UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA

EN INFORMÁTICA

PROYECTO FIN DE CARRERA

Implementación y evaluación de un modelo para la gestión de la tabla de arbitraje en InfiniBand

Joaquín Laborda Camacho

Julio, 2005

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA POLITÉCNICA SUPERIOR

(Departamento de Informática)

PROYECTO FIN DE CARRERA

Implementación y evaluación de un modelo para la gestión de la tabla de arbitraje en InfiniBand

Autor: Joaquín Laborda Camacho
Directores: José Luís Sánchez García y Francisco José Alfaro Cortés

Julio, 2005

Reunido en la fecha el Tribunal evaluador, que más abajo se cita, del Proyecto Fin de Carrera titulado:

_____

_____

presentado por D/Dª

_____

y siendo su/s tutor/es

_____

se otorga la calificación de _____

Y para que así conste, se firma la presente acta en

Albacete a ..... de ..... de 20.....

PRESIDENTE: _____

SECRETARIO: _____

VOCAL: _____

SECRETARIO PRESIDENTE VOCAL

Resumen

InfiniBand es una tecnología de interconexión entre sistemas de procesamiento y dispositivos de E/S que permite formar una red de área de sistema. Desde que en Octubre del año 2000 se publicara la primera versión, de las especificaciones de InfiniBand, muchos han sido los trabajos que se han realizado en torno a esta tecnología. Los estudios son muy diversos y van desde propuestas para garantizar calidad de servicio a las aplicaciones, hasta técnicas de encaminamiento o tolerancia a fallos, pasando por el desarrollo de software de administración y de librerías de comunicaciones específicas para InfiniBand.

Para proporcionar calidad de servicio a las aplicaciones, InfiniBand proporciona varios mecanismos que permiten al administrador de la subred gestionar distintas garantías de calidad de servicio, tanto para servicios con conexión como sin conexión. En los servicios con conexión se puede intentar garantizar calidad de servicio. Sin embargo, en los servicios sin conexión no es posible dar garantías, pues no se conoce a priori los requisitos que necesitan las aplicaciones. Los mecanismos que incorpora InfiniBand para conseguir proporcionar calidad de servicio son básicamente: niveles de servicio, canales virtuales, arbitraje en dichos canales virtuales y particiones.

El trabajo desarrollado en este Proyecto Fin de Carrera está relacionado con uno de esos mecanismos, concretamente el arbitraje de los canales virtuales. Los puertos de salida (de conmutadores, encaminadores y terminales) incorporan una tabla de arbitraje que permite especificar la prioridad que tendrá cada canal virtual a la hora de enviar paquetes a través de ese puerto de salida. Es obligatorio implementar esta tabla si el puerto tiene más de dos canales virtuales. El arbitraje se realiza sólo entre los canales virtuales de datos, pues el canal virtual del tráfico de control siempre tiene preferencia.

El trabajo que aquí se presenta ha consistido, básicamente, en la implementación de unos algoritmos para el manejo de la tabla de arbitraje, y su evaluación de prestaciones. Esos algoritmos reflejan el comportamiento de un modelo de gestión de la tabla de arbitraje que había sido previamente propuesto de una manera formal.

A mi padre.

Agradecimientos

A mi **madre**, que con todo lo que ha leído mientras he estado haciendo el proyecto, ya podría examinar a sus chiquillos sobre tablas de arbitraje. También agradecer a mis **hermanos** el apoyo que me han dado.

Por último, agradecerle a mi amigo **Juanan** todo lo que me ha ayudado, dejando a veces lo suyo sin importarle.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

INTRODUCCIÓN Y MOTIVACIÓN

El trabajo que aquí se presenta ha consistido fundamentalmente en la realización y evaluación de un conjunto de algoritmos que permiten gestionar la tabla de arbitraje de los puertos de salida de conmutadores, encaminadores y terminales de una red InfiniBand. Los algoritmos obtenidos reflejan el comportamiento de un modelo particular de gestión de la tabla de arbitraje que ha sido propuesto en la tesis doctoral realizada por el profesor Francisco José Alfaro Cortés, y que también es tutor de este Proyecto Fin de Carrera.

1.1 INTRODUCCIÓN

Los clusters de computadores se han convertido en una plataforma ampliamente utilizada por diferentes tipos de usuarios para la ejecución de sus aplicaciones. Sus buenas prestaciones, unidas a su reducido coste comparado con el de los grandes supercomputadores, lo han hecho posible.

A ese buen nivel de prestaciones ha contribuido de una forma importante la mejora que ha experimentado la red de interconexión utilizada. Ello ha sido posible gracias a la aparición de nuevas tecnologías de red, como fue el caso de Myrinet hace unos años y más recientemente el de InfiniBand o PCI Express Advanced Switching.

Con sus especificaciones publicadas hace unos años, InfiniBand se ha convertido en el centro de atención de fabricantes e investigadores, de la misma forma que lo será PCI

Express Advanced Switching en breve espacio de tiempo. Esta ventaja de unos pocos años hace que de momento haya muchos más trabajos sobre InfiniBand.

Los estudios son muy diversos y van desde propuestas para garantizar calidad de servicio a las aplicaciones, hasta técnicas de encaminamiento o tolerancia a fallos, pasando por el desarrollo de software de administración y de librerías de comunicaciones específicas para InfiniBand.

Nos encontramos pues en un momento en el que tiene gran interés el estudio de los diferentes aspectos que caracterizan a las nuevas tecnologías de red y en especial el de aquellas de más reciente aparición como es el caso de InfiniBand o PCI Express Advanced Switching.

1.2 MOTIVACIÓN

Como se ha indicado anteriormente, muchos están siendo los estudios que se están realizando en torno a la tecnología de red InfiniBand, algunos de los cuales tienen que ver con los aspectos que incorpora para proporcionar calidad de servicio a las aplicaciones. Uno de ellos es el que constituyó la tesis doctoral que con el título "*Una sencilla metodología para garantizar calidad de servicio en subredes InfiniBand"* se presentó en el Departamento de Informática de esta universidad. En ese trabajo se plantea un modelo formal para la gestión de la tabla de arbitraje de los puertos de salida de conmutadores, encaminadores y terminales de una red InfiniBand. Se muestra además, a través de una serie de propiedades y teoremas, el adecuado funcionamiento de la tabla para conseguir proporcionar calidad de servicio a las aplicaciones que se usan en ese tipo de redes.

Hay que indicar que el trabajo citado plantea una metodología para garantizar latencia máxima y ancho de banda a los flujos de datos que requieren el uso de unos puertos dados de la red. Esa metodología indica la forma en la que se tienen que asignar las entradas de las tablas de arbitraje correspondientes a esos puertos. Todo ello queda plasmado en el modelo formal antes citado. Para el completo tratamiento de la tabla de arbitraje se plantea un conjunto de procedimientos o algoritmos de los cuales se indica cuál debe ser su funcionamiento, y cómo deben ser utilizados.

Dadas las propiedades del modelo propuesto y el hecho de que se haya asegurado su correcto funcionamiento de una manera formal, parece lógico dar continuidad al trabajo cubriendo los diferentes aspectos que se dejaron como trabajo futuro. En este Proyecto Fin de Carrera se aborda uno de ellos, concretamente el que se refiere a la implementación de

los algoritmos para el manejo de la tabla. La idea es obtener versiones eficientes de esos algoritmos y evaluar las prestaciones que ofrecen.

Por tanto, el objetivo principal de este proyecto es elaborar y evaluar un conjunto de algoritmos que permitan gestionar la tabla de arbitraje de los puertos de salida de conmutadores, encaminadores y terminales de una red InfiniBand, y hacerlo de acuerdo con un determinado modelo que será el de referencia.

Capítulo 2

OBJETIVOS Y METODOLOGÍA

Se incluyen en este capítulo los objetivos que se debían cubrir con este trabajo cuando éste se planteó. Así mismo, se indican las tareas fundamentales que se plantearon en su momento para conseguir esos objetivos.

2.1 OBJETIVOS

El objetivo principal de este trabajo es en realidad doble: por un lado elaborar o crear unos métodos, correspondientes a los algoritmos de un determinado modelo de gestión de la tabla de arbitraje de los puertos de salida de conmutadores, encaminadores y terminales de una red InfiniBand; y por otro realizar un estudio del rendimiento de dichos métodos.

Para lograr estos dos objetivos principales se plantearon en un principio varios objetivos parciales, que se describen brevemente en las siguientes líneas:

- **Conocer las características básicas de la tecnología InfiniBand.** Se trata de revisar y comprender las principales características de InfiniBand, y de forma especial las propias de la tabla de arbitraje de los puertos de salida de conmutadores, encaminadores y terminales de una red InfiniBand.

- **Comprender el modelo de gestión de la tabla de arbitraje en la que se basa este trabajo.** Se trata de estudiar con detalle el modelo formal de la tabla de arbitraje

considerado y comprender el funcionamiento de los algoritmos propuestos en ese modelo.

- **Elaborar los métodos correspondientes a los algoritmos del modelo**. A partir de la descripción del comportamiento de los algoritmos propuestos en el modelo para la gestión de la tabla de arbitraje, se trata de obtener los métodos, en un lenguaje de alto nivel, que reflejen ese comportamiento y lo hagan de la forma más eficiente posible.

- **Evaluar los métodos realizados**. Una vez comprobado el adecuado funcionamiento del código generado, se debe desarrollar un proceso de evaluación que permita extraer conclusiones sobre el comportamiento individual de cada algoritmo implementado, así como del que se produce como consecuencia de la combinación de ellos. En este segundo caso se deben comparar las diferentes opciones que a partir del modelo formal se presentaron.

Cubriendo estos objetivos parciales en su totalidad se habrá conseguido, de una forma estructurada y progresiva, alcanzar el objetivo principal del proyecto.

2.2 METODOLOGÍA

Para alcanzar los objetivos anteriores se plantearon diversas tareas. A continuación se resumen las más importantes, teniendo en cuenta que algunas de ellas se han solapado en el tiempo:

- **Consulta de documentación.** Básicamente han sido dos grupos de documentos los que se han manejado. Para conocer las características básicas de InfiniBand se han consultado algunos de los textos disponibles (especificaciones [Inf00], [Inf01], resúmenes, capítulos de otros proyectos [Vil04], etc.). En lo que se refiere al modelo de gestión de la tabla de arbitraje, la fuente ha sido la tesis doctoral en la que se propone [Alf03].

- **Estudio del modelo de la tabla de arbitraje.** Una de las tareas fundamentales para conseguir los objetivos de este trabajo ha sido el estudio del modelo en el que se basa el mismo. Al tratarse de un modelo formal, acompañado de la correspondiente carga matemática, requiere de una pormenorizada revisión, para comprender la esencia de su formulación y con ello asimilar correctamente el funcionamiento que se propone para manejar la tabla de arbitraje.

- **Implementación de los algoritmos para la gestión de la tabla.** A partir de las características del modelo de tabla, los algoritmos y las propiedades que se derivan a partir de su uso, se deben diseñar las estructuras de datos y los métodos que mejor reflejen el funcionamiento global de la tabla de arbitraje de acuerdo con el modelo. Será importante que el código obtenido sea lo más eficiente posible, tanto en cuanto a recursos que debe requerir para su implementación como en lo que se refiere a la velocidad de su procesamiento.

- **Evaluación del código obtenido.** A través de un conjunto de pruebas se determinará el comportamiento y nivel de prestaciones de los algoritmos realizados. El proceso de evaluación debe arrojar datos sobre el comportamiento individual de cada uno de los métodos, pero también debe proporcionar datos sobre el funcionamiento conjunto de todos ellos. Para este segundo caso, se considerarán las diferentes alternativas que se proponen en el documento de referencia [Alf03].

En los siguientes capítulos se muestra el resultado de aplicar cada una de estas tareas en busca de los objetivos antes enumerados. Así, el Capítulo 3 incluye un resumen de las características principales de InfiniBand. En el Capítulo 4 se incluye un breve resumen del modelo en el que se basa este trabajo. A lo largo del Capítulo 5 se presentan los métodos realizados para reflejar su funcionamiento. Estos últimos son evaluados en el Capítulo 6, presentando y analizando los resultados de dicha evaluación. Finalmente, en el Capítulo 7 se incluyen las conclusiones del trabajo realizado.

Capítulo 3

INFINIBAND

En este capítulo se presenta una breve descripción de InfiniBand analizando sus características principales. No se pretende realizar una descripción exhaustiva de la especificaciones de InfiniBand. En caso de necesitar más detalles del contenido de este capítulo se pueden consultar en las especificaciones de InfiniBand [Inf01], [IBA99], o algunos libros publicados sobre InfiniBand [Shan02].

3.1 DESCRIPCIÓN

InfiniBand es una tecnología de interconexión entre sistemas de procesamiento y dispositivos de E/S que permite formar una red de área de sistema (Figura 3.1). La arquitectura definida por InfiniBand es independiente del sistema operativo y de la plataforma.

A finales de los años noventa la idea de InfiniBand surge de la fusión de dos iniciativas existentes: NGIO *(Next Generation I/O)* y FIO *(Future I/O)*. Ambas compartían una buena parte de sus metas, y pronto se vio que no había mercado para las dos. Por ello, se decidió unir ambas propuestas para intentar hacerse hueco dentro del mercado de las tecnologías de interconexión. De esta forma, en octubre de 1999 se fundó la IBTA *(InfiniBand Trade Associattion)*.

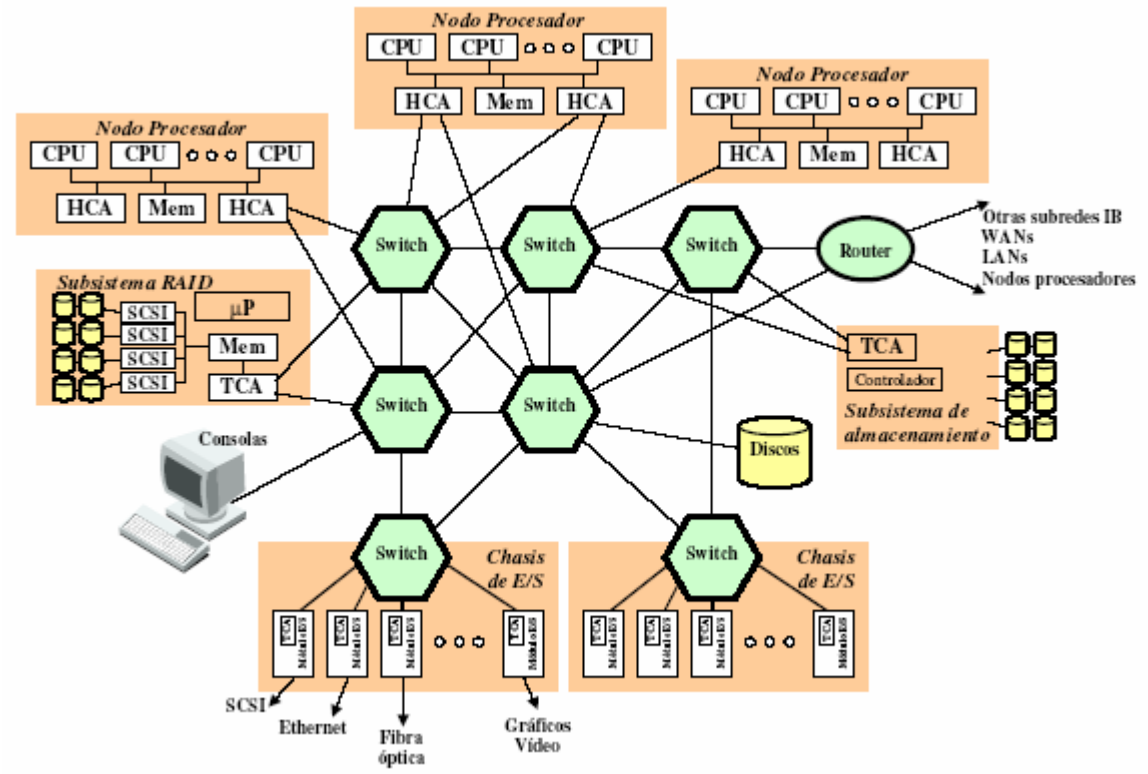


Figura 3.1: Red de área de sistema con InfiniBand.

Las dos principales metas que en un principio se plantea InfiniBand son: salvar las limitaciones que presentan los actuales buses PCI (por ejemplo, cuellos de botella, fiabilidad, escalabilidad, etc.), y estandarizar las emergentes tecnologías propietarias en el terreno de los clusters (por ejemplo, Servernet, Myricom, Gigante, etc.).

Aun así, InfiniBand quiere ir más allá de ser una simple sustitución del típico bus PCI. InfiniBand incorpora características que hasta ahora sólo podían encontrarse en supercomputadores grandes y costosos. Estas características son importantes para el montaje de clusters de altas prestaciones y permiten aprovechar las posibilidades de la tecnología actual.

En Octubre del año 2000 se publicó la primera versión, 1.0, de las especificaciones de Infiniband [Inf00]. En Junio del año 2001 se publicó la primera modificación a las especificaciones, 1.0a [Inf01], que consistió en corrección de errores e introducción de algunas anotaciones. En Noviembre del año 2002 se publicó una versión con todas las características adicionales 1.1 [Inf02]. En Junio del año 2003 se publicó las correcciones a erratas en la especificación 1.1. Por último, en Septiembre del año 2004 se ha presentado la versión 1.2, que introduce una serie de mejoras que hacen que se pueda obtener un ancho de banda de hasta 120Gbps en cada dirección así como otras nuevas características.

3.2 ARQUITECTURA

InfiniBand define una red de área de sistema (System Area Network, SAN) para conectar ordenadores, sistemas de E/S y dispositivos de E/S tal y como se muestra en la Figura 3.1. InfiniBand proporciona la infraestructura adecuada para comunicación y gestión, tanto para transacciones de E/S como para comunicación entre ordenadores. Un sistema InfiniBand puede variar desde un pequeño servidor formado por un procesador y unos cuantos dispositivos de E/S conectados, hasta un supercomputador masivamente paralelo con miles de procesadores y dispositivos de E/S que está conectado vía Internet a otras plataformas de procesamiento y/o sistemas de E/S.

InfiniBand define una interconexión conmutada que permite a muchos dispositivos intercambiar datos de forma simultánea, con gran ancho de banda y baja latencia. Al ser un sistema conmutado, se pueden conseguir características como protección, fiabilidad, escalabilidad, seguridad, etc, hasta ahora impensables en sistemas de E/S, e incluso en la mayoría de las redes habituales para conexión de computadores.

Un nodo final en una SAN InfiniBand puede comunicarse por medio de múltiples puertos del conmutador al que está conectado, pudiéndose habilitar de esta manera caminos alternativos. Así, se podría aprovechar la disponibilidad de caminos alternativos tanto para incrementar el ancho de banda real, como para permitir tolerancia a fallos.

InfinBand permite a las unidades de E/S comunicarse entre ellas y con cualquier sistema de procesamiento existente en el sistema. De esta manera, una unidad de E/S tiene la misma capacidad de comunicación que cualquier otro nodo de procesamiento.

3.2.1 Topología

InfiniBand tiene una topología conmutada con conexiones punto a punto, lo que permite tanto topologías regulares como irregulares. El sistema de administración será capaz de identificar cualquier topología formada, y construir las tablas de encaminamiento adecuadas para permitir el intercambio de información entre dos elementos cualquiera conectados a través de la red.

Desde el punto de vista de alto nivel, InfiniBand sólo es un medio para interconectar nodos entre sí (Figura. 3.2), donde un nodo puede ser un sistema de procesamiento, una unidad de E/S o un encaminador hacia otra red.

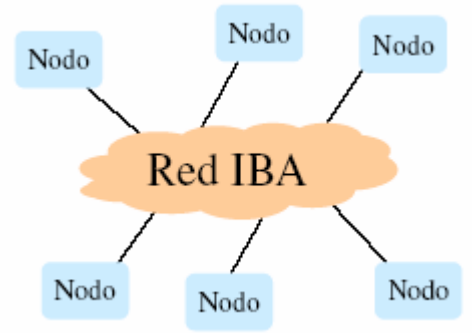


Figura 3.2: Red InfiniBand.

Una red InfiniBand está dividida en subredes interconectadas entre sí mediante routers o encaminadores (Figura. 3.3). Los nodos finales estarán conectados a una única subred o a múltiples subredes por medio de distintas interfaces.

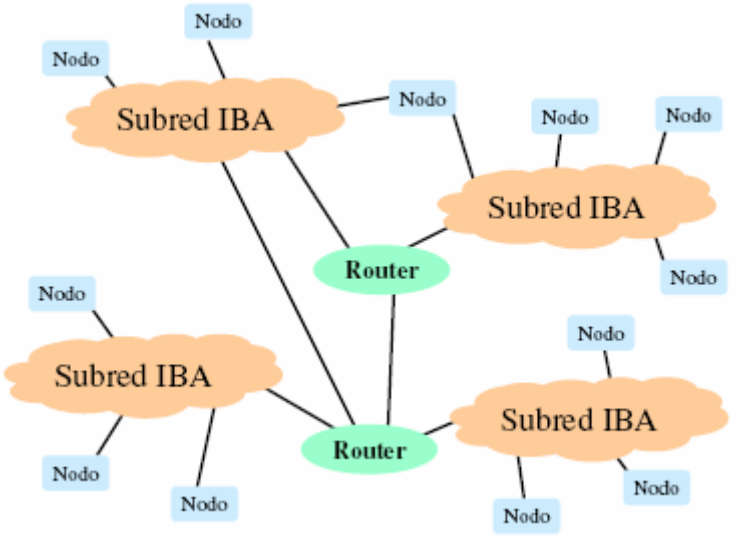


Figura 3.3: Componentes en una red InfiniBand.

En cada una de las subredes, debe haber un Subnet Manager que se encargue de ciertas tareas de control de la subred. Es importante decir que no todos los dispositivos pertenecientes a la subred tienen la capacidad de realizar dicha tarea, pero el que sí la tenga, puede llegar a actuar de Subnet Manager, aunque sólo uno de ellos puede hacerlo en un momento dado.

3.2.2 Componentes de InfiniBand

Una subred InfiniBand puede estar compuesta por nodos finales, conmutadores, encaminadores y el Subnet Manager, todos ellos interconectados por enlaces (Figura 3.4). Estos enlaces pueden ser de cable, de fibra óptica o grabados en la placa base.

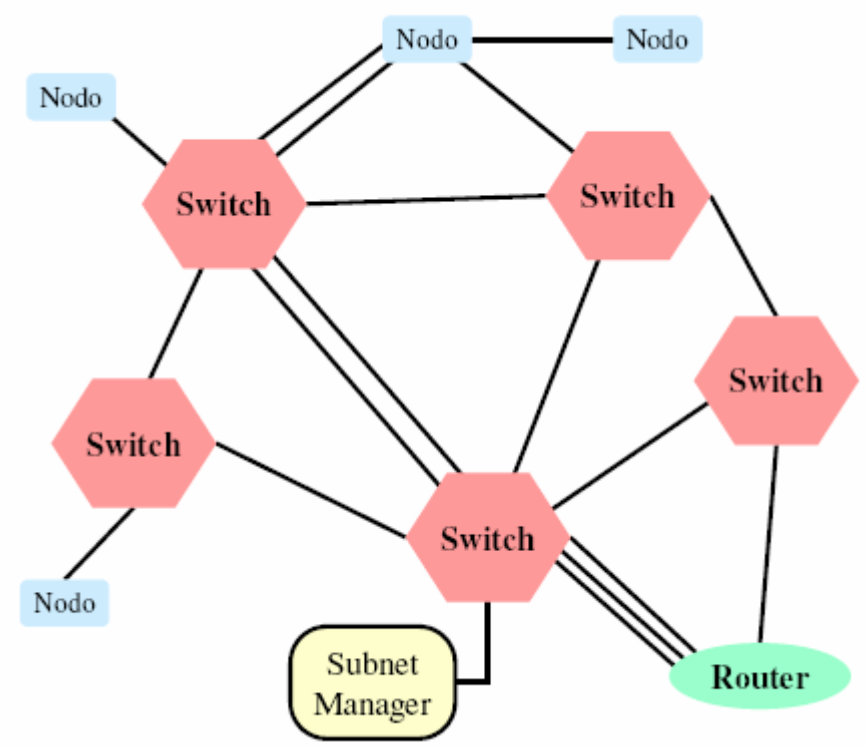


Figura 3.4: Componentes en una subred InfiniBand.

Entre cualquiera de los componentes de la red pueden existir múltiples enlaces, consiguiendo de esta forma incrementar el ancho de banda, así como mejorar la tolerancia a fallos. Además, cualquiera de los componentes de la red InfiniBand puede estar conectado a un único conmutador, a varios, o directamente con cualquier otro dispositivo sin pasar por un conmutador. Sin embargo, la conexión directa entre dos nodos finales forma una subred independiente, sin conexión con el resto de dispositivos. En este caso, ambos se pondrán de acuerdo para que uno de ellos actúe como Subnet Manager.

Los dispositivos en un sistema InfiniBand están clasificados como:

Enlaces y repetidores

Los *enlaces* (Links) interconectan channels adapters, switches, routers y repetidores formando una estructura. Los enlaces son cables de cobre, de fibra óptica o impresos en circuitos.

Los *repetidores* (Repeaters) son dispositivos transparentes que extienden el alcance de los enlaces. Sólo participan en los protocolos de la capa física y los nodos no detectan su presencia. En el volumen 2 de la especificación [Inf01] se describe los cables y repetidores para varios tipos de medios físicos.

Channel adapters

Un *channel adapter* (CA) es el componente en un nodo final que lo conecta a la red. Un channel adapter es un dispositivo DMA programable con características especiales de protección, que permite que las operaciones de DMA sean iniciadas de forma local o remota. Cada channel adapter tiene uno o varios puertos, tal y como puede verse en la Figura 3.5. Normalmente cada uno de ellos suele estar conectado a un puerto de un conmutador, aunque también es posible conectarlos entre sí directamente.

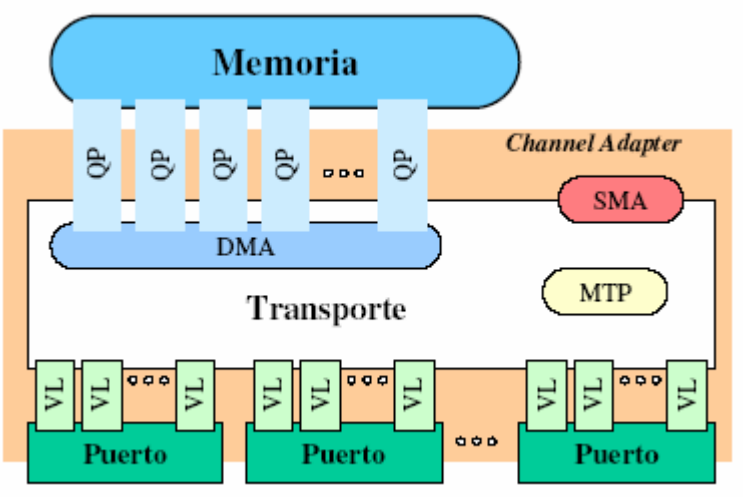


Figura 3.5: Estructura de un channel adapter.

Cada puerto del channel adapter tiene su propia dirección local configurada por el Subnet Manager. La entidad local que se comunica con el Subnet Manager para la configuración del channel adapter es el Subnet Management Agent (SMA). Por otra parte, cada puerto tiene su propio conjunto de buffers de envío y recepción, y es capaz de estar enviando y recibiendo información al mismo tiempo. El almacenamiento de los buffers

está distribuido en varios canales virtuales (VL), teniendo cada VL su propio control de flujo.

InfiniBand permite el particionado virtual de la red por medio de la definición de dominios de acceso lógico. Cada puerto de cada channel adapter pertenece al menos a una partición y puede comunicarse sólo con otros puertos que también pertenezcan a alguna de sus particiones. De esta forma, la propia red proporciona un nivel de protección.

Los channel adapter pueden clasificarse en:

Host Channel Adapter (HCA). Un HCA es para sistemas de procesamiento con gran capacidad para ejecutar distinto tipo de software.

Target Channel Adapter (HCA). Un TCA es para dispositivos de E/S, que por su simplicidad, no suelen tener capacidad para ejecutar software.

Conceptualmente tanto el HCA como el TCA son idénticos. Desde un punto de vista arquitectónica, el HCA difiere del TCA porque el HCA tiene una interfaz más estructurada que ofrece al sistema operativo.

Conmutadores

Los *conmutadores* (Switches) (Figura 3.6) disponen de una serie de puertos donde se conectan los enlaces. A todos los efectos, los conmutadores son completamente transparentes para los nodos finales. Es decir, ellos no saben de su existencia, ni muchos menos les son accesibles.

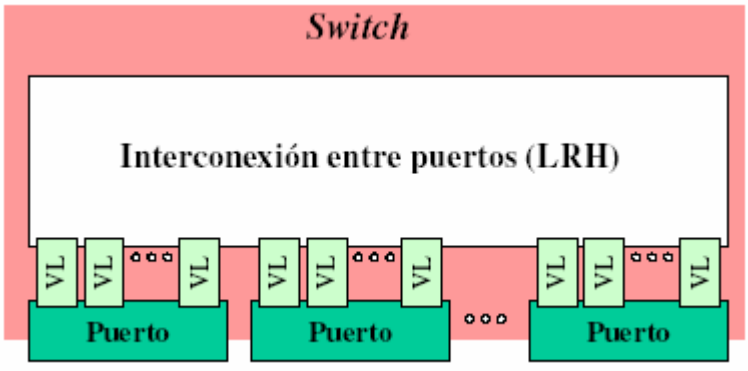


Figura 3.6: Estructura de un conmutador de InfiniBand.

En contraste con los channel adapters, los conmutadores ni generan ni consumen paquetes, salvo los propios de gestión de la subred. Los conmutadores simplemente transmiten paquetes desde un puerto de entrada hacia un puerto de salida, en base a la dirección destino que el paquete lleva en su cabecera.

Los paquetes que cruzan los conmutadores permanecen inalterados en su contenido, aunque sí puede que sufran cambios en su cabecera.

Los conmutadores tienen unas tablas de encaminamiento para decidir por qué puerto sacar cada paquete. Estas tablas se cargan en el momento del arranque , o tras un cambio en la topología, por el Subnet Manager. El conmutador, en base a esas tablas y a la dirección destino que viene especificada en la cabecera del paquete, debe decidir por qué puerto reenviar cada paquete que le llega. Según las especificaciones, no todos los conmutadores deben tener capacidad de reenvío multidestino, dejando este aspecto a criterio de los fabricantes.

Encaminadores

Un *encaminador* (Router) se encarga de comunicar entre sí distintas subredes. Los encaminadores tampoco tienen capacidad para generar o consumir paquetes (sin contar los de control). Nuevamente, se limitan a trasvasarlos desde uno de sus puertos de entrada a uno de los de salida. En este caso la decisión de qué puerto de salida se toma en base a una dirección global (GID). Esta dirección global es única en todas las redes InfiniBand, mientras que la dirección local (LID) es propia de cada subred.

El esquema básico de un encaminador se muestra en la Figura 3.7. Al igual que los conmutadores, los encaminadores tienen varios puertos con capacidad de recibir y transmitir de forma simultánea. El espacio para almacenamiento también está distribuido entres varios canales virtuales.

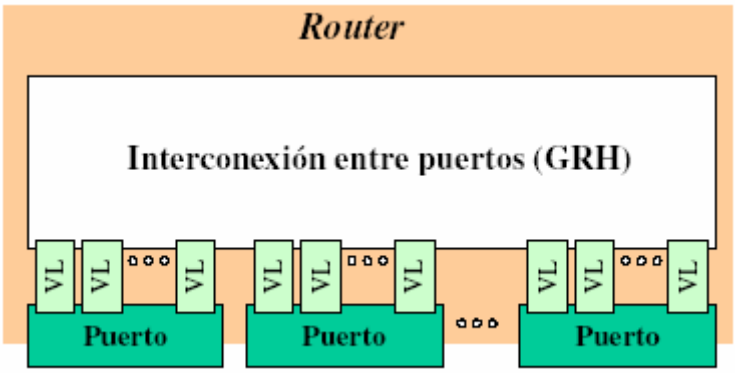


Figura 3.7: Estructura de un encaminador de InfiniBand.

