public boolean handleEvent(Event evt) {
    if
((!recibiendo_doc)&&(suscrito_escritor)&&(fragmento_concedido)&&(evt.id
== Event.ACTION_EVENT &&
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        ((evt.target == suprimir)||(evt.target == insertar)||(evt.target
== sustituir))) {
      int auxindice = Integer.parseInt(indice_local.getText());
        int len = documento_local.length;

      if
((auxindice<longitud_fragmento)&&(!((longitud_fragmento==1)&&(evt.target
== suprimir)))){
      //solo pueden modificar el fragmento asignado y no pueden reducir
su tamaño a menos de 1 caracter
        int op = 0;
        if(evt.target == insertar){
          if(longitud_fragmento == 0){//si el documento es vacio,
aceptamos la inserción de un caracter en la pos 0
            auxindice = 0;
            longitud_fragmento = 0;
          }
          op = 1;
          longitud_fragmento++;
        }else if((longitud_fragmento != 0)&&(evt.target == suprimir)){
          op = 0;
          longitud_fragmento--;
        }else if((longitud_fragmento != 0)&&(evt.target == sustituir))
          op = 2;
        char auxchar = (char)(new
String(caracter_nuevo.getText())).charAt(0);
        short auxshort = (short)(auxchar);
        int[] cambio = new int[1];

        cambio[0] = (int)((((int)(auxshort<<2)+op)<<14) + auxindice);

        fragmento_local = aplicar_cambios(fragmento_local, cambio, 0);

        tabla_cambios.put(new Integer(indice_cambio),new
Integer(cambio[0]));
        tabla_tiempos_cambios.put(new Integer(indice_cambio),new
Tiempo(thisNode().time()));
          indice_cambio++;    //lo actualizamos al valor del próximo
cambio
//      estado.setText("longitud_fragmento="+longitud_fragmento);
        refresca_pantalla_fragmento();
        return true;
      }else if((longitud_fragmento==1)&&(evt.target == suprimir)){
        estado.setText("Imposible hacer 0 la longitud del fragmento");
        return true;
      }else{
        estado.setText("error en la introducción del índice");
//      refresca_pantalla();
        return true;
      }
    }else if ((evt.id == Event.ACTION_EVENT && evt.target ==
suscribirse_lector)&&(!suscrito_escritor)) {
      suscrito_lector = !suscrito_lector;
      if(suscrito_lector)
        suscribirse_lector.setLabel("Desus. Lector");
      else
        suscribirse_lector.setLabel("Sus. Lector");
      send(new ReaderSCapsule(Rgroup,target,port,port,target,
suscrito_lector));
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      //si suscrito_lector = true:suscripción , si
suscrito_lector=false:desuscripción
      if (suscrito_lector) { //cuando nos suscribimos, una copia del doc
se pide
        esperando_primer_frag = true;
        recibiendo_doc = true;
        fragmentos_recibidos = new Hashtable();
        nfrag_recibidos = 0;
        documento_local = new short[0];
          ultimos_cambios_global = new Hashtable();
      }
        return true;
    }else if (evt.id == Event.ACTION_EVENT && evt.target ==
suscribirse_escritor && ((suscrito_lector)&&(!recibiendo_doc))) {
      suscrito_escritor = !suscrito_escritor;
      fragmento_concedido = false;
        fragmento_local = new short[0];
      refresca_pantalla_fragmento();
      if(suscrito_escritor){
        suscribirse_escritor.setLabel("Desus. Escritor");
      }else{
        suscribirse_escritor.setLabel("Sus. Escritor");
      }
      send(new
WriterSCapsule(getWGroup(),target,getPort(),getPort(),suscrito_escritor))
;
      liberando_fragmento = !suscrito_escritor;
      //si suscrito_escritor = true:suscripción , si
suscrito_escritor=false:desuscripción
      if (suscrito_escritor) { //cuando nos suscribimos, se pide atencion
para pedir un fragmento
        esperando_para_pedir_frag = true;
      }
      estado.setText("...");
        return true;

    }else if (evt.id == Event.ACTION_EVENT && evt.target ==
Refrescar_global) {
      refresca_pantalla_global();
        return true;
    }else if (evt.id == Event.ACTION_EVENT && evt.target == FIN) {
      System.exit(0);
        return true;
    } else return super.handleEvent(evt);
  }

  public void run(){ //el thread se encarga de suscribirnos cuando sea
preciso
    while (true) {
      try {
      Thread.sleep(vg.tiempoPeticionDoc);
      } catch (InterruptedException e) {
      continue;
      }
      if ((esperando_primer_frag)&&(suscrito_lector)) // si no hemos
recibido el primer fragmento
      send(new ReaderSCapsule(Rgroup,target,port,port,target,
suscrito_lector));
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      else if (esperando_para_pedir_frag)
      send(new
WriterSCapsule(getWGroup(),target,getPort(),getPort(),suscrito_escritor))
;
    }
  }//run

  public void setArgs(KeyArgs k)
    throws Exception
  {
    k.merge(defaults);

    for (int i = 0; i < k.length(); i++) {
      if (k.key(i).equals("-target")) {
        target = NodeAddress.fromString(k.arg(i));
        k.strike(i);
      } else if (k.key(i).equals("-Rgroup")) {
      Rgroup = NodeAddress.fromString(k.arg(i));
      k.strike(i);
      } else if (k.key(i).equals("-Wgroup")) {
      Wgroup = NodeAddress.fromString(k.arg(i));
      k.strike(i);
      }
    }

    super.setArgs(k);
  }

  public void start()
    throws Exception
  {
    thisNode().register(new MCastECProtocol());

    recibiendo_doc = false;
    documento_local = new short[0];
    fragmento_local = new short[0];
    parar_envio_cambios = false;

    Panel interfaz = new Panel();
    interfaz.setLayout(new GridLayout(3,1));

    Panel norte = new Panel();
    norte.setLayout(new GridLayout(2,2));
    nombre = new Label
("Node:"+NodeAddress.toString(thisNode().getAddress()),Label.CENTER);
    norte.add(nombre);
    estado = new Label("De momento naita"+target,Label.CENTER);
    norte.add(estado);
    longitud_documento_label = new Label ("longitud doc global =
0",Label.CENTER);
    longitud_fragmento_label = new Label ("longitud fragmento =
0",Label.CENTER);
    norte.add(longitud_documento_label);
    norte.add(longitud_fragmento_label);
    interfaz.add(norte);

    Panel centro = new Panel();
    centro.setLayout(new GridLayout(1,3));
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    doc_local = new TextArea(" ",10
,20,TextArea.SCROLLBARS_VERTICAL_ONLY);
    doc_local.setEditable(false);
    Refrescar_global = new Button("Refresco!");
    frag_local = new TextArea(" ",10
,20,TextArea.SCROLLBARS_VERTICAL_ONLY);
    frag_local.setEditable(false);
    centro.add(doc_local);
    centro.add(Refrescar_global);
    centro.add(frag_local);
    interfaz.add(centro);

    Panel teclado = new Panel();
    teclado.setLayout(new GridLayout(3,4));
    suscribirse_lector = new Button("Sus. Lector");
    teclado.add(suscribirse_lector);
    teclado.add(new Label("Indice global",Label.CENTER));
    teclado.add(new Label("Indice local",Label.CENTER));
    teclado.add(new Label("Caracter",Label.CENTER));
    suscribirse_escritor = new Button("Sus. Escritor");
    teclado.add(suscribirse_escritor);
    indice_global = new TextField("0000");
    indice_local = new TextField("0000");
    caracter_nuevo = new TextField("0000");
    teclado.add(indice_global);
    teclado.add(indice_local);
    teclado.add(caracter_nuevo);
    FIN = new Button ("FIN");
    teclado.add(FIN);
    suprimir = new Button("Suprimir");
    insertar = new Button("Insertar");
    sustituir = new Button("Sustituir");
    teclado.add(suprimir);
    teclado.add(insertar);
    teclado.add(sustituir);
    interfaz.add(teclado);

    add("Center",interfaz);

    new Thread(this).start();
    new Thread(new EnviaCambios(this)).start();
    new Thread(new Liberar_fragmento(this)).start();

    pack();
    show();
  }

  public ReceiverApplication() {
    super();
  }
}
```

StopSendingChangesCapsule.java

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
/**                              StopSendingChangesCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                  Comenzado: 19/02/01
 */

package MCastECv2.apps;

import ants.*;
import ants.wrapper.*;

/** Cápsula de notificación de encolamiento de cambios de manera local
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class StopSendingChangesCapsule
  extends DataCapsule
{
  final private static byte[] MID =
findMID("apps.StopSendingChangesCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID =
findPID("apps.StopSendingChangesCapsule");
  protected byte[] pid() { return PID; }

  boolean parando; // '1' cuando se solicita parar y '0' cuando se
solicita reanudar la transmisión.
  int Wgroup;
  int target;

  public int length() {
    return super.length() + Xdr.BOOLEAN + Xdr.INT + Xdr.INT;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(parando);
    xdr.PUT(Wgroup);
    xdr.PUT(target);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    parando = xdr.BOOLEAN();
    Wgroup = xdr.INT();
    target = xdr.INT();

    return xdr;
  }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public boolean evaluate(Node n) {  //enrutamiento multicast desde el
Sender
    W_JAI m = (W_JAI)n.getCache().get(Wgroup, target);
    if (m != null) {
      if(m.ai1 == null)
        n.deliverToApp(this,dpt); //nodo Writer
      else   //nodo intermedio
        //se distribuye por multicast
        for (int i = 0; i < m.ai1.length; i++)
          n.routeForNode(this, m.ai1[i]);
    }
    return true;
  }

  public StopSendingChangesCapsule() { }

  public StopSendingChangesCapsule(int ga,int ta,short sa,short da,
boolean p) {
    super (sa,da,0,new ByteArray(0));

    Wgroup = ga;
    target = ta;
    parando = p;
  }

}
```

SendYourRequestCapsule.java

```
/**                              SendYourRequestCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                  Comenzado: 19/02/01
 */

package MCastECv2.apps;

import ants.*;
import ants.wrapper.*;

/** Cápsula de señalización a los aspirantes a Writer para que envíen sus
peticiones
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class SendYourRequestCapsule
  extends DataCapsule
{
  final private static byte[] MID =
findMID("apps.SendYourRequestCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID =
findPID("apps.SendYourRequestCapsule");
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  protected byte[] pid() { return PID; }

  public int length() {
    return super.length();
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    return xdr;
  }

  public boolean evaluate(Node n) {  //enrutamiento unicast hacia el
aspirante a Writer
    if ( n.getAddress() == getDst())
      return n.deliverToApp(this, dpt);
    else
      return n.routeForNode(this, getDst());

  }

  public SendYourRequestCapsule() { }

  public SendYourRequestCapsule(short sa, short da, int na) {
    super (sa,da,na,new ByteArray(0));
  }
}
```