Los encaminadores no son completamente transparentes para los nodos finales, puesto que el nodo fuente debe especificar la dirección local (LID) del encaminador de salida al exterior de su subred, además de indicar la dirección global (GID) del nodo destino.

Cada encaminador envía los paquetes hacia la siguiente subred a otro encaminador, hasta que el paquete llega a la subred destino. El último encaminador envía el paquete al nodo final utilizando la dirección local (LID) que se especificó en el origen.

3.2.3 Características de InfiniBand

Pares de colas

Son la interfaz virtual que el hardware proporciona a un productor de información en InfiniBand, y el puerto de comunicación virtual que proporciona para el consumidor de dicha información. De esta forma, la comunicación tiene lugar entre un QP fuente y un QP destino. Cada QP permite tener aislada y protegida esa comunicación de la del resto de QPs u otros consumidores.

Tipos de servicio

Cada QP se configura para proporcionar un determinado tipo de servicio. Ese tipo de servicio está basado en cómo interactúan los QPs fuente y destino, que deben estar configurados con el mismo tipo de servicio. Hay tres características que identifican a cada tipo de servicio que son:

- *Servicio orientado a conexión* (cada QP fuente tiene a un único QP destino, y viceversa) frente a *no orientado a conexión* (cada QP puede enviar/recibir paquetes a/desde cualquier otro QP).

- *Servicio con confirmación* (un QP debe confirmar la recepción de un paquete al QP que lo envió) frente a *sin confirmación* (no se asegura que la información llegue a su destino).

- *Servicio de transporte de InfiniBand* (permite transmitir paquetes en bruto encapsulando paquetes de otros protocolos de transporte) frente a otro *tipo de transporte*.

Claves

Las claves (keys) proporcionan un cierto nivel de aislamiento y protección del tráfico, aunque por sí solas no proporcionan seguridad puesto que están disponibles en la cabecera de los paquetes que fluyen por la red. Son unos valores asignados por las entidades de administración y que luego se insertarán en los paquetes según la función y destino al que vayan dirigidos. Las aplicaciones sólo podrán acceder a los paquetes que contengan claves para los que ellas estén habilitadas. Los diferentes tipos de claves son:

- *Management Key, M_Key*. Las asigna el Subnet Manager a cada puerto tal que todo los paquetes de control con ese puerto deberían llevar esa clave insertada en los paquetes.

- *Baseboard Management Key*, *B_Key*. Permite que actúe el gestor de subred en placa. Se encuentra en algunos paquetes encargados de la gestión de la subred.

- *Partition Key*, *P_Key*. Se utilizan para dividir la subred en distintas zonas, tal que cada adaptador contiene una tabla de estas claves para definir las particiones para las que está habilitado.

- *Queue Key*, *Q_Key*. Permite controlar el derecho de acceso a las colas para los servicios sin conexión. Esta clave identifica unívocamente a los nodos que no hayan establecido previamente una conexión.

- *Memory Keys*, *K_Key* y *R_Key*. Permiten el uso de direcciones de memoria virtual y dota al consumidor de un mecanismo para controlar el acceso a dicha memoria.

Direcciones de memoria virtual

La arquitectura de InfiniBand está optimizada para manejar direccionamiento virtual (virtual memory adresses). Un consumidor puede especificar direcciones virtuales y es el adaptador el que convierte la dirección virtual a una dirección física. Un consumidor registra regiones de memoria virtual con su adaptador, y éste le devuelve dos claves llamadas L_Key (acceso a una zona de memoria de esa región) y R_key (acceso remoto a una región de memoria previamente registrada en el adaptador).

Dominios protegidos

Los dominios protegidos permiten conceder distintos modos de acceso a las zonas de memoria registrada. Un consumidor establece qué zonas de memoria de las que tiene registradas son accesibles por sus colas.

Particiones

Las particiones (partitions) permiten aislar de forma virtual zonas de una subred InfiniBand. Este particionamiento no está ligado a límites establecidos por subredes, conmutadores o encaminadores. Son zonas virtuales que permiten aislar ciertas zonas de la subred.

Control de la tasa de inyección.

De cara a ser capaz de utilizar componentes que funcionen a diferentes velocidades, InfiniBand define un mecanismo de control de inyección que previene que un puerto pueda llegar a saturar a otro más lento. Por ejemplo, el control de flujo puede provocar congestión hacia atrás en la ruta hasta tal punto que congestione a otros enlaces que estén siendo afectados por ese cuello de botella. Por ello, cada puerto que funcione a velocidades mayores de 1X (2,5 Gb/s) tal que cada fuente tiene un tiempo de time-out asociado a cada destino, siendo ese valor el ratio entre las velocidades de la fuente y de los enlaces intermedios hasta llegar al destino. Por tanto, cada vez que el origen envía un paquete, debe esperar a que transcurra ese tiempo antes de enviar otro paquete a la misma fuente.

Verbs.

InfiniBand describe el comportamiento de la interfaz entre el adaptador del host y el sistema operativo por medio de un conjunto de comportamientos llamados verbos. Estos describen los parámetros necesarios para configurar y gestionar el adaptador, crear y liberar colas, configurar operaciones con las colas, hacer peticiones de envío a las colas, consultar el estado de las peticiones, etc. Con toda esta información, el diseñador de sistemas operativos deberá definir APIs adecuados.

Direccionamiento.

Cada nodo final puede tener uno o más adaptadores y a su vez estos pueden tener uno o más puertos. Además, cada adaptador tiene un número variable de pares de colas (QPs). Cada QP tiene asignado un número (queue pair number, QPN) por parte del adaptador, que lo identifica de forma única.

Cada puerto tiene un identificador local (LID) asignado por el Subnet Manager además de tener una dirección global asignada (GID). Dentro de la subred local los LIDs son únicos y se usan para encaminar los paquetes mientras que los GIDs son únicas en cualquier subred InfiniBand.

Cada conmutador de la subred tiene una tabla de encaminamiento que está basada los identificadores locales, tal que cada paquete cuando viaja por la subred, lleva incorporadas la dirección del puerto fuente (SLID) y la del puerto destino (DLID). En caso de que viaje entre varias subredes InfiniBand, las direcciones globales fuente (SGID) y destino (DGID) serán usadas por los encaminadores intermedios.

Por tanto, la dirección que identifica a dos entidades que se comunican será la combinación de las direcciones de los puertos (GID + LID) y de los pares de colas que interviene (QPN).

Multicast.

Cada grupo multidestino está identificado por una dirección LID y GID única. De esta forma, puede haber grupos multidestino en la misma subred o también entre distintas subredes. Cada nodo se une a un grupo multidestino indicando los LID de los puertos que van a participar. Esta información se distribuye a los conmutadores, que se configuran para saber a qué puertos deben viajar los paquetes de ese grupo multidestino. Evidentemente, se tiene en cuenta que no se formen ciclos en esas rutas multidestino.

Cuando un conmutador o un encaminador recibe un paquete multidestino lo reenvía por todos aquellos puertos a través de los que se accede a puertos que participen en el grupo multidestino, salvo por el puerto por donde llegó dicho paquete. De esta forma, el paquete llegará una sola vez a cada miembro del grupo multidestino.

Hasta ahora se han expuesto lo más brevemente posible algunas de las características de InfiniBand. A continuación se va a desarrollar una de las características que tiene alto grado de relación con todo lo expuesto en este trabajo.

Canales virtuales

Los canales virtuales (VL) constituyen un mecanismo para crear múltiples enlaces virtuales con un único enlace físico. Un canal virtual representa un conjunto de buffers de transmisión y recepción en un puerto (Figura 3.8).

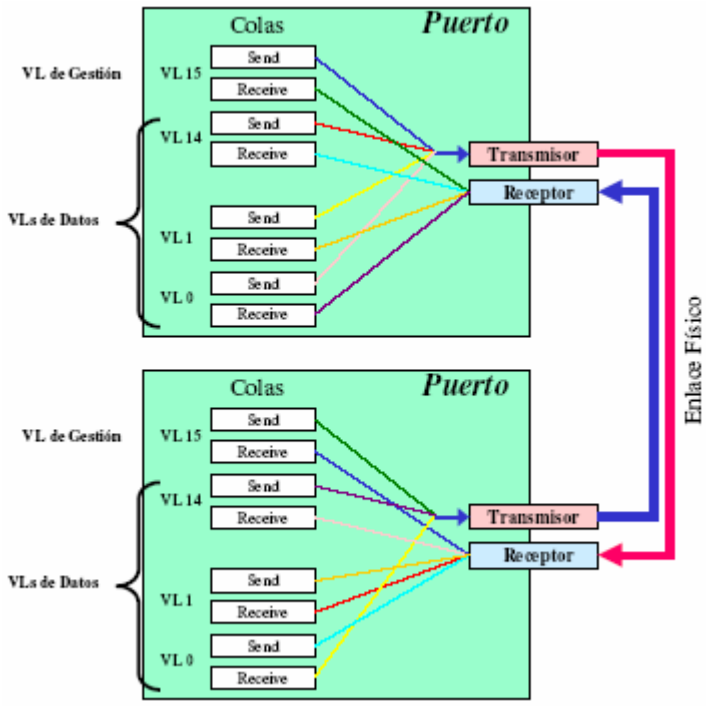


Figura 3.8: Multiplexación del enlace físico en canales virtuales.

Todos los puertos soportan un VL15 exclusivamente para la gestión de la subred. También deben soportar como mínimo un canal virtual para datos que es el VL0. El resto de canales virtuales de datos (VL1 a VL14) son opcionales. Dependiendo de la implementación, un puerto puede tener 16, 8, 4 ó 2 canales virtuales implementados.

El VL que en cada momento se usa viene configurado por el Subnet Manager, y esta decisión está basada en el campo nivel de servicio (Service Level, SL) que llevan los paquetes. En concreto, existe una tabla SLtoVLMappingTable que especifica la correspondencia entre los SLs y VLs.

El tráfico de cada VL no influye en el tráfico de otros VLs. Cada puerto mantiene control de flujo separado para cada VL de datos. Cada paquete incorpora en su cabecera un SL. Por otra parte, cada puerto mantiene la tabla de correspondencia entre SLs y VLs. Así, cuando el paquete va fluyendo por la subred es ese SL que tiene en su cabecera el que va indicando el VL a usar en cada caso.

Cada puerto debe adaptarse al mínimo número entre los que él incorpora y los que tiene el puerto con el que está conectado. Esto se realiza durante la fase de establecimiento de la subred.

3.3 ESTRUCTURA ARQUITECTÓNICA

El funcionamiento de InfiniBand se describe mediante la interacción de una serie de niveles. El protocolo que gobierna cada nivel es independiente del resto de niveles. Cada nivel usa los servicios que le proporciona el nivel inferior y a su vez proporciona una serie de servicios al nivel inmediatamente superior.

Los distintos niveles que distingue InfiniBand, y sus modos de actuación se muestran en la Figura 3.9. Por encima del nivel de transporte están los manejadores de los sistemas operativos.

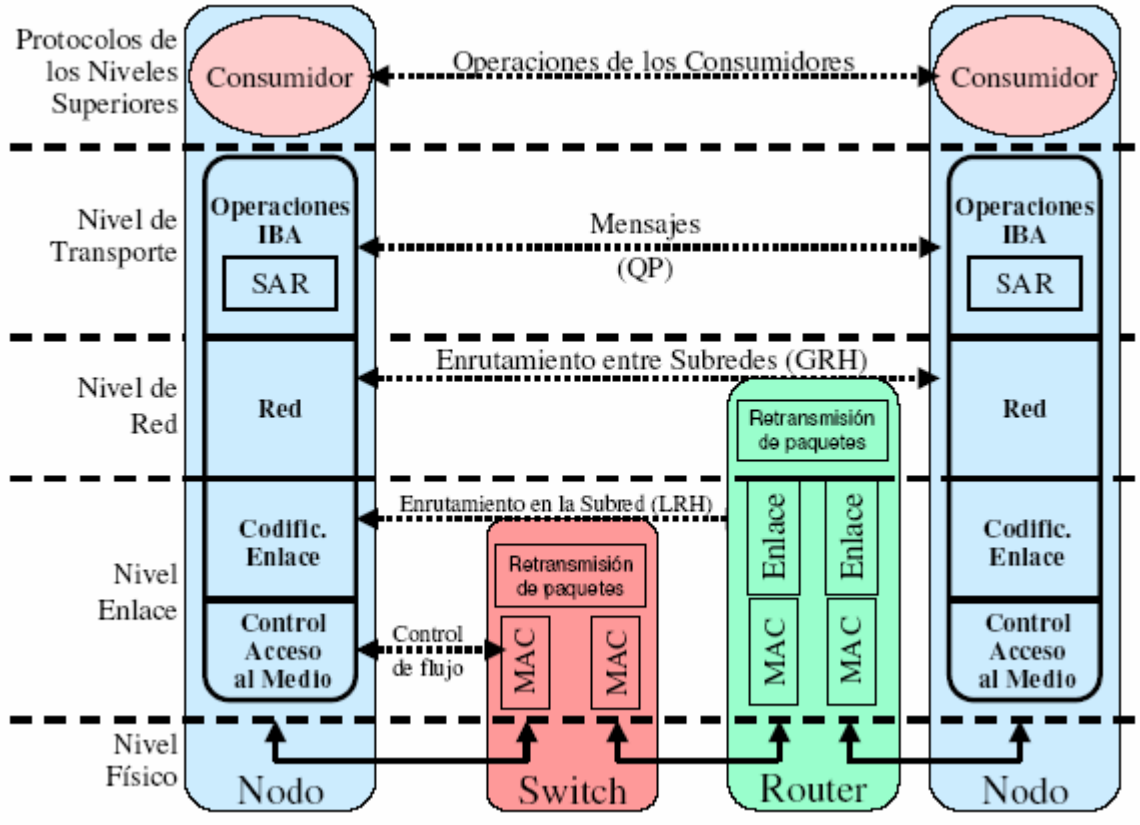


Figura 3.9: Niveles en la arquitectura de InfiniBand.

3.3.1 Nivel físico

El nivel físico especifica cómo se trasladan los bits al cable. InfiniBand utiliza el sistema de codificación 8B/10B. El nivel físico varía dependiendo del medio físico que se vaya a utilizar. InfiniBand especifica tres tipos de medios físicos: par trenzado, fibra óptica o circuito en placa. El nivel físico también especifica los símbolos que se usarán para indicar principio y final de paquete, datos, espacio entre paquetes, el protocolo de señalización a utilizar, etc.

La capa física está descrita en el volumen 2 de la especificación de InfiniBand [Inf01].

3.3.2 Nivel de enlace

El nivel de enlace describe los formatos de paquete a usar y los protocolos para las operaciones con ellos. También son tareas de este nivel el control de flujo y el encaminamiento de los paquetes dentro de la misma subred.

Se pueden distinguir dos tipos de paquetes:

- **Paquetes de gestión**. Se utilizan para mantenimiento de las operaciones de enlace. Estos paquetes se crean y se consumen en el nivel de enlace, y no están sujetos a control de flujo. Se suelen usar para operaciones de negociación de parámetros entre los puertos de cada lado del enlace, como velocidad, tasa de inyección, etc. También son estos paquetes los que se utilizan para control de flujo y mantenimiento de la integridad del enlace. Estos paquetes sólo se envían entre ambos extremos de un enlace, y nunca se retransmiten hacia otro destino.

- **Paquetes de datos.** Estos son los paquetes que intercambian las operaciones de InfiniBand y pueden contener varias cabeceras, algunas de las cuales son opcionales.

La Figura 3.10 muestra el formato del paquete del nivel de enlace. Se observa como se encapsulan los paquetes de los niveles superiores en los paquetes que envían los niveles inferiores de la arquitectura.

La *Local Route Header* (LRH) siempre está presente e identifica los puertos origen y destino donde los conmutadores intermedios encaminarán el paquete. Esta cabecera también contiene el nivel de servicio y el canal virtual que debe utilizar el paquete. Este

canal virtual podría cambiar a lo largo de la ruta, pero el resto de campos de esta cabecera permanecerán inalterados durante todo el trayecto.

El control de flujo que realiza la capa de enlace está basado en créditos. En cada enlace, el extremo receptor envía al origen los créditos disponibles. Estos créditos se contabilizan para cada canal virtual, e indican el número de paquetes de datos que el receptor puede aceptar en un canal virtual. El extremo origen no enviará paquetes a menos que el receptor le haya indicado previamente que tiene espacio disponible para alojarlos. Por otra parte, el canal virtual número 15 (el canal virtual de gestión) no está sujeto a control de flujo.

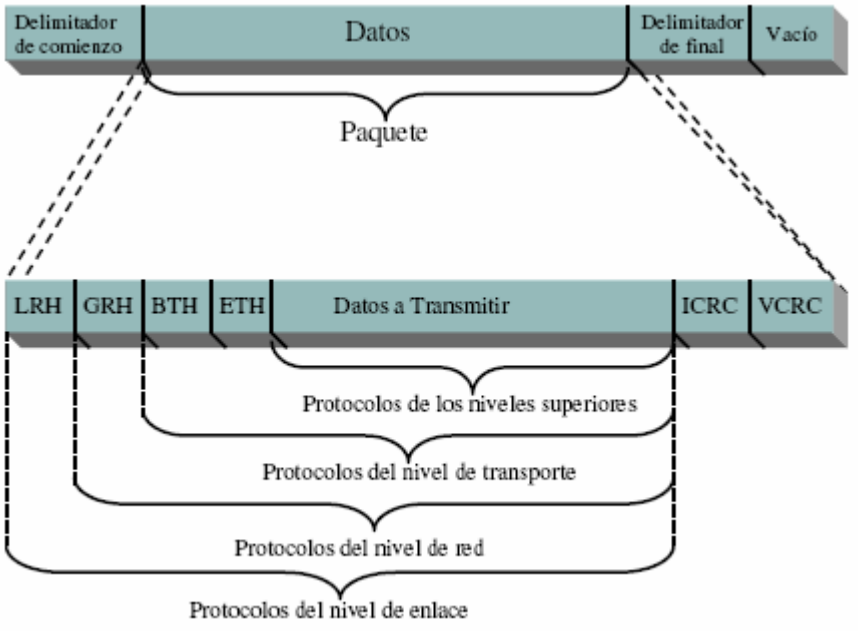


Figura 3.10: Formato del paquete de datos en InfiniBand.

3.3.3 Nivel de red

El nivel de red describe el protocolo para encaminar un paquete entre distintas subredes. Este nivel define una *Global Route Header* (GRH) que debe estar presente en los paquetes que tengan que ser encaminados entre dos o más subredes. La GRH identifica los puertos origen y destino usando direcciones globales (GID) en el formato de una dirección IPv6.

Los encaminadores se basan en la información que contiene la GRH para encaminar los paquetes hacia su destino. Conforme los paquetes van avanzando entre subredes los encaminadores van modificando el contenido del GRH y reemplazando el LRH. Sin

embargo, los identificadores globales de los nodos fuente y destino no variarán a lo largo de toda la ruta.

El nodo origen pone el identificador global (GID) del destino en la GRH pero el identificador local (LID) del encaminador en la LRH. Cada conmutador intermedio hace avanzar el paquete hacia la siguiente subred hasta que llega a la subred destino. El último encaminador reemplaza la LRH usando el identificador local del destino.

3.3.4 Nivel de transporte

Los protocolos de enlace y de red llevan un paquete al destino deseado. La parte de nivel de transporte del paquete se encarga de hacer que se entregue el paquete en la cola (QP) apropiada, indicándole además cómo procesar los datos contenidos en el paquete.

El nivel de transporte es el responsable de segmentar una operación en varios paquetes si los datos a transmitir exceden el tamaño máximo de paquete (MTU). El QP en el extremo final reensambla los paquetes para formar la secuencia de datos que se quiso enviar.

La *Base Transport Header* (BTH) está en todos los paquetes, salvo en los datagramas RAW. Esta cabecera especifica el QP destino e indica el código de operación, el número de secuencia del paquete y la partición a la que pertenece.

Cuando ocurre un error en la secuencia es que se ha perdido algún paquete intermedio. En ese caso, el receptor descartará los paquetes siguientes que pudieran llegar hasta que llegue el paquete perdido. En el caso del servicio sin confirmación el receptor detecta un paquete perdido, aborta la operación y descarta todos los paquetes siguientes hasta que comience otra operación nueva.

El nivel de transporte es el núcleo central de la arquitectura de InfiniBand. Los servicios del nivel de transporte se acceden desde los QPs, donde cada QP se configura para una clase de servicio especifico.

3.3.5 Protocolos de los niveles superiores

InfiniBand soporte cualquier número de protocolos de nivel superior utilizados por varios usuarios (Figura 3.11). También define los mensajes y protocolos para ciertas

funciones de gestión. Estos protocolos están separados en Servicios de Subred y Gestión de la Subred.

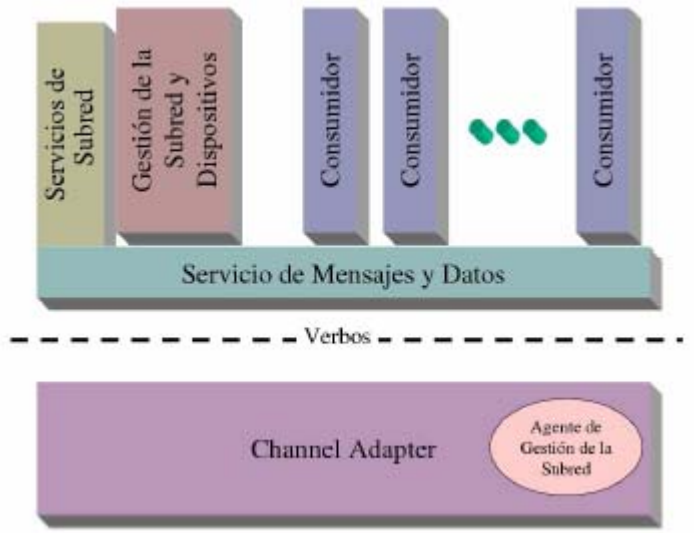


Figura 3.11: Niveles superiores en InfiniBand.

InfiniBand es una plataforma para que las aplicaciones intercambien información.

3.4 INFRAESTRUCTURA DE GESTIÓN

La especificación de InfiniBand define los mensajes y protocolos para muchas de las tareas de gestión de la red. Estos protocolos de gestión se pueden agrupar en gestiones de la subred y servicios de la subred, teniendo cada uno de esos grupos propiedades distintas.

En cada subred debe existir un *Subnet Manager* encargado de configurar la red y administrar la información necesaria para las operaciones de la subred. InfiniBand define la infraestructura básica para permitir diversas clases de gestión (*Management Classes*). Cada clase de gestión permite gestionar diversos aspectos de la red de forma independiente.

Para permitir la consistencia entre distintas clases, la infraestructura global de gestión define un conjunto común de acciones de gestión llamados métodos que pueden usar todas las clases. Cada clase puede definir los atributos a los que se les podrán aplicar esos métodos. De hecho, una clase define qué atributos son apropiados para cada método.

3.4.1 Clases de gestión

Algunas de estas clases son obligatorias y otras opcionales, y depende de las implementaciones el que aparezcan o no en los productos finales. Ciertas clases necesitan un agente especial que proporcione ese servicio, mientras que otras las proporciona directamente el Subnet Manager (Ejemplo: *Subnet Management*. Especifica los procedimientos que permiten al Subnet Manager descubrir, configurar y gestionar la subred).

3.4.2 Elementos de gestión

Sobre tres tipos de elementos de gestión se basa la infraestructura de gestión de la red: agentes de servicio, gestores de clase (*class managers*) y clientes y aplicaciones de gestión (Figura 3.12).

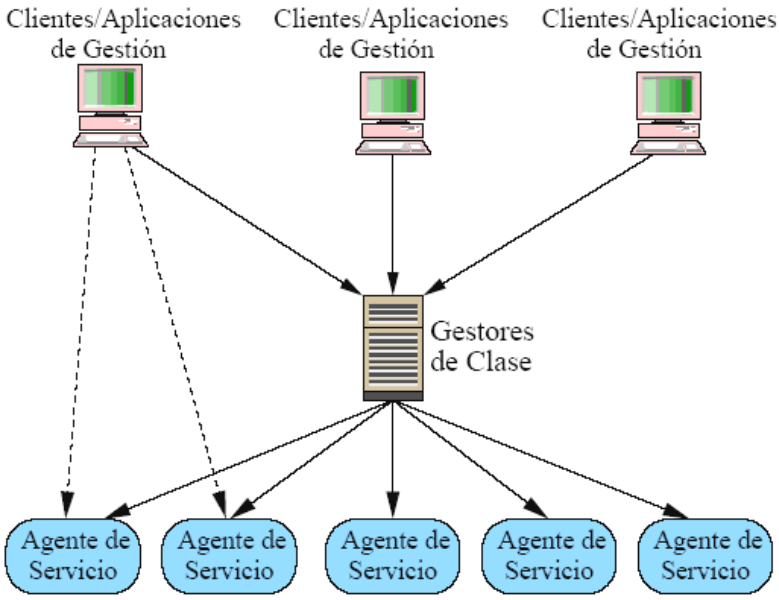


Figura 3.12: Elementos de gestión en InfiniBand.

Un agente de servicio es una entidad que reside en los nodos gestionados. Tiene una interfaz conocida para recibir y responder a las peticiones relacionadas con la gestión. La arquitectura de InfiniBand permite cualquier número de agentes de gestión en cada nodo. Lo usual es que en cada puerto haya un agente por cada clase de gestión. InfiniBand especifica cómo descubrir y configurar esos agentes de gestión.

Por otra parte, hay un gestor de clase en aquellos casos en que se necesite tener una entidad central de gestión que interactúe con los agentes de gestión. Estos gestores de clase

son útiles por ejemplo para realizar una recolección eficiente de información de gestión distribuida entre los nodos, o para asegurar la atomicidad en el acceso a ciertos agentes de servicio.

También comentar que los clientes y aplicaciones de gestión suelen estar ubicados normalmente en hosts. Estos clientes y aplicaciones de gestión acceden a los agentes de servicio y a los gestores de clase para obtener información relativa a la gestión y solicitar funciones de gestión.

3.4.3 Mensajes y métodos de gestión

La infraestructura para la gestión define la estructura y formato de los paquetes de gestión (*Management Datagram*, MAD). Estos paquetes de gestión son los que intercambian los elementos de gestión. Estos mensajes se envían usando la clase de servicio no fiable y sin conexión (UD). Todos los paquetes MAD deben tener exactamente 256 bytes, y consisten en una cabecera MAD y los datos correspondientes. El contenido del campo de datos variará en función de la clase concreta para la que se esté usando.

Las especificaciones también definen los métodos que usarán los elementos de gestión para intercambiar información. Estos procedimientos incluyen:

- **Gets y Sets:** Leen y modifican los atributos de los objetos gestionados.

- **Traps:** Permiten obtener información de ciertos eventos de forma asíncrona por parte de un agente de servicio.

- **Subscription:** Es la forma en que las aplicaciones piden recibir grupos.

- **Reports:** Es la forma en que el gestor de una clase puede distribuir traps y otras informaciones a las aplicaciones que lo hubieran solucionado.

3.4.4 Subnet Manager

El *Subnet Manager* (SM) se encarga de configurar y mantener la subred. El SM utiliza un formato especial de paquete llamado *Subnet Management Packet* (SMP) que tiene prioridad sobre el resto de paquetes. Un paquete SMP es un tipo especial de paquete MAD.

Algunas de las funciones del SM son descubrir la topología física de la subred, configurar los nodos y los conmutadores (LIDs, GIDs, Claves, etc.), calcular y distribuir a los conmutadores las tablas de encaminamiento en la subred, recibir peticiones de los agentes locales de gestión, etc.

Cada nodo contiene un Agente de Gestión de la Subred (Subred Management Agent, SMA) que responde a los paquetes de gestión enviados por el SM. Cada puerto tiene una cola llamada QP0 dedicada a enviar y recibir paquetes de control. La interfaz que utiliza QP0 es el *Subnet Management Interface* (SMI) (Figura 3.13).

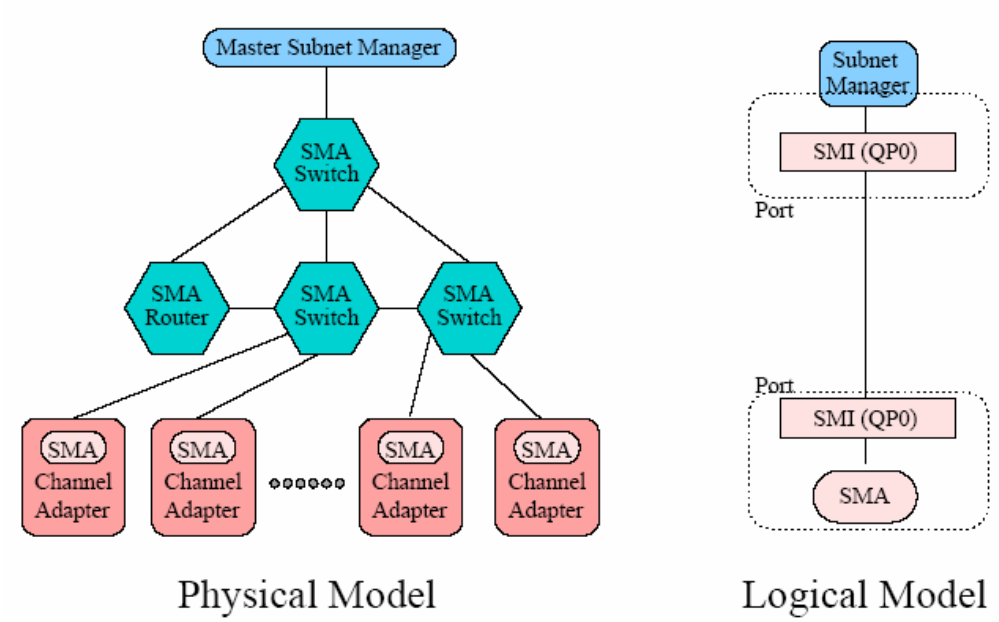


Figura 3.13: Modelo de gestión de la subred en InfiniBand.

Los paquetes de control sólo se envían desde y hacia un SMI, y no se pueden enviar otro tipo de paquetes a un SMI. Los paquetes de control nunca saldrán de la subred (los encaminadores no los reenvían) y siempre utilizan el canal virtual VL_{15} que está totalmente dedicado a este tipo de tráfico, siempre tiene prioridad sobre el resto de canales virtuales y no está sujeto a control de flujo.

Pueden existir varios SMs, pero sólo uno de ellos (llamado *Master*) puede estar activo en un momento dado. Si un SM descubre otro SM utiliza un protocolo específico para determinar cuál de ellos actuará como Master, quedando el resto a la espera por si el Master dejara de funcionar, pasar a actuar como tal. Solamente el Subnet Manager Master es el autorizado para configurar los nodos y conmutadores.

El Subnet Manager Master se asegura el control exclusivo sobre los SMA al establecer una clave de control (M_Key) en cada puerto. Posteriormente, los paquetes de

control que intercambie con ese puerto contendrán la clave que se les suministró. Sólo serán válidos los paquetes de control que contengan la clave correcta, ignorándose el resto de paquetes de control. La clave M_Key expira transcurrido un cierto periodo, tras el cual, el SMA vuelve a permitir establecer una nueva clave. De este modo, si el SM Master muere, otro SM puede actuar como Master.

3.4.5 Subnet Administration

Sólo puede haber un *Subnet Administrador* (SA) en cada subred, y se supone que debe estar situado en el mismo nodo que el Subnet Manager Master.

El administrador de la subred actúa como gestor de clase para la gestión de la subred. Este SA proporciona a los clientes y aplicaciones de control los medios para obtener y enviar información de control. Este SA es el encargado de interactuar con el Subnet Manager Master. Además también proporciona a los nodos la posibilidad de que estos le soliciten recibir notificación sobre las traps.

3.4.6 Servicios generales

Las especificaciones de InfiniBand describen una infraestructura de servicio general para las tareas de control llamadas no críticas. Como tareas no críticas se entiende todo lo necesario para el control de la subred excepto la configuración y su mantenimiento posterior. Todas las clases de servicios (excepto la SM) usan el modelo de servicio general (Figura 3.14).

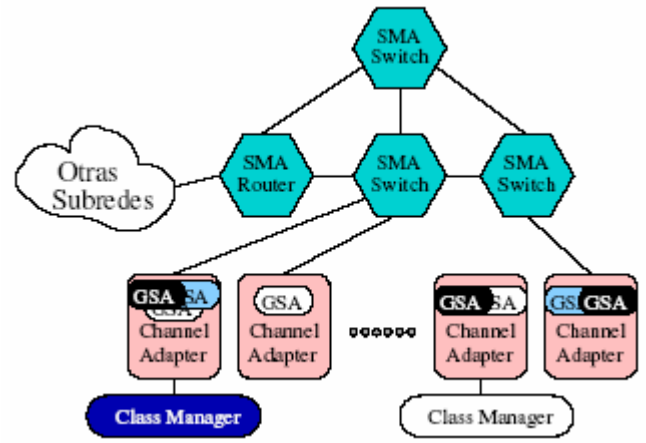


Figura 3.14: Modelo de servicios generales para la gestión en InfiniBand.

El modelo de servicios generales se basa en intercambiar información entre:

- Un gestor de clase y cliente/aplicaciones de control.

- Un gestor de clase y un agente de servicio general (*Global Service Agent*, GSA).

- Directamente entre un cliente/aplicación de control y un GSA.

Lo normal es que para cada clase haya varios clientes o aplicaciones de control, un gestor de la clase, y muchos GSAs (al menos uno por puerto). El GSA es la entidad de un puerto que habilita esa clase de gestión para ese puerto. Un GSA responde a peticiones relativas a su clase del gestor de la clase o de aplicaciones de control. Un ejemplo podría ser el agente de gestión de dispositivo que tienen todos los nodos de E/S para responder al gestor de recursos de E/S de cada host.

Las especificaciones de InfiniBand definen una interfaz para los servicios generales llamada *General Service Interface* (GSI), que viene representado por un QP reservado (el QP1). Cualquier petición, sea ésta de la clase que sea, se envía al QP1 del destino, y es el GSI quien se lo hace llegar al GSA correspondiente. El GSI permite redirigir la petición a otro QP y/u otro puerto.

Las aplicaciones de gestión, los gestores de clase y los GSAs pueden estar ubicados en cualquier nodo (Figura 3.15).

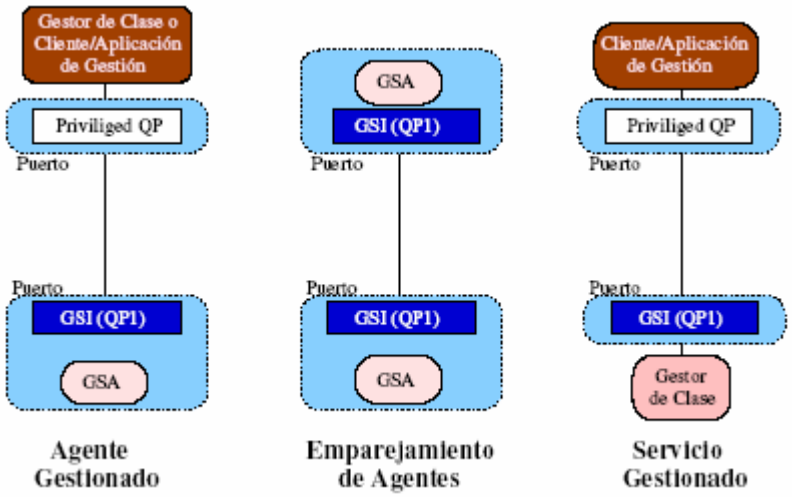


Figura 3.15: Modelo de servicios generales para la gestión en InfiniBand.

Como ejemplo de agente gestionado se podría citar un gestor de recursos de E/S en un host que accede al agente de gestión de dispositivo en una unidad de E/S. Como ejemplo

de emparejamiento de agentes se podría citar el intercambio entre gestores de comunicación cuando se establece una conexión entre dos adaptadores. Por último, un ejemplo de un servicio gestionado podría ser el SA como un gestor de la clase de administración de la subred, que responde a los hosts sobre la información de gestión.

3.4.7 Comunication management

La gestión de comunicaciones especifica los protocolos y mecanismos para establecer, mantener y liberar conexiones. También se encarga de la resolución de identificadores, que permite a los usuarios del servicio no fiable y sin conexión identificar el QP que soporta un servicio en particular.

En cada adaptador hay un *Communication Manager* (CM) accesible por cualquier de sus puertos por medio del GSI de dicho puerto. Un CM interactúa con otros CMs por medio de los paquetes de la clase CommMgt. Las especificaciones de InfiniBand definen el comportamiento que deben tener los CM que interactúan. Sin embargo, la interfaz entre el QP cliente y el CM dependerá del sistema operativo y queda fuera de las especificaciones de InfiniBand.

3.5 CALIDAD DE SERVICIO

InfiniBand proporciona varios mecanismos que permiten al administrador de la subred gestionar distintas garantías de calidad de servicio, tanto para servicios con conexión como sin conexión. En los servicios con conexión se puede intentar garantizar calidad de servicio. Sin embargo, en los servicios sin conexión no es posible dar garantías, pues no se conoce a priori los requisitos que necesitan las aplicaciones. Para este tipo de servicio se puede configurar la subred con algún tipo de mecanismo similar a los que hacen los servicios diferenciados.

Los mecanismos que proporciona InfiniBand para conseguir proporcionar calidad de servicio son básicamente: niveles de servicio, canales virtuales, arbitraje en dichos canales virtuales y particiones.

3.5.1 Niveles de servicio

InfiniBand define el atributo nivel de servicio que deben incorporar todos los paquetes. Se permiten 16 niveles de servicio distintos, aunque el propósito de cada uno de

ellos no está especificado. De esta forma, dependerá de las implementaciones o del administrador de la subred cómo distribuir los tipos de tráficos existentes entre los niveles de servicio disponibles.

Los niveles de servicio permiten segregar los distintos tipos de tráfico existentes de forma que se pueda señalar cada tipo de tráfico distinto, para luego ser capaz de proporcionar a cada tipo de tráfico un trato distinto, en función de sus necesidades.

3.5.2 Correspondencia SL a VL

Tal y como se indicó en la Sección 3.2.3, los puertos en InfiniBand tienen canales virtuales. Un canal virtual representa un conjunto de buffers para transmisión y recepción en un puerto.

Cada puerto puede tener un número de canales virtuales diferentes (Figura 3.16). Durante la fase de configuración ambos extremos de enlace deben ponerse de acuerdo para funcionar con el número mínimo de enlaces de los dos.

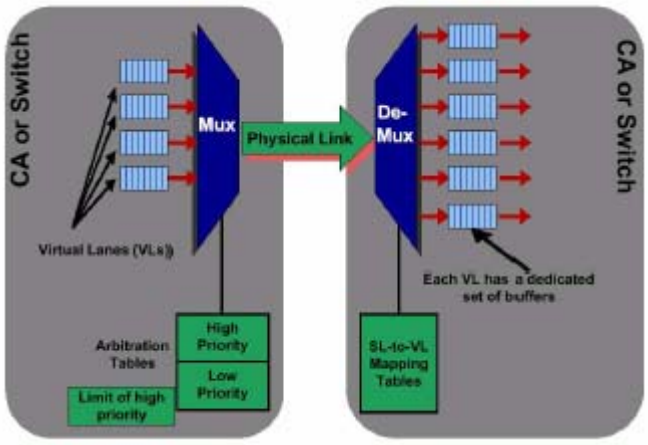


Figura 3.16: Funcionamiento de los canales virtuales en un canal físico.

Por otra parte, cada paquete contiene un Service Level (SL) que determina qué VL se usará en cada uno de los enlaces por donde pasa. De esta forma, es posible que puedan funcionar en la misma subred dispositivos que implementan distinto número de canales virtuales.

Cada puerto de salida (en conmutadores, encaminadores y nodos terminales) tiene una tabla de correspondencia entre los SLs y los VLs (SLtoVLMappingTable) configurada por las entidades de gestión de la red. Una correcta configuración de esta tabla hará que cada

tipo de tráfico, previamente segregado en distintos niveles de servicio, use canales virtuales diferentes.

3.5.3 Particiones

Utilizando esta característica de InfiniBand se puede conseguir aislar determinadas zonas de la subred del tráfico destinado a otras zonas.

Cada puerto o interfaz de la subred es miembro de al menos de una partición y puede pertenecer a varias de ellas. A cada adaptador de la subred le es asignada por el gestor de particiones cierto número de *claves de partición* (P_Keys). Estas claves se insertan en los paquetes que se envían luego a la red y cuando una entidad de la subred recibe un paquete con una P_Key para la que no está habilitado, debe descartar dicho paquete.

Los conmutadores y los encaminadores pueden ser configurados por el gestor de particiones para que también descarten paquetes aislando físicamente zonas de la subred.

Para utilizar las particiones como mecanismo para mejorar la calidad de servicio, se pueden dedicar algunos SLs a ciertas particiones. Esto, junto con lo anteriormente expuesto, permite aislar el tráfico de las distintas particiones en distintos canales virtuales, y garantizar a cada partición una calidad de servicio distinta.

3.5.4 Arbitraje de los puertos de salida

Los puertos de salida (tanto en los conmutadores, como en los encaminadores y en los nodos terminales) incorporan una tabla de arbitraje que permite especificar la prioridad que tendrá cada canal virtual a la hora de enviar paquetes a través de ese puerto de salida.

Es obligatorio implementar esta tabla si el puerto tiene más de dos canales virtuales. El arbitraje se realiza sólo entre los canales virtuales de datos, pues el canal virtual del tráfico de control siempre tiene preferencia.

Cada tabla de arbitraje (Figura 3.17) está formada por dos tablas, una para gestionar el tráfico de los canales virtuales de alta prioridad y otra para los canales virtuales de baja prioridad. Sin embargo, InfiniBand no especifica qué es alta y baja prioridad, quedando este aspecto a criterio del administrador de la subred.

Cada una de esas tablas tiene un funcionamiento cíclico y ponderado. Tanto la tabla de alta prioridad como la de baja prioridad tienen un máximo de 64 entradas. Cada entrada especifica un canal virtual y un peso. El peso indica la cantidad de unidades de 64 bytes a enviar por ese canal virtual. El peso debe estar entre 0 y 255, y siempre se redondea a un paquete completo.

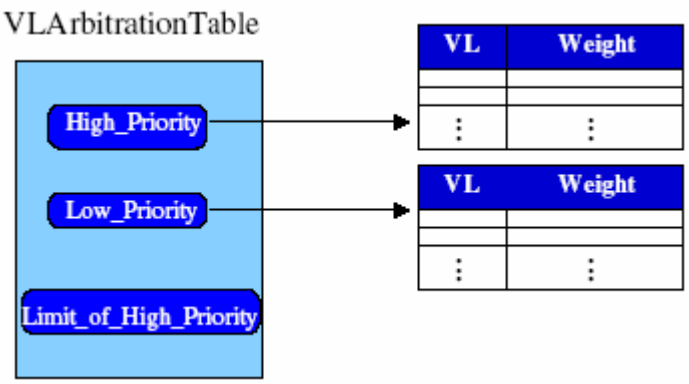


Figura 3.17: Estructura de la tabla de arbitraje.

El valor LimitOfHighPriority determina la cantidad máxima de información que puede sacarse de los canales virtuales de alta prioridad, antes de pasar a enviar un paquete de un canal virtual de los considerados de baja prioridad. En concreto, los canales virtuales situados en la tabla de alta prioridad pueden enviar LimitOfHighPriority × 4096 bytes antes de enviar un paquete de baja prioridad. Una vez enviado el paquete de un canal virtual de baja prioridad (si había alguno listo para enviarse), se vuelve a la tabla de alta prioridad y se continúa por la entrada donde se quedó y con el peso que le restaba.

Una característica importante del funcionamiento de la tabla de arbitraje es que si no hay paquetes de alta prioridad para ser transmitidos, se puede pasar a transmitir paquetes de baja prioridad. De esta forma no se desperdicia ancho de banda, y cuando no haya tráfico de alta prioridad se puede pasar a transmitir otro tipo de tráfico de menos prioridad aprovechando el ancho de banda disponible.

3.6 RESUMEN

En este capítulo se ha realizado un breve recorrido por la especificación de InfiniBand para conocer mejor parte de su funcionamiento, aunque este proyecto se basa principalmente en algunas serán utilizadas. El estándar InfiniBand nació como la unión de dos propuestas con objetivos comunes, que yendo por separado, hubieran tenido un futuro poco prometedor. Uno de esos objetivos era el de sustituir la tecnología PCI que tantos cuellos de botella provoca, pero yendo aún más allá de ser una simple sustitución,

considerándose más como una apuesta revolucionaria capaz de igualar las prestaciones que proporcionaban hasta ese momento los supercomputadores y a un mayor coste económico.

Se han comentado ciertos componentes de InfiniBand como *channel adapter*, además de algunas características que posee, de las cuales, la más importante en lo que a este proyecto se refiere es la de *canal virtual* (*virtual lane*). También se ha presentado el sistema de niveles que InfiniBand plantea viendo en cierta medida la funcionalidad que cada uno de los niveles proporciona.

Se ha hecho un recorrido por la infraestructura de gestión de InfiniBand, destacando la figura del *Subnet Manager*, que es el encargado de realizar la configuración de la red así como de administrar la información necesaria para las operaciones de la subred.

Para terminar este recorrido por InfiniBand, siendo además la parte más relacionada con el contenido de este proyecto, se han indicado las características que proporciona InfiniBand con el fin de proporcionar al consumidor calidad de servicio. Estas son *niveles de servicio*, *correspondencia entre niveles de servicio y canales virtuales, particiones* y *arbitraje de los canales virtuales*. Esta última característica de InfiniBand será la que más se va a considerar ya que la finalidad de las propuestas realizadas son para el manejo de la tabla de arbitraje de los canales virtuales.

Capítulo 4

INTRODUCCIÓN A LA TABLA DE ARBITRAJE: CONCEPTOS, ALGORITMOS Y GESTIÓN

Como se indicó en el capítulo de Introducción, el objetivo de este proyecto es elaborar y evaluar un conjunto de algoritmos. Estos algoritmos deben servir para gestionar la tabla de arbitraje de los puertos de salida de conmutadores, encaminadores y terminales de una red InfiniBand. La forma en la que se debe producir esa gestión viene determinada por el trabajo presentado en [Alf03]. En ese trabajo se desarrolla un modelo formal para el tratamiento de la tabla de arbitraje, así como una serie de propiedades y teoremas derivados.

Hay que indicar que el trabajo citado plantea una metodología para garantizar latencia máxima y ancho de banda a los flujos de datos que requieren el uso de la red. Esa metodología indica la forma en la que se tienen que asignar las entradas de las tablas de arbitraje correspondientes a los puertos de salida de conmutadores, encaminadores y terminales. Todo ello queda plasmado en el modelo formal previamente mencionado.

Así pues, y antes de presentar los algoritmos que se han desarrollado durante este proyecto, parece lógico revisar aunque sea brevemente ese modelo formal, en la medida en que dichos algoritmos se ajustan a dicho modelo.

4.1 MODELO FORMAL DE LA TABLA DE ARBITRAJE

La tabla de arbitraje está compuesta por un número fijo de entradas. En InfiniBand son 64, aunque la teoría desarrollada es válida para cualquier número de entradas. En el modelo propuesto en [Alf03] usa para identificar las entradas una numeración basada en la *permutación bit-reversal*. Esta permutación es tal que aplicada a una secuencia $m = b_{n-1}b_{n-2}...b_1b_0$ de dígitos, obtiene como resultado la secuencia $\Re(m) = b_0b_1...b_{n-2}b_{n-1}$. En la Figura 4.1 se ve un ejemplo de la numeración utilizada y su comparación con la numeración correlativa habitual.

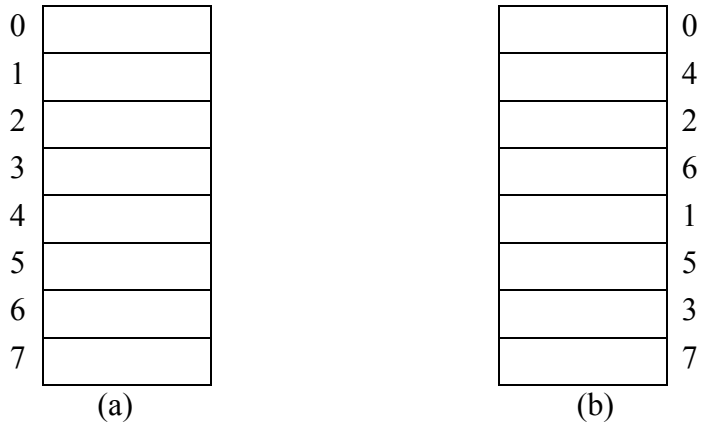


Figura 4.1: Tabla de 8 entradas, (a) numeración correlativa de las entradas, (b) numeración basada en la aplicación de la permutación bit-reversal.

En la tabla de arbitraje se pueden distinguir diferentes secuencias de entradas en función de la distancia entre dos entradas consecutivas de la secuencia. En el ejemplo de la Figura 4.1 (b) podemos tener 1 secuencia de 8 entradas separadas a distancia 1 formada por todas las entradas, 2 secuencias disjuntas formadas por 4 entradas y separadas entre sí a distancia 2 y que son (0, 2, 1, 3) y (4, 6, 5, 7), 4 secuencias de 2 entradas a distancia 4 que son (0, 1), (4, 5), (2, 3) y (6, 7), 8 secuencias de distancia 8 formadas cada una por una única entrada. Cada una de estas secuencias forma un conjunto, que son los encargados de atender las peticiones que reciba la tabla de arbitraje.

Conjuntos con entradas a distancia 1 entre sí: {0, 1, 2, 3, 4, 5, 6, 7}
Conjuntos con entradas a distancia 2 entre sí: {0, 1, 2, 3} {4, 5, 6, 7}
Conjuntos con entradas a distancia 4 entre sí: {0, 1} {2, 3} {4, 5} {6, 7}
Conjuntos con entradas a distancia 8 entre sí: {0} {1} {2}{3} {4} {5} {6} {7}

En la Figura 4.2, se han representado los conjuntos que se pueden formar utilizando todas las entradas que forman la tabla de arbitraje. Además, se puede apreciar como los

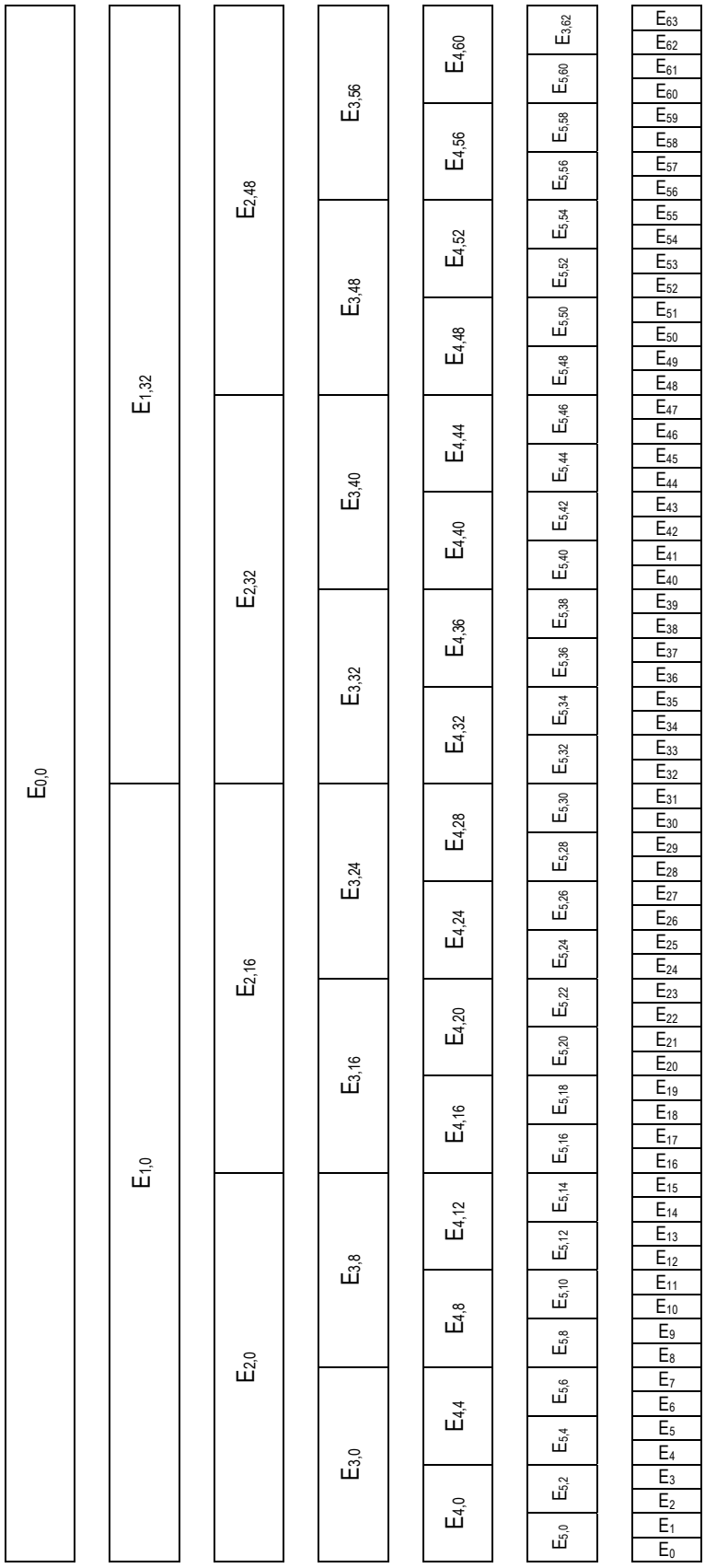


Figura 4.2: Descomposición en forma de árbol binario de todos los conjuntos posibles de entradas de la tabla de arbitraje, donde cada nivel se corresponde con un tipo de petición distinto.

conjuntos forman una estructura de árbol binario con siete niveles. Un rectángulo del nivel 6 de la Figura 4.2 representa a un conjunto con una sola entrada de la tabla, mientras que los de niveles menores de 6, son conjuntos con varias entradas. Los conjuntos de la tabla se identifican como $E_{i,j}$ siendo *i* el nivel al que pertenece el conjunto y *j* el número de la primera entrada perteneciente a ese conjunto. Es importante indicar que el valor de *i* depende de la distancia que haya entre dos entradas consecutivas de un mismo conjunto.

En la Tabla 4.1 se utilizan algunos de los conjuntos de la Figura 4.2 como ejemplos en los que se indican las entradas que poseen.

Conjunto	Nº Entradas	Entradas que lo forman
$\mathbf{E_{2,16}}$	16	16, 24, 20, 28, 18, 26, 22, 30, 17, 25, 21, 29, 19, 27, 23, 31
$\mathbf{E_{6,12}}$	1	12
$\mathbf{E_{4,60}}$	4	60, 62, 61, 63
$\mathbf{E_{3,40}}$	8	40, 44, 42, 46, 41, 45, 43, 47

Tabla 4.1: Entradas que conforman algunos conjuntos de la Figura 4.2.

A partir de ahora, y con el objetivo de hacer más sencillos y claros los comentarios aquí presentados, se ha considerado una tabla con 5 niveles únicamente, y no con los 7 que realmente tienen las tablas de arbitraje en InfiniBand. De esta forma, las representaciones de la tabla y de la estructura auxiliar que más adelante se explica, podrán tener un tamaño que permita apreciar mucho mejor lo que se quiere indicar con ellas.

Cada conjunto se puede dividir en dos subconjuntos disjuntos, teniendo cada uno de ellos la mitad de entradas. Así por ejemplo, el conjunto $E_{2,16}$ puede dividirse en dos subconjuntos que son $E_{3,16}$ y $E_{3,24}$ (ver Figura 4.2), siendo además disjuntos entre sí. A partir de esto, se puede decir que un conjunto es el *padre* de los dos que hay en el nivel inferior al suyo, siendo denominados éstos *hijos* del anterior y *hermanos* entre sí. Siguiendo con el conjunto $E_{2,16}$, éste forma parte de los conjuntos $E_{0,0}$ y $E_{1,0}$. Por tanto, todos los conjuntos de niveles superiores que contengan a uno dado se dice que son conjuntos *ancestros* de este último.

Un conjunto está *libre* si todas sus entradas no están siendo utilizadas (ocupada en caso contrario), y es denominado conjunto *singular* si a la vez de estar libre, su hermano no lo está. Como el conjunto $E_{0,0}$ no tiene hermano, no podrá ser, por definición, un conjunto singular.

En la Figura 4.3 se puede ver una tabla de arbitraje (de sólo 16 entradas) en la que se han sombreado los conjuntos que están libres, estando el resto ocupados. De los conjuntos

libres, destacan $E_{3,0}$, $E_{2,4}$ y $E_{1,8}$, los cuales son conjuntos singulares, y están sombreados con un color más oscuro que el resto de conjuntos libres.

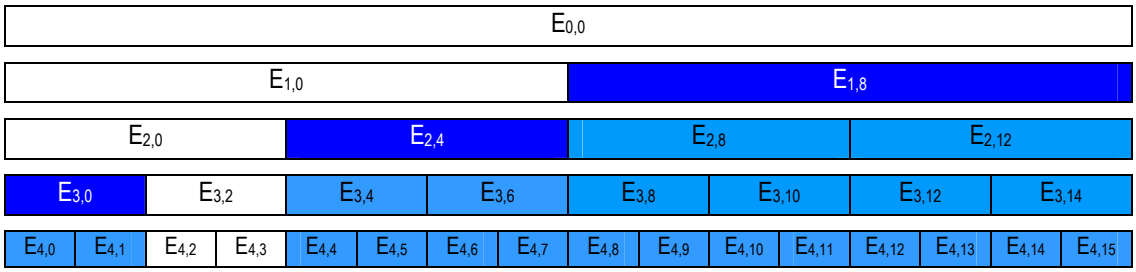


Figura 4.3: Tabla de arbitraje con tres conjuntos singulares.

Utilizando el concepto de conjunto singular se explican los términos de tabla *normalizada* y de tabla *ordenada*, pues dichas situaciones están marcadas por el número de conjuntos singulares que haya y por la situación que tengan entre ellos en la tabla de arbitraje.

Una tabla está *normalizada* si a lo sumo hay un conjunto singular por nivel. Es decir, en un nivel puede que no haya un conjunto singular, pero en el caso de que haya, éste ha de ser único. La tabla mostrada en la Figura 4.3 está normalizada porque en el nivel 4 no hay ningún conjunto singular, y los niveles 1, 2 y 3 poseen uno ($E_{1,8}$, $E_{2,4}$ y $E_{3,0}$). Sin embargo, la tabla de arbitraje de la Figura 4.4 no está normalizada, ya que el nivel 3 posee dos conjuntos singulares ($E_{3,0}$ y $E_{3,4}$).

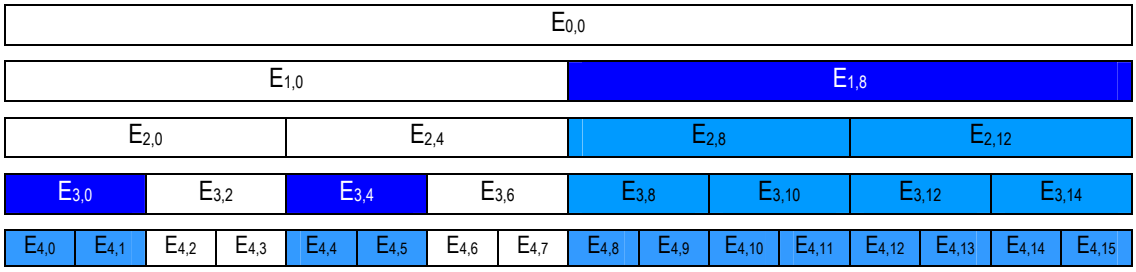


Figura 4.4: Tabla de arbitraje no normalizada y ordenada.

Una tabla está *ordenada* si los conjuntos singulares están ordenados de menor a mayor tamaño de izquierda a derecha. En la Figura 4.4 se ve que la tabla está ordenada, puesto que se respeta el orden de menor a mayor que implica que una tabla esté ordenada. Los conjuntos $E_{3,0}$ y $E_{3,4}$ están más a la izquierda que el conjunto $E_{1,8}$ que es de mayor tamaño. Un ejemplo de tabla no ordenada es la tabla de la Figura 4.5, en la que se puede observar como $E_{2,0}$ está más a la izquierda que $E_{3,4}$ que es más pequeño que él.

Ahora bien, si se observan las Figuras 4.3, 4.4 y 4.5, se puede llegar a la conclusión de que los conceptos de tabla normalizada y tabla ordenada son totalmente independientes, ya

que hay un caso de tabla no normalizada pero ordenada (Figura 4.4), otra que está normalizada pero no ordenada (Figura 4.5) y otra que sí está tanto normalizada como ordenada (Figura 4.3).

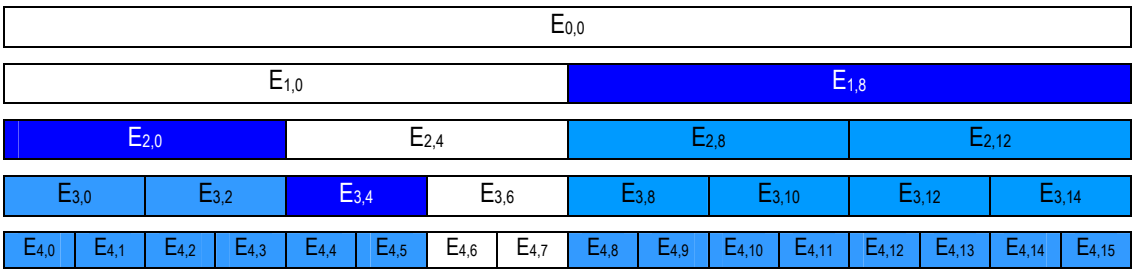


Figura 4.5: Ejemplo de tabla de arbitraje no ordenada.

Tras ser descritos los conceptos básicos necesarios para poder comprender todo el trabajo que se desarrolla a continuación, se recuerdan las ideas básicas que se expusieron en [Alf03] para la gestión de la tabla de arbitraje y a partir de las cuales se ha desarrollado este proyecto.

El primero de los algoritmos propuestos es el de *asignación o inserción,* que se encarga de encontrar un conjunto libre para ubicar una petición, la cual está indicada por el nivel al que pertenece el conjunto solicitado. El algoritmo de inserción busca el conjunto libre capaz de atender la petición y que esté situado lo máximo posible a la izquierda en la representación en árbol de la tabla de arbitraje de la Figura 4.2. En [Alf03] se demuestra que siempre que haya un número suficiente de entradas, el algoritmo de inserción encuentra una secuencia de entradas que asignar para un nivel específico.