RequestCapsule.java

```
/**                              RequestCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                  Comenzado: 19/02/01
 */

package MCastECv2.apps;

import ants.*;
import ants.wrapper.*;

/** Cápsula con petición de fragmento para editarlo, al Sender
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class RequestCapsule
  extends DataCapsule
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
{
  final private static byte[] MID = findMID("apps.RequestCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.RequestCapsule");
  protected byte[] pid() { return PID; }

  int indice;

  public int length() {
    return super.length() + Xdr.INT;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(indice);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    indice = xdr.INT();

    return xdr;
  }

  public boolean evaluate(Node n) {  //enrutamiento unicast hacia el
Sender
    n.log("Voy a pedir el indice "+indice+ "para el nodo "+getSrc());
    if ( n.getAddress() == getDst())
      return n.deliverToApp(this, dpt);
    else
      return n.routeForNode(this, getDst());
  }

  public RequestCapsule() { }

  public RequestCapsule(short sa, short da,int na, int i) {
    super (sa,da,na,new ByteArray(0));

    indice = i;
  }
}
```

GCACKCapsule.java

```
/**                                GCACKCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                Comenzado: 19/02/01
 */
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
package MCastECv2.apps;

import ants.*;
import ants.wrapper.*;
import MCastECv2.apps.variablesglobales;

/** Confirmación de la recepción de cápsula de cambios globales. Se
realiza fussing de estas cápsulas
 *  en los nodos intermedios para ahorrar ancho de banda
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class GCACKCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.GCACKCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.GCACKCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int sello; //entero enviado por el Sender como sello de fragmento de
doc

  int Rgroup;
  int target;
  int reverse;

  public int length() {
    int s = Xdr.INT + Xdr.INT + Xdr.INT + Xdr.INT;
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(Rgroup);
    xdr.PUT(target);
    xdr.PUT(reverse);

    xdr.PUT(sello);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    Rgroup = xdr.INT();
    target = xdr.INT();
    reverse = xdr.INT();

    sello = xdr.INT();
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    return xdr;
  }

  public boolean evaluate(Node n) {
    if (n.getAddress() == getSrc()){ //si estoy en el nodo origen
      reverse = n.getAddress();
      return n.routeForNode(this, target);
    }else{ //si no estoy en el nodo origen
      n.log("estoy aki, en otro sitio");
      W_JAIIAI m = (W_JAIIAI)n.getCache().get(Rgroup, target);
      if (m == null)
        return true; //no hay nodos suscritos
      else {
      int i;
      for(i=0;i<m.ai1.length;i++){ //buscamos la entrada de este nodo

System.out.println("m.ai1["+i+"]="+NodeAddress.toString(m.ai1[i]));
        if (m.ai1[i]==reverse) break;
      }
      m.ai2[i] = 1; //ponemos confirmación del nodo origen
        boolean recibido_de_todos = true;
        for (int j = 0; j<m.ai1.length; j++){
        if(m.ai2[j]==0){ //si hay alguno del que no se haya recibido
confirmación
          recibido_de_todos = false;
          break;
        }//if
        }//for
        if (recibido_de_todos) {
        m.i1 = 0; //esperando_confirmacion = 0
          for (int j = 0; j<m.ai1.length; j++)
          m.ai2[j] = 0; //ponemos a cero todas las confirmaciones
        return n.deliverToApp(this,dpt); //y le pasamos a la aplicación
la confirmación definitiva
      }
      return true;
      }
    }//
  }

  public GCACKCapsule() { }

  public GCACKCapsule(int ga, int ta, short sa, short da, int na, int s)
{
    super (sa,da,na,new ByteArray(0));
    Rgroup = ga;
    target = ta;
    reverse = 0;
    sello = s;
  }
}
```

Resultado_peticion.java

```
/**                                Resultado_peticion.java
 *
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                  Comenzado: 19/02/01
 */

package MCastECv2.apps;
import MCastECv2.apps.*;

/** Clase que almacena la información de la aceptación o no de un
apetición de pasarse a Writer.
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class Resultado_peticion {
   public boolean aceptacion; //si se acepta o deniega la peticion
   public int indice_inicial; //indice inicial del fragmento que puede
modificar
   public int no_indices; //indice final del fragmento que puede
modificar
   public int sello; //sello que identifica la resolución
   public Resultado_peticion(int s){
     aceptacion = false;
     indice_inicial = -1;
     no_indices = 0;
     sello = s;
   }

   public Resultado_peticion(int i_i, int n_i, int s){
     aceptacion = true;
     indice_inicial = i_i;
     no_indices = n_i;
     sello = s;
   }
}
```

ResolutionCapsule.java

```
/**                              ResolutionCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                  Comenzado: 19/02/01
 */

package MCastECv2.apps;

import ants.*;
import ants.wrapper.*;

/** Cápsula con resolución a la peticion de incorporarse como Writer
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
public class ResolutionCapsule
  extends DataCapsule
{
  final private static byte[] MID = findMID("apps.ResolutionCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.ResolutionCapsule");
  protected byte[] pid() { return PID; }

  boolean aceptacion; // '1' cuando se admite la petición y '0' cuando se
deniega
  int indice_inicial;
  int no_indices;
  int sello;

  public int length() {
    return super.length() + Xdr.BOOLEAN + Xdr.INT + Xdr.INT + Xdr.INT;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(aceptacion);
    xdr.PUT(indice_inicial);
    xdr.PUT(no_indices);
    xdr.PUT(sello);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    aceptacion = xdr.BOOLEAN();
    indice_inicial = xdr.INT();
    no_indices = xdr.INT();
    sello = xdr.INT();

    return xdr;
  }

  public boolean evaluate(Node n) {  //enrutamiento unicast hacia el
aspirante
    n.log("Voy hacia "+getDst()+" y la resolución es
ACEPTACION="+aceptacion);
    if(n.getAddress() == getDst())//si estoy en el nodo aspirante
      return n.deliverToApp(this,dpt);
    else {
      return n.routeForNode(this,getDst());
    }
  }

  public ResolutionCapsule() { }

  public ResolutionCapsule(short sa, short da,int aspirante,boolean
acept,int i_i,int n_i,int s) {
    super (sa,da,aspirante,new ByteArray(0));
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    aceptacion = acept;
    indice_inicial = i_i;
    no_indices = n_i;
    sello = s;
  }

}
```

ResolutionACKCapsule.java

```
/**                                          ResolutionACKCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                   Comenzado: 19/02/01
 */

package MCastECv2.apps;

import ants.*;
import ants.wrapper.*;

/** Cápsula de confirmación de la recepción la resolución de una petición
para ser Writer
 *  Y da entrada en las tablas intermedias a los nuevos Writer
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class ResolutionACKCapsule
  extends Capsule
{
  final private static byte[] MID = findMID("apps.ResolutionACKCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.ResolutionACKCapsule");
  protected byte[] pid() { return PID; }

  variablesglobales vg = new variablesglobales();

  int Wgroup;
  int target;
  short spt;
  short dpt;
  int sello; //sello de la ResolutionCapsule recibida
  int reverse;  //nodo anterior
  boolean aceptacion; //si se acepto la peticion de incorporacion como
Writer


  public int length() {
    return super.length() + Xdr.INT + Xdr.INT + Xdr.SHORT + Xdr.SHORT +
Xdr.INT + Xdr.INT + Xdr.BOOLEAN;
  }

  public Xdr encode() {
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    Xdr xdr = super.encode();

    xdr.PUT(Wgroup);
    xdr.PUT(target);
    xdr.PUT(spt);
    xdr.PUT(dpt);
    xdr.PUT(sello);
    xdr.PUT(reverse);
    xdr.PUT(aceptacion);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    Wgroup = xdr.INT();
    target = xdr.INT();
    spt = xdr.SHORT();
    dpt = xdr.SHORT();
    sello = xdr.INT();
    reverse = xdr.INT();
    aceptacion = xdr.BOOLEAN();

    return xdr;
  }


  public boolean evaluate(Node n) {
    n.log("Aceptacion = "+aceptacion);
    if(aceptacion){
      W_JAI m = (W_JAI)n.getCache().get(Wgroup, target);
      if (m == null) {
        m = new W_JAI();
        n.getCache().put(Wgroup, target, m,
vg.TIEMPO_MAXIMO_SUSCRIPCION);
      }
      if (reverse != 0) {//no estoy en el nodo origen
        boolean found = false;
        for (int i = 0; m.ai1 != null && i < m.ai1.length; i++) {
        if (m.ai1[i] == reverse) { found = true; break; }
        }
        if (!found) {
        int len = (m.ai1 != null) ? m.ai1.length : 0;
        int[] nn = new int[len+1];
        if (len > 0) System.arraycopy(m.ai1, 0, nn, 0, len);
        nn[len] = reverse;
        m.ai1 = nn;
        }
      }
    }//if(aceptacion)
    if(n.getAddress()!=target){//
      reverse = n.getAddress();
      return n.routeForNode(this,target);
    }else
      return n.deliverToApp(this,dpt);
  }
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  public ResolutionACKCapsule() { }

  public ResolutionACKCapsule(int Wga, int ta, short sa, short da, int
se, boolean acep) {
    Wgroup = Wga;
    target = ta;
    spt = sa;
    dpt = da;
    reverse = 0;
    sello = se;
    aceptacion = acep;
  }
}
```