Ejemplo 1 *Como situación inicial se tiene la mostrada en la Figura 4.6. En ese momento llega una petición que requiere cuatro entradas, o lo que es lo mismo, se necesita un conjunto de nivel 2. Como se ha indicado anteriormente, el algoritmo selecciona el primer conjunto libre que encuentre buscando de izquierda a derecha. En este caso será elegido el conjunto* $E_{2,8}$*. Tras ser ocupado el conjunto que el algoritmo ha seleccionado, el estado de la tabla de arbitraje es el mostrado en la Figura 4.7.*

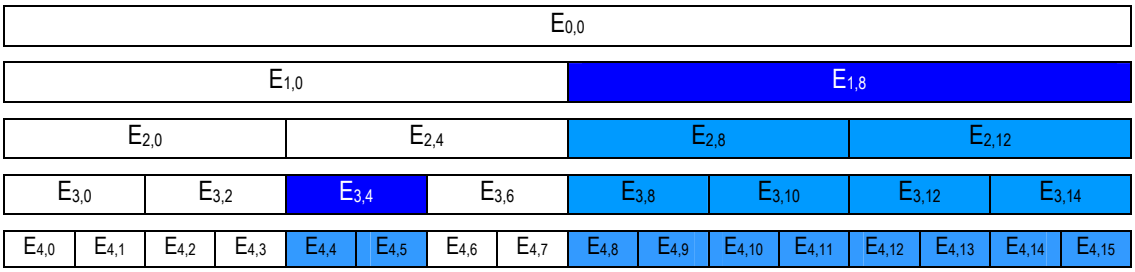


Figura 4.6: Tabla de arbitraje del Ejemplo 1 antes de aplicar el algoritmo de asignación.

Cuando se realiza una eliminación, se liberan entradas ocupadas. Al liberarlas, se puede dar el caso de que la tabla deje de estar ordenada, normalizada o ambas cosas de manera simultánea.

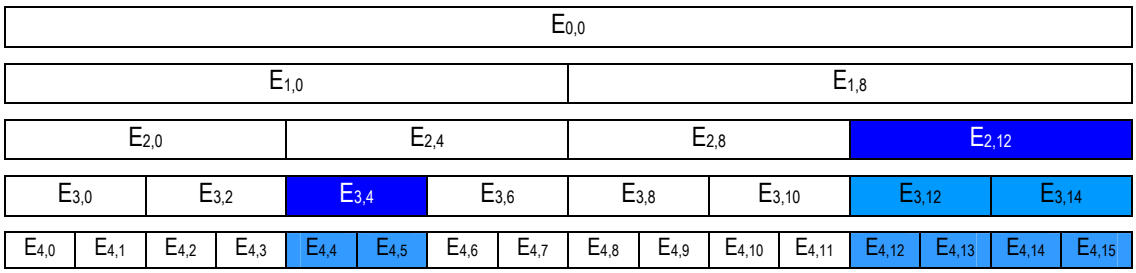


Figura 4.7: Tabla de arbitraje del Ejemplo 1 tras asignar el conjunto $E_{2,8}$.

Ejemplo 2 *Partiendo de la Figura 4.7, se da la situación en la que se ha de eliminar el conjunto que atiende a una petición, en concreto, el conjunto* $E_{2,0}$*. La tabla inicialmente está tanto normalizada como ordenada, pero tras liberar el conjunto indicado, queda tal como se ve en la Figura 4.8. Se observa que en esta nueva situación la tabla no está normalizada, ya que hay dos conjuntos singulares de nivel 2 (*$E_{2,0}$ *y* $E_{2,12}$*) y tampoco está ordenada, al estar* $E_{2,0}$ *más a la izquierda que* $E_{3,4}$*, que es más pequeño.*

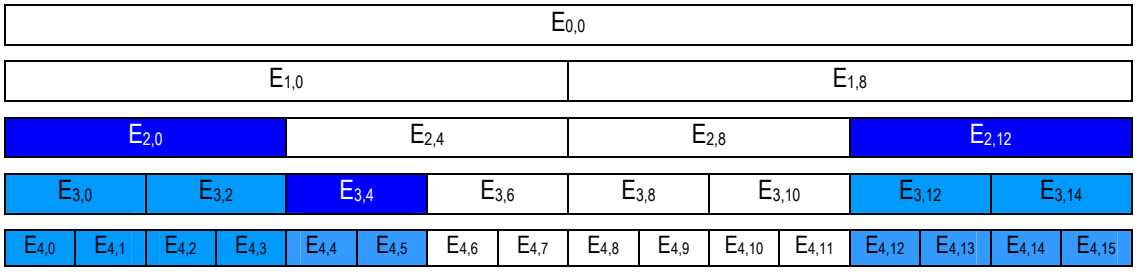


Figura 4.8: Tabla de arbitraje no ordenada ni normalizada del Ejemplo 2 tras liberar el conjunto $E_{2,0}$.

Estando la tabla no normalizada, puede que se dé el caso en el que el algoritmo de asignación no pueda ocupar una secuencia de entradas para una petición que las solicite, aún habiendo un número de entradas suficientes para ello, por ejemplo, una petición que requiera 8 entradas en la Figura 4.8. Este caso es consecuencia de que las entradas no pertenecen a un mismo conjunto. Casos así son el motivo de la necesidad de un procedimiento capaz de unir conjuntos libres para formar uno del máximo tamaño posible. Este algoritmo se denomina *desfragmentación*.

Ejemplo 3 *Partiendo de la situación del Ejemplo 2, cuya situación está reflejada en la Figura 4.8, llega una petición que requiere 8 entradas libres (petición de tipo 1). Ésta no podrá ser atendida aún habiendo 10 entradas libres debido a que esas entradas no están agrupadas en un mismo conjunto, haciendo que el algoritmo de asignación no pueda*

asignarle un conjunto libre a esa petición. Sin embargo, al unir los conjuntos libres $E_{2,0}$ y $E_{2,12}$ para formar otro conjunto, también libre, de mayor tamaño, ya se puede asignar un conjunto a la petición que llegó solicitando 8 entradas, puesto que $E_{1,8}$, que es el conjunto obtenido de la unión de los conjuntos $E_{2,0}$ y $E_{2,12}$, es capaz de albergarla. El resultado de esa unión se refleja en la tabla de arbitraje de la Figura 4.9.

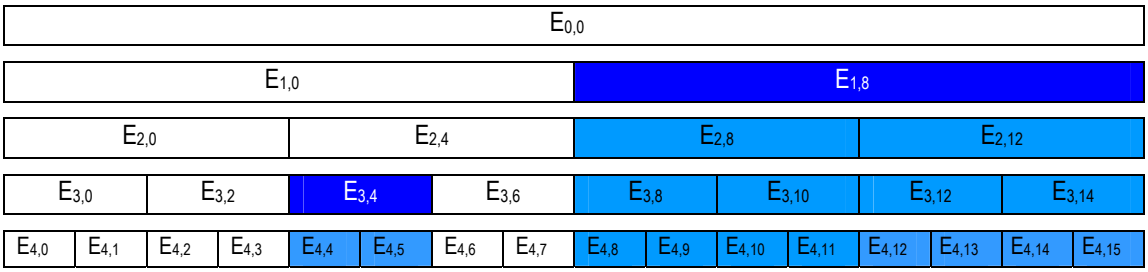


Figura 4.9: Tabla de arbitraje del Ejemplo 3 tras aplicar el algoritmo de desfragmentación.

Una vez explicada la importancia del algoritmo de desfragmentación dentro de la gestión de la tabla de arbitraje, también es necesario explicar la funcionalidad que tiene dentro de dicha gestión el algoritmo de reordenación, y para ello se utilizará el Ejemplo 4.

Ejemplo 4 *Este ejemplo parte de la tabla de arbitraje de la Figura 4.10, en la que se ha liberado previamente el conjunto $E_{3,14}$. Si a continuación se solicitara una petición que requiriese 2 entradas, de acuerdo con el funcionamiento del algoritmo de inserción, se seleccionaría el conjunto $E_{3,8}$, dejando así la tabla de estar normalizada como se puede ver en la Figura 4.11.*

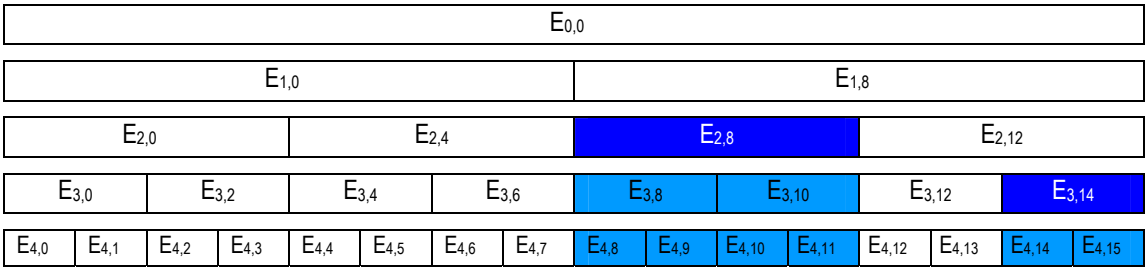


Figura 4.10: Situación inicial de los Ejemplos 4 y 5.

Así pues, para conseguir detectar un conjunto singular que se encuentre desordenado respecto a otro y deshacer esa situación, se utiliza el algoritmo de *reordenación*. Este algoritmo consiste en recorrer la tabla y comprobar si dado un conjunto singular $E_{i,j}$, existe otro conjunto singular $E_{k,m}$ tal que este último sea de mayor tamaño (k<i) y esté situado en la tabla más a la izquierda que el primero (m<j). En ese caso se procede al intercambio de $E_{k,m}$ y del ancestro de $E_{i,j}$ en el nivel k.

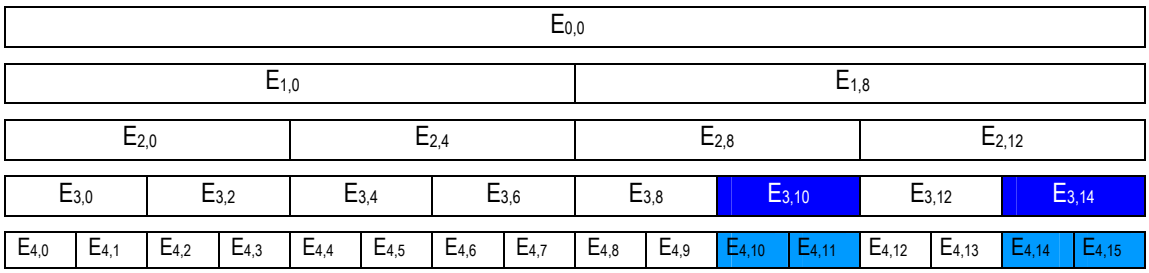


Figura 4.11: Situación final de la tabla de arbitraje en el Ejemplo 4.

Ejemplo 5 *Volviendo a la tabla de arbitraje de la Figura 4.10, en la que se había liberado previamente el conjunto $E_{3,14}$, la tabla no está ordenada, pero lo estaría si $E_{2,8}$ estuviera a la derecha del $E_{3,14}$. Si se intercambian $E_{2,8}$ y $E_{2,12}$ (ancestro de $E_{3,14}$ en el nivel 2) quedaría la situación de la Figura 4.12. Puede observarse que la tabla ahora está ordenada, y por tanto si recibiera cualquier petición que pudiera atender, seguiría estando ordenada y normalizada tras la asignación, tal y como se demuestra en [Alf03] al explicar las propiedades del algoritmo de inserción.*

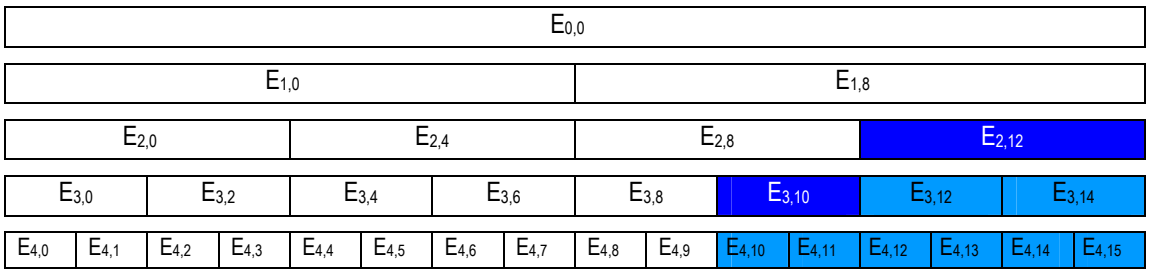


Figura 4.12: Situación tras realizarse la reordenación en el Ejemplo 5.

El uso adecuado de los algoritmos de inserción, desfragmentación y reordenación permite gestionar de una forma eficiente la tabla de arbitraje en InfiniBand. En [Alf03] se incluye una serie de teoremas, y sus correspondientes demostraciones, que así lo acreditan.

4.2 GESTIÓN DE LA TABLA DE ARBITRAJE

Tras haber introducido diversos conceptos relacionados con la tabla de arbitraje y los algoritmos que se propusieron en [Alf03] para su gestión, ahora es el turno de indicar cómo se combinan los algoritmos para el funcionamiento global de la tabla de arbitraje. En [Alf03] se presentaron tres combinaciones, o *modelos* como se denominarán de aquí en adelante, que eran los siguientes:

- **MODELO 1:** Tras realizar una eliminación o una inserción, se aplica el algoritmo de desfragmentación.

- **MODELO 2:** Tras llevar a cabo una eliminación, se aplica el algoritmo de reordenación y después el de desfragmentación. Tras la inserción no se realiza ninguna acción adicional.
- **MODELO 3:** Tras realizar una eliminación, esta vez se aplica primero el algoritmo de desfragmentación de la tabla de arbitraje y después el de reordenación. En este caso tampoco se realiza nada tras la inserción.

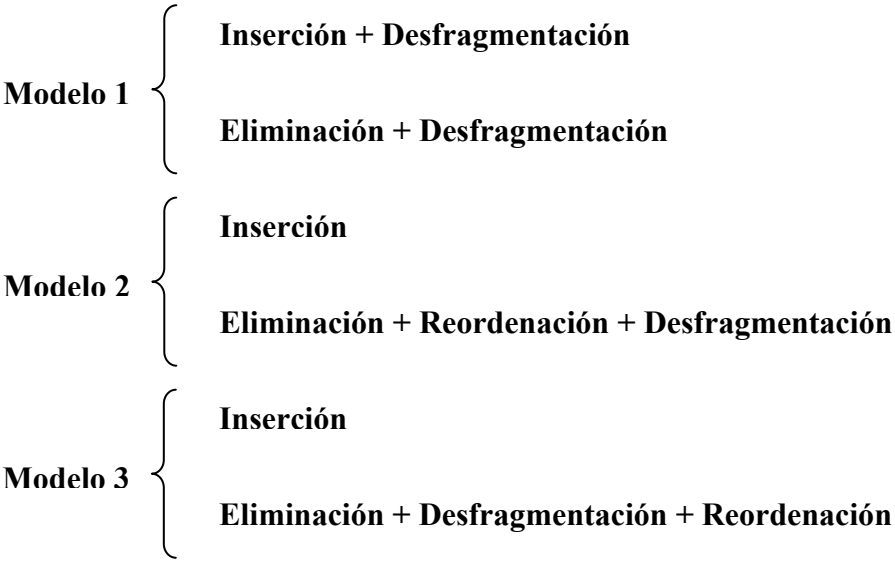


Figura 4.13: Combinaciones de los algoritmos propuestos.

Para explicar cada modelo, se van a utilizar los Ejemplos 6, 7 y 8 que muestran cuál sería el funcionamiento de los tres modelos ante una misma situación. En todos los casos, el punto de partida es el indicado en la Figura 4.14.

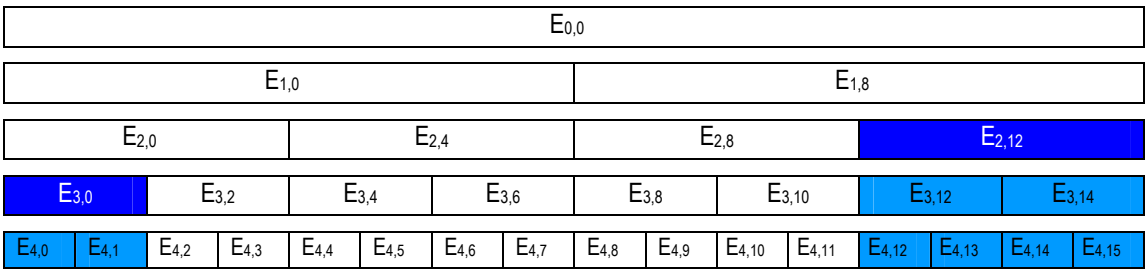


Figura 4.14: Situación inicial de la tabla de arbitraje en los Ejemplos 6, 7 y 8.

La situación va a consistir en la liberación del conjunto $E_{4,7}$ seguida de la petición de un conjunto de nivel 4.

Ejemplo 6 *En este ejemplo se va a utilizar el modelo 1. Por tanto, a la hora de liberar el conjunto* $E_{4,7}$*, la tabla de arbitraje queda tal como muestra la Figura 4.15, donde se puede observar que la tabla no está ordenada. Tras la eliminación se aplica el algoritmo de*

desfragmentación, que no tiene efecto pues la tabla está normalizada. En este caso no se realiza ninguna acción más sobre la tabla hasta la petición que le va a suceder.

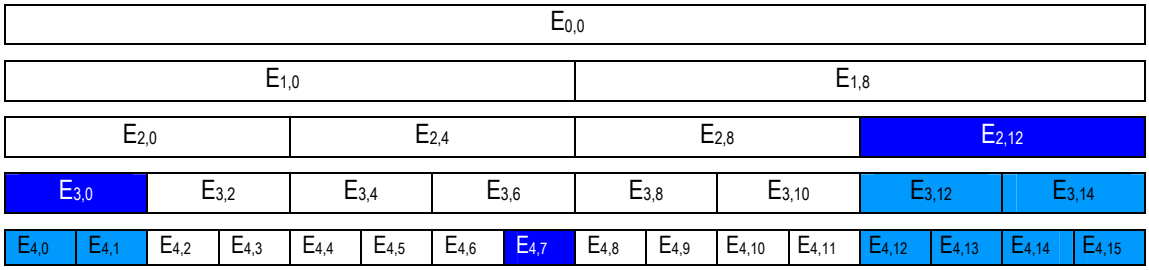


Figura 4.15: Situación de la tabla de arbitraje en los Ejemplos 6, 7 y 8 tras eliminar el conjunto $E_{4,7}$.

Cuando llega la petición que requiere conjunto de nivel 4, el algoritmo de inserción elegirá el conjunto $E_{4,0}$. Por tanto, la tabla de arbitraje queda tal y como refleja la Figura 4.16, donde se puede ver una tabla no normalizada.

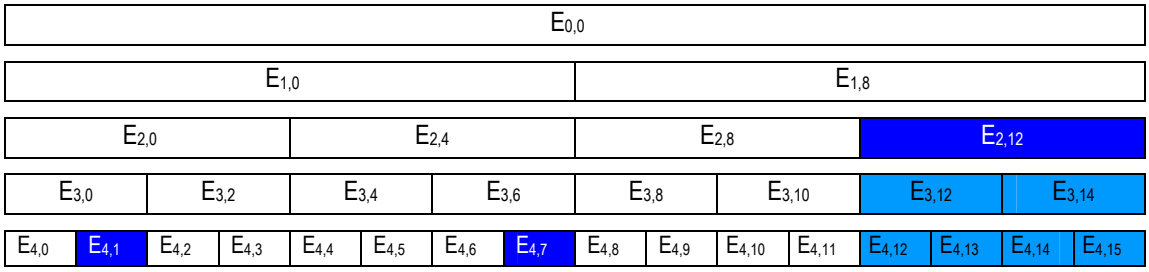


Figura 4.16: Situación de la tabla de arbitraje del Ejemplo 6 tras ocupar el conjunto $E_{4,0}$.

La aplicación del algoritmo de desfragmentación hace que la tabla vuelva a estar normalizada, como se ven en la Figura 4.17. Esto permite poder atender hasta tres peticiones que necesiten dos entradas mientras que en la tabla de la Figura 4.16 no sería posible.

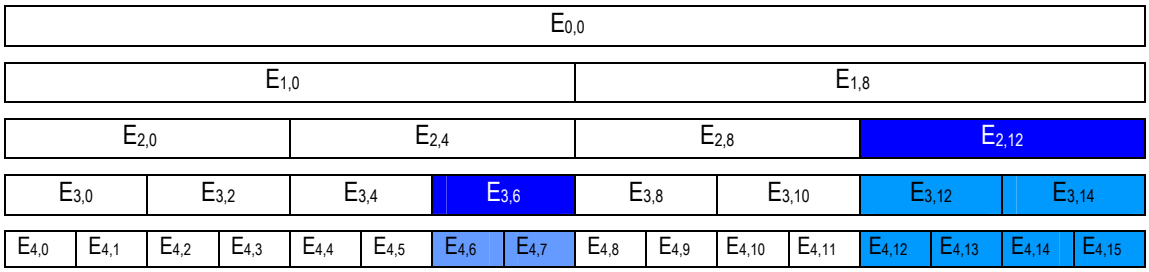


Figura 4.17: Situación final de la tabla de arbitraje de los Ejemplos 6, 7 y 8.

Ejemplo 7 *Ahora se va a seguir el modelo 2 . Tras eliminar el conjunto $E_{4,7}$, la tabla queda tal como muestra la Figura 4.15, al igual que en el Ejemplo 6, donde se puede observar*

que la tabla no está ordenada puesto que $E_{3,0}$ es un conjunto singular mayor que $E_{4,7}$ y el primero está más a la izquierda que el segundo. Siguiendo el modelo, tras eliminar hay que ejecutar el algoritmo de reordenación, de manera que el estado de la tabla de arbitraje resultante es el de la Figura 4.18.

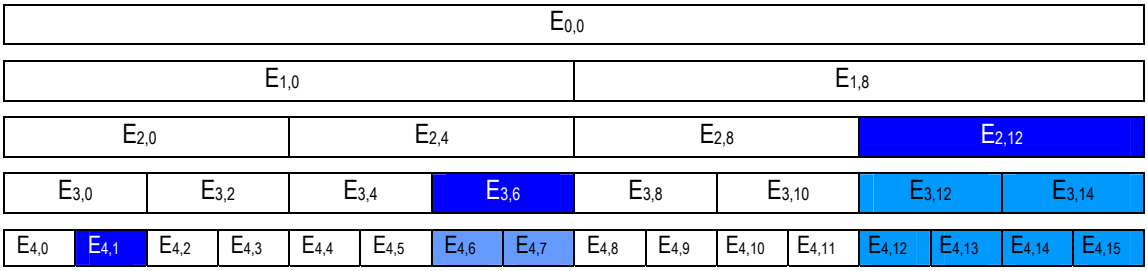


Figura 4.18: Tabla de arbitraje de los Ejemplos 7 y 8 tras aplicar la reordenación sobre ella.

Como la tabla está normalizada, la desfragmentación no tiene efecto. Así pues, se puede continuar con la situación propuesta para este ejemplo con la petición de un conjunto de nivel 4, de manera que el algoritmo de inserción en este ejemplo elige el conjunto $E_{4,1}$ para ser ocupado. Como tras la inserción ya no se realiza nada, la tabla de arbitraje final de este ejemplo es la misma que la del Ejemplo 6 (Figura 4.17).

Ejemplo 8 *Por último, se considera el modelo 3. En este caso, tras eliminar el conjunto $E_{4,7}$, la tabla también queda como muestra la de la Figura 4.15 al igual que en los dos ejemplos anteriores. Siguiendo el orden de algoritmos que se han de aplicar tras eliminar, primero es ahora el de desfragmentación. Como la tabla está normalizada, la tabla de arbitraje sigue en el mismo estado. Al aplicar el algoritmo de reordenación, se produce el mismo efecto que en el ejemplo anterior, al igual que la posterior ocupación de un conjunto de nivel 4, por lo que el resultado final también es el mismo (Figura 4.17).*

4.3 RESUMEN

En este capítulo se han revisado básicamente los conceptos más importantes relacionados con la tabla de arbitraje que fueron desarrollados en [Alf03], y una vez conocidos dichos conceptos, se han podido describir los algoritmos que también se propusieron en [Alf03] para optimizar el rendimiento de la tabla de arbitraje con varios ejemplos que revelaban la importancia de su utilización.

Por último, se han mostrado los tres modelos en que las implementaciones de los algoritmos que se utilizan para el tratamiento de la tabla de arbitraje y tras describirlos, se han realizado ejemplos que reflejaban su funcionamiento ante una misma situación.

Capítulo 5

IMPLEMENTACIÓN DE LA TABLA DE ARBITRAJE

En el capítulo anterior se ha presentado el modelo formal desarrollado en [Alf03], que marca la pauta de los algoritmos que se van a implementar en este proyecto. A lo largo de este capítulo se detalla la implementación que se ha realizado para este Proyecto Fin de Carrera de dichos algoritmos.

Durante el tiempo empleado en el estudio de los algoritmos utilizados para gestionar la tabla de arbitraje, se han ido planteando posibles mejoras sobre dichos algoritmos, bien modificando el mecanismo esencial del algoritmo, o bien añadiendo ciertas estructuras de datos que pudieran mejorar su rendimiento. De todas ellas, la más significativa ha sido incorporar una estructura de datos denominada "*Singulares*". De esta forma, todos los algoritmos desarrollados en el modelo formal serán implementados usando dicha estructura. Además, con dicha estructura se ha propuesto un nuevo modelo de gestión de la tabla de arbitraje como una posible mejora respecto a los tres modelos explicados en [Alf03].

Por tanto, a lo largo de este capítulo se presentará la estructura "*Singulares*", se mostrarán los métodos que contendrán la implementación de los algoritmos desarrollados en el modelo formal previamente explicado, así como otros métodos que son utilizados por ellos. Además, se introduce una nueva implementación del método de búsqueda de conjuntos para el método de inserción. Por último, se presentará el nuevo modelo

propuesto como mejora a los tres que se plantearon en [Alf03] y que fueron revisadas en el capítulo anterior.

5.1 ESTRUCTURA SINGULARES

De todos los conceptos relacionados con la tabla de arbitraje que se han expuesto hasta ahora, uno de los más importantes es el concepto de conjunto *singular*, pues se maneja en varios de los procesos para el tratamiento de la tabla. Es por ello que se ha pensado mantener información basada en el concepto de conjunto singular que facilite la gestión de la tabla de arbitraje de InfiniBand.

Veamos, con un ejemplo, la información que podemos obtener a partir de los conjuntos singulares de la tabla.

Ejemplo 9 *Se tiene una tabla de arbitraje con 16 entradas. De ella sólo se sabe que tiene como conjuntos singulares a* $E_{2,4}$, $E_{2,12}$, $E_{3,0}$ *y* $E_{4,11}$. *Con esta información se puede saber fácilmente qué conjuntos están libres y cuáles no. En la Tabla 5.1 se muestran los conjuntos que están libres como consecuencia de los conjuntos singulares anteriormente citados.*

Conjunto singular	Conjuntos libres dentro del conjunto singular
$\mathbf{E_{2,4}}$	$E_{2,4} - E_{3,4}$, $E_{3,6} - E_{4,4}$, $E_{4,5}$, $E_{4,6}$, $E_{4,7}$
$\mathbf{E_{2,12}}$	$E_{2,12} - E_{3,12}$, $E_{3,14} - E_{4,12}$, $E_{4,13}$, $E_{4,14}$, $E_{4,15}$
$\mathbf{E_{3,0}}$	$E_{3,0} - E_{4,0}$, $E_{4,1}$
$\mathbf{E_{4,11}}$	$E_{4,11}$

Tabla 5.1: Conjuntos libres en la tabla de arbitraje del Ejemplo 9.

Como es lógico, el resto de conjuntos que no están incluidos en la Tabla 5.1 están ocupados. Así pues, se puede obtener una imagen del estado de la tabla de arbitraje a partir de los conjuntos singulares, que es la mostrada en la Figura 5.1.

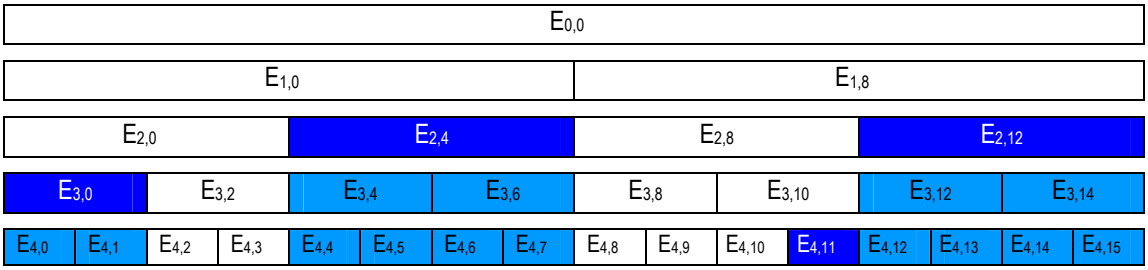


Figura 5.1: Estado de la tabla de arbitraje del Ejemplo 9.

Por tanto, teniendo sólo la información de los conjuntos singulares de la tabla se puede llegar a obtener una imagen completa de su situación. De esta forma, se ha decidido usar la estructura de datos *Singulares* para almacenar esa información de una tabla de arbitraje. Esta estructura de datos almacena, para cada nivel, los índices, en concreto dos, de los conjuntos singulares que posee ordenándolos de izquierda a derecha, así como el número de ellos que hay en cada momento. La razón de que sólo sea necesario almacenar los índices de, como máximo, dos conjuntos por nivel se deriva directamente de las demostraciones realizadas en [Alf03]. De manera informal, podría explicarse como que una tabla de arbitraje antes de que se le aplique una inserción o eliminación siempre está normalizada, con lo que el número máximo de conjuntos singulares en un nivel es uno. Sin embargo, cuando se produce una eliminación o una inserción (en este último caso puede que sea sobre una tabla no ordenada), la tabla puede quedar no normalizada. Así pues, el número máximo de conjuntos singulares que puede haber en un nivel es 2, el que había anteriormente y el que se acaba de formar. Inmediatamente después se realizará una desfragmentación, dejando la tabla nuevamente normalizada.

Evidentemente, la principal utilidad de esta estructura es saber la localización en la tabla de arbitraje de los conjuntos singulares. Conociendo esta información se puede saber el estado de la tabla, ya que sabiendo donde se sitúa un conjunto singular, se pueden saber todos los conjuntos que hay libres y por tanto, también se pueden saber los que están ocupados.

La estructura de datos que se ha elegido para representar los conjuntos singulares es una tabla con tantas columnas[1] como tipos de peticiones haya (niveles en el árbol). Cada una de esas columnas tiene tres filas: dos de ellas para almacenar los índices de los dos posibles conjuntos singulares que pueda haber por nivel, y la fila 0 que indica el número de conjuntos singulares que contiene cada nivel en un momento dado. Esta estructura puede verse gráficamente en la Figura 5.2.

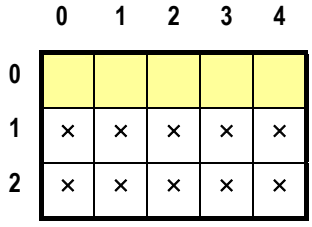


Figura 5.2: Representación de la estructura *Singulares*.

Suponiendo que se implementase en lenguaje C, su declaración sería:

[1] Es importante decir que aún sabiendo que el conjunto de nivel 0 no puede ser singular por definición, para el buen funcionamiento de los métodos implementados, será considerado singular cuando esté libre.

```
typedef   char    singulares[emax+1][3];
```

siendo *emax* el valor del nivel más alto. Por tanto, el tamaño concreto de la estructura, para una tabla de 64 entradas (en ese caso, el árbol tendría 7 niveles) sería de:

$$7\,niveles \times 3\frac{entradas}{nivel} \times 8\frac{bits}{entrada} = 168\,bits = 21\,bytes$$

Ejemplo 10 *A partir de los datos que se dan en el Ejemplo 9 y la Figura 5.2, la estructura Singulares para la tabla de arbitraje sería la mostrada en la Figura 5.3.*

	0	1	2	3	4
0	0	0	2	1	1
1	×	×	4	0	11
2	×	×	12	×	×

Figura 5.3: Estructura *Singulares* para la tabla mostrada en la Figura 5.1.