WriterSACKCapsule.java

```
/**                                        WriterSACKCapsule.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa v2
 *                                    Comenzado: 19/02/01
 */

package MCastECv2.apps;

import ants.*;
import ants.wrapper.*;

/** Cápsula de confirmación de la recepción una liberacion de fragmento
 *
 *  @author Javier Jesús Sánchez Medina
 *
 */

public class WriterSACKCapsule
  extends Capsule
{
  final private static byte[] MID = findMID("apps.WriterSACKCapsule");
  protected byte[] mid() { return MID; }

  final private static byte[] PID = findPID("apps.WriterSACKCapsule");
  protected byte[] pid() { return PID; }

  short spt;
  short dpt;

  public int length() {
    int s = Xdr.SHORT + Xdr.SHORT;
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
    return super.length() + s;
  }


  public Xdr encode() {
    Xdr xdr = super.encode();

    xdr.PUT(spt);
    xdr.PUT(dpt);

    return xdr;
  }

  public Xdr decode() {
    Xdr xdr = super.decode();

    spt = xdr.SHORT();
    dpt = xdr.SHORT();

    return xdr;
  }

  public boolean evaluate(Node n) {  //enrutamiento unicast hacia el ex-
writer
    if ( n.getAddress() == getDst())
      return n.deliverToApp(this, dpt);
    else
      return n.routeForNode(this, getDst());

  }

  public WriterSACKCapsule() { }

  public WriterSACKCapsule(int fw, short sa, short da) {
    setDst(fw);
    spt = sa;
    dpt = da;
  }
}
```

Tiempo.java

```
package MCastECv2.apps;
/** Tiempo
*/
public class Tiempo {
   private long t;
   long get (){
      return t;
   }
   Tiempo (long time){
      this.t = time;
   }//constructor
}//Tiempo
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

MCastECProtocol.java

```
/**                              MCastECProtocol.java
 *
 *          Ejemplo de protocolo multicast de edición cooperativa
 *                                Comenzado: 19/02/01
 */

package MCastECv2.apps;

import ants.*;
/**
 *
 *            protocolo multicast de procesamiento de texto distribuido
v2
 *                        @author Javier Jesús Sánchez Medina
 */

public class MCastECProtocol extends Protocol {

  public MCastECProtocol() throws Exception {
    startProtocolDefn();

    startGroupDefn();
    addCapsule("apps.ChangesCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.GChangesCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.GCACKCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.ReaderSCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.WriterSCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.WriterSACKCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.DocCapsule");
    endGroupDefn();

    startGroupDefn();
    addCapsule("apps.DocACKCapsule");
    endGroupDefn();

    startGroupDefn();
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
        addCapsule("apps.SendYourRequestCapsule");
        endGroupDefn();

        startGroupDefn();
        addCapsule("apps.RequestCapsule");
        endGroupDefn();

        startGroupDefn();
        addCapsule("apps.ResolutionCapsule");
        endGroupDefn();

        startGroupDefn();
        addCapsule("apps.ResolutionACKCapsule");
        endGroupDefn();

        startGroupDefn();
        addCapsule("apps.StopSendingChangesCapsule");
        endGroupDefn();

        endProtocolDefn();
    }
}
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

ANEXO II. Configuraciones de las Pruebas

En este anexo se muestran los ficheros de configuración de todas las pruebas realizadas.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

MatrizXMatriz v1

Configuración A

```
#                    Pruebas_MxM_configuracion_A.config
#
#        Pruebas con el protocolo de producto distribuido de Matriz X
Matriz v1
#                    Autor: Javier Jes£s S nchez Medina
#
#

#Maquina A-wireless
node 0.0.0.1 -routes Pruebas_MxM_configuracion_A.routes
channel 0.0.0.1 192.168.1.2:8001 -log 255
application 0.0.0.1 apps.MApplication -lado 6 -target 0.0.0.4 -target
0.0.0.5 -target 0.0.0.6 -target 0.0.0.7
manager 0.0.0.1 -gui true -log 255

#pcmovil1
node 0.0.0.2 -routes Pruebas_MxM_configuracion_A.routes
channel 0.0.0.2 192.168.1.3:8002 -log 255
application 0.0.0.2 apps.CApplication -lado 6
manager 0.0.0.2 -gui true -log 255

#pcmovil1
node 0.0.0.3 -routes Pruebas_MxM_configuracion_A.routes
channel 0.0.0.3 192.168.1.3:8003 -log 255
application 0.0.0.3 apps.CApplication -lado 6
manager 0.0.0.3 -gui true -log 255

#Maquina A-wireless
node 0.0.0.4 -routes Pruebas_MxM_configuracion_A.routes
channel 0.0.0.4 192.168.1.2:8004 -log 255
application 0.0.0.4 apps.CApplication -lado 6
manager 0.0.0.4 -gui true -log 255

#Maquina A-wireless
node 0.0.0.5 -routes Pruebas_MxM_configuracion_A.routes
channel 0.0.0.5 192.168.1.2:8005 -log 255
application 0.0.0.5 apps.CApplication -lado 6
manager 0.0.0.5 -gui true -log 255

#pcmovil2
node 0.0.0.6 -routes Pruebas_MxM_configuracion_A.routes
channel 0.0.0.6 192.168.1.4:8006 -log 255
application 0.0.0.6 apps.CApplication -lado 6
manager 0.0.0.6 -gui true -log 255

#pcmovil2
node 0.0.0.7 -routes Pruebas_MxM_configuracion_A.routes
channel 0.0.0.7 192.168.1.4:8007 -log 255
application 0.0.0.7 apps.CApplication -lado 6
manager 0.0.0.7 -gui true -log 255
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.1 0.0.0.4
connect 0.0.0.1 0.0.0.5
connect 0.0.0.1 0.0.0.6
connect 0.0.0.1 0.0.0.7
```

Configuración B

```
#                       Pruebas_MxM_configuracion_B.config
#
#        Pruebas con el protocolo de producto distribuido de Matriz X
Matriz v1
#                       Autor: Javier Jes£s S nchez Medina
#
#

#Maquina A-wired
node 0.0.0.1 -routes Pruebas_MxM_configuracion_B.routes
channel 0.0.0.1 192.168.0.4:8001 -log 255
application 0.0.0.1 apps.MApplication -lado 15 -target 0.0.0.2 -target
0.0.0.3 -target 0.0.0.4 -target 0.0.0.5 -target 0.0.0.6 -target 0.0.0.7
manager 0.0.0.1 -gui true -log 255

#Maquina B
node 0.0.0.2 -routes Pruebas_MxM_configuracion_B.routes
channel 0.0.0.2 192.168.0.2:8002 -log 255
application 0.0.0.2 apps.CApplication -lado 15
manager 0.0.0.2 -gui true -log 255

#Maquina B
node 0.0.0.3 -routes Pruebas_MxM_configuracion_B.routes
channel 0.0.0.3 192.168.0.2:8003 -log 255
application 0.0.0.3 apps.CApplication -lado 15
manager 0.0.0.3 -gui true -log 255

#Maquina A-wired
node 0.0.0.4 -routes Pruebas_MxM_configuracion_B.routes
channel 0.0.0.4 192.168.0.4:8004 -log 255
application 0.0.0.4 apps.CApplication -lado 15
manager 0.0.0.4 -gui true -log 255

#Maquina A-wired
node 0.0.0.5 -routes Pruebas_MxM_configuracion_B.routes
channel 0.0.0.5 192.168.0.4:8005 -log 255
application 0.0.0.5 apps.CApplication -lado 15
manager 0.0.0.5 -gui true -log 255

#Maquina C
node 0.0.0.6 -routes Pruebas_MxM_configuracion_B.routes
channel 0.0.0.6 192.168.0.3:8006 -log 255
application 0.0.0.6 apps.CApplication -lado 15
manager 0.0.0.6 -gui true -log 255

#Maquina C
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
node 0.0.0.7 -routes Pruebas_MxM_configuracion_B.routes
channel 0.0.0.7 192.168.0.3:8007 -log 255
application 0.0.0.7 apps.CApplication -lado 15
manager 0.0.0.7 -gui true -log 255


connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.1 0.0.0.4
connect 0.0.0.1 0.0.0.5
connect 0.0.0.1 0.0.0.6
connect 0.0.0.1 0.0.0.7
```

Configuración C

```
#                                Pruebas_MxM_configuracion_C.config
#
#        Pruebas con el protocolo de producto distribuido de Matriz X
Matriz v1
#                      Autor: Javier Jes£s S nchez Medina
#
#

#Maquina A
node 0.0.0.1 -routes Pruebas_MxM_configuracion_C.routes
channel 0.0.0.1 192.168.0.4:8001 -log 255
application 0.0.0.1 apps.MApplication -lado 19 -target 0.0.0.2 -target
0.0.0.3 -target 0.0.0.4 -target 0.0.0.5 -target 0.0.0.6 -target 0.0.0.7
manager 0.0.0.1 -gui true -log 255

#Maquina A
node 0.0.0.2 -routes Pruebas_MxM_configuracion_C.routes
channel 0.0.0.2 192.168.0.4:8002 -log 255
application 0.0.0.2 apps.CApplication -lado 19
manager 0.0.0.2 -gui true -log 255

#Maquina A
node 0.0.0.3 -routes Pruebas_MxM_configuracion_C.routes
channel 0.0.0.3 192.168.0.4:8003 -log 255
application 0.0.0.3 apps.CApplication -lado 19
manager 0.0.0.3 -gui true -log 255

#Maquina A
node 0.0.0.4 -routes Pruebas_MxM_configuracion_C.routes
channel 0.0.0.4 192.168.0.4:8004 -log 255
application 0.0.0.4 apps.CApplication -lado 19
manager 0.0.0.4 -gui true -log 255

#Maquina A
node 0.0.0.5 -routes Pruebas_MxM_configuracion_C.routes
channel 0.0.0.5 192.168.0.4:8005 -log 255
application 0.0.0.5 apps.CApplication -lado 19
manager 0.0.0.5 -gui true -log 255

#Maquina A
node 0.0.0.6 -routes Pruebas_MxM_configuracion_C.routes
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
channel 0.0.0.6 192.168.0.4:8006 -log 255
application 0.0.0.6 apps.CApplication -lado 19
manager 0.0.0.6 -gui true -log 255

#Maquina A
node 0.0.0.7 -routes Pruebas_MxM_configuracion_C.routes
channel 0.0.0.7 192.168.0.4:8007 -log 255
application 0.0.0.7 apps.CApplication -lado 19
manager 0.0.0.7 -gui true -log 255


connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.1 0.0.0.4
connect 0.0.0.1 0.0.0.5
connect 0.0.0.1 0.0.0.6
connect 0.0.0.1 0.0.0.7
```