Esta estructura se utiliza, entre otras muchas cosas, para comprobar si un conjunto está ocupado o libre, para detectar posibles casos de tabla no normalizada cuando hay dos singulares en un mismo nivel y para encontrar conjuntos desordenados al reordenar. Con esto se consigue, por ejemplo, no tener que recorrer por completo la tabla de arbitraje para realizar todas esas tareas.

5.2 IMPLEMENTACIÓN DE LOS ALGORITMOS PARA EL TRATAMIENTO DE LA TABLA DE ARBITRAJE

En esta sección se describen las implementaciones para los algoritmos descritos de manera básica en [Alf03]. En la medida de lo posible se han buscado implementaciones eficientes y sencillas. También se describen otros algoritmos auxiliares que se usan en aquellos.

5.2.1 Método de *asignación* o *inserción*

Este método de asignación o inserción es el encargado de buscar las entradas de la tabla para una nueva petición. Dada una petición de tipo *n*, las entradas han de estar

separadas entre sí una distancia $d = 2^n$. Por esto, se necesitan $\frac{max}{d}$ entradas, siendo *max* el número de entradas de la tabla de arbitraje. El código básico de este método se muestra en la Figura 5.4. La entrada y salidas de dicho método son:

Entradas:

- **nivel:** Nivel al cual pertenece el conjunto que va a atender la petición.

Salida:

- Devuelve verdadero si ha encontrado algún conjunto que pueda albergar a la petición recién llegada, y en caso contrario devuelve falso.
- En "*indice"* se devuelve el índice del conjunto asignado en caso de encontrar un conjunto disponible para la petición realizada.

```
1:  booleano inserción (nivel, *indice)
2:  nuevasEntradas=2^(emax-nivel);
3:  si (nuevasEntradas+numEntradasOcupadas > max) entonces
4:      resultado= falso;
5:  sino
6:      *indice=buscaPrimeraEntradaNuevaPeticion(nivel);
7:      ocupa(E_nivel,*indice);
8:      actualizarSingularesTrasAnnadirPeticion(E_nivel,*indice);
9:      resultado=verdadero;
10: finsi
11: devolver resultado;
12: Fin inserción
```

Figura 5.4: Método de *inserción*.

El método propuesto inicialmente comprueba si hay suficientes entradas libres para atender en la tabla de arbitraje una petición del nivel que se le ha indicado (Figura 5.4, líneas 2-4). En caso de que no sea posible, es decir, no hay entradas suficientes, devuelve directamente falso (Figura 5.4, línea 4). En el caso de que sí pueda ser atendida, busca el conjunto de entradas adecuado.

La búsqueda del conjunto adecuado la realiza otro método llamado *buscaPrimeraEntradaNuevaPeticion* (Figura 5.4, línea 6) almacenando en *indice* el valor del índice del conjunto que se va a ocupar. Tras encontrar el conjunto, el método procede a ocuparlo (procedimiento *ocupa*, Figura 5.4, línea 7). Como el conjunto ha sido ocupado,

sólo falta actualizar la estructura *Singulares*, (Figura 5.4, línea 8) de manera que muestre los cambios que se han realizado dentro de la tabla de arbitraje. De esta manera, cualquier método que tenga que consultar el estado de la tabla de arbitraje más adelante, lo hará con el estado actual de la tabla. Por último, como en este caso se ha encontrado un conjunto que atienda la petición, se devuelve verdadero.

Veamos de forma más detallada el método que hemos utilizado para realizar la búsqueda concreta del conjunto de entradas más adecuado para atender la petición.

5.2.2 Método *buscaPrimeraEntradaNuevaPeticion*

Este procedimiento se encarga de recorrer los conjuntos en busca de uno que esté libre, sea capaz de atender una petición que ha solicitado $\frac{\max}{d}$ entradas libres y además ha de estar situado lo más a la izquierda posible en el árbol de la Figura 4.2. La entrada y salida para este método son las siguientes:

Entrada:

- **nivel:** Nivel del conjunto que se está buscando.

Salida:

- Índice *j* del conjunto libre $E_{i,j}$ situado más a la izquierda y que se ha seleccionado para atender la petición realizada.

```
 1:  entero   buscaPrimeraEntradaNuevaPeticion (nivel)
 2:  menorIndice=max;
 3:  para i=nivel hasta 0 hacer
 4:      si (daNumConjuntosSingulares(i) == 1) entonces
 5:          auxIndice=daIndiceConjuntoSingular(i,1);
 6:          si (auxIndice<menorIndice) entonces
 7:              menorIndice=auxIndice;
 8:          finsi
 9:      finsi
10:  finpara
11:  devolver menorIndice;
 8:  Fin buscaPrimeraEntradaNuevaPeticion
```

Figura 5.5: Método *buscaPrimeraEntradaNuevaPeticion*.

El método recorre la estructura *Singulares* desde el nivel solicitado por la petición hasta el nivel 0 (Figura 5.5, líneas 3-10). Para los niveles donde haya algún conjunto singular se mira el índice del primero. De todos ellos, se queda con el índice de menor valor, o sea, el situado más a la izquierda (Figura 5.5, líneas 4-7), y al final de la búsqueda, devuelve su valor (Figura 5.5, línea 11). Hay que señalar que en la línea 4 de la Figura 5.5 se asume que sólo puede haber a lo sumo un conjunto singular por nivel. Esto es debido a que se supone que la tabla está normalizada, tal y como se demuestra en [Alf03].

Ejemplo 11 *El punto de partida de este ejemplo es la tabla de arbitraje de la Figura 5.6, a la que llega una petición de un conjunto de nivel 3. Se ha visto en la sección anterior que para asignarle un conjunto, el algoritmo de asignación llama primero al método buscaPrimeraEntradaNuevaPeticion(3). Este método, mientras recorre la estructura Singulares desde el nivel 3 hasta el nivel 0, va consultando los índices de los conjuntos singulares de cada nivel.*

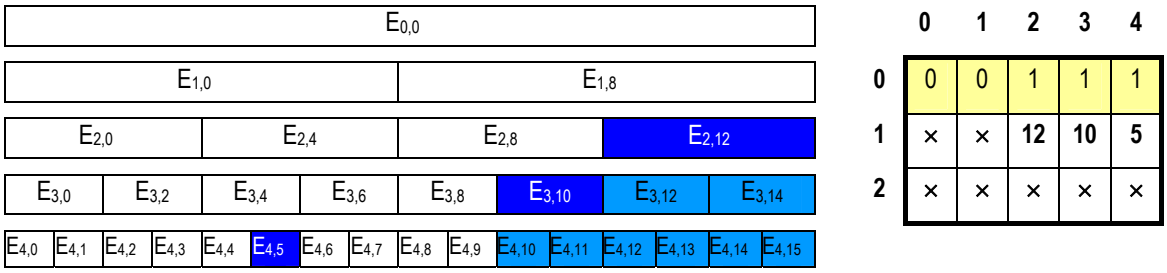


Figura 5.6: Situación inicial del Ejemplo 11.

Tras su aplicación, el resultado obtenido es que el conjunto elegido para atender la petición de tipo 3 recibida es el conjunto $E_{3,10}$. Por ello, el método de inserción procede a ocupar dicho conjunto, quedando la tabla de arbitraje tal y como se puede ver en la Figura 5.7.

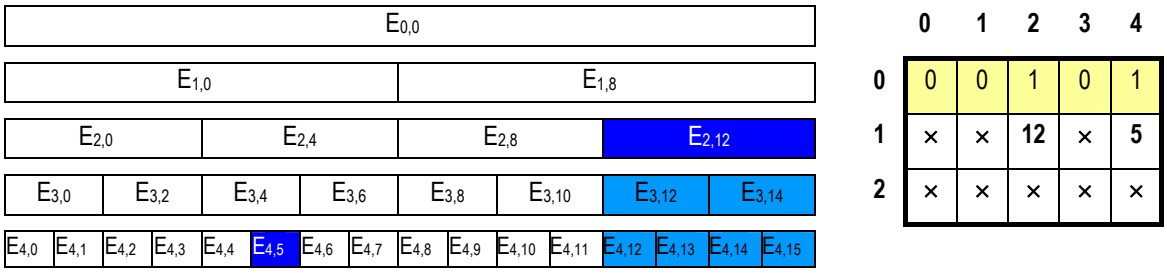


Figura 5.7: Situación final del Ejemplo 11 tras ocupar el conjunto $E_{3,10}$.

Ejemplo 12 *Para este ejemplo se toma como punto de partida la tabla de arbitraje de la Figura 5.8. En este ejemplo, al igual que en el anterior, llega una petición de tipo 3. Por tanto, repitiendo la mecánica a realizar para llevar a cabo una inserción, primero se*

busca mediante el método buscaPrimeraEntradaNuevaPeticion(3) un conjunto candidato para albergar dicha petición. De esta manera, durante el transcurso de la aplicación de este método, al ir consultando los índices de los primeros conjuntos singulares desde el nivel 3 hasta el nivel 0, se ve que en los niveles 3 y 2 no hay ningún conjunto singular. Al llegar al nivel 1 comprueba que sí hay un conjunto singular, viendo que el conjunto es el $E_{1,8}$. De esta manera, 8 es el valor devuelto por el método de búsqueda.

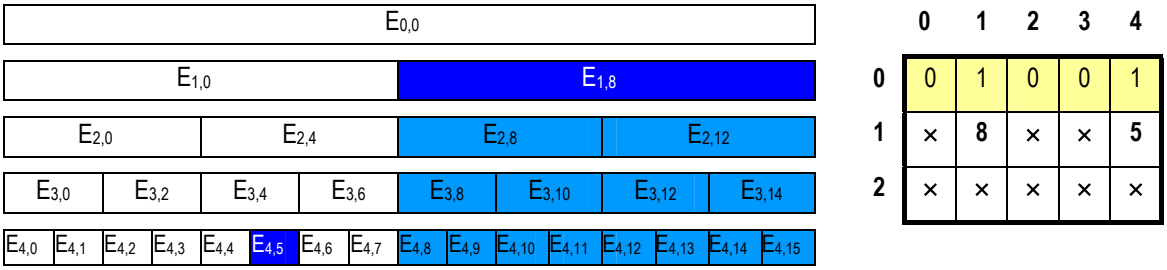


Figura 5.8: Tabla de arbitraje inicial del Ejemplo 12.

Así pues, se ocupa el conjunto $E_{3,8}$, quedando la situación de la tabla de arbitraje como se muestra en la Figura 5.9.

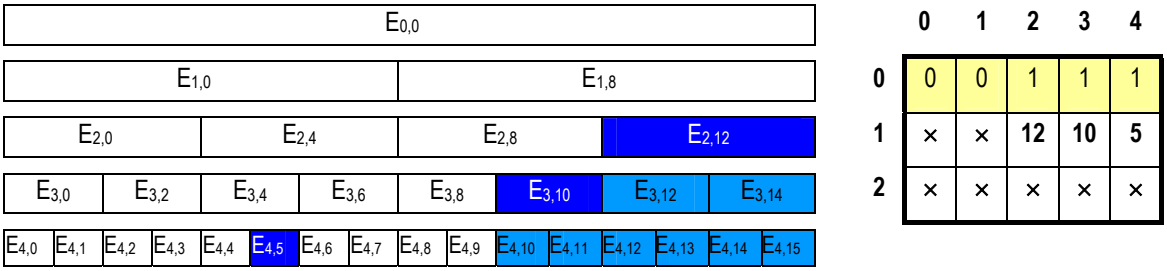


Figura 5.9: Situación final del Ejemplo 12.

5.2.3 Método de *desfragmentación*

El propósito de este método, que ya se ha explicado en el capítulo anterior a la hora de describir el modelo formal, es el de normalizar una tabla que ha dejado de estarlo debido a una eliminación o una inserción. Para ello, se procede a agrupar dos conjuntos singulares de un mismo nivel, repitiendo la operación tantas veces como sea necesario. En concreto, esta operación hay que repetirla hacia niveles superiores porque al unir dos conjuntos en un nivel, dicha unión puede provocar que haya otros dos conjuntos singulares en algún nivel superior. La entrada para este método es:

Entrada:

- **nivel:** Nivel donde se ha añadido o eliminado una petición en la tabla.

```
1:  Procedimiento desfragmentación (nivel)
2:  para i= nivel hasta 1 hacer
3:      numSingulares=daNumConjuntosSingulares(i);
4:      si (numSingulares == 2) entonces
5:          indice1=daIndiceConjuntoSingular(i,1);
6:          indice2=daIndiceConjuntoSingular(i,2);
7:          E_{i,k}=hermano(E_{i,indice2});
8:          intercambioConjuntos(E_{i,indice1},E_{i,k});
9:      finsi
10: finpara
11: Fin desfragmentación
```

Figura 5.10: Método *desfragmentación*.

El método recorre los niveles desde el que se le ha pasado como argumento hasta el nivel 1 (Figura 5.10, líneas 2-10). En ese recorrido consulta el número de conjuntos singulares que tiene cada nivel (Figura 5.10, línea 3). En caso de que en un nivel haya dos conjuntos singulares, simplemente toma sus índices de la estructura, calcula el hermano del segundo y lo intercambia con el primero llamando al procedimiento *intercambioConjuntos* (Figura 5.10, líneas 4-8). Tras el intercambio, continúa con el nivel superior. Hay que indicar que los conjuntos para realizar la desfragmentación se eligen tal que el nuevo conjunto de mayor tamaño quede más a la derecha. Con ello se quiere mantener, cuando sea posible, la tabla ordenada.

Ejemplo 13 *Supongamos que en la situación de la Figura 5.9 se produce la liberación del conjunto $E_{2,0}$, con lo que se alcanza la situación mostrada en la Figura 5.11. En ese caso, la tabla pasa a no estar normalizada ya que $E_{2,0}$ y $E_{2,12}$ son conjuntos singulares de un mismo nivel. El algoritmo de desfragmentación consulta el contador de conjuntos singulares del nivel 2 que hay en la estructura Singulares. El resultado de esa consulta es que hay dos conjuntos singulares.*

Así pues, se ha de realizar un intercambio entre dos conjuntos para unir a los dos conjuntos singulares y así formar uno solo. A continuación se intercambian los conjuntos $E_{2,0}$ y $E_{2,8}$, siendo este último hermano del segundo conjunto singular ($E_{2,12}$). Una vez se ha realizado el intercambio, la situación queda tal y como indica la Figura 5.12.

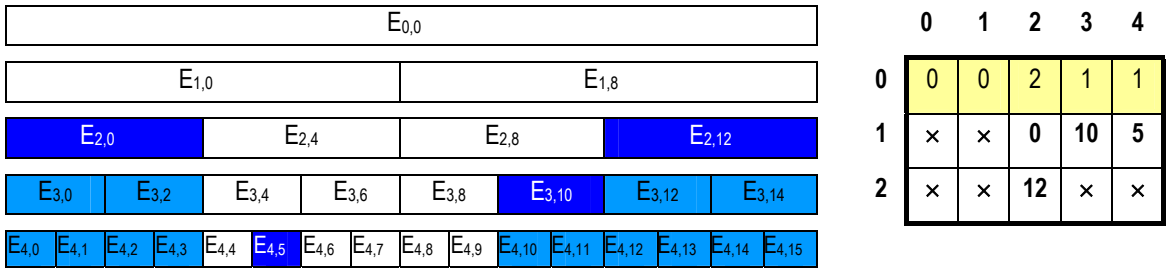


Figura 5.11: Situación inicial del Ejemplo 13.

Después, el procedimiento sigue investigando sobre si hay que realizar más intercambios motivados por la posible existencia de dos conjuntos singulares en niveles superiores. Tras llegar al nivel 1 y ver que no se han de realizar más intercambios, concluye su ejecución.

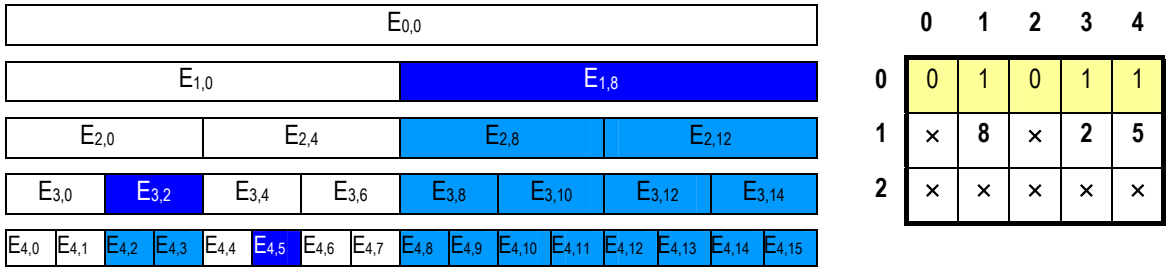


Figura 5.12: Situación final del Ejemplo 13.

Ejemplo 14 *Partiendo de la Figura 5.13, tras la eliminación o bien del conjunto $E_{4,3}$ o del $E_{4,5}$, la tabla pasa a no estar normalizada ya que $E_{4,3}$ y $E_{4,5}$ son conjuntos singulares de un mismo nivel. El método de desfragmentación consulta el contador de conjuntos singulares de nivel 4 que hay en la estructura Singulares. El resultado de esa consulta es que hay dos conjuntos singulares.*

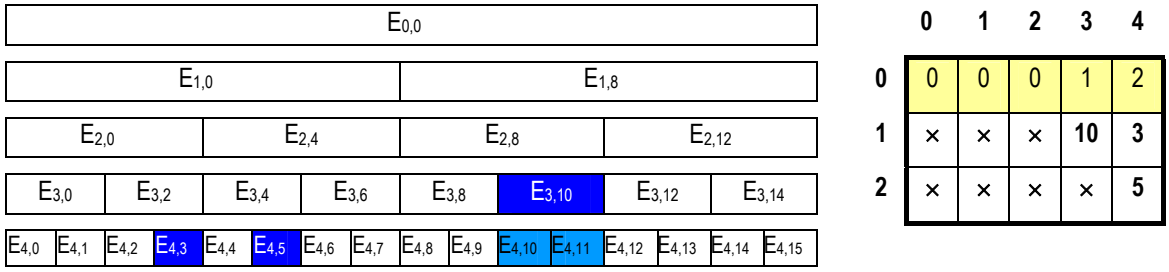
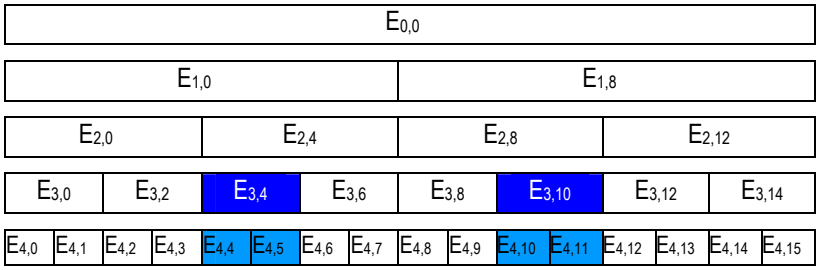


Figura 5.13: Situación inicial del Ejemplo 14.

Por tanto, se ha de realizar un intercambio entre dos conjuntos para unir a los dos conjuntos singulares y así formar uno sólo. De esta forma, se intercambian los conjuntos $E_{4,3}$ y $E_{4,4}$. Una vez se ha realizado el intercambio, la situación queda tal y como indica la Figura 5.14. El procedimiento sigue por el nivel 3 comprobando si hay nuevamente dos

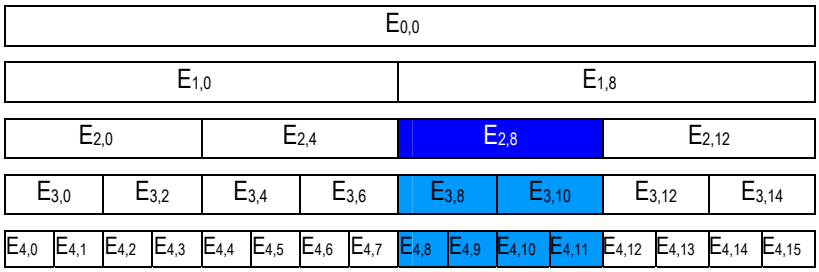
conjuntos singulares en ese nivel. Al consultar el contador de conjuntos singulares del nivel 3, se comprueba que también hay dos ($E_{3,4}$ y $E_{3,10}$).



	0	1	2	3	4
0	0	0	0	2	0
1	×	×	×	4	×
2	×	×	×	10	×

Figura 5.14: Situación tras el primer intercambio en el Ejemplo 14.

Así pues, se vuelve a realizar otro intercambio para conseguir que la tabla vuelva a estar normalizada, siendo esta vez los conjuntos implicados $E_{3,4}$ y $E_{3,8}$. El resultado es la tabla de arbitraje de la Figura 5.15. Tras este último intercambio, el algoritmo sigue comprobando si hay dos conjuntos singulares en niveles superiores, siendo esta vez negativo el resultado de dicha búsqueda.



	0	1	2	3	4
0	0	0	1	0	0
1	×	×	8	×	×
2	×	×	×	×	×

Figura 5.15: Situación final del Ejemplo 14.

5.2.4 Método de *reordenación*

La finalidad de este método fue descrita en el Apartado 4.1 del capítulo anterior. En concreto, la idea es conseguir que los conjuntos singulares estén dispuestos en la tabla de arbitraje de la Figura 4.2 tal que los conjuntos singulares de mayor tamaño estén situados más a la derecha. Este método también hace uso de la estructura *Singulares*, pues su uso evita tener que ir recorriendo todos los conjuntos de la tabla para buscar los conjuntos singulares.

El pseudocódigo del método implementado puede verse en la Figura 5.16. Puede observarse que para el funcionamiento de este método no es necesario pasar ningún parámetro y tampoco devuelve un resultado. El método se limita a modificar adecuadamente la tabla de arbitraje y la estructura *Singulares*. Recorre todos los niveles

```
 1:  Procedimiento reordenación ()
 2:  para nivel=emax hasta 2 hacer
 3:      numSingulares=daNumConjuntosSingulares(nivel);
 4:      para posicion=1 hasta numSingulares hacer
 5:          encontrados=2;
 6:          mientras (encontrados>=2) hacer
 7:              encontrados=0;
 8:              menor=daIndiceConjuntoSingular(nivel,posición);
 9:              menorNivel=nivel;
10:              auxIndex=menor;
11:              para j=1 hasta (nivel-1) hacer
12:                  auxPos=1;
13:                  auxNumSingulares=daNumConjuntosSingulares(j);
14:                  para auxPos=1 hasta auxNumConjuntosSingulares hacer
15:                      actual=daIndiceConjuntoSingular(j,auxPos);
16:                      si (auxIndex>actual) entonces
17:                          encontrados++;
18:                      finsi
19:                      si (actual<menor) entonces
20:                          menor=actual;
21:                          menorNivel=i;
22:                      finsi
23:                  finpara
24:              finpara
25:              si (menor!=auxIndex) entonces
26:                  intercambioConjuntos(E_menorNivel,menor, E_menorNivel,n); //n=ancestro(auxIndex,menorNivel)
27:              finsi
28:          finmientras
29:      finpara
30:  finpara
31:  Fin reordenacion
```

Figura 5.16: Método de *reordenación*.

desde *emax* hasta 2 (Figura 5.16, líneas 2-30) y para cada uno de los conjuntos singulares $E_{i,j}$, busca algún otro que esté desordenado respecto a él (Figura 5.16, líneas 11-24). Durante la búsqueda, de entre todos los conjuntos desordenados va quedándose con el situado más a la izquierda (Figura 5.16, líneas 19-24). Además cuenta los que ha encontrado (Figura 5.16, líneas 16-17). Por último, termina la búsqueda y si ha encontrado

algún conjunto $E_{k,m}$ de menor nivel situado más a su izquierda, lo intercambia con el ancestro de $E_{i,j}$ en el nivel k (Figura 5.16, líneas 25-26). Si encuentra más de un conjunto desordenado, vuelve a repetir la búsqueda por si alguno aún sigue desordenado después de haber cambiado la ubicación de $E_{i,j}$.

Ejemplo 15 *Partiendo de la Figura 5.17, tras la liberación del conjunto $E_{2,0}$, la tabla pasa a no estar ordenada, puesto que $E_{2,0}$ es un conjunto singular mayor tanto de $E_{4,4}$ como de $E_{3,8}$, y está más a la izquierda que ellos en la tabla de arbitraje. Por ello, hay que aplicar una reordenación sobre la tabla. Lo primero que se hace es buscar el primer conjunto singular de la tabla, y tomarlo como referencia para buscar otros que estén desordenados respecto a él, utilizando para ello la estructura Singulares. De esta forma, se comprueba si hay algún conjunto singular en el nivel 4, siendo el resultado de esta búsqueda afirmativo. Tras consultar cuál es el índice de ese conjunto, se tiene que el $E_{4,4}$ será el primer conjunto referencia.*

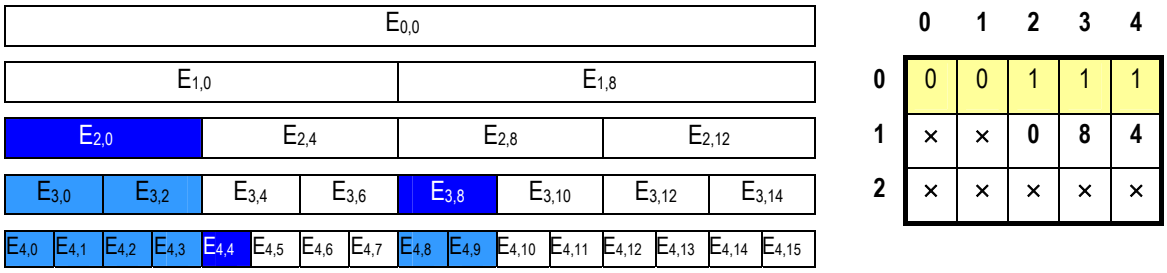


Figura 5.17: Tabla de arbitraje inicial del Ejemplo 15.

Tras encontrar un conjunto de referencia, $E_{4,4}$, busca un posible conjunto singular que esté desordenado respecto a él. Para ello recorre los contadores de número de conjuntos singulares por nivel que hay en la estructura Singulares, consultando los índices de estos desde el nivel 1 hasta el nivel 3 ("nivel-1" de la Figura 5.16, línea 11). En nuestro ejemplo, en el nivel 2 hay uno, y además está desordenado puesto que su índice es menor que el de $E_{4,4}$. Ese conjunto es el recién liberado $E_{2,0}$ y tras encontrarlo, debería seguir buscando otro posible conjunto singular desordenado, hasta descubrir que ya no hay ninguno más. Por tanto procede a intercambiar $E_{2,0}$ con el conjunto ancestro de $E_{4,4}$ en el nivel 2 ($E_{2,4}$) quedando la tabla de arbitraje como se muestra en la Figura 5.18.

A continuación, sigue con el conjunto $E_{3,8}$ como conjunto singular de referencia, en busca de otros conjuntos que estén desordenados respecto de él. En este caso es solamente el conjunto $E_{2,4}$. Se procede al intercambio de $E_{2,4}$ y el ancestro de $E_{3,8}$ en el nivel 2 ($E_{2,8}$), siendo el resultado de dicho intercambio la tabla de la Figura 5.19.

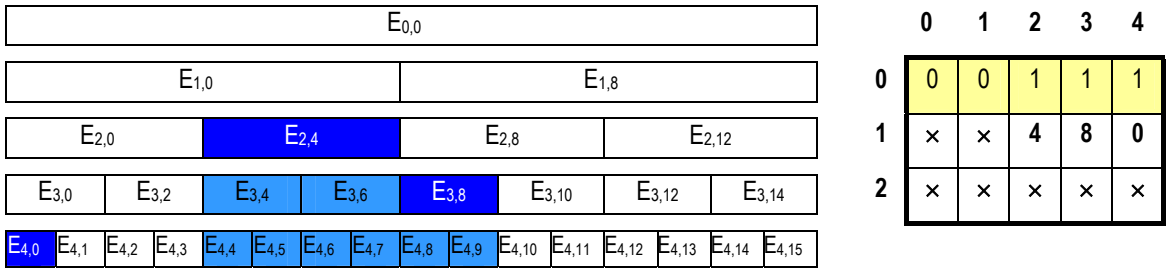


Figura 5.18: Estado de la tabla de arbitraje tras el primer intercambio del Ejemplo 15.

Tras ese intercambio, el método continua buscando algún conjunto singular desordenado respecto a $E_{2,8}$, pero al no haber ninguno concluye su aplicación.

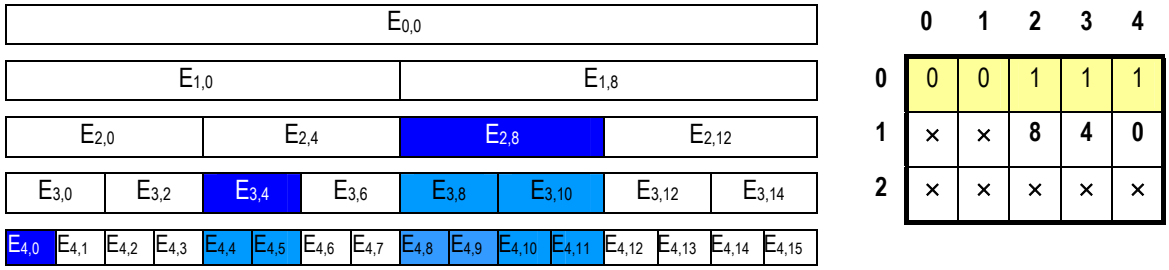


Figura 5.19: Tabla de arbitraje final del Ejemplo 15.

5.2.5 Método *esConjuntoLibre*

Este método comprueba si un conjunto está libre. Aunque no haya aparecido en los métodos anteriores, se utiliza de manera indirecta al ser llamado por otros métodos como *ocupa, intercambioConjuntos* y otros métodos más que no son directamente mencionados en este capítulo. La entrada y salida de este método son:

Entrada:

- $\boldsymbol{E_{i,j}}$**:** Conjunto del que se quiere saber si está libre u ocupado.

Salida:

- Devuelve verdadero si $E_{i,j}$ está libre, y en caso de que no lo esté, devuelve falso.

El pseudocódigo del método *esConjuntoLibre* se muestra en la Figura 5.20.

Aunque esta comprobación se podría realizar recorriendo todas y cada una de las entradas que forman un conjunto y viendo si están libres o no, sin embargo, se ha implementado aprovechando la estructura auxiliar *Singulares*. Esta propuesta parte de la

```
1:  booleano esConjuntoLibre (E_i,j)
2:  si esConjuntoSingular(E_i,j) entonces
3:      resultado=verdadero;
4:  sino
5:      aux=daNivelConjuntoAncestroSingular (E_i,j);
6:      si ((aux>=0) AND (aux<emax)) entonces
7:          resultado=verdadero;
8:      sino
9:          resultado=falso;
10:     finsi
11: finsi
12: devolver resultado;
13: Fin esConjuntoLibre
```

Figura 5.20: Método *esConjuntoLibre*.

idea de que un conjunto está libre si es singular o está incluido en otro conjunto singular[2]. Así pues, el procedimiento que sigue este método es primero comprobar si el conjunto que se ha pasado como argumento es singular (Figura 5.20, líneas 2-3). En caso de no serlo, comprueba si el conjunto del que se quiere saber si está libre tiene algún conjunto ancestro que sea singular. Esta consulta la realiza llamando a *daNivelConjuntoAncestroSingular* (Figura 5.20, línea 5). Si ese conjunto ancestro singular existe, la función devuelve, como se verá más adelante. un valor que esté dentro del rango {-1, 0, 1, 2, …, *emax-1*} (Figura 5.20, líneas 6-10).