PingBroadcast v1

Configuración A

```
#                                        Pruebas_PBv1_configuracion_A.config
#
#                                    Pruebas con el protocolo de ping
Broadcast v2
#                                         Autor: Javier Jes£s S nchez Medina

node 0.0.0.1 -routes Pruebas_PBv1_configuracion_A.routes -log 255
channel 0.0.0.1 localhost:8001 -log 255
application 0.0.0.1 apps.PingBApplication
manager 0.0.0.1 -gui true -log 255

node 0.0.0.2 -routes Pruebas_PBv1_configuracion_A.routes -log 255
channel 0.0.0.2 localhost:8002 -log 255
application 0.0.0.2 apps.PingBApplication
manager 0.0.0.2 -gui true -log 255

node 0.0.0.3 -routes Pruebas_PBv1_configuracion_A.routes -log 255
channel 0.0.0.3 localhost:8003 -log 255
application 0.0.0.3 apps.PingBApplication
manager 0.0.0.3 -gui true -log 255

connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.2 0.0.0.3
```

Configuración B

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
#                                        Pruebas_PBv1_configuracion_B.config
#
#                                    Pruebas con el protocolo de ping
Broadcast v2
#                                          Autor: Javier Jes£s S nchez Medina

node 0.0.0.1 -routes Pruebas_PBv1_configuracion_B.routes -log 255
channel 0.0.0.1 localhost:8001 -log 255
application 0.0.0.1 apps.PingBApplication
manager 0.0.0.1 -gui true -log 255

node 0.0.0.2 -routes Pruebas_PBv1_configuracion_B.routes -log 255
channel 0.0.0.2 localhost:8002 -log 255
application 0.0.0.2 apps.PingBApplication
manager 0.0.0.2 -gui true -log 255

node 0.0.0.3 -routes Pruebas_PBv1_configuracion_B.routes -log 255
channel 0.0.0.3 localhost:8003 -log 255
application 0.0.0.3 apps.PingBApplication
manager 0.0.0.3 -gui true -log 255

node 0.0.0.4 -routes Pruebas_PBv1_configuracion_B.routes -log 255
channel 0.0.0.4 localhost:8004 -log 255
application 0.0.0.4 apps.PingBApplication
manager 0.0.0.4 -gui true -log 255

node 0.0.0.5 -routes Pruebas_PBv1_configuracion_B.routes -log 255
channel 0.0.0.5 localhost:8005 -log 255
application 0.0.0.5 apps.PingBApplication
manager 0.0.0.5 -gui true -log 255

connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.1 0.0.0.4
connect 0.0.0.1 0.0.0.5
connect 0.0.0.2 0.0.0.3
connect 0.0.0.2 0.0.0.4
connect 0.0.0.2 0.0.0.5
connect 0.0.0.3 0.0.0.4
connect 0.0.0.3 0.0.0.5
connect 0.0.0.4 0.0.0.5
```

Configuración C

```
#                                        Pruebas_PBv1_configuracion_C.config
#
#                                    Pruebas con el protocolo de ping
Broadcast v2
#                                          Autor: Javier Jesœs Sÿnchez Medina

node 0.0.0.1 -routes Pruebas_PBv1_configuracion_C.routes -log 255
channel 0.0.0.1 localhost:8001 -log 255
application 0.0.0.1 apps.PingBApplication
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
manager 0.0.0.1 -gui true -log 255

node 0.0.0.2 -routes Pruebas_PBv1_configuracion_C.routes -log 255
channel 0.0.0.2 localhost:8002 -log 255
application 0.0.0.2 apps.PingBApplication
manager 0.0.0.2 -gui true -log 255

node 0.0.0.3 -routes Pruebas_PBv1_configuracion_C.routes -log 255
channel 0.0.0.3 localhost:8003 -log 255
application 0.0.0.3 apps.PingBApplication
manager 0.0.0.3 -gui true -log 255

node 0.0.0.4 -routes Pruebas_PBv1_configuracion_C.routes -log 255
channel 0.0.0.4 localhost:8004 -log 255
application 0.0.0.4 apps.PingBApplication
manager 0.0.0.4 -gui true -log 255

node 0.0.0.5 -routes Pruebas_PBv1_configuracion_C.routes -log 255
channel 0.0.0.5 localhost:8005 -log 255
application 0.0.0.5 apps.PingBApplication
manager 0.0.0.5 -gui true -log 255

node 0.0.0.6 -routes Pruebas_PBv1_configuracion_C.routes -log 255
channel 0.0.0.6 localhost:8006 -log 255
application 0.0.0.6 apps.PingBApplication
manager 0.0.0.6 -gui true -log 255

connect 0.0.0.1 0.0.0.2
connect 0.0.0.2 0.0.0.3
connect 0.0.0.3 0.0.0.4
connect 0.0.0.3 0.0.0.6
connect 0.0.0.4 0.0.0.5
```

Configuración D

```
#                                    Pruebas_PBv1_configuracion_D.config
#
#                                Pruebas con el protocolo de ping
Broadcast v2
#                                     Autor: Javier Jesos S~nchez Medina

node 0.0.0.1 -routes Pruebas_PBv1_configuracion_D.routes -log 255
channel 0.0.0.1 localhost:8001 -log 255
application 0.0.0.1 apps.PingBApplication
manager 0.0.0.1 -gui true -log 255

node 0.0.0.2 -routes Pruebas_PBv1_configuracion_D.routes -log 255
channel 0.0.0.2 localhost:8002 -log 255
application 0.0.0.2 apps.PingBApplication
manager 0.0.0.2 -gui true -log 255

node 0.0.0.3 -routes Pruebas_PBv1_configuracion_D.routes -log 255
channel 0.0.0.3 localhost:8003 -log 255
application 0.0.0.3 apps.PingBApplication
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
manager 0.0.0.3 -gui true -log 255

node 0.0.0.4 -routes Pruebas_PBv1_configuracion_D.routes -log 255
channel 0.0.0.4 localhost:8004 -log 255
application 0.0.0.4 apps.PingBApplication
manager 0.0.0.4 -gui true -log 255

node 0.0.0.5 -routes Pruebas_PBv1_configuracion_D.routes -log 255
channel 0.0.0.5 localhost:8005 -log 255
application 0.0.0.5 apps.PingBApplication
manager 0.0.0.5 -gui true -log 255

node 0.0.0.6 -routes Pruebas_PBv1_configuracion_D.routes -log 255
channel 0.0.0.6 localhost:8006 -log 255
application 0.0.0.6 apps.PingBApplication
manager 0.0.0.6 -gui true -log 255

node 0.0.0.7 -routes Pruebas_PBv1_configuracion_D.routes -log 255
channel 0.0.0.7 localhost:8007 -log 255
application 0.0.0.7 apps.PingBApplication
manager 0.0.0.7 -gui true -log 255

node 0.0.0.8 -routes Pruebas_PBv1_configuracion_D.routes -log 255
channel 0.0.0.8 localhost:8008 -log 255
application 0.0.0.8 apps.PingBApplication
manager 0.0.0.8 -gui true -log 255

node 0.0.0.9 -routes Pruebas_PBv1_configuracion_D.routes -log 255
channel 0.0.0.9 localhost:8009 -log 255
application 0.0.0.9 apps.PingBApplication
manager 0.0.0.9 -gui true -log 255

connect 0.0.0.1 0.0.0.4
connect 0.0.0.2 0.0.0.4
connect 0.0.0.3 0.0.0.4
connect 0.0.0.4 0.0.0.5
connect 0.0.0.5 0.0.0.6
connect 0.0.0.5 0.0.0.9
connect 0.0.0.6 0.0.0.7
connect 0.0.0.6 0.0.0.8
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

PingBroadcast v2

Configuración A

```
#                                        Pruebas_PBv2_configuracion_A.config
#
#                                    Pruebas con el protocolo de ping
Broadcast v2
#                                          Autor: Javier Jes£s S nchez Medina

#Maquina C
node 0.0.0.1 -routes Pruebas_PBv2_configuracion_A.routes -log 255
channel 0.0.0.1 192.168.0.3:8001 -log 255
application 0.0.0.1 apps.PingBApplication
manager 0.0.0.1 -gui true -log 255

#Maquina B
node 0.0.0.2 -routes Pruebas_PBv2_configuracion_A.routes -log 255
channel 0.0.0.2 192.168.0.2:8002 -log 255
application 0.0.0.2 apps.PingBApplication
manager 0.0.0.2 -gui true -log 255

#Maquina A-wired
node 0.0.0.3 -routes Pruebas_PBv2_configuracion_A.routes -log 255
channel 0.0.0.3 192.168.0.4:8003 -log 255
application 0.0.0.3 apps.PingBApplication
manager 0.0.0.3 -gui true -log 255

#Maquina A-wireless
node 0.0.0.4 -routes Pruebas_PBv2_configuracion_A.routes -log 255
channel 0.0.0.4 192.168.1.2:8004 -log 255
application 0.0.0.4 apps.PingBApplication
manager 0.0.0.4 -gui true -log 255

#pcmovil1
node 0.0.0.5 -routes Pruebas_PBv2_configuracion_A.routes -log 255
channel 0.0.0.5 192.168.1.3:8005 -log 255
application 0.0.0.5 apps.PingBApplication
manager 0.0.0.5 -gui true -log 255

#pcmovil2
node 0.0.0.6 -routes Pruebas_PBv2_cconfiguracion_A.routes -log 255
channel 0.0.0.6 192.168.1.4:8006 -log 255
application 0.0.0.6 apps.PingBApplication
manager 0.0.0.6 -gui true -log 255

connect 0.0.0.1 0.0.0.3
connect 0.0.0.2 0.0.0.3
connect 0.0.0.3 0.0.0.4
connect 0.0.0.4 0.0.0.5
connect 0.0.0.4 0.0.0.6
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

E.T.S.I.T. BIBLIOTECA DE TELECOMUNICACIONES

MulticastRT v1

Configuración A