En los dos siguientes ejemplos se toma como punto de partida la Figura 5.21:

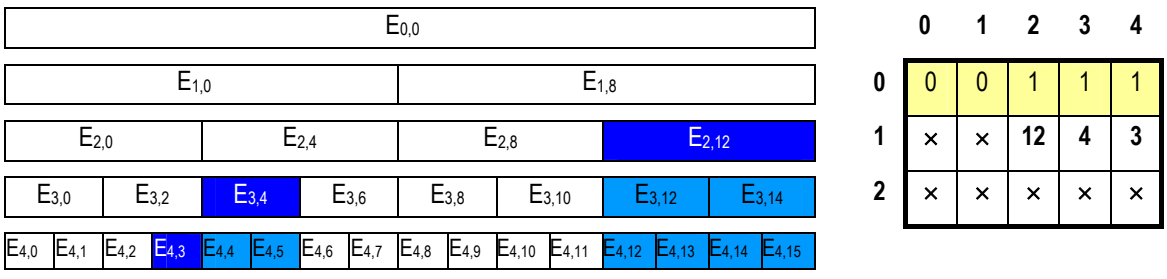


Figura 5.21: Situación inicial de los Ejemplos 16 y 17.

[2] Como se ha indicado anteriormente, aún sabiendo que el conjunto $E_{0,0}$, por definición, no puede ser considerado como conjunto singular, para el buen funcionamiento de los métodos implementados, se va a suponer en el caso de estar libre, que se pueda tomar como conjunto singular.

Ejemplo 16 *Sobre la tabla de arbitraje de la Figura 5.21, se invoca a esConjuntoLibre($E_{3,12}$). Como se ha podido ver, el método primero comprueba si éste es un conjunto singular, siendo el resultado de esa consulta negativo. A continuación, comprueba si tiene algún conjunto ancestro singular llamando a daNivelConjuntoAncestroSingular($E_{3,12}$), devolviendo este último el valor 2, que corresponde con el nivel de su ancestro singular, ya que el conjunto $E_{2,12}$ es un conjunto ancestro suyo y singular a la vez. Por lo tanto, el conjunto $E_{3,12}$ sí está libre.*

Ejemplo 17 *Sobre el mismo ejemplo anterior de la Figura 5.21, ahora se invoca a esConjuntoLibre($E_{4,9}$). Aquí también mira primero si el conjunto $E_{4,9}$ es un conjunto singular y comprueba que no lo es. A continuación, llama a daNivelConjuntoAncestroSingular($E_{4,9}$) para comprobar si tiene algún conjunto singular ancestro, devolviendo el valor "-1", que indica que no posee ninguno. Por tanto, el conjunto $E_{4,9}$ no está libre y se devolverá falso.*

5.2.6 Método *daNivelConjuntoAncestroSingular*

Este método busca a través de los conjuntos ancestros del conjunto pasado como argumento, uno que sea singular. Si dicho ancestro singular existe, entonces devuelve el nivel del citado conjunto, y si no existe devuelve el valor -1. La entrada y salida de este método son:

Entrada:

- $\boldsymbol{E_{i,j}}$**:** Conjunto del que se quiere conocer el nivel de su ancestro singular, si lo tiene.

Salida:

- El nivel del citado conjunto ancestro en caso de que exista, y -1 si $E_{i,j}$ no tiene ningún conjunto ancestro singular.

La salida de este método es conveniente comentarla más a fondo, ya que tiene varios casos especiales. En concreto, el valor de salida que puede devolver este método pertenece al rango {-1, 0, 1, 2, … , *emax-1*}. El primer caso especial se da cuando devuelve el valor -1, indicando con ello que el conjunto pasado como parámetro no tiene ningún ancestro singular. Otro caso especial es cuando el método devuelve el valor 0, ya que en una situación normal, significaría que el conjunto $E_{0,0}$ es el conjunto ancestro singular del conjunto pasado como parámetro, cuando por definición, $E_{0,0}$ no puede ser singular. Sin embargo devolviendo el valor 0, el método da cuenta de que la tabla de arbitraje está vacía. Por último, hay que señalar que el método nunca puede devolver el valor *emax*. El motivo es porque ningún conjunto de ese nivel puede ser ancestro de otro conjunto, ya que los

conjuntos de ese nivel son los de menor tamaño de toda la tabla, y un conjunto no puede ser ancestro de sí mismo.

```
1:  entero   daNivelConjuntoAncestroSingular (Ei,j)
2:  si esConjuntoSingular(Ei,j) entonces
3:    resultado=-1;
4:  sino
5:    nivel=i;
6:    continuar=verdadero;
7:    mientras ((nivel>0) AND (continuar)) hacer
8:          Enivel-1,k=conjuntoPadre(nivel-1,Ei,j);
9:          si esConjuntoSingular(Enivel-1,k) entonces
10:               resultado=nivel-1;
11:               continuar=false;
12:         finsi
13:         nivel--;
14:   finmientras
15:   si (continuar) entonces
16:         resultado=-1;
17:   finsi
18: finsi
19: devolver resultado;
20: Fin daNivelConjuntoAncestroSingular
```

Figura 5.22: Método *daNivelConjuntoAncestroSingular*.

Primero se comprueba si $E_{i,j}$ es un conjunto singular (Figura 5.22, línea 2). Si el mismo $E_{i,j}$ es ya singular, es que no tiene ningún ancestro que lo sea, con lo que vamos a devolver que no existe dicho ancestro singular. Si no lo es, se obtienen sus ancestros uno a uno, y con la ayuda de la estructura *Singulares* se va comprobando si alguno de ellos es singular (Figura 5.22, líneas 5-14). En caso de encontrarlo, se devuelve el nivel al que pertenece el conjunto que cumple el requisito de ser singular (Figura 5.22, líneas 9-11). Si el conjunto pasado como argumento no tiene ningún ancestro singular, se devuelve el valor "-1" para indicar esa situación.

Ejemplo 18 *Partiendo de la Figura 5.23, se hace una llamada a daNivelConjuntoAncestroSingular($E_{4,13}$). El método primero comprueba si el conjunto $E_{4,13}$ es singular, viendo que no lo es. Por tanto, se han de calcular sus ancestros mientras no se encuentre ninguno que sea singular. El primer conjunto ancestro de $E_{4,13}$ es el*

conjunto $E_{3,12}$, y utilizando la estructura Singulares, se comprueba que éste no es el ancestro que se está buscando. Lo mismo ocurre con el siguiente ancestro en un nivel inferior, que es $E_{2,12}$. Por último, al subir al siguiente nivel, el conjunto ancestro es $E_{1,8}$, y en este caso la consulta sobre si es un conjunto singular da resultado afirmativo, dando fin a la búsqueda y devolviendo el nivel 1.

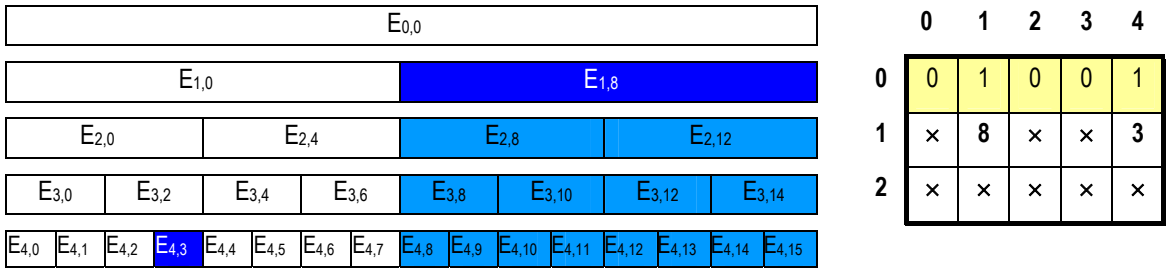


Figura 5.23: Situación del Ejemplo 18.

5.2.7 Método esConjuntoSingular

Este método comprueba si un conjunto es singular. Esta función se utiliza en los algoritmos implementados tanto directa como indirectamente. La entrada y salida de este método son las siguientes:

Entrada:

- ***E*$_{i,j}$:** Conjunto del que se quiere comprobar si es singular.

Salida:

- Devuelve verdadero si $E_{i,j}$ es un conjunto singular, y falso en caso de que no lo sea.

El pseudocódigo del método implementado puede verse en la Figura 5.24.

Puede observarse que todo el proceso se lleva a cabo utilizando la estructura *Singulares*. El algoritmo primero examina si hay conjuntos singulares en el nivel del que está realizando la consulta (Figura 5.24, líneas 3-4). Si hay alguno, consulta los índices de los conjuntos singulares de ese nivel para buscar uno que coincida con el conjunto que se ha pasado como parámetro (Figura 5.24, líneas 4-9). Si alguno de los conjuntos singulares coincide con el que se busca, el método devuelve verdadero, en caso contrario, simplemente devuelve falso.

```
1:  booleano     esConjuntoSingular (E_i,j)
2:  resultado=false;
3:  numSingulares=daNumConjuntosSingulares(i);
4:  k=1;
5:  mientras (k<=numSingulares) AND (NOT (resultado)) hacer
6:          auxIndex=daIndiceConjuntoSingular(i, k);
7:          si (auxIndex==j) entonces
8:              resultado=verdadero;
9:          finsi
10:     k++;
11: finmientras
12: devolver resultado;
13: Fin esConjuntoSingular
```

Figura 5.24: Método *esConjuntoSingular*.

Para los siguientes ejemplos, se toma como situación inicial la Figura 5.25.

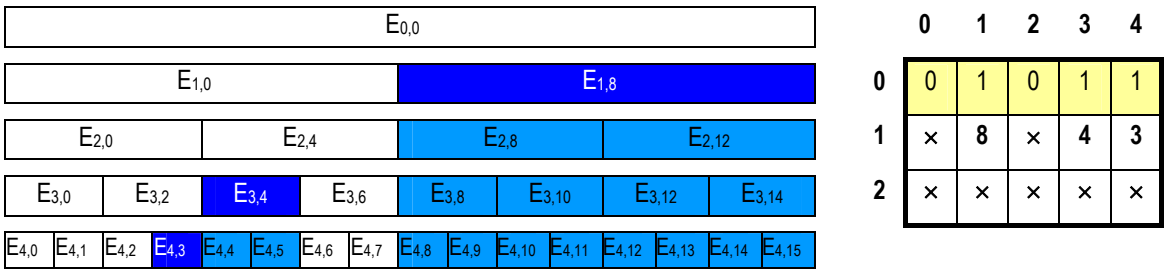


Figura 5.25: Situación para los Ejemplos 19 y 20.

Ejemplo 19 *Se invoca a esConjuntoSingular($E_{4,13}$). El procedimiento comprueba si en el nivel 4 hay algún conjunto singular y ve que sí lo hay. Entonces procede a buscar el índice 13 entre los conjuntos singulares de ese nivel. Como no se tiene éxito en dicha búsqueda, se devuelve falso, indicando así que no es un conjunto singular.*

Ejemplo 20 *Esta vez se invoca a esConjuntoSingular($E_{3,4}$). Siguiendo otra vez los pasos, comprueba si en el nivel 3 hay algún conjunto singular consultando el contador que hay en la estructura Singulares para ese nivel. Una vez se ha comprobado que hay uno, se examina si el índice del conjunto singular de ese nivel es el 4, que corresponde con el índice del conjunto que se ha pasado como argumento. Como ambos índices coinciden, se procede a informar que $E_{3,4}$ sí es un conjunto singular.*

5.3 INSERCIÓN INTELIGENTE

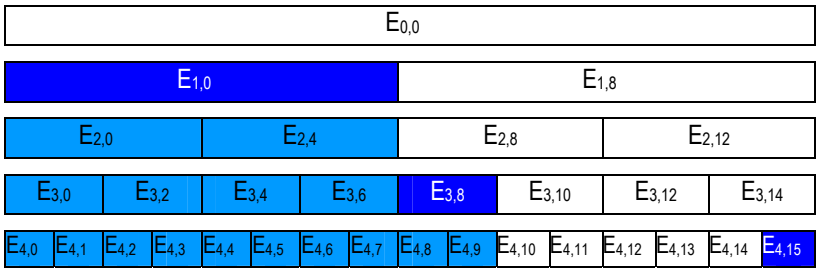
En la propuesta que se revisó en el capítulo anterior, tanto el método de desfragmentación como el de reordenación surgen para que la asignación no cause ningún problema debido a su funcionamiento. A continuación, se suponen dos casos en los que se deja de utilizar uno de ellos (desfragmentación o reordenación), viendo qué efectos se producen y posibles soluciones para suplir su ausencia.

Si se dejase de utilizar el método de desfragmentación, en muchos casos la asignación no podría atender la petición más restrictiva posible, ya que la tabla estaría en muchas ocasiones no normalizada y aún habiendo el número de entradas necesario, no estarían formando un conjunto capaz de atender esa petición. A este inconveniente se suma el hecho de que al poderse dar casos en los que la tabla de arbitraje no esté normalizada, puede haber más de dos conjuntos singulares en algunos niveles. En ese caso sería preferible implementar la estructura de datos *Singulares* de manera dinámica. Eso ralentizaría todo lo referido al mantenimiento de dicha estructura de datos. La única solución posible sería coger cierto número de entradas que puedan atender la petición solicitada y que a su vez, no pertenezcan a un mismo conjunto. Al analizar esa posible solución, se ve que no está de acuerdo con la idea de tabla de arbitraje que se propuso utilizando el concepto de conjunto. Por lo tanto, si se dejara de utilizar la desfragmentación, no es posible alcanzar otra solución que pueda mejorar a priori el rendimiento de lo propuesto.

Por otra parte, si se eliminara el método de reordenación se tendrían situaciones donde habría que aplicar inserciones sobre una tabla no ordenada. Al producirse esto, podría suceder que la tabla pasase a estar también no normalizada debido al funcionamiento del método de asignación, que busca el conjunto libre capaz de albergar la petición solicitada situado lo más a la izquierda posible. Esto haría necesario desfragmentar también tras una inserción. Sin embargo, en este caso sí hay solución factible. Esa solución se llama *inserción inteligente* y se desarrolla a continuación. Primero se verá un ejemplo de su funcionamiento para pasar a continuación a estudiar la implementación propuesta.

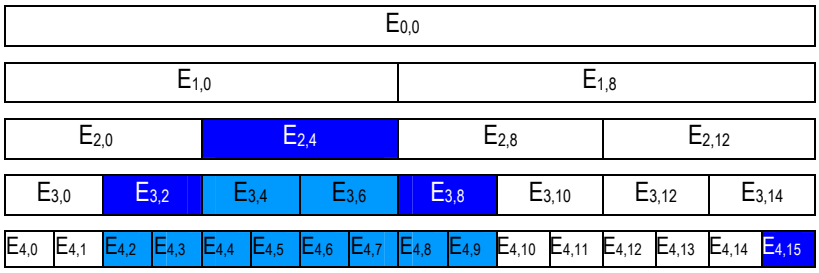
Ejemplo 21 *El punto de partida de este ejemplo es la tabla de arbitraje de la Figura 5.26. Se observa que la tabla no está ordenada, ya que al no haberse aplicado el método de reordenación, se tiene el conjunto singular* $E_{1,0}$ *que está más a la izquierda que el conjunto* $E_{3,8}$*, y a su vez más a la izquierda que el conjunto singular* $E_{4,15}$*.*

En esta situación, si ahora se recibiera una petición de tipo 3, el algoritmo de asignación que se utilizaba antes elegiría al conjunto $E_{3,0}$*. Esto provocaría que la tabla quedara tal y como se ve en la Figura 5.27.*



	0	1	2	3	4
0	0	1	0	1	1
1	×	0	×	8	15
2	×	×	×	×	×

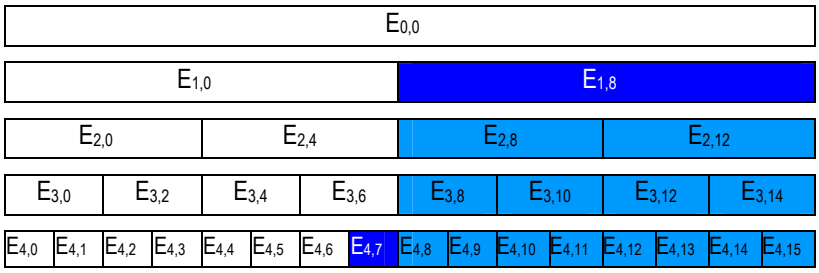
Figura 5.26: Situación inicial del Ejemplo 21.



	0	1	2	3	4
0	0	0	1	2	1
1	×	×	4	2	15
2	×	×	×	8	×

Figura 5.27: Situación tras ocupar el conjunto $E_{3,0}$ en el Ejemplo 21.

Puede observarse que la tabla de la Figura 5.27 no está normalizada, por lo que hay que llevar a cabo una desfragmentación, siendo el resultado final de este ejemplo la tabla de la Figura 5.28.



	0	1	2	3	4
0	0	1	0	0	1
1	×	8	×	×	7
2	×	×	×	×	×

Figura 5.28 Situación final tras realizar la desfragmentación en el Ejemplo 21.

Sin embargo, la situación podría haberse solventado con éxito si al recibir la petición de tipo 3, estando la tabla en la situación de la Figura 5.26, se hubiera elegido el conjunto $E_{3,8}$. Para ello, en este caso sería necesario elegir el conjunto singular de ese nivel, y no el primer conjunto libre de izquierda a derecha. En ese caso, se hubiera alcanzado la situación mostrada en la Figura 5.29, donde se puede observar que la tabla sigue estando normalizada, y por tanto, no es necesario aplicar desfragmentación. En el siguiente Ejemplo, se muestra cómo se llegaría a la situación mostrada en la Figura 5.29.

Ejemplo 22 *Volviendo a la tabla de arbitraje de la Figura 5.26 como punto de partida de este ejemplo, y se recibe otra vez una petición de tipo 3. En este caso, en vez de buscar conjuntos libres de izquierda a derecha, como se hacía antes, se comienza a buscar un*

conjunto singular en el nivel solicitado. Mientras no lo encuentre sigue buscando en los niveles superiores al solicitado, seleccionando al primero que encuentre. En este caso el conjunto que seleccionaría el método de búsqueda sería el conjunto $E_{3,8}$, puesto que es singular y pertenece al nivel solicitado, y el estado de la tabla tras ocupar el conjunto $E_{3,8}$ sería el mostrado en la Figura 5.29.

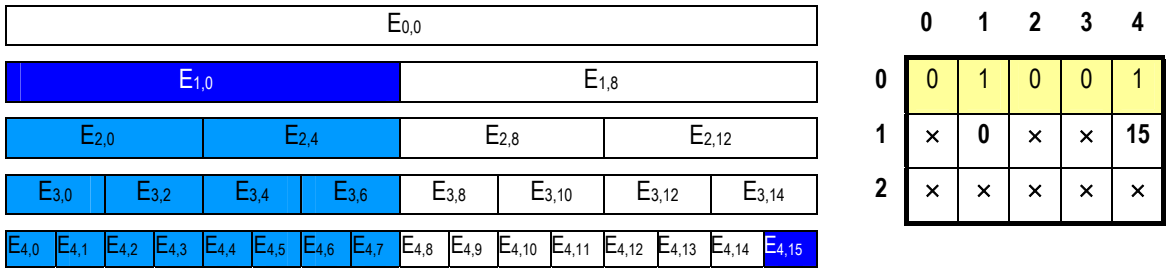


Figura 5.29 Situación final del Ejemplo 22.

En esta idea se apoya el siguiente método de búsqueda de un conjunto libre en un determinado nivel, que ha sido llamado *busquedaInteligenteDeConjunto*. Se verá en primer lugar la implementación realizada y a continuación su aplicación sobre un ejemplo.

Hay que señalar que ahora el método de asignación (se seguirá denominando así) hay que modificarlo para que en vez de llamar a *buscaPrimeraEntradaNuevaPeticion* (Figura 5.4, línea 6) para buscar un conjunto que cumpla las condiciones solicitadas, llame a *busquedaInteligenteDeConjunto*. En este caso también se elimina la utilización del método *reordenación* tras una eliminación, reflejando así la situación que se supuso al comienzo de este capítulo. Así pues, el método *busquedaInteligenteDeConjunto* devolverá el índice del conjunto singular de menor tamaño que sea capaz de albergar una petición del nivel indicado. La entrada y la salida de este método son las siguientes:

Entrada:

- **nivel:** Nivel del conjunto que se está buscando.

Salida:

- Índice *j* del conjunto libre $E_{i,j}$ que se asigna.

El pseudocódigo implementado del método *busquedaInteligenteDeConjunto* puede verse en la Figura 5.30. Puede observarse que se sigue utilizando la estructura *Singulares*. El procedimiento a seguir por este método es simplemente recorrer a partir del nivel pasado como argumento hasta el nivel 0 (Figura 5.30, líneas 2-6) los contadores de conjuntos singulares que poseen (Figura 5.30, línea 3), y en el momento en que se encuentre un nivel en que exista un conjunto singular (al estar la tabla normalizada, a lo

```
1:  entero   busquedaInteligenteDeConjunto (nivel)
2:  para i=nivel hasta 0 hacer
3:      si (daNumConjuntosSingulares(i) == 1) entonces
4:          devolver daIndiceConjuntoSingular(i,1);
5:      finsi
6:  finpara
7:  devolver -1;
8:  Fin busquedaInteligenteDeConjunto
```

Figura 5.30: Método *busquedaInteligenteDeConjunto*.

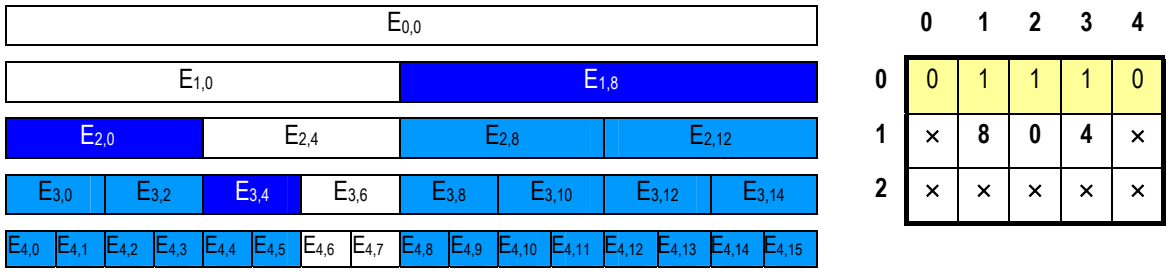
sumo habrá un conjunto en cada nivel), devuelve el índice del conjunto singular que posea (Figura 5.30, línea 4). Si no hay ningún conjunto singular apto para la petición que lo ha solicitado, devuelve el valor -1.

De esta forma, el pseudocódigo del método *inserción* quedaría como se muestra en la Figura 5.31. Hay que señalar que una vez encontrado el conjunto singular de menor tamaño que sea capaz de albergar la petición solicitada, se llama al método *ocupa* con ese índice. En ese caso, va a ocupar el primer conjunto del nivel solicitado entre los descendientes, en ese nivel, del conjunto singular elegido, que coincide ese índice con el del propio conjunto singular.

```
1:  booleano inserción (nivel, *indice)
2:  nuevasEntradas=2^(emax-nivel);
3:  si (nuevasEntradas+numEntradasOcupadas > max) entonces
4:      resultado= falso;
5:  sino
6:      *indice=busquedaInteligenteDeConjunto(nivel);
7:      ocupa(E_(nivel,*indice));
8:      actualizarSingularesTrasAnnadirPeticion(E_(nivel,*indice));
9:      resultado=verdadero;
10: finsi
11: devolver resultado;
12: Fin inserción
```

Figura 5.31: Método de *inserción* utilizando *busquedaInteligenteDeConjunto* como método de búsqueda.

Para los siguientes ejemplos, se toma como situación inicial la Figura 5.32.

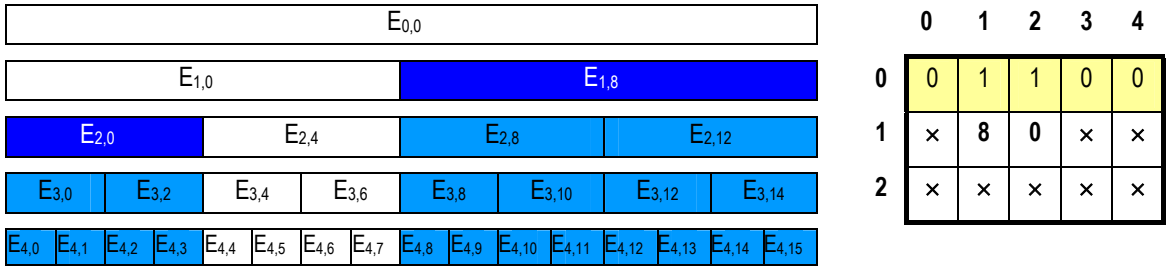


	0	1	2	3	4
0	0	1	1	1	0
1	×	8	0	4	×
2	×	×	×	×	×

Figura 5.32 Tabla de arbitraje inicial de los Ejemplos 23 y 24.

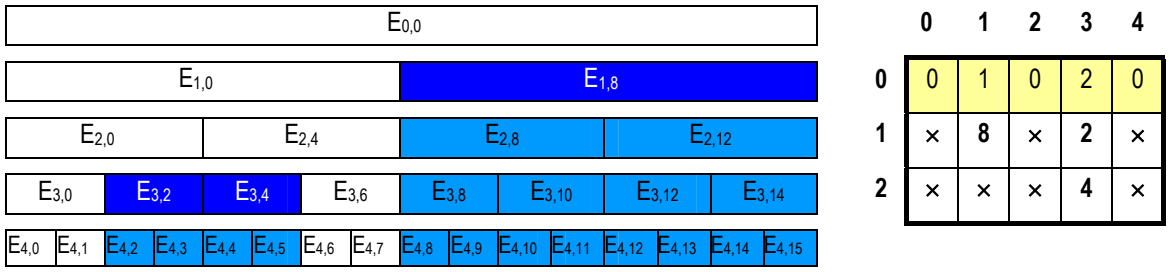
Ejemplo 23 *Inicialmente, la tabla de arbitraje de este ejemplo es igual a la de la Figura 5.32, y llega una petición que requiere un conjunto de nivel 3. El método inserción invoca a busquedaInteligenteDeConjunto para buscar un conjunto que asignarle a esa petición.*

El método busquedaInteligenteDeConjunto empieza a recorrer la estructura Singulares en el nivel 3. En ese nivel, ya hay un conjunto singular y por ello, selecciona el primer conjunto singular de ese nivel, que es $E_{3,4}$. Más tarde, en el método de inserción, la tabla de arbitraje pasa a estar como la de la Figura 5.33(a). Sin embargo, si en vez de utilizar este método de búsqueda, se hubiera utilizado la búsqueda anterior, la tabla de arbitraje resultante sería igual a la de la Figura 5.33(b).



	0	1	2	3	4
0	0	1	1	0	0
1	×	8	0	×	×
2	×	×	×	×	×

(a)



	0	1	2	3	4
0	0	1	0	2	0
1	×	8	×	2	×
2	×	×	×	4	×

(b)

Figura 5.33: (a) Situación final del Ejemplo 23 utilizando inserción inteligente.
(b) Situación final del Ejemplo 23 utilizando la inserción anterior.

Ejemplo 24 *Partiendo de la tabla de arbitraje mostrada en la Figura 5.32, llega una petición que requiere un conjunto de nivel 4. Siguiendo los pasos del método inserción, se*

invoca a busquedaInteligenteDeConjunto para buscar un conjunto que asignarle a esa petición.

El método de búsqueda comienza la búsqueda del conjunto candidato consultando en la estructura Singulares si hay algún conjunto singular en el nivel 4, que es el que ha sido solicitado. Comprueba que no hay ningún conjunto singular en ese nivel, por lo que pasa a buscar al nivel 3. Por tanto, el método busquedaInteligenteDeConjunto consulta en la estructura Singulares si hay algún conjunto singular en el nivel 3, obteniendo un resultado afirmativo para dicha consulta. El valor del índice de ese conjunto singular resulta ser 4. Como el conjunto $E_{3,4}$ es demasiado grande para atender la petición que se ha solicitado, el método inserción decide ocupar primer conjunto de nivel 4 descendiente de $E_{3,4}$, que es $E_{4,4}$. Así pues, el resultado de ocupar el conjunto $E_{4,4}$ es el mostrado en la Figura 5.34(a). Sin embargo, si se hubiera utilizado el método de búsqueda anterior, la tabla de arbitraje resultante sería la mostrada en la Figura 5.34(b)

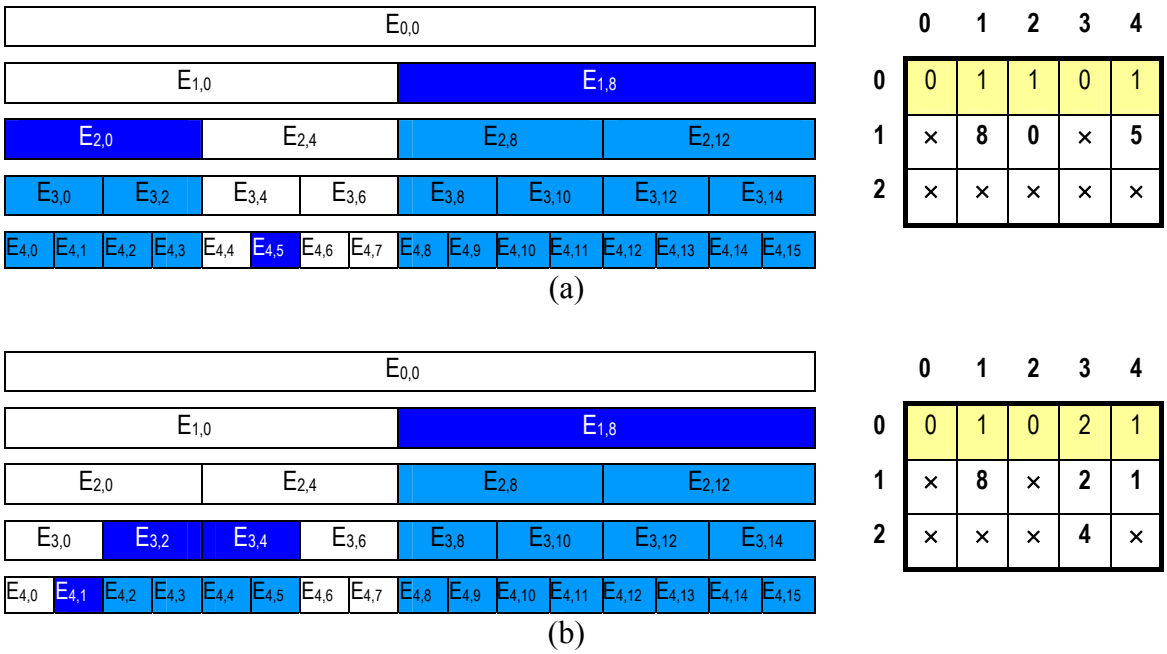


Figura 5.34: (a) Situación final del Ejemplo 24 utilizando inserción inteligente.
(b) Situación final del Ejemplo 24 utilizando la inserción anterior.

5.4 NUEVO MODELO PARA GESTIONAR LA TABLA DE ARBITRAJE

En el capítulo anterior se expusieron tres modelos en los que se combinaban de distintas maneras los algoritmos de gestión de la tabla de arbitraje. Gracias a la utilización de la estructura *Singulares* y a la definición de la *inserción inteligente*, se ha podido

desarrollar otro nuevo modelo con el que llevar a cabo la gestión de la tabla de arbitraje de mejor forma que en los modelos anteriormente planteados.