```
#
Pruebas_MCRTv1_configuracion_A.config
#
#                                   Pruebas con el protocolo de ping
Broadcast v2
#                                        Autor: Javier Jesos S~nchez Medina

node 0.0.0.1 -routes Pruebas_MCRTv1_configuracion_A.routes -log 255
channel 0.0.0.1 localhost:8001 -log 255
application 0.0.0.1 apps.SenderApplication
manager 0.0.0.1 -gui true -log 255

node 0.0.0.2 -routes Pruebas_MCRTv1_configuracion_A.routes -log 255
channel 0.0.0.2 localhost:8002 -log 255
application 0.0.0.2 apps.NodeApplication
manager 0.0.0.2 -gui true -log 255

node 0.0.0.3 -routes Pruebas_MCRTv1_configuracion_A.routes -log 255
channel 0.0.0.3 localhost:8003 -log 255
application 0.0.0.3 apps.ReceiverApplication
manager 0.0.0.3 -gui true -log 255

node 0.0.0.4 -routes Pruebas_MCRTv1_configuracion_A.routes -log 255
channel 0.0.0.4 localhost:8004 -log 255
application 0.0.0.4 apps.NodeApplication
manager 0.0.0.4 -gui true -log 255

node 0.0.0.5 -routes Pruebas_MCRTv1_configuracion_A.routes -log 255
channel 0.0.0.5 localhost:8005 -log 255
application 0.0.0.5 apps.ReceiverApplication
manager 0.0.0.5 -gui true -log 255

node 0.0.0.6 -routes Pruebas_MCRTv1_configuracion_A.routes -log 255
channel 0.0.0.6 localhost:8006 -log 255
application 0.0.0.6 apps.ReceiverApplication
manager 0.0.0.6 -gui true -log 255

node 0.0.0.7 -routes Pruebas_MCRTv1_configuracion_A.routes -log 255
channel 0.0.0.7 localhost:8007 -log 255
application 0.0.0.7 apps.PingBApplication
manager 0.0.0.7 -gui true -log 255

node 0.0.0.8 -routes Pruebas_MCRTv1_configuracion_A.routes -log 255
channel 0.0.0.8 localhost:8008 -log 255
application 0.0.0.8 apps.PingBApplication
manager 0.0.0.8 -gui true -log 255

node 0.0.0.9 -routes Pruebas_MCRTv1_configuracion_A.routes -log 255
channel 0.0.0.9 localhost:8009 -log 255
application 0.0.0.9 apps.PingBApplication
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
manager 0.0.0.9 -gui true -log 255

connect 0.0.0.1 0.0.0.4
connect 0.0.0.2 0.0.0.4
connect 0.0.0.3 0.0.0.4
connect 0.0.0.4 0.0.0.5
connect 0.0.0.5 0.0.0.6
connect 0.0.0.5 0.0.0.9
connect 0.0.0.6 0.0.0.7
connect 0.0.0.6 0.0.0.8
```

Configuración B

```
#
Pruebas_MCRTv1_configuracion_B.config
#
#                                   Pruebas con el protocolo de ping
Broadcast v2
#                                        Autor: Javier Jesos S~nchez Medina

node 0.0.0.1 -routes Pruebas_MCRTv1_configuracion_B.routes -log 255
channel 0.0.0.1 localhost:8001 -log 255
application 0.0.0.1 apps.SenderApplication
manager 0.0.0.1 -gui true -log 255

node 0.0.0.2 -routes Pruebas_MCRTv1_configuracion_B.routes -log 255
channel 0.0.0.2 localhost:8002 -log 255
application 0.0.0.2 apps.NodeApplication
manager 0.0.0.2 -gui true -log 255

node 0.0.0.3 -routes Pruebas_MCRTv1_configuracion_B.routes -log 255
channel 0.0.0.3 localhost:8003 -log 255
application 0.0.0.3 apps.ReceiverApplication
manager 0.0.0.3 -gui true -log 255

node 0.0.0.4 -routes Pruebas_MCRTv1_configuracion_B.routes -log 255
channel 0.0.0.4 localhost:8004 -log 255
application 0.0.0.4 apps.NodeApplication
manager 0.0.0.4 -gui true -log 255

node 0.0.0.5 -routes Pruebas_MCRTv1_configuracion_B.routes -log 255
channel 0.0.0.5 localhost:8005 -log 255
application 0.0.0.5 apps.ReceiverApplication
manager 0.0.0.5 -gui true -log 255

node 0.0.0.6 -routes Pruebas_MCRTv1_configuracion_B.routes -log 255
channel 0.0.0.6 localhost:8006 -log 255
application 0.0.0.6 apps.NodeApplication
manager 0.0.0.6 -gui true -log 255

node 0.0.0.7 -routes Pruebas_MCRTv1_configuracion_B.routes -log 255
channel 0.0.0.7 localhost:8007 -log 255
application 0.0.0.7 apps.ReceiverApplication
manager 0.0.0.7 -gui true -log 255
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
node 0.0.0.8 -routes Pruebas_MCRTv1_configuracion_B.routes -log 255
channel 0.0.0.8 localhost:8008 -log 255
application 0.0.0.8 apps.NodeApplication
manager 0.0.0.8 -gui true -log 255

node 0.0.0.9 -routes Pruebas_MCRTv1_configuracion_B.routes -log 255
channel 0.0.0.9 localhost:8009 -log 255
application 0.0.0.9 apps.ReceiverApplication
manager 0.0.0.9 -gui true -log 255

connect 0.0.0.1 0.0.0.2
connect 0.0.0.2 0.0.0.4
connect 0.0.0.2 0.0.0.3
connect 0.0.0.4 0.0.0.6
connect 0.0.0.4 0.0.0.5
connect 0.0.0.6 0.0.0.7
connect 0.0.0.6 0.0.0.8
connect 0.0.0.8 0.0.0.9
```

Configuración C

```
#
Pruebas_MCRTv1_configuracion_C.config
#
#                                  Pruebas con el protocolo de ping
Broadcast v2
#                                        Autor: Javier Jesos S~nchez Medina

node 0.0.0.1 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.1 localhost:8001 -log 255
application 0.0.0.1 apps.SenderApplication
manager 0.0.0.1 -gui true -log 255

node 0.0.0.2 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.2 localhost:8002 -log 255
application 0.0.0.2 apps.NodeApplication
manager 0.0.0.2 -gui true -log 255

node 0.0.0.3 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.3 localhost:8003 -log 255
application 0.0.0.3 apps.NodeApplication
manager 0.0.0.3 -gui true -log 255

node 0.0.0.4 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.4 localhost:8004 -log 255
application 0.0.0.4 apps.ReceiverApplication
manager 0.0.0.4 -gui true -log 255

node 0.0.0.5 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.5 localhost:8005 -log 255
application 0.0.0.5 apps.ReceiverApplication
manager 0.0.0.5 -gui true -log 255
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
node 0.0.0.6 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.6 localhost:8006 -log 255
application 0.0.0.6 apps.NodeApplication
manager 0.0.0.6 -gui true -log 255

node 0.0.0.7 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.7 localhost:8007 -log 255
application 0.0.0.7 apps.NodeApplication
manager 0.0.0.7 -gui true -log 255

node 0.0.0.8 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.8 localhost:8008 -log 255
application 0.0.0.8 apps.NodeApplication
manager 0.0.0.8 -gui true -log 255

node 0.0.0.9 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.9 localhost:8009 -log 255
application 0.0.0.9 apps.NodeApplication
manager 0.0.0.9 -gui true -log 255

node 0.0.0.10 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.10 localhost:8010 -log 255
application 0.0.0.10 apps.ReceiverApplication
manager 0.0.0.10 -gui true -log 255

node 0.0.0.11 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.11 localhost:8011 -log 255
application 0.0.0.11 apps.ReceiverApplication
manager 0.0.0.11 -gui true -log 255

node 0.0.0.12 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.12 localhost:8012 -log 255
application 0.0.0.12 apps.ReceiverApplication
manager 0.0.0.12 -gui true -log 255

node 0.0.0.13 -routes Pruebas_MCRTv1_configuracion_C.routes -log 255
channel 0.0.0.13 localhost:8013 -log 255
application 0.0.0.13 apps.ReceiverApplication
manager 0.0.0.13 -gui true -log 255

connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.2 0.0.0.4
connect 0.0.0.3 0.0.0.5
connect 0.0.0.2 0.0.0.6
connect 0.0.0.3 0.0.0.8
connect 0.0.0.3 0.0.0.9
connect 0.0.0.6 0.0.0.10
connect 0.0.0.7 0.0.0.11
connect 0.0.0.8 0.0.0.12
connect 0.0.0.9 0.0.0.13
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

MulticastRT v2

Configuración A

```
#                    Pruebas_MCRTv2_configuracion_A.config
#
#            Ejemplo de protocolo multicast orientado a tiempo real
#                     Autor: Javier Jes£s S nchez Medina
#                            Comenzado: 19/10/00
#

#Maquina A-wired
node 0.0.0.1 -routes Pruebas_MCRTv2_configuracion_A.routes
channel 0.0.0.1 192.168.0.4:8001 -log 255
application 0.0.0.1 apps.SenderApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.1 -gui true -log 255

#Maquina B
node 0.0.0.2 -routes Pruebas_MCRTv2_configuracion_A.routes
channel 0.0.0.2 192.168.0.2:8002 -log 255
application 0.0.0.2 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.2 -gui true -log 255

#Maquina A-wireless
node 0.0.0.3 -routes Pruebas_MCRTv2_configuracion_A.routes
channel 0.0.0.3 192.168.1.2:8003 -log 255
application 0.0.0.3 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.3 -gui true -log 255

#Maquina C
node 0.0.0.4 -routes Pruebas_MCRTv2_configuracion_A.routes
channel 0.0.0.4 192.168.0.3:8004 -log 255
application 0.0.0.4 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.4 -gui true -log 255

#Maquina B
node 0.0.0.5 -routes Pruebas_MCRTv2_configuracion_A.routes
channel 0.0.0.5 192.168.0.2:8005 -log 255
application 0.0.0.5 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.5 -gui true -log 255

#pcmovil2
node 0.0.0.6 -routes Pruebas_MCRTv2_configuracion_A.routes
channel 0.0.0.6 192.168.1.4:8006 -log 255
application 0.0.0.6 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.6 -gui true -log 255

#pcmovil1
node 0.0.0.7 -routes Pruebas_MCRTv2_configuracion_A.routes
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
channel 0.0.0.7 192.168.1.3:8007 -log 255
application 0.0.0.7 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.7 -gui true -log 255

connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.2 0.0.0.4
connect 0.0.0.2 0.0.0.5
connect 0.0.0.3 0.0.0.6
connect 0.0.0.3 0.0.0.7
```

Configuración B

```
#                      Pruebas_MCRTv2_configuracion_B.config
#
#            Ejemplo de protocolo multicast orientado a tiempo real
#                     Autor: Javier Jes£s S nchez Medina
#                            Comenzado: 19/10/00
#

#Maquina A-wired
node 0.0.0.1 -routes Pruebas_MCRTv2_configuracion_B.routes
channel 0.0.0.1 192.168.0.4:8001 -log 255
application 0.0.0.1 apps.SenderApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.1 -gui true -log 255

#Maquina B
node 0.0.0.2 -routes Pruebas_MCRTv2_configuracion_B.routes
channel 0.0.0.2 192.168.0.2:8002 -log 255
application 0.0.0.2 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.2 -gui true -log 255

#Maquina A-wireless
node 0.0.0.3 -routes Pruebas_MCRTv2_configuracion_B.routes
channel 0.0.0.3 192.168.1.2:8003 -log 255
application 0.0.0.3 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.3 -gui true -log 255

#pcmovil1
node 0.0.0.4 -routes Pruebas_MCRTv2_configuracion_B.routes
channel 0.0.0.4 192.168.1.3:8004 -log 255
application 0.0.0.4 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.4 -gui true -log 255

#Maquina C
node 0.0.0.5 -routes Pruebas_MCRTv2_configuracion_B.routes
channel 0.0.0.5 192.168.0.3:8005 -log 255
application 0.0.0.5 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.5 -gui true -log 255
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
#Maquina B
node 0.0.0.6 -routes Pruebas_MCRTv2_configuracion_B.routes
channel 0.0.0.6 192.168.0.2:8006 -log 255
application 0.0.0.6 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.6 -gui true -log 255

#pcmovil2
node 0.0.0.7 -routes Pruebas_MCRTv2_configuracion_B.routes
channel 0.0.0.7 192.168.1.4:8007 -log 255
application 0.0.0.7 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.7 -gui true -log 255

node 0.0.0.8 -routes Pruebas_MCRTv2_configuracion_B.routes
channel 0.0.0.8 192.168.1.4:8008 -log 255
application 0.0.0.8 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.8 -gui true -log 255

connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.3 0.0.0.4
connect 0.0.0.2 0.0.0.5
connect 0.0.0.2 0.0.0.6
connect 0.0.0.4 0.0.0.7
connect 0.0.0.4 0.0.0.8
```

Configuración C

```
#                    Pruebas_MCRTv2_configuracion_C.config
#
#            Ejemplo de protocolo multicast orientado a tiempo real
#                     Autor: Javier Jes£s S nchez Medina
#                             Comenzado: 19/10/00
#

#pcmovil1
node 0.0.0.1 -routes Pruebas_MCRTv2_configuracion_C.routes
channel 0.0.0.1 192.168.1.3:8001 -log 255
application 0.0.0.1 apps.SenderApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.1 -gui true -log 255

#Maquina A-wireless
node 0.0.0.2 -routes Pruebas_MCRTv2_configuracion_C.routes
channel 0.0.0.2 192.168.1.2:8002 -log 255
application 0.0.0.2 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.2 -gui true -log 255

node 0.0.0.5 -routes Pruebas_MCRTv2_configuracion_C.routes
channel 0.0.0.5 192.168.1.2:8005 -log 255
application 0.0.0.5 apps.NodeApplication -group 255.255.0.0 -target
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
0.0.0.1
manager 0.0.0.5 -gui true -log 255

#pcmovil2
node 0.0.0.3 -routes Pruebas_MCRTv2_configuracion_C.routes
channel 0.0.0.3 192.168.1.4:8003 -log 255
application 0.0.0.3 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.3 -gui true -log 255

node 0.0.0.6 -routes Pruebas_MCRTv2_configuracion_C.routes
channel 0.0.0.6 192.168.1.4:8006 -log 255
application 0.0.0.6 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.6 -gui true -log 255

#Maquina A-wired
node 0.0.0.4 -routes Pruebas_MCRTv2_configuracion_C.routes
channel 0.0.0.4 192.168.0.4:8004 -log 255
application 0.0.0.4 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.4 -gui true -log 255

node 0.0.0.9 -routes Pruebas_MCRTv2_configuracion_C.routes
channel 0.0.0.9 192.168.0.4:8009 -log 255
application 0.0.0.9 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.9 -gui true -log 255

#Maquina B
node 0.0.0.8 -routes Pruebas_MCRTv2_configuracion_C.routes
channel 0.0.0.8 192.168.0.2:8008 -log 255
application 0.0.0.8 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.8 -gui true -log 255

node 0.0.0.11 -routes Pruebas_MCRTv2_configuracion_C.routes
channel 0.0.0.11 192.168.0.2:8011 -log 255
application 0.0.0.11 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.11 -gui true -log 255

#Maquina C
node 0.0.0.7 -routes Pruebas_MCRTv2_configuracion_C.routes
channel 0.0.0.7 192.168.0.3:8007 -log 255
application 0.0.0.7 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.7 -gui true -log 255

node 0.0.0.10 -routes Pruebas_MCRTv2_cconfiguracion_C.routes
channel 0.0.0.10 192.168.0.3:8010 -log 255
application 0.0.0.10 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.10 -gui true -log 255

connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.2 0.0.0.4
connect 0.0.0.3 0.0.0.5
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
connect 0.0.0.3 0.0.0.6
connect 0.0.0.4 0.0.0.7
connect 0.0.0.4 0.0.0.8
connect 0.0.0.5 0.0.0.9
connect 0.0.0.9 0.0.0.10
connect 0.0.0.9 0.0.0.11
```

Configuración D

```
#                       Pruebas_MCRv2_configuracion_D.config
#
#            Ejemplo de protocolo multicast orientado a tiempo real
#                     Autor: Javier Jes£s S nchez Medina
#                             Comenzado: 19/10/00
#

#pcmovil1
node 0.0.0.1 -routes Pruebas_MCRv2_configuracion_D.routes
channel 0.0.0.1 192.168.1.3:8001 -log 255
application 0.0.0.1 apps.SenderApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.1 -gui true -log 255

node 0.0.0.2 -routes Pruebas_MCRv2_configuracion_D.routes
channel 0.0.0.2 192.168.1.3:8002 -log 255
application 0.0.0.2 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.2 -gui true -log 255

node 0.0.0.5 -routes Pruebas_MCRv2_configuracion_D.routes
channel 0.0.0.5 192.168.1.3:8005 -log 255
application 0.0.0.5 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.5 -gui true -log 255

#pcmovil2
node 0.0.0.3 -routes Pruebas_MCRv2_configuracion_D.routes
channel 0.0.0.3 192.168.1.4:8003 -log 255
application 0.0.0.3 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.3 -gui true -log 255

node 0.0.0.6 -routes Pruebas_MCRv2_configuracion_D.routes
channel 0.0.0.6 192.168.1.4:8006 -log 255
application 0.0.0.6 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.6 -gui true -log 255

node 0.0.0.7 -routes Pruebas_MCRv2_configuracion_D.routes
channel 0.0.0.7 192.168.1.4:8007 -log 255
application 0.0.0.7 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.7 -gui true -log 255

#Maquina A-wireless
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
node 0.0.0.4 -routes Pruebas_MCRv2_configuracion_D.routes
channel 0.0.0.4 192.168.1.2:8004 -log 255
application 0.0.0.4 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.4 -gui true -log 255

node 0.0.0.8 -routes Pruebas_MCRv2_configuracion_D.routes
channel 0.0.0.8 192.168.1.2:8008 -log 255
application 0.0.0.8 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.8 -gui true -log 255

node 0.0.0.9 -routes Pruebas_MCRv2_configuracion_D.routes
channel 0.0.0.9 192.168.1.2:8009 -log 255
application 0.0.0.9 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.9 -gui true -log 255

node 0.0.0.10 -routes Pruebas_MCRv2_configuracion_D.routes
channel 0.0.0.10 192.168.1.2:8010 -log 255
application 0.0.0.10 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.10 -gui true -log 255

node 0.0.0.11 -routes Pruebas_MCRv2_configuracion_D.routes
channel 0.0.0.11 192.168.1.2:8011 -log 255
application 0.0.0.11 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.11 -gui true -log 255

connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.2 0.0.0.4
connect 0.0.0.2 0.0.0.5
connect 0.0.0.3 0.0.0.6
connect 0.0.0.3 0.0.0.7
connect 0.0.0.4 0.0.0.8
connect 0.0.0.4 0.0.0.9
connect 0.0.0.7 0.0.0.10
connect 0.0.0.7 0.0.0.11
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

MulticastEC v2

Configuración A

```
#                              Pruebas_MCECv2_configuracion_A.config
#
#        Ejemplo de protocolo multicast de procesamiento de texto
distribuido v2
#                      Autor: Javier Jes£s S nchez Medina
#                              Comenzado: 19/02/01
#


#Maquina A
node 0.0.0.1 -routes Pruebas_MCECv2_configuracion_A.routes
channel 0.0.0.1 192.168.0.4:8001 -log 255
application 0.0.0.1 apps.SenderApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.1 -gui true -log 255

#Maquina B
node 0.0.0.2 -routes Pruebas_MCECv2_configuracion_A.routes
channel 0.0.0.2 192.168.0.2:8002 -log 255
application 0.0.0.2 apps.NodeApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.2 -gui true -log 255

#Maquina A-wireless
node 0.0.0.3 -routes Pruebas_MCECv2_configuracion_A.routes
channel 0.0.0.3 192.168.1.2:8003 -log 255
application 0.0.0.3 apps.NodeApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.3 -gui true -log 255

#Maquina C
node 0.0.0.4 -routes Pruebas_MCECv2_configuracion_A.routes
channel 0.0.0.4 192.168.0.3:8004 -log 255
application 0.0.0.4 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.4 -gui true -log 255

#Maquina B
node 0.0.0.5 -routes Pruebas_MCECv2_configuracion_A.routes
channel 0.0.0.5 192.168.0.2:8005 -log 255
application 0.0.0.5 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.5 -gui true -log 255

#pcmovil1
node 0.0.0.6 -routes Pruebas_MCECv2_configuracion_A.routes
channel 0.0.0.6 192.168.1.3:8006 -log 255
application 0.0.0.6 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.6 -gui true -log 255
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
#pcmovil2
node 0.0.0.7 -routes Pruebas_MCECv2_configuracion_A.routes
channel 0.0.0.7 192.168.1.4:8007 -log 255
application 0.0.0.7 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.7 -gui true -log 255

connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.2 0.0.0.4
connect 0.0.0.2 0.0.0.5
connect 0.0.0.3 0.0.0.6
connect 0.0.0.3 0.0.0.7
```

Configuración B

```
#                    Pruebas_MCECv2_configuracion_B.config
#
#        Ejemplo de protocolo multicast de procesamiento de texto
distribuido v2
#                    Autor: Javier Jesœs Sÿnchez Medina
#                           Comenzado: 19/02/01
#