El hecho de haber introducido la inserción inteligente, hace que ya no sea necesaria la utilización de la reordenación tras una eliminación. Esto es debido a que el conjunto a ser ocupado ya no ha de ser el libre situado más a la izquierda de la tabla de arbitraje, si no que busca el conjunto singular de menor tamaño capaz de albergar dicha petición. Además, en caso de que el conjunto singular sea de mayor tamaño que el requerido, se ocupa el conjunto del tamaño requerido que pertenezca al singular seleccionado y que esté lo más a la izquierda. Todo esto hace que ahora ya sea indiferente el orden entre conjuntos singulares.

Por tanto, se propone ahora un nuevo modelo que, continuando la numeración anterior, se denominará *Modelo 4. En este caso,* tras realizar una eliminación solamente se realiza la desfragmentación, mientras que tras una inserción no se realiza ninguna acción más para recomponer la situación.

Modelo 4 { **Inserción**; **Eliminación + Desfragmentación**

Figura 5.35: Último modelo propuesto.

Al igual que se hizo al explicar los tres primeros modelos, se va a probar este modelo con la misma situación que se utilizó entonces (Ejemplos 6, 7 y 8, página 46).

La situación inicial que se va a utilizar para explicar este modelo, es una tabla de arbitraje como la de la Figura 5.36. En esta situación se produce la liberación del conjunto $E_{4,7}$, seguida de la petición de un conjunto de nivel 4.

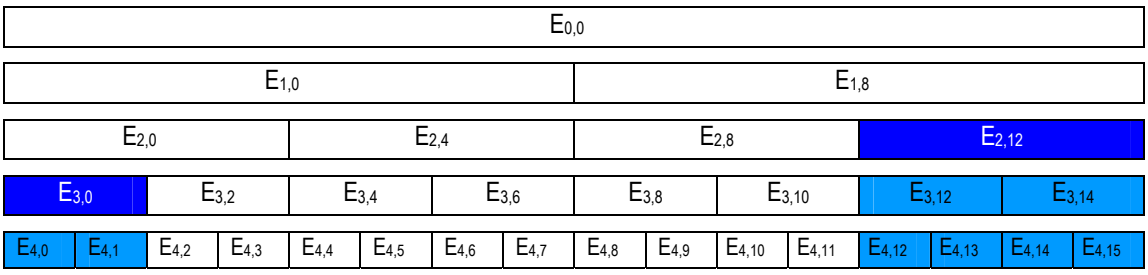


Figura 5.36: Situación inicial de la tabla de arbitraje en el Ejemplo 25.

Ejemplo 25 *Como se ha comentado anteriormente, se va a liberar el conjunto $E_{4,7}$ de la tabla de arbitraje de la Figura 5.36. El resultado de esta liberación se ve en la Figura 5.37. Puede observarse que la tabla sigue estando normalizada, por lo que al desfragmentar no se realiza ningún intercambio. Como no se produce reordenación, la situación final tras la eliminación sigue siendo la mostrada en la Figura 5.37.*

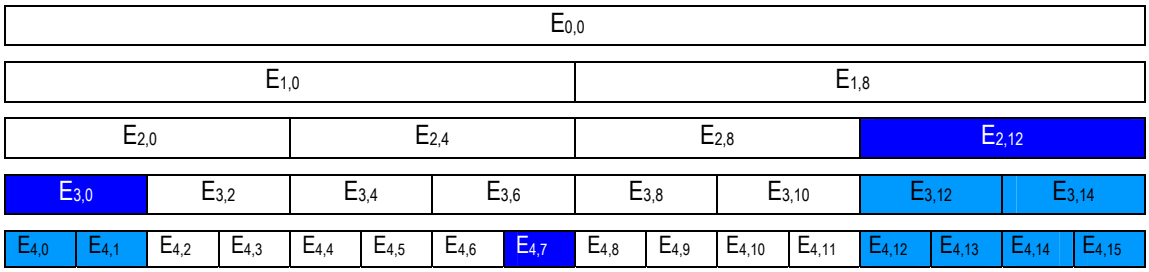


Figura 5.37: Tabla de arbitraje del Ejemplo 25 tras liberar el conjunto $E_{4,7}$.

En esta situación, la tabla de arbitraje ha recibido una petición de tipo 4. Al aplicar la inserción inteligente se le asigna el recién liberado conjunto $E_{4,7}$, de manera que la tabla es igual que la que se tenía de inicio para este ejemplo.

5.5 RESUMEN

A lo largo de este capítulo se han desarrollado las implementaciones para los algoritmos propuestos en [Alf03] y que ya fueron revisados en el capítulo anterior. Antes de describir la implementación de los algoritmos, se ha presentado la estructura de datos *Singulares*, a partir de la cual se plantean todos los algoritmos. Se presentaron varios ejemplos de su uso y una posible implementación en lenguaje C.

Tras esto, se han presentado las implementaciones de los algoritmos propuestos para la gestión de la tabla de arbitraje en el modelo formal descrito en [Alf03]. Los métodos presentados se pueden agrupar en dos grupos según su importancia. El primero, y más importante, es el compuesto por los métodos de *asignación*, *desfragmentación* y *reordenación*. El método de *asignación* se encarga de buscar el conjunto que atenderá una petición solicitada. Por otra parte, el método *desfragmentación* se encarga de unir secuencias de entradas aisladas para formar otras secuencias capaces de atender peticiones que requieran un conjunto de mayor tamaño. Para ello, se llevan a cabo una serie de intercambios entre conjuntos para obtener un conjunto de mayor número de entradas. El último de los métodos es el de *reordenación*, que también mediante intercambios dispone los conjuntos singulares de la tabla de arbitraje tal que cumplan un determinado orden entre ellos.

Entre los métodos auxiliares utilizados para implementar los anteriores tenemos *esConjuntoLibre*, *esConjuntoSingular* y *daNivelConjuntoSingularAncestro*. El método *esConjuntoLibre* se encarga de comprobar si un conjunto está libre. El método *esConjuntoSingular* comprueba si un conjunto es singular utilizando para ello la estructura *Singulares*. Por último, el método *daNivelConjuntoAncestroSingular* devuelve el nivel del conjunto ancestro singular de uno dado.

También se ha presentado una nueva alternativa para el algoritmo de inserción denominado *inserción inteligente*. En este caso, se ha cambiado el método de búsqueda del conjunto candidato, basándolo esta vez en la búsqueda de un conjunto singular en vez de buscar un conjunto libre situado lo más a la izquierda posible.

Para concluir el capítulo, se presentó una nueva alternativa de modelo de gestión de la tabla de arbitraje. A partir de la estructura *Singulares* y de la *inserción inteligente* se ha explicado con un ejemplo su funcionamiento.

Capítulo 6

EVALUACIÓN Y ANÁLISIS DE RESULTADOS

Después de estudiar el modelo de gestión de la tabla de arbitraje y de realizar los algoritmos capaces de llevar a cabo dicha gestión, es el turno de evaluar los métodos que se implementaron en base a los algoritmos del modelo formal propuesto en [Alf03]. Éste es el objetivo principal de este capítulo.

A la hora de evaluar un sistema, hay varios métodos de evaluación que pueden utilizarse, de los que se ha elegido el método basado en llevar a cabo *simulaciones*. En este capítulo se va a comentar todo lo relacionado con el simulador que se va a utilizar para obtener resultados que permitan evaluar los modelos de gestión de la tabla implementados: características generales, tipo de carga, resultados ofrecidos, etc. Tras esto, se detallará en qué van a consistir las pruebas a realizar con el simulador.

Por último, tras llevar a cabo las pruebas, con los resultados obtenidos se va a comentar el comportamiento de cada prueba realizada, para después compararlas entre sí. Para el modelo de gestión de la tabla de arbitraje que mejores resultados obtenga, se realizará un estudio más exhaustivo.

6.1 MÉTODO DE EVALUACIÓN

Para realizar la evaluación de prestaciones de un sistema, lo más recomendado es utilizar alguno de los métodos más comunes, y que básicamente son:

- **Mediciones reales:** Consisten en tomar medidas directamente sobre el sistema o sistemas implicados en la evaluación a realizar, es decir, se necesita un sistema existente y en caso de no existir, como mínimo se necesita un prototipo sobre el que llevar a cabo las mediciones. Cuando se inició este trabajo no había ninguna máquina InfiniBand disponible en la Escuela, haciendo imposible utilizar este método de evaluación.

- **Modelos analíticos:** Los modelos analíticos son una alternativa bastante económica porque no necesitan de la existencia previa de ningún sistema ni ninguna herramienta de medición en concreto, y además, proporcionan unos resultados de forma casi inmediata. El inconveniente de este método es que la precisión que se pueda obtener con él no sea muy grande al tener que abstraer mucho el sistema modelado.

- **Simulaciones:** La simulación es un método utilizado ampliamente en el campo de investigación, puesto que se puede modelar con precisión cualquier sistema. Además, es bastante útil cuando el sistema a evaluar no puede ser modelado analíticamente, y también permite obtener diferentes niveles de detalle del sistema. Por estos motivos se ha optado por este método de evaluación para analizar los métodos de gestión de la tabla de arbitraje que se han propuesto en este trabajo.

6.2 SIMULADOR

Como se ha indicado en el apartado anterior, se ha utilizado un simulador para evaluar los métodos de gestión de la tabla de arbitraje propuestos en el capítulo anterior. En líneas generales el simulador modela:

- **Estado de la tabla de arbitraje:** Mantiene información sobre el estado de la tabla de arbitraje y el de las estructuras auxiliares que utiliza.

- **Funcionamiento de la tabla de arbitraje:** Métodos que realizan la gestión de la tabla de arbitraje.

El simulador está dirigido por eventos, que son las peticiones de entradas o eliminación de conjuntos sobre la tabla de arbitraje. Además, el simulador ha sido desarrollado por el autor de este proyecto, utilizando el paradigma de programación orientada a objetos, e implementado por medio del lenguaje de programación C++ en un entorno Linux.

6.2.1 Arquitectura simulada

El simulador representa la tabla de arbitraje de un puerto de salida de un conmutador, encaminador o terminal de una red InfiniBand. Como se ha dicho al introducir la simulación como método de evaluación, ésta permite obtener diferentes niveles de detalle del sistema que está siendo simulado. Por ello, en el simulador desarrollado en este proyecto, las peticiones de inserción ocupan por completo las entradas de la tabla que les son asignadas, es decir, se les asigna el máximo peso de esas entradas, cuando en realidad, en una misma entrada, puede haber agrupadas varias peticiones que hayan sido solicitadas por varias conexiones del mismo nivel de servicio hasta alcanzar el peso máximo [Alf03]. Se ha decidido que el simulador no llegue a ese nivel de detalle respecto a la realidad puesto que no es necesario para el tipo de estudio que se tiene que hacer.

El tamaño de la tabla de arbitraje modelada en el simulador es de 64 entradas, que coincide con el tamaño máximo de una tabla de arbitraje en InfiniBand. Además de la tabla en sí, se han implementado otras estructuras auxiliares que permiten gestionar las entradas, almacenando información sobre la petición que la está ocupando en ese momento, de tal manera que permiten identificar la petición y decir de qué tipo es. También se ha modelado la estructura de datos *Singulares* puesto que va a ser bastante utilizada por los métodos.

6.2.2 Eventos

Al introducir el simulador al principio de este capítulo, se ha comentado que éste está dirigido por eventos. Por ello, es conveniente explicar cuáles son los eventos que provocan que el estado de la tabla de arbitraje del simulador sufra cambios. En concreto, al simulador llegan dos tipos de eventos:

- **Petición de inserción:** Este evento indica al simulador que se quiere ocupar un conjunto de la tabla de arbitraje. Se denominará, a partir de ahora, como *Operación de Inserción* al conjunto de acciones que se tengan que llevar a cabo en el simulador para ocupar ese conjunto de entradas. Si trasladásemos este evento a la realidad, éste se daría cuando llegase una conexión al puerto de salida de cualquier

terminal, encaminador o conmutador de una red InfiniBand, a través de uno de sus canales virtuales. Según el nivel de servicio al que pertenezca esta conexión, se ha de realizar una petición de inserción a la tabla de arbitraje, para que ésta la tenga en cuenta a la hora de arbitrar las salidas del puerto.

- **Petición de eliminación:** Este evento provoca que el simulador libere un conjunto de entradas de la tabla de arbitraje que hasta ese momento estaban ocupadas, o lo que es lo mismo, provoca una *Operación de Eliminación*, que consiste en el conjunto de acciones que se han de realizar para liberar el conjunto anteriormente mencionado. En la realidad, este evento se daría cuando una aplicación terminara y no hubiese otras aplicaciones agregadas en ese canal virtual. En ese caso debe procederse a liberar las entradas de la tabla de arbitraje que ese canal virtual estaba ocupando.

Así pues, una vez explicados los eventos que recibe el simulador, ya se puede explicar qué tipo de carga se va a utilizar para realizar las pruebas.

6.2.3 Carga utilizada

En este proyecto se realizarán simulaciones con **carga sintética**, con una distribución de operaciones como se indica a continuación. Se generarán aleatoriamente las peticiones de inserción y de eliminación con una probabilidad del 50% para cada tipo. Una vez se ha decidido si se va a producir una petición de inserción o de eliminación, necesitamos generar qué tipo de inserción o eliminación se va a hacer. Así pues, se va a comentar de manera más detallada qué información es necesaria en cada caso:

- **Petición de inserción:** Tras decidirse que el evento a introducir en el simulador va a ser éste, faltaría decidir de qué nivel va a ser el conjunto que se va a solicitar. Para ello, se obtiene aleatoriamente el nivel de entre todos los pertenecientes al rango [1-*emax*[3]]. Una situación especial que se podría dar es que no hubiera entradas suficientes en la tabla de arbitraje para ocupar un conjunto del nivel que se acaba de elegir. En ese caso se desestima la petición y se genera un nuevo evento.

- **Petición de eliminación:** Una vez se ha decidido que se va a llevar a cabo una eliminación en el simulador, falta saber qué petición de las que están siendo atendidas dentro de la tabla de arbitraje ha de ser eliminada de ella. Una situación especial que se puede dar es que cuando se solicite que se elimine una petición, la tabla esté vacía, por lo que no se podría eliminar ninguna petición.

[3] El valor *emax* es igual al valor del nivel de la tabla de arbitraje más alto.

6.2.4 Parámetros de entrada

A la hora de ejecutar el simulador desarrollado en este proyecto es necesario proporcionarle ciertos parámetros para establecer la configuración a evaluar y para indicar los resultados que se quieren obtener. Los principales parámetros de entrada del simulador son:

- **Modelo a ejecutar.** Se introduce el número que identifica el modelo de ejecución que se quiere ejecutar. Éste ha de ser uno de los 4 modelos de ejecución propuestos en capítulos anteriores, y que más adelante se describirán con más detalle.

- **Número de simulaciones a ejecutar.** Especifica el número de simulaciones que se van a repetir para una misma prueba. Esto permite descartar los valores extraños que se obtenían por motivos externos.

- **Fichero de semillas.** Se indica el nombre del fichero de texto que contiene los valores que servirán como semillas para los generadores aleatorios que permiten obtener la secuencia de eventos que recibirá el simulador, utilizándose cada una de ellas para cada una de las simulaciones.

- **Identificador de la ejecución.** Este nombre sirve para poder identificar los resultados obtenidos por cada una de las simulaciones.

- **Fichero de configuración del modo debug.** Este parámetro es opcional, y es el nombre del fichero que contiene las operaciones para las que el usuario quiere conocer el estado de la tabla de arbitraje durante su ejecución.

6.2.5 Resultados ofrecidos

Una vez realizadas las simulaciones, la herramienta desarrollada debe ofrecer cualquier tipo de resultado que pueda llegar a hacer falta para después poder evaluar los métodos implementados. Estos resultados que se obtienen se pueden agrupar en tres grupos, que serán desarrollados en los próximos subapartados (del 6.2.5.1 al 6.2.5.3).

6.2.5.1 Resultados relacionados con las peticiones

Los resultados que se van a listar en este subapartado tienen que ver con el número de peticiones que ha generado el simulador. Para cada resultado se calcula el total, y además

ese total se desglosa en el número de ocurrencias que ha habido en cada uno de los niveles considerados en la tabla de arbitraje.

- *Número de peticiones solicitadas.* Número de peticiones de inserción que se han generado.

- *Número de peticiones atendidas.* Número de peticiones solicitadas que han sido aceptadas en el simulador.

- *Número de peticiones no atendidas.* Número de peticiones que han sido denegadas porque no había suficiente número de entradas para que pudieran ser atendidas.

- *Número de peticiones eliminadas.* Número de peticiones de eliminación que se han podido realizar.

- *Número de peticiones eliminadas fallidas.* Número de peticiones de eliminación que no se han podido realizar porque no había ningún conjunto ocupado dentro de la tabla de arbitraje.

6.2.5.2 Resultados relacionados con los intercambios

Los resultados que se van a listar en este subapartado se refieren a todo lo relacionado con los intercambios que se produzcan durante el desarrollo de una simulación. Para los tres primeros resultados se ofrece el valor total, y también la distribución del número de ocurrencias por cada uno de los niveles.

- *Número de intercambios.* Intercambios llevados a cabo en la tabla de arbitraje.

- *Número de intercambios durante la desfragmentación.* Número de intercambios que se han realizado a lo largo de la aplicación del método de desfragmentación.

- *Número de intercambios durante la reordenación.* Número de intercambios llevados a cabo sobre la tabla de arbitraje a lo largo de la aplicación del método de reordenación.

- *Intercambios por cada desfragmentación.* Número medio de intercambios por desfragmentación aplicada.

- *Intercambios por cada reordenación.* Número medio de intercambios por reordenación aplicada.

- *Desfragmentaciones con intercambio.* Porcentaje de desfragmentaciones en las que ha habido al menos un intercambio durante su aplicación.

- *Reordenaciones con intercambio.* Porcentaje de reordenaciones en las que se ha producido al menos un intercambio durante su aplicación.

6.2.5.3 Resultados relacionados con los tiempos de ejecución

Todos los resultados que a continuación se van a listar están referidos a los tiempos de ejecución de las *operaciones* realizadas durante la simulación y de las *tareas* que las conforman, tomando como unidad de medida el microsegundo. Antes de comentar los resultados que se obtienen, y puesto que se han mencionado los términos de *Tarea* y de *Operación*, es importante diferenciarlos.

El término *Tarea* coincide con alguno de los métodos implementados en el simulador o parte de ellos, coincidiendo estos métodos, la mayoría de las veces, con alguno de los que se propusieron en los capítulos 4 y 5. Además, las tareas pueden tener incluida alguna *Subtarea.*

Por ejemplo, la tarea *Desfragmentación* está referida a la aplicación del método *desfragmentación* que se propuso en el Apartado 5.2.3. Esta tarea contiene dos subtareas que son: "*Búsqueda Desfragmentación"* e *"Intercambio Desfragmentación"*. La primera tarea se refiere a la parte del método *desfragmentación* en la que se realiza la búsqueda de dos conjuntos singulares de un mismo nivel. Mientras que la segunda comprende la parte del método *desfragmentación* en la que se invoca al método *intercambioConjuntos*, que es el encargado de realizar el intercambio entre dos conjuntos con el propósito de normalizar la tabla. Para evitar posibles confusiones a la hora de distinguir estos dos últimos términos (*tarea* y *subtarea*), a partir de ahora se denominará *Tarea* a ambos.

Por otro lado, el término *Operación* indica el total de tareas que realiza el simulador como respuesta a un evento que ha recibido.

Así pues, tras distinguir los conceptos de *operación* y de *tarea*, se van a comentar cuáles son estas operaciones y tareas de las que el simulador obtiene información relativa a sus tiempos de ejecución. Estas son:

- *Operación de Inserción.* Operación provocada por una petición de inserción que ha sido aceptada. En este caso se mide el tiempo empleado por las tareas que conforman esta operación.

- *Operación de Inserción Fallida.* Operación de inserción en la que la petición de inserción no ha podido ser aceptada pues no había entradas suficientes para ello.

- *Inserción.* Tiempo empleado por el método *inserción*.

- *Búsqueda de Inserción.* Tiempo empleado por el método *buscaPrimeraEntradaNuevaPeticion* o por el método *busquedaInteligenteDeConjunto*, según se utilice el uno o el otro.

- *Actualización de Singulares en la inserción.* Tiempo empleado en actualizar la estructura de datos *Singulares* tras haberse realizado una inserción en la tabla de arbitraje.

- *Operación de Eliminación.* Operación en la que la petición de eliminación se ha podido llevar a cabo. Al igual que en la operación de inserción, se mide el tiempo empleado por las tareas que forman parte de esta operación.

- *Operación de Eliminación Fallida.* Operación en la que el simulador recibió una petición de eliminación, pero al estar vacía la tabla de arbitraje, no se podía eliminar nada.

- *Eliminación.* Tiempo empleado en liberar un conjunto de la tabla de arbitraje.

- *Actualización de Singulares en la Eliminación.* Tiempo empleado en actualizar la estructura de datos *Singulares* tras haberse realizado una eliminación en la tabla de arbitraje.

- *Desfragmentación.* Tiempo empleado en aplicar el método *desfragmentación*.

- *Búsqueda de Desfragmentación.* Tiempo empleado en la búsqueda de conjuntos singulares que puedan provocar que la tabla quede no normalizada durante el método *desfragmentación*.

- *Intercambio de Desfragmentación.* Tiempo utilizado para realizar intercambios durante la aplicación del método *desfragmentación*.

- *Reordenación.* Tiempo empleado en aplicar el método *reordenación*.

- *Búsqueda de Reordenación.* Tiempo empleado en la búsqueda de conjuntos singulares que estén desordenados entre sí durante el método *reordenación*.

- *Intercambio de Reordenación.* Tiempo utilizado para realizar intercambios a lo largo de la aplicación del método *reordenación*.

- *Actualización de Singulares en Intercambio***.** Tiempo empleado en actualizar la estructura *Singulares* tras haberse realizado un intercambio entre dos conjuntos de la tabla de arbitraje.

Para todas estas operaciones y tareas, se han calculado los siguientes valores:

- *Tiempo total de ejecución.*

- *Tiempo medio de ejecución.*

- *Desviación típica de la muestra.*

- *Peor tiempo.* Mayor tiempo obtenido de todas las aplicaciones realizadas de la operación o tarea.

- *Operación de peor tiempo.* Número de la operación en la que se ha dado el mayor tiempo.

Todos estos resultados ayudarán a evaluar los métodos que se han implementado.

6.3 PRUEBAS REALIZADAS

Tras presentar el simulador y los resultados que se obtienen de él, se describen a continuación las pruebas que se van a realizar y qué configuraciones se han considerado para esas pruebas.

6.3.1 Modelos simulados

En los capítulos 4 y 5 se han presentado 4 modelos en los que se combinan los diferentes métodos de gestión de la tabla de arbitraje. En este apartado se van a volver a comentar esos modelos propuestos, entrando en más detalle e indicando qué tareas conforman cada una de las operaciones que realiza el simulador.

6.3.1.1 Modelo 1

Este es el primer modelo que se presentó en el modelo formal desarrollado en [Alf03], y se comentó en el Apartado 4.2 de este trabajo (página 45). La Tabla 6.1(a) muestra este modelo desglosando cada operación en las tareas que contiene.

La Operación de Inserción está formada por la tarea de Inserción seguida de la Desfragmentación, mientras que en la Operación de Eliminación se aplicaría primero la tarea de Eliminación seguida de la Desfragmentación.

6.3.1.2 Modelo 2

Este modelo es el segundo que se presentó junto al modelo anterior en el Apartado 4.2 de este trabajo, siendo su estructura la mostrada por la Tabla 6.1(b). La Operación de Inserción está formada solamente por la tarea de Inserción, mientras que en la Operación de Eliminación se llevan a cabo, en este orden, las tareas de Eliminación, Reordenación y Desfragmentación.

6.3.1.3 Modelo 3

Este modelo es el último que se presentó en el Apartado 4.2. Las operaciones y tareas que lo forman se pueden ver en la Tabla 6.2(a), siendo en esta ocasión la Operación de Inserción igual que en el modelo anterior, y en la Operación de Eliminación se intercambia el orden de las tareas de Reordenación y Desfragmentación.

6.3.1.4 Modelo 4

Este modelo se presentó en el Apartado 5.4 de este proyecto. Hay que señalar que este modelo no se deriva directamente del trabajo desarrollado en [Alf03] y es una propuesta directa de este Proyecto Fin de Carrera. La principal novedad de este modelo es la utilización del método *busquedaInteligenteDeConjunto* en la tarea "Búsqueda Inserción", ya que en los modelos 1, 2 y 3 se ha utilizado el método *buscaPrimeraEntradaNuevaPeticion*. La estructura de este ejemplo está reflejada en la Tabla 6.2(b), la cual muestra todas las tareas utilizadas por las dos operaciones que maneja el simulador. Puede observarse que se ha vuelto a dejar de utilizar la tarea de Reordenación.

Modelo 1	Op. Inserción	Inserción	Búsqueda Inserción
			Actualización Singulares Inserción
		Desfragmentación	Búsqueda Desfragmentación
			Intercambio Desfragmentación
			Actualización Singulares Intercambio
	Op. Eliminación	Eliminación	Actualización Singulares Eliminación
		Desfragmentación	Búsqueda Desfragmentación
			Intercambio Desfragmentación
			Actualización Singulares Intercambio

(a)

Modelo 2	Op. Inserción	Inserción	Búsqueda Inserción
			Actualización Singulares Inserción
	Op. Eliminación	Eliminación	Actualización Singulares Eliminación
		Reordenación	Búsqueda Reordenación
			Intercambio Reordenación
			Actualización Singulares Intercambio
		Desfragmentación	Búsqueda Desfragmentación
			Intercambio Desfragmentación
			Actualización Singulares Intercambio

(b)

Tabla 6.1: (a) Modelo 1 desglosado por tareas. (b) Modelo 2 desglosado por tareas.

Modelo 3	Op. Inserción	Inserción	Búsqueda Inserción	
			Actualización Singulares Inserción	
	Op. Eliminación	Eliminación	Actualización Singulares Eliminación	
		Desfragmentación	Búsqueda Desfragmentación	
			Intercambio Desfragmentación	Actualización Singulares Intercambio
		Reordenación	Búsqueda Reordenación	
			Intercambio Reordenación	Actualización Singulares Intercambio

(a)

Modelo 4	Op. Inserción	Inserción	Búsqueda Inserción	
			Actualización Singulares Inserción	
	Op. Eliminación	Eliminación	Actualización Singulares Eliminación	
		Desfragmentación	Búsqueda Desfragmentación	
			Intercambio Desfragmentación	Actualización Singulares Intercambio

(b)

Tabla 6.2: (a) Modelo 3 desglosado por tareas. (b) Modelo 4 desglosado por tareas.

6.3.2 Descripción de las pruebas

Hasta ahora, en este capítulo se han descrito el simulador (características, eventos, operaciones y tareas) y los modelos que se van a probar sobre él. Ahora falta describir cómo se van a probar en el simulador los cuatro modelos propuestos. Pero antes de ello, es importante definir de una forma más precisa los términos *Simulación* y *Ejecución*.

Se va a denominar *Simulación* a un conjunto de un millón de operaciones realizadas por el simulador, mientras que se denomina *Ejecución* a un conjunto de simulaciones efectuadas durante una misma llamada al simulador.

Cuando empezaron a realizarse las pruebas sobre el simulador, podía observarse que en los tiempos de ejecución de casi todas la operaciones/tareas, se daban algunas ocurrencias en las que los valores obtenidos eran excesivamente grandes respecto al resto de tiempos, en concreto sobre 10^4 veces mayor. Al repetir una misma prueba varias veces, los mayores tiempos empleados por cada operación/tarea eran muy diferentes a los obtenidos en pruebas anteriores, y se daban en operaciones/tareas distintas. Además seguían siendo demasiado altos en comparación con el resto de tiempos. Tras varias pruebas, seguía pasando lo mismo, y por ello, era necesario estudiar más a fondo este problema.

Así pues, era necesario implementar un sistema de debug en el simulador. En concreto, la información que debía proporcionar el debug para cada operación era la siguiente: estado inicial y final de la tabla de arbitraje, tareas realizadas a lo largo de la operación y el tiempo de ejecución empleado por cada una de ellas.

Tras implementar el sistema de debug en el simulador, se realizó una prueba para obtener los peores tiempos y las operaciones/tareas en las que se daban. Después, se repitió la prueba, utilizando esta vez el sistema de debug, con el fin de estudiar las operaciones/tareas en las que se habían dado los peores casos. Los resultados obtenidos fueron los siguientes:

- ✓ Las operaciones/tareas estudiadas no eran tan complejas como para emplear tiempos de tan alta magnitud.

- ✓ La mayoría de los tiempos empleados en realizar las operaciones/tareas que en la prueba anterior habían sido los peores casos, en esta prueba eran muchísimo menores.

Aún así, se decidió repetir la misma prueba varias veces para comprobar si el comportamiento de las operaciones/tareas seguía algún patrón. Tras repetir varias veces la

prueba, se observó que los tiempos empleados para llevar a cabo las operaciones/tareas eran similares.

La conclusión que se sacó de este estudio fue que esos valores extremadamente exagerados podían ser causados por posibles cambios de contexto del sistema operativo llevados a cabo a lo largo de la ejecución, o también se cuestionó si se podían dar por fallos de caché durante la ejecución del simulador, haciendo necesario el acceso a dispositivos de almacenamiento de menor velocidad. Fuera cual fuera el motivo, provocaba los grandes retrasos que distorsionaban los resultados que el simulador obtenía.

La solución que se propuso fue la de realizar 10 veces la misma simulación, y una vez obtenidos los resultados, seleccionar aquellos en los que no existan valores provocados por causas ajenas al simulador. Por consiguiente, cada operación/tarea que se realiza durante la ejecución del simulador, se repite 10 veces, con lo que podemos tomar los tiempos de cada una de esas veces, y se calcula la mediana de dichos valores. Así pues, los resultados que se muestran en las secciones siguientes son las medianas obtenidas de cada operación/tarea. De esta forma se eliminan los valores extremos y se realizan las comparaciones con un valor estadísticamente significativo.

En resumen, se van a lanzar simulaciones consistentes en un millón de operaciones. En concreto, dichas operaciones son inserciones y eliminaciones que se aplican sobre la tabla de arbitraje con una probabilidad del 50%.

6.4 ANÁLISIS DE LOS RESULTADOS

A lo largo de este capítulo, se ha ido detallando todo lo referente a las pruebas a realizar para evaluar los métodos implementados, como han sido: modelo de evaluación, descripción del simulador y descripción de las pruebas a realizar. A continuación, se van a comentar los resultados obtenidos tras simular los 4 modelos propuestos, para después compararlos y elegir el que proporcione mejores resultados. Para cada modelo, se mostrará una tabla con los tiempos de ejecución obtenidos y una figura con estas cuatro gráficas:

- Tiempo total de ejecución empleado en cada operación.

- Tiempo total de ejecución empleado en cada tarea importante.

- Distribución de los tiempos de ejecución de *Operación de Inserción* y de las principales tareas que la componen.

- Distribución de los tiempos de ejecución de *Operación de Eliminación* y de las principales tareas que la componen.

Estas gráficas serán etiquetadas como (a), (b), (c) y (d), respectivamente.

Los resultados que se van a comentar fueron obtenidos tras realizar las pruebas en un ordenador con procesador modelo AMD Athlon™ XP 1800+ con una velocidad de 1,53 GHz, 512 Mb de Memoria RAM DDR y sistema operativo Linux (Fedora Core 3).