#Maquina A
node 0.0.0.1 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.1 192.168.0.4:8001 -log 255
application 0.0.0.1 apps.SenderApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.1 -gui true -log 255

#Maquina B
node 0.0.0.2 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.2 192.168.0.2:8002 -log 255
application 0.0.0.2 apps.NodeApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.2 -gui true -log 255

node 0.0.0.9 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.9 192.168.0.2:8009 -log 255
application 0.0.0.9 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.9 -gui true -log 255

node 0.0.0.10 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.10 192.168.0.2:8010 -log 255
application 0.0.0.10 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.10 -gui true -log 255

#Maquina A-wireless
node 0.0.0.3 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.3 192.168.1.2:8003 -log 255
application 0.0.0.3 apps.NodeApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
manager 0.0.0.3 -gui true -log 255

#Maquina C
node 0.0.0.4 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.4 192.168.0.3:8004 -log 255
application 0.0.0.4 apps.NodeApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.4 -gui true -log 255

node 0.0.0.7 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.7 192.168.0.3:8007 -log 255
application 0.0.0.7 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.7 -gui true -log 255

node 0.0.0.8 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.8 192.168.0.3:8008 -log 255
application 0.0.0.8 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.8 -gui true -log 255

#pcmovil1
node 0.0.0.5 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.5 192.168.1.3:8005 -log 255
application 0.0.0.5 apps.NodeApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.5 -gui true -log 255

node 0.0.0.11 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.11 192.168.1.3:8011 -log 255
application 0.0.0.11 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.11 -gui true -log 255

node 0.0.0.12 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.12 192.168.1.3:8012 -log 255
application 0.0.0.12 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.12 -gui true -log 255

#pcmovil2
node 0.0.0.6 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.6 192.168.1.4:8006 -log 255
application 0.0.0.6 apps.NodeApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.6 -gui true -log 255

node 0.0.0.13 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.13 192.168.1.4:8013 -log 255
application 0.0.0.13 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.13 -gui true -log 255

node 0.0.0.14 -routes Pruebas_MCECv2_configuracion_B.routes
channel 0.0.0.14 192.168.1.4:8014 -log 255
application 0.0.0.14 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.14 -gui true -log 255
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
connect 0.0.0.1 0.0.0.2
connect 0.0.0.1 0.0.0.3
connect 0.0.0.2 0.0.0.4
connect 0.0.0.2 0.0.0.9
connect 0.0.0.2 0.0.0.10
connect 0.0.0.3 0.0.0.5
connect 0.0.0.3 0.0.0.6
connect 0.0.0.4 0.0.0.7
connect 0.0.0.4 0.0.0.8
connect 0.0.0.5 0.0.0.11
connect 0.0.0.5 0.0.0.12
connect 0.0.0.6 0.0.0.13
connect 0.0.0.6 0.0.0.14
```

Configuración C

```
#                    Pruebas_MCECv2_configuracion_C.config
#
#        Ejemplo de protocolo multicast de procesamiento de texto
distribuido v2
#                     Autor: Javier Jes£s S nchez Medina
#                            Comenzado: 19/02/01
#

#pcmovil1
node 0.0.0.1 -routes Pruebas_MCECv2_configuracion_C.routes
channel 0.0.0.1 192.168.1.3:8001 -log 255
application 0.0.0.1 apps.SenderApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.1 -gui true -log 255

#pcmovil2
node 0.0.0.2 -routes Pruebas_MCECv2_configuracion_C.routes
channel 0.0.0.2 192.168.1.4:8002 -log 255
application 0.0.0.2 apps.NodeApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.2 -gui true -log 255

node 0.0.0.3 -routes Pruebas_MCECv2_configuracion_C.routes
channel 0.0.0.3 192.168.1.4:8003 -log 255
application 0.0.0.3 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.3 -gui true -log 255

#Maquina A-wireless
node 0.0.0.4 -routes Pruebas_MCECv2_configuracion_C.routes
channel 0.0.0.4 192.168.1.2:8004 -log 255
application 0.0.0.4 apps.NodeApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.4 -gui true -log 255

#Maquina A
node 0.0.0.5 -routes Pruebas_MCECv2_configuracion_C.routes
channel 0.0.0.5 192.168.0.4:8005 -log 255
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
application 0.0.0.5 apps.NodeApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.5 -gui true -log 255

node 0.0.0.6 -routes Pruebas_MCECv2_configuracion_C.routes
channel 0.0.0.6 192.168.0.4:8006 -log 255
application 0.0.0.6 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.6 -gui true -log 255

#Maquina B
node 0.0.0.7 -routes Pruebas_MCECv2_configuracion_C.routes
channel 0.0.0.7 192.168.0.2:8007 -log 255
application 0.0.0.7 apps.NodeApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.7 -gui true -log 255

node 0.0.0.8 -routes Pruebas_MCECv2_configuracion_C.routes
channel 0.0.0.8 192.168.0.2:8008 -log 255
application 0.0.0.8 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.8 -gui true -log 255

#Maquina C
node 0.0.0.9 -routes Pruebas_MCECv2_configuracion_C.routes
channel 0.0.0.9 192.168.0.3:8009 -log 255
application 0.0.0.9 apps.ReceiverApplication -Rgroup 0.255.0.0 -Wgroup
255.0.0.0 -target 0.0.0.1
manager 0.0.0.9 -gui true -log 255

connect 0.0.0.1 0.0.0.2
connect 0.0.0.2 0.0.0.3
connect 0.0.0.2 0.0.0.4
connect 0.0.0.4 0.0.0.5
connect 0.0.0.5 0.0.0.6
connect 0.0.0.5 0.0.0.7
connect 0.0.0.7 0.0.0.8
connect 0.0.0.7 0.0.0.9
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

ANEXO III: Ejemplos de ficheros necesarios para la ejecución del ANTS

En este anexo mostramos los ficheros necesarios para ejecutar un prototipo con el ANTS. En el pliego de condiciones se muestra el significado de cada uno.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

Makefile

```
SUBDIRS = `find . -type d ! -name "." ! -name "CVS" -print -prune`

all:
      for i in *.config; \
        do \
        ./makeroutes -route < $$i > `basename $$i .config`.routes ; \
        if [ -d ../mcast ]; then \
          java mcast.RouteUtil `basename $$i .config`.routes  $$i
`basename $$i .config`.mroutes ; \
        fi; \
        done
```

makeroutes

```
#!/bin/sh
# the next line restarts using tclsh \
exec tclsh "$0" "$@"
#
#  $Id: makeroutes,v 1.2 1997/08/30 14:37:57 djw Exp $
#
#  Copyright 1996 Massachusetts Institute of Technology
#
#  Permission to use, copy, modify, distribute, and sell this software
and
#  its documentation for any purpose is hereby granted without fee,
provided
#  that the above copyright notice appear in all copies and that both
that
#  copyright notice and this permission notice appear in supporting
#  documentation, and that the name of M.I.T. not be used in advertising
or
#  publicity pertaining to distribution of the software without specific,
#  written prior permission.  M.I.T. makes no representations about the
#  suitability of this software for any purpose.  It is provided "as is"
#  without express or implied warranty.
#


#
# primitive/slow matrix data structure
#

proc matrix {m i j args} {
  global $m
  set idx [list $i $j]
  if {$args == {}} then {
    set [set m]($idx)
  } else {
    set [set m]($idx) [lindex $args 0]
  }
}
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
#
# distance and route matrices for path algorithm
#

proc D {k i j} {
  matrix D$k $i $j
}

proc setD {k i j v} {
  matrix D$k $i $j $v
}

proc II {k i j} {
  matrix II$k $i $j
}

proc setII {k i j v} {
  matrix II$k $i $j $v
}


#
# procs to express the topology
#

proc channel {n addr args} {
  global nodes
  set nodes($n) $addr
}

proc application {node name args} {}
proc extension {args} {}
proc manager {node args} {}
proc group {args} {}

proc node {name args} {
  global nodes
  set nodes($name) {}

  setD 0 $name $name 0

  foreach i [nodes] {
    foreach j [nodes] {
      if {[catch {D 0 $i $j}] != 0} {
      setD 0 $i $j infinity
      }
      if {[catch {II 0 $i $j}] != 0} {
      setII 0 $i $j nil
      }
    }
  }

  return $name
}

proc path {a b w} {
  setD 0 $a $b $w
  setII 0 $a $b $a
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
}

proc connect {a b} {
  path $a $b 1
  path $b $a 1
}

proc nodes {} {
  global nodes
  lsort [array names nodes]
}

proc addr {n} {
  global nodes
  return $nodes($n)
}

proc pred {i j} {
  II [lindex [nodes] end] $i $j
}

proc route {i j} {
  if {[pred $i $j] == "nil"} then {
    return "nil"
  }
  set last $j
  while {[pred $i $last] != $i} {
    set last [pred $i $last]
  }
  return $last
}

proc cost {i j} {
  D [lindex [nodes] end] $i $j
}

proc topology {i j} {
  II 0 $i $j
}


#
# the topology data
#

proc topology-booktest {} {
  for {set i 1} {$i<=5} {incr i} {
    node $i "localhost:8000"
  }

  path 1 2 3
  path 1 3 8
  path 1 5 -4

  path 2 4 1
  path 2 5 7

  path 3 2 4
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
  path 4 1 2
  path 4 3 -5

  path 5 4 6
}


#
# compute paths based on floyd-warshall
#

proc shortest-paths {} {
  set last 0
  foreach k [nodes] {
    foreach i [nodes] {
      foreach j [nodes] {

      set a [D $last $i $j]
      set b [D $last $i $k]
      set c [D $last $k $j]

      if {[catch {set d [expr {$b + $c}]}] != 0} {
        set d infinity
      }

      set e [II $last $i $j]
      set f [II $last $k $j]

      if {$a == "infinity"} {
        setD $k $i $j $d
        setII $k $i $j $f
      } elseif {$d == "infinity"} {
        setD $k $i $j $a
        setII $k $i $j $e
      } elseif {$a <= $d} {
        setD $k $i $j $a
        setII $k $i $j $e
      } else {
        setD $k $i $j $d
        setII $k $i $j $f
      }

      }
    }
    set last $k
  }
}


#
# the main deal
#

proc main {} {
  global argv argv0

  #
  # get a topology from stdin
  #
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
eval [read stdin]

#
# now calculate routes, etc.
#

shortest-paths

#
# and write requested info to stdout
#

foreach format $argv {
  switch -- $format {

    -topology {
    puts "\nTopology:\n"

    foreach x [nodes] {
      foreach y [nodes] {
        puts -nonewline stdout "[topology $x $y]\t"
      }
      puts stdout ""
    }
    }

    -route {
    puts "#"
    puts "# shortest routes, automatically generated"
    puts "# by $argv0 on [clock format [clock seconds] -format %D]"
    puts "#"
    puts ""

    puts "# source\tdestination\tnext   \taddr"
    puts ""

    foreach x [nodes] {
      foreach y [nodes] {
        set r [route $x $y]
        if {$r != "nil"} {
          puts "$x\t$y\t$r\t[addr $r]"
        }
      }
    }
    }

    -length {
    puts "\nShortest path lengths:\n"

    foreach x [nodes] {
      foreach y [nodes] {
        puts -nonewline stdout "[cost $x $y]\t"
      }
      puts stdout ""
    }
    }

    default {
    puts stderr "usage: $argv0 "
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
      puts "  -topology"
      puts "  -route"
      puts "  -length"
      exit 1
      }
    }
  }
}

main


# Local Variables:
# mode: tcl
# tcl-indent-level: 2
# End:
```

mcastRT.config

```
#                                mcastRT.config
#
#            Ejemplo de protocolo multicast orientado a tiempo real
#                     Autor: Javier Jesús Sánchez Medina
#                              Comenzado: 19/10/00
#


#
# configuración multicast
#
#
#
#             +--Receiver(3) +--Receiver(5) +--Receiver(7) +--Receiver(9)
#             |              |              |              |
#      +--intermedio(2)--intermedio(4)--intermedio(6)--intermedio(8)
#      |
#   Sender(1)
#
#
#

node 0.0.0.1 -routes mcastRT.routes
channel 0.0.0.1 localhost:8001 -log 255
application 0.0.0.1 apps.SenderApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.1 -gui true -log 255

node 0.0.0.2 -routes mcastRT.routes
channel 0.0.0.2 localhost:8002 -log 255
application 0.0.0.2 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.2 -gui true -log 255

node 0.0.0.4 -routes mcastRT.routes
channel 0.0.0.4 localhost:8004 -log 255
application 0.0.0.4 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
manager 0.0.0.4 -gui true -log 255

node 0.0.0.6 -routes mcastRT.routes
channel 0.0.0.6 localhost:8006 -log 255
application 0.0.0.6 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.6 -gui true -log 255

node 0.0.0.8 -routes mcastRT.routes
channel 0.0.0.8 localhost:8008 -log 255
application 0.0.0.8 apps.NodeApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.8 -gui true -log 255

node 0.0.0.3 -routes mcastRT.routes
channel 0.0.0.3 localhost:8003 -log 255
application 0.0.0.3 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.3 -gui true -log 255

node 0.0.0.5 -routes mcastRT.routes
channel 0.0.0.5 localhost:8005 -log 255
application 0.0.0.5 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.5 -gui true -log 255

node 0.0.0.7 -routes mcastRT.routes
channel 0.0.0.7 localhost:8007 -log 255
application 0.0.0.7 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.7 -gui true -log 255

node 0.0.0.9 -routes mcastRT.routes
channel 0.0.0.9 localhost:8009 -log 255
application 0.0.0.9 apps.ReceiverApplication -group 255.255.0.0 -target
0.0.0.1
manager 0.0.0.9 -gui true -log 255


connect 0.0.0.1 0.0.0.2
connect 0.0.0.2 0.0.0.3
connect 0.0.0.2 0.0.0.4
connect 0.0.0.4 0.0.0.5
connect 0.0.0.4 0.0.0.6
connect 0.0.0.6 0.0.0.7
connect 0.0.0.6 0.0.0.8
connect 0.0.0.8 0.0.0.9
```

mcastRT.routes

```
#
# shortest routes, automatically generated
# by ./makeroutes on 04/23/01
#

# source    destination next        addr
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

0.0.0.1	0.0.0.2	0.0.0.2	localhost:8002
0.0.0.1	0.0.0.3	0.0.0.2	localhost:8002
0.0.0.1	0.0.0.4	0.0.0.2	localhost:8002
0.0.0.1	0.0.0.5	0.0.0.2	localhost:8002
0.0.0.1	0.0.0.6	0.0.0.2	localhost:8002
0.0.0.1	0.0.0.7	0.0.0.2	localhost:8002
0.0.0.1	0.0.0.8	0.0.0.2	localhost:8002
0.0.0.1	0.0.0.9	0.0.0.2	localhost:8002
0.0.0.2	0.0.0.1	0.0.0.1	localhost:8001
0.0.0.2	0.0.0.3	0.0.0.3	localhost:8003
0.0.0.2	0.0.0.4	0.0.0.4	localhost:8004
0.0.0.2	0.0.0.5	0.0.0.4	localhost:8004
0.0.0.2	0.0.0.6	0.0.0.4	localhost:8004
0.0.0.2	0.0.0.7	0.0.0.4	localhost:8004
0.0.0.2	0.0.0.8	0.0.0.4	localhost:8004
0.0.0.2	0.0.0.9	0.0.0.4	localhost:8004
0.0.0.3	0.0.0.1	0.0.0.2	localhost:8002
0.0.0.3	0.0.0.2	0.0.0.2	localhost:8002
0.0.0.3	0.0.0.4	0.0.0.2	localhost:8002
0.0.0.3	0.0.0.5	0.0.0.2	localhost:8002
0.0.0.3	0.0.0.6	0.0.0.2	localhost:8002
0.0.0.3	0.0.0.7	0.0.0.2	localhost:8002
0.0.0.3	0.0.0.8	0.0.0.2	localhost:8002
0.0.0.3	0.0.0.9	0.0.0.2	localhost:8002
0.0.0.4	0.0.0.1	0.0.0.2	localhost:8002
0.0.0.4	0.0.0.2	0.0.0.2	localhost:8002
0.0.0.4	0.0.0.3	0.0.0.2	localhost:8002
0.0.0.4	0.0.0.5	0.0.0.5	localhost:8005
0.0.0.4	0.0.0.6	0.0.0.6	localhost:8006
0.0.0.4	0.0.0.7	0.0.0.6	localhost:8006
0.0.0.4	0.0.0.8	0.0.0.6	localhost:8006
0.0.0.4	0.0.0.9	0.0.0.6	localhost:8006
0.0.0.5	0.0.0.1	0.0.0.4	localhost:8004
0.0.0.5	0.0.0.2	0.0.0.4	localhost:8004
0.0.0.5	0.0.0.3	0.0.0.4	localhost:8004
0.0.0.5	0.0.0.4	0.0.0.4	localhost:8004
0.0.0.5	0.0.0.6	0.0.0.4	localhost:8004
0.0.0.5	0.0.0.7	0.0.0.4	localhost:8004
0.0.0.5	0.0.0.8	0.0.0.4	localhost:8004
0.0.0.5	0.0.0.9	0.0.0.4	localhost:8004
0.0.0.6	0.0.0.1	0.0.0.4	localhost:8004
0.0.0.6	0.0.0.2	0.0.0.4	localhost:8004
0.0.0.6	0.0.0.3	0.0.0.4	localhost:8004
0.0.0.6	0.0.0.4	0.0.0.4	localhost:8004
0.0.0.6	0.0.0.5	0.0.0.4	localhost:8004
0.0.0.6	0.0.0.7	0.0.0.7	localhost:8007
0.0.0.6	0.0.0.8	0.0.0.8	localhost:8008
0.0.0.6	0.0.0.9	0.0.0.8	localhost:8008
0.0.0.7	0.0.0.1	0.0.0.6	localhost:8006
0.0.0.7	0.0.0.2	0.0.0.6	localhost:8006
0.0.0.7	0.0.0.3	0.0.0.6	localhost:8006
0.0.0.7	0.0.0.4	0.0.0.6	localhost:8006
0.0.0.7	0.0.0.5	0.0.0.6	localhost:8006
0.0.0.7	0.0.0.6	0.0.0.6	localhost:8006
0.0.0.7	0.0.0.8	0.0.0.6	localhost:8006
0.0.0.7	0.0.0.9	0.0.0.6	localhost:8006
0.0.0.8	0.0.0.1	0.0.0.6	localhost:8006
0.0.0.8	0.0.0.2	0.0.0.6	localhost:8006
0.0.0.8	0.0.0.3	0.0.0.6	localhost:8006
0.0.0.8	0.0.0.4	0.0.0.6	localhost:8006

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004

```
0.0.0.8     0.0.0.5     0.0.0.6     localhost:8006
0.0.0.8     0.0.0.6     0.0.0.6     localhost:8006
0.0.0.8     0.0.0.7     0.0.0.6     localhost:8006
0.0.0.8     0.0.0.9     0.0.0.9     localhost:8009
0.0.0.9     0.0.0.1     0.0.0.8     localhost:8008
0.0.0.9     0.0.0.2     0.0.0.8     localhost:8008
0.0.0.9     0.0.0.3     0.0.0.8     localhost:8008
0.0.0.9     0.0.0.4     0.0.0.8     localhost:8008
0.0.0.9     0.0.0.5     0.0.0.8     localhost:8008
0.0.0.9     0.0.0.6     0.0.0.8     localhost:8008
0.0.0.9     0.0.0.7     0.0.0.8     localhost:8008
0.0.0.9     0.0.0.8     0.0.0.8     localhost:8008
```

mcastRT.start

```
java ants.ConfigurationManager mcastRT.config 0.0.0.9 >& 0.0.0.9.log &
java ants.ConfigurationManager mcastRT.config 0.0.0.8 >& 0.0.0.8.log &
java ants.ConfigurationManager mcastRT.config 0.0.0.7 >& 0.0.0.7.log &
java ants.ConfigurationManager mcastRT.config 0.0.0.6 >& 0.0.0.6.log &
java ants.ConfigurationManager mcastRT.config 0.0.0.5 >& 0.0.0.5.log &
java ants.ConfigurationManager mcastRT.config 0.0.0.4 >& 0.0.0.4.log &
java ants.ConfigurationManager mcastRT.config 0.0.0.3 >& 0.0.0.3.log &
java ants.ConfigurationManager mcastRT.config 0.0.0.2 >& 0.0.0.2.log &
java ants.ConfigurationManager mcastRT.config 0.0.0.1 >& 0.0.0.1.log &
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2004