6.4.1 Resultados obtenidos para el Modelo 1

Operación/Tarea	Tpo. total	Tpo. medio	Desviación típica
Operación de Inserción	**13063258 µs**	**30,365 µs**	**14,621**
Inserción	6372569 µs	14,813 µs	1,653
Búsqueda Inserción	883823 µs	2,054 µs	0,227
Act. Singulares Inserción	1305578 µs	3,035 µs	1,163
Desfragmentación Inserción	3849510 µs	8,948 µs	14,504
Búsqueda Desfragmentación	896048 µs	2,083 µs	0,471
Intercambio Desfragmentación	840208 µs	51,629 µs	22,592
Operación de Eliminación	**15428036 µs**	**35,862 µs**	**29,013**
Eliminación	4970904 µs	11,555 µs	2,132
Act. singulares Eliminación	1173646 µs	2,728 µs	0,614
Desfragmentación Eliminación	7504323 µs	17,443 µs	29,378
Búsqueda Desfragmentación	1014093 µs	2,357 µs	0,905
Intercambio Desfragmentación	4089010 µs	55,060 µs	20,410
Operación Inserción Fallida	**316064 µs**	**4,582 µs**	**0,493**
Operación Eliminación Fallida	**148708 µs**	**2,106 µs**	**0,308**

Tabla 6.3: Resultados de tiempos obtenidos para el Modelo 1.

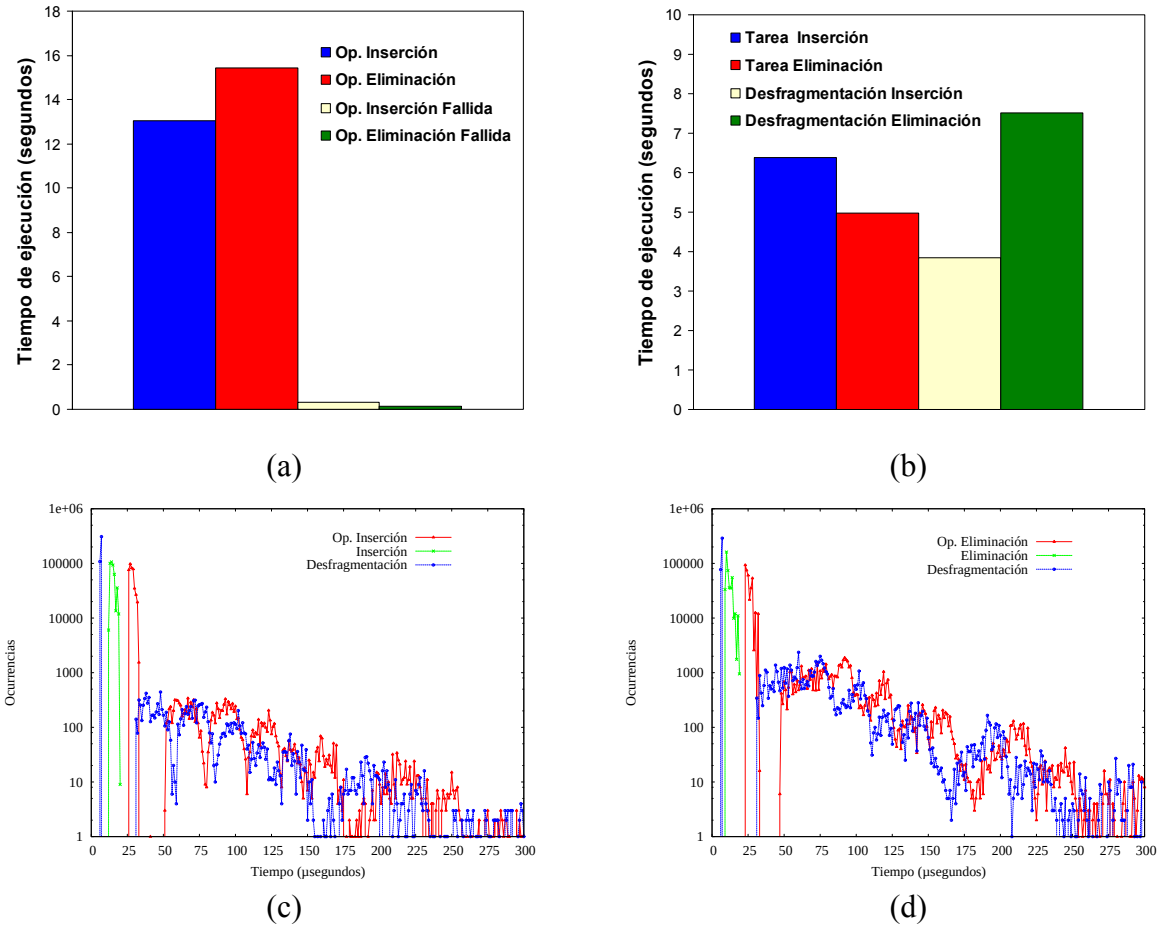


Figura 6.1: Tiempos de operaciones y tareas y su distribución para el Modelo 1.

En este apartado se van a describir los resultados obtenidos tras simular el Modelo 1. Hay que recordar que en *Operación de Inserción* se aplicaría primero la tarea *Inserción* seguida de *Desfragmentación*. Mientras tanto, en *Operación de Eliminación* se realiza primero la tarea *Eliminación* y a continuación *Desfragmentación*.

En la Tabla 6.3 se muestran los tiempos totales, medios y la desviación típica de cada operación y sus tareas asociadas. En la Figura 6.1(a) y (b) se pueden observar los tiempos totales de forma gráfica para las operación y las tareas, respectivamente. En concreto, la Figura 6.1(a) muestra cómo se ha distribuido el tiempo entre las operaciones durante toda la ejecución de la prueba. Gran parte de éste se ha repartido entre las *Operaciones de Inserción* y las *Operaciones de Eliminación*, siendo estas últimas las que más tiempo han empleado. Para explicar estos datos, la Figura 6.1(b) muestra cómo la tarea *Desfragmentación* aplicada tras una Inserción ha consumido menos tiempo que cuando ha sido aplicada tras una tarea de Eliminación. La diferencia de tiempo es mayor de la que hay entre los tiempos utilizados por las tareas *Inserción* y *Eliminación*, ya que la primera utiliza más tiempo.

Este mayor empleo de tiempo por parte de *Desfragmentación* tras una *Eliminación* es debido al número de intercambios realizados durante su aplicación, tal y como se observa en la Tabla 6.4. El 82,03% de los intercambios realizados durante la tarea *Desfragmentación*, se han producido cuando se ha aplicado durante una *Operación de Eliminación*. Así pues, queda despejar por qué se produce ese alto porcentaje de intercambios en las desfragmentaciones realizadas durante *Operación de Eliminación*.

	Número de intercambios	Porcentajes
Desfragmentación tras Inserción	16274	17,97%
Desfragmentación tras Eliminación	74264	82,03%
Total Desfragmentación	90537	

Tabla 6.4: Intercambios realizados durante la Desfragmentación en el Modelo 1.

Una posible causa sería que se dieran más operaciones de eliminación que operaciones de inserción, pero la Tabla 6.5 lo descarta, ya que el número de ocurrencias es prácticamente el mismo.

La única conclusión que se puede obtener es que la tarea *Inserción* no provoca que la tabla de arbitraje pase a estar no normalizada tantas veces como se pensaba. El motivo de haber pensado esto era porque no se aplicaba el método *reordenación* tras una tarea de *Eliminación* sobre la tabla de arbitraje, como se indicó en el Apartado 4.1 (Página 44, Ejemplo 4), ya que al utilizarlo, el método de *inserción* nunca dejaría no normalizada la tabla de arbitraje

	Número de ocurrencias
Peticiones atendidas	430201
Peticiones no atendidas	68973
Peticiones eliminadas	430210
Peticiones eliminadas fallidas	70616

Tabla 6.5: Peticiones recibidas en el Modelo 1.

Para explicar el comportamiento de *Operación de Inserción* se utiliza la Figura 6.1(c), que representa la distribución de los tiempos empleados por esta operación y las tareas que la componen: *Inserción* y *Desfragmentación*, que también se muestran en la figura. Se pueden observar dos partes diferenciadas: una comprendida por las ocurrencias cuya duración está dentro del rango [26 µs, 33 µs], y la otra formada por las ocurrencias que han necesitado para ejecutarse un tiempo superior a 51 µs.

Observando la primera parte, se toma como hipótesis que todas las *Operaciones de Inserción* que se encuentran dentro de ella, son aquellas en las que no se ha dado ningún intercambio durante la aplicación de la tarea *Desfragmentación*. Hay varios indicios que hacen tomar esta decisión:

- Esta parte tiene un gran porcentaje de las *Operaciones de Inserción* que se aplicaron durante la simulación y el porcentaje de ocurrencias de esta operación sin intercambio es muy alto.

- El comportamiento de la gráfica tiene, al igual que la tarea *Desfragmentación*, dos partes diferenciadas.

Viendo la Tabla 6.6, puede observarse como tan sólo un 3,16% de las desfragmentaciones realizadas durante *Operación de Inserción*, ha llevado a cabo al menos un intercambio.

	Porcentajes
Desfragmentaciones tras Inserción con intercambio	3,16%
Desfragmentaciones tras Eliminación con intercambio	15%

Tabla 6.6: Porcentaje de *Desfragmentaciones* del Modelo 1 en las que se ha producido algún intercambio.

La Figura 6.2 muestra los porcentajes acumulados de las ocurrencias de *Operación de Inserción*, y se puede observar como se llega al 97% de ocurrencias cuando el tiempo empleado es de 33µs, justo cuando termina la primera parte que se ha diferenciado de

Operación de Inserción. Ese 97% es también el porcentaje de casos de esta operación en las que no se ha aplicado ningún intercambio. De todos modos, la justificación de la división del comportamiento de *Operación de Inserción* en dos partes no queda totalmente clara. Para completarla, hay que fijarse además en el comportamiento de las dos tareas que hay incluidas dentro de *Operación de Inserción*.

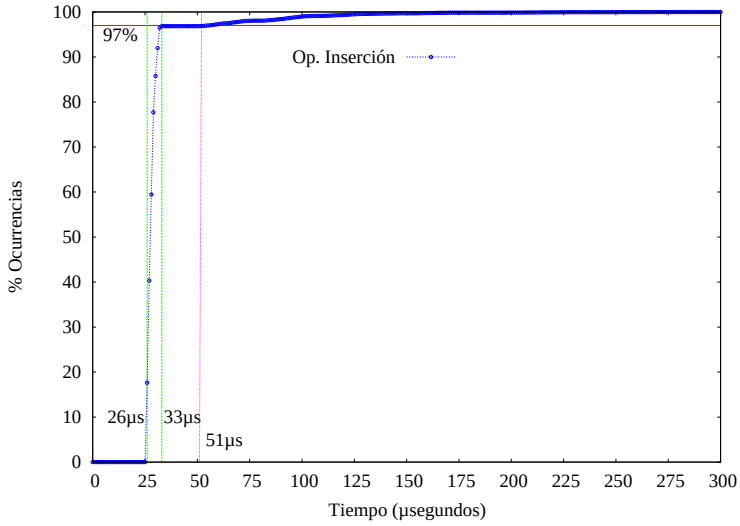


Figura 6.2: Porcentajes acumulados de ocurrencias de *Operación de Inserción* en el Modelo 1.

La Figura 6.3 muestra la distribución por tiempos de las ocurrencias que se han llevado a cabo de la tarea *Desfragmentación* aplicada tras una *Inserción*, y de las tareas que se realizan dentro de ella: *Búsqueda Desfragmentación* e *Intercambio Desfragmentación*.

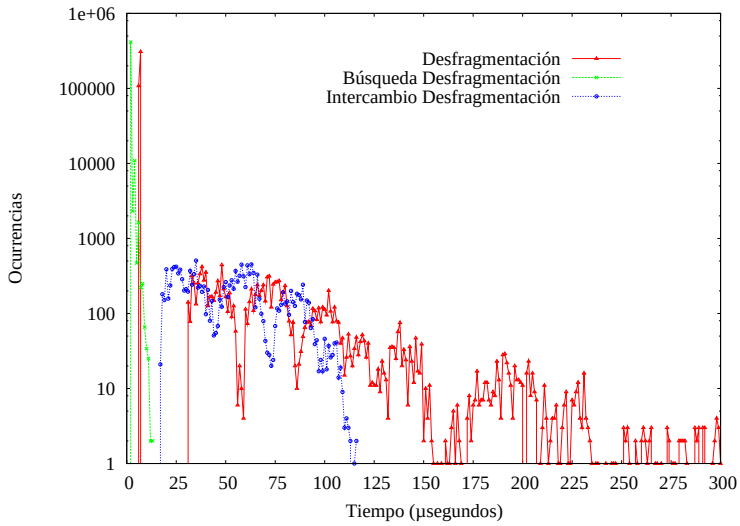


Figura 6.3: Distribución de los tiempos de ejecución de la tarea *Desfragmentación* aplicada tras una *Inserción* y de las tareas que la componen. Modelo 1.

La tarea *Desfragmentación* está dividida en dos grupos, el primero va de los 6 a los 7 microsegundos y el segundo de 31 microsegundos en adelante. El primer grupo se ha supuesto también que está formado por las ocurrencias en las que no se han dado intercambios. Esta hipótesis se puede comprobar viendo que ningún intercambio se lleva a cabo en menos de 21 µs, cuando el máximo tiempo del primer grupo de ocurrencias es de 7 µs.

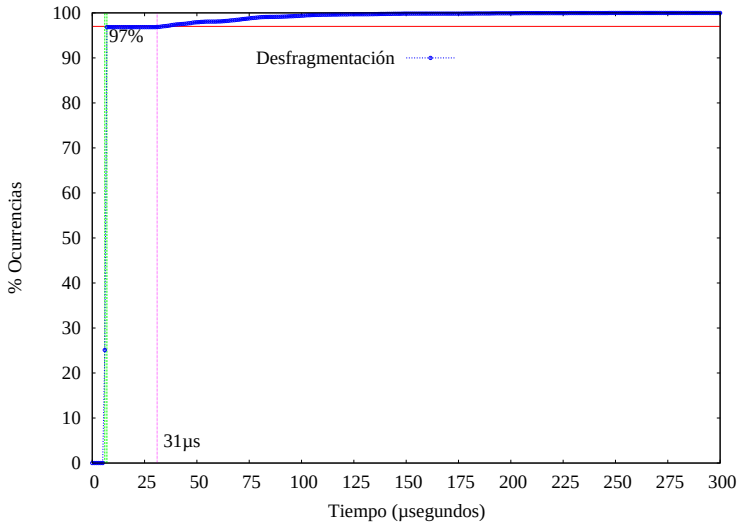


Figura 6.4: Porcentajes acumulados de ocurrencias de *Desfragmentación* tras una *Inserción* en el Modelo 1.

Un caso de *Operación de Inserción* sin intercambios consiste en una *Inserción* seguida de una *Desfragmentación*. En la Figura 6.1(d) estaría representada dentro del primer bloque de casos de la tarea *Desfragmentación*, tal y como se ha comprobado anteriormente. Si se suman los máximos tiempos de la tarea *Inserción*, 7 µs, y del primer grupo de casos de *Desfragmentación*, 20 µs, el valor nunca sería mayor o igual que el menor de los valores del segundo bloque de casos de *Operación de Inserción*, 51 µs.

	Tiempos (µs)
Tiempo máximo del primer bloque de *Desfragmentación*	7
Tiempo máximo de la tarea *Inserción*	20
Tiempo mínimo del segundo bloque de *Operación de Inserción*	51

Tabla 6.7: Tiempos utilizados para verificar el comportamiento de *Operación de Inserción* en el Modelo 1.

Por todo esto, es imposible que se haya representado algún caso en el que no se haya producido un intercambio dentro del segundo grupo. Además, como el primer grupo está formado por el mismo porcentaje de ocurrencias de esta operación en las que no se han

dado intercambios, también es imposible que haya alguna en la que se haya dado un intercambio.

Ahora toca el turno de comentar la Figura 6.1(d), donde se observa el comportamiento de *Operación de Eliminación* y de las tareas que la componen: *Eliminación* y *Desfragmentación*. En este caso también se diferencian dos partes, una que está dentro del rango [23 µs, 33 µs] y la otra comprende todas las ocurrencias que han invertido un tiempo de 48 µs en adelante. Los motivos de la existencia de estas dos partes se han supuesto igual que en *Operación de Inserción*, es decir, la primera está formada por ocurrencias de *Operación de Eliminación* en las que no se hayan dado ningún intercambio durante la tarea *Desfragmentación*, mientras que la segunda está formada por aquellas en las que sí se ha aplicado algún intercambio al llevar a cabo la tarea *Desfragmentación*.

Al igual que ocurrió en el caso de *Operación de Inserción*, los motivos para realizar esta suposición son los siguientes:

- Esta parte tiene un gran porcentaje de las operaciones de Eliminación que se aplicaron durante la simulación, al igual que es alto el porcentaje de ocurrencias de esta operación en las que no se ha realizado intercambio alguno.

- El comportamiento de la gráfica tiene al igual que la tarea *Desfragmentación*, dos partes diferenciadas.

La Tabla 6.6 (Página 94) muestra que sólo el 15% de las ocurrencias de la tarea *Desfragmentación* han llevado a cabo al menos un intercambio, por lo que el 85% restante serán las que pertenecen a la primera parte de la gráfica. La Figura 6.5 muestra como el 85% de las ocurrencias están dentro del rango [23 µs, 33 µs].

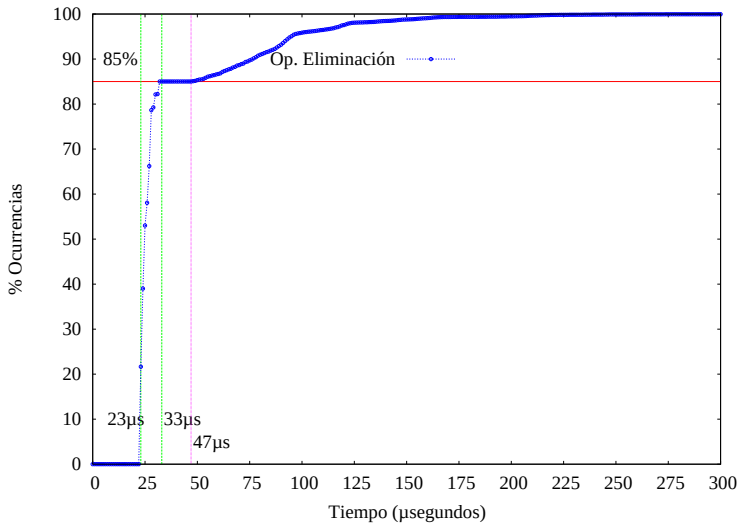


Figura 6.5: Porcentajes acumulados de *Operación de Eliminación* en el Modelo 1.

Aún así, es necesario conocer el comportamiento de la tarea *Desfragmentación*. La Figura 6.6 muestra la distribución de ocurrencias de esta tarea y de sus subtareas. En ella se observa que también está dividida en dos partes: De 6 a 7 microsegundos y a partir de 31 µs. Se vuelve a suponer que la división está provocada por la aplicación de intercambios. Las ocurrencias en las que no se realizan intercambios, se aglomeran en el primer grupo de casos mientras que en el segundo grupo están aquellas que si han aplicado algún intercambio entre dos conjuntos de la tabla de arbitraje.

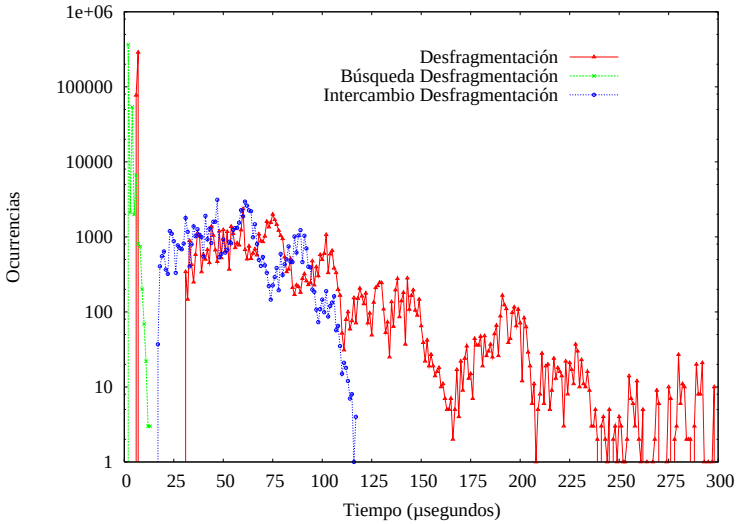


Figura 6.6: Distribución de los tiempos de ejecución de la tarea *Desfragmentación* aplicada tras una *Eliminación* y de las tareas que la componen. Modelo 1.

Observando la Figura 6.6, se ve que ninguna de las ocurrencias del primer grupo tarda más en aplicarse que el intercambio de conjuntos más rápido que se haya dado en cualquier ocurrencia de la tarea *Desfragmentación*. Esto hace imposible que una *Desfragmentación* en la que se haya dado algún intercambio esté representada en el primer grupo. Por tanto, queda claro que aquellos casos de *Desfragmentación* en los que no se den intercambios, están representados en el primer grupo.

En la Figura 6.7 además se observa como el porcentaje de casos en ese primer conjunto es igual al de casos en los que no se han dado intercambios, que es del 85%, lo cual aclara aún más lo que se ha supuesto.

Para cualquier caso de *Operación de Eliminación* en el que no se aplique un intercambio, su *Desfragmentación* está representada en el primer grupo. Si se sumara el tiempo máximo de la tarea *Eliminación*, 19 µs, con el de la tarea *Desfragmentación* en el que no se ha dado ningún intercambio, 7 µs, el resultado es menor al tiempo invertido por el intercambio más rápido efectuado durante cualquier *Operación de Eliminación*, 47 µs. Por tanto, es totalmente imposible que se de esta operación con intercambios dentro del

primer grupo, y además, como el porcentaje de ocurrencias de ese grupo es igual al de operaciones de eliminación que se han dado sin intercambio, queda aún más clara la diferenciación de los dos grupos existentes.

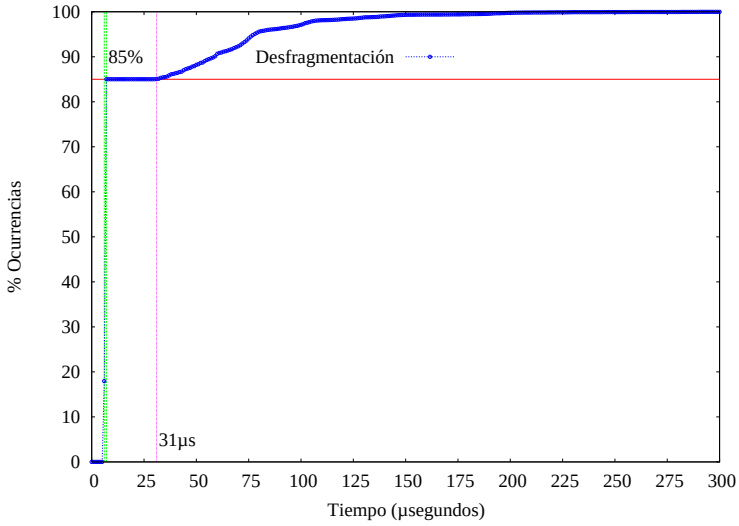


Figura 6.7: Porcentajes acumulados de ocurrencias de *Desfragmentación* tras una *Eliminación* en el Modelo 1.

	Tiempos (µs)
Tiempo máximo del primer bloque de *Desfragmentación*	7
Tiempo máximo de la tarea *Eliminación*	19
Tiempo mínimo del segundo bloque de la *Operación de Eliminación*	47

Tabla 6.8: Tiempos utilizados para confirmar el comportamiento de *Operación de Eliminación* en el Modelo 1.

6.4.2 Resultados obtenidos para los Modelos 2 y 3

Se han agrupado estos dos modelos ya que su comportamiento es prácticamente el mismo. La Figura 6.8 muestra la similitud entre los comportamientos de las operaciones de inserción y de las operaciones de eliminación en ambos modelos, así como también entre los tiempos invertidos. Esto está causado porque sólo se diferencian en el orden de aplicación de los métodos *desfragmentación* y *reordenación*.

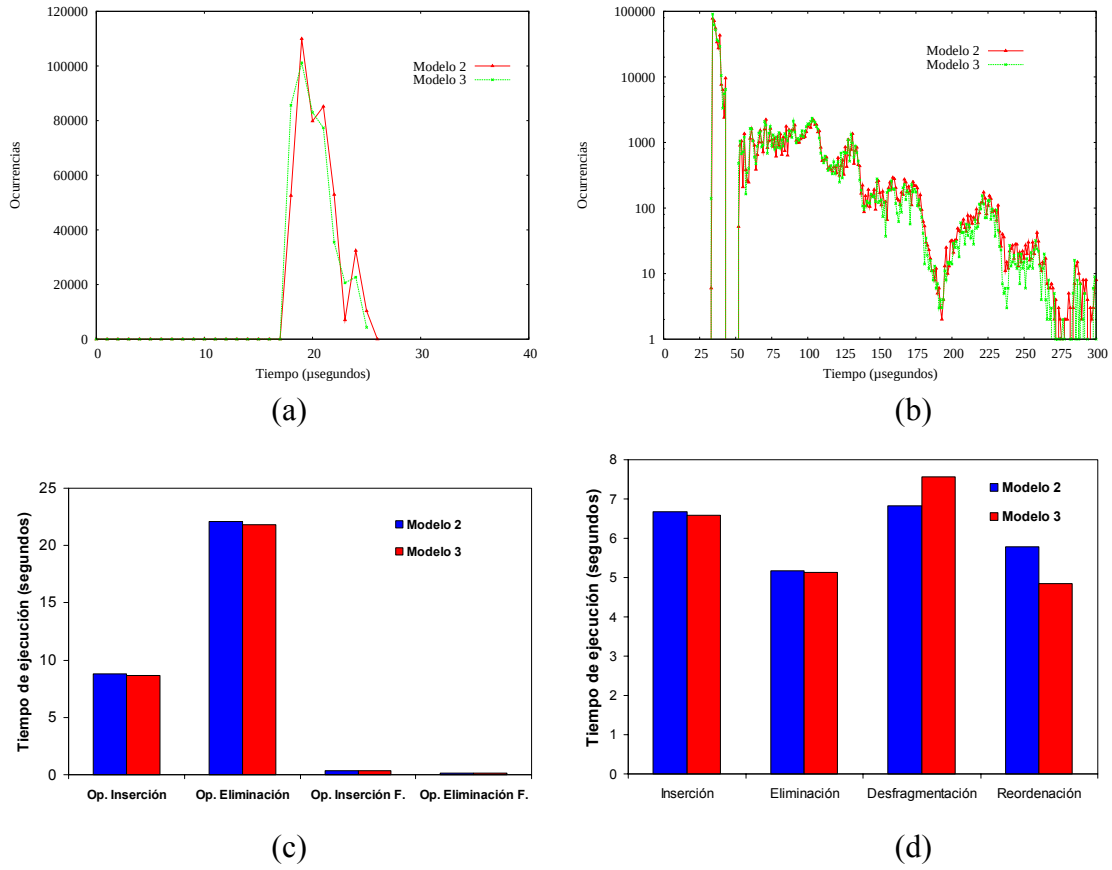


Figura 6.8: (a) Comparativa del comportamiento de O*peración de Inserción* en los modelos 2 y 3.
(b) Comparativa del comportamiento de *Operación de Eliminación* en los modelos 2 y 3.
(c) Comparativa de los tiempos invertidos en las distintas operaciones en los modelos 2 y 3.
(d) Comparativa de los tiempos invertidos en las distintas tareas en los modelos 2 y 3.

La única diferencia se puede encontrar en la Figura 6.8(d), donde se observa que tanto en *Desfragmentación* como en *Reordenación*, el tiempo invertido por los modelos es diferente. El Modelo 2 invierte más tiempo que el Modelo 3 en *Reordenación*, y ocurre todo lo contrario para *Desfragmentación*. La explicación de esta situación es tan simple como el hecho de que en el Modelo 2 se aplica antes *Reordenación* que *Desfragmentación*. Esto conlleva que la probabilidad de llevarse a cabo un intercambio en *Reordenación* durante el Modelo 2 sea mayor que en el Modelo 3. Lo mismo se podría decir para *Desfragmentación* para el Modelo 3 respecto al 2.

De todas maneras, lo observado en la Figura 6.8(c) no es motivo para negar que el comportamiento entre ambos modelos no es similar, puesto que las Figuras 6.8 (a), (b) y (c) sí muestran la semejanza entre el Modelo 2 y el Modelo 3.

Por tanto, se utilizará el Modelo 2 como referencia para explicar el comportamiento de ambos. Así, aunque únicamente se nombre al Modelo 2, todo lo que se exponga en este apartado se ha de trasladar al Modelo 3. Es preciso recordar que el Modelo 2 consiste en la aplicación de *Operación de Inserción*, compuesta únicamente por la tarea *Inserción*, y de *Operación de Eliminación*, que consiste en la utilización de la tarea *Eliminación*, seguida de *Reordenación* y *Desfragmentación*.

La Figura 6.9(a) muestra como se ha distribuido el tiempo total de ejecución entre las distintas operaciones que se han llevado a cabo en el Modelo 2. *Operación de Eliminación* es la operación que más tiempo ha invertido para llevarse a cabo. La causa se puede encontrar al observar la Figura 6.9(b), donde se expone el tiempo de ejecución de cada una de las principales tareas que se aplican en este modelo. El hecho de que dentro de *Operación de Eliminación* se apliquen las tareas *Eliminación*, *Desfragmentación* y *Reordenación*, implica que el tiempo que ha necesitado para su ejecución haya sido tan grande respecto al de las demás operaciones. En concreto, las dos últimas tareas mencionadas son las principales causantes, ya que pueden darse dentro de ellas intercambios entre conjuntos y esta tarea necesita bastante tiempo para llevarse a cabo, como muestra la Tabla 6.9, donde se indican los tiempos totales, medios y la desviación típica de cada operación y sus tareas asociadas.

En la Figura 6.9(c) se observa el comportamiento de *Operación de Inserción* y de la tarea que contiene: *Inserción*. Al contener únicamente la tarea *Inserción*, la demostración de que su comportamiento está supeditado al de la tarea *Inserción*, es trivial sólo con ver la figura.

Por otro lado, la Figura 6.9(d) muestra el comportamiento *Operación de Eliminación*. En este caso también se observan dos bloques de ocurrencias bien diferenciadas. El primero está formado por todas las ocurrencias que invirtieron un tiempo perteneciente al rango [33 μs, 43 μs], y el segundo está formado por las ocurrencias que necesitaron un tiempo de 52 μs en adelante. Por tanto, se va a suponer, al igual que se hizo en el Modelo 1, que el primer bloque está compuesto por las ocurrencias en las que no se ha llevado a cabo ningún intercambio entre dos conjuntos de la tabla de arbitraje, mientras que en las del segundo bloque sí se ha realizado algún intercambio.

Operación/Tarea	Tpo. total	Tpo. medio	Desviación típica
Operación de Inserción	**8782460 µs**	**20,415 µs**	**1,793**
Inserción	6678711 µs	15,525 µs	1,769
Búsqueda Inserción	986310 µs	2,293 µs	0,455
Act. Singulares Inserción	1370408 µs	3,186 µs	1,176
Operación de Eliminación	**22065482 µs**	**51,290 µs**	**33,853**
Eliminación	5172454 µs	12,023 µs	2,214
Act. singulares Eliminación	1279360 µs	2,974 µs	0,511
Desfragmentación	6815659 µs	15,843 µs	27,417
Búsqueda Desfragmentación	1037383 µs	2,411 µs	0,992
Intercambio Desfragmentación	3356594 µs	55,223 µs	20,600
Reordenación	5790483 µs	13,460 µs	20,150
Búsqueda Reordenación	1199973 µs	2,789 µs	0,916
Intercambio Reordenación	2373373 µs	48,467 µs	23,980
Operación Inserción Fallida	**342901 µs**	**4,972 µs**	**0,165**
Operación Eliminación Fallida	**167693 µs**	**2,375 µs**	**0,484**

Tabla 6.9: Resultados de tiempos obtenidos para el Modelo 2.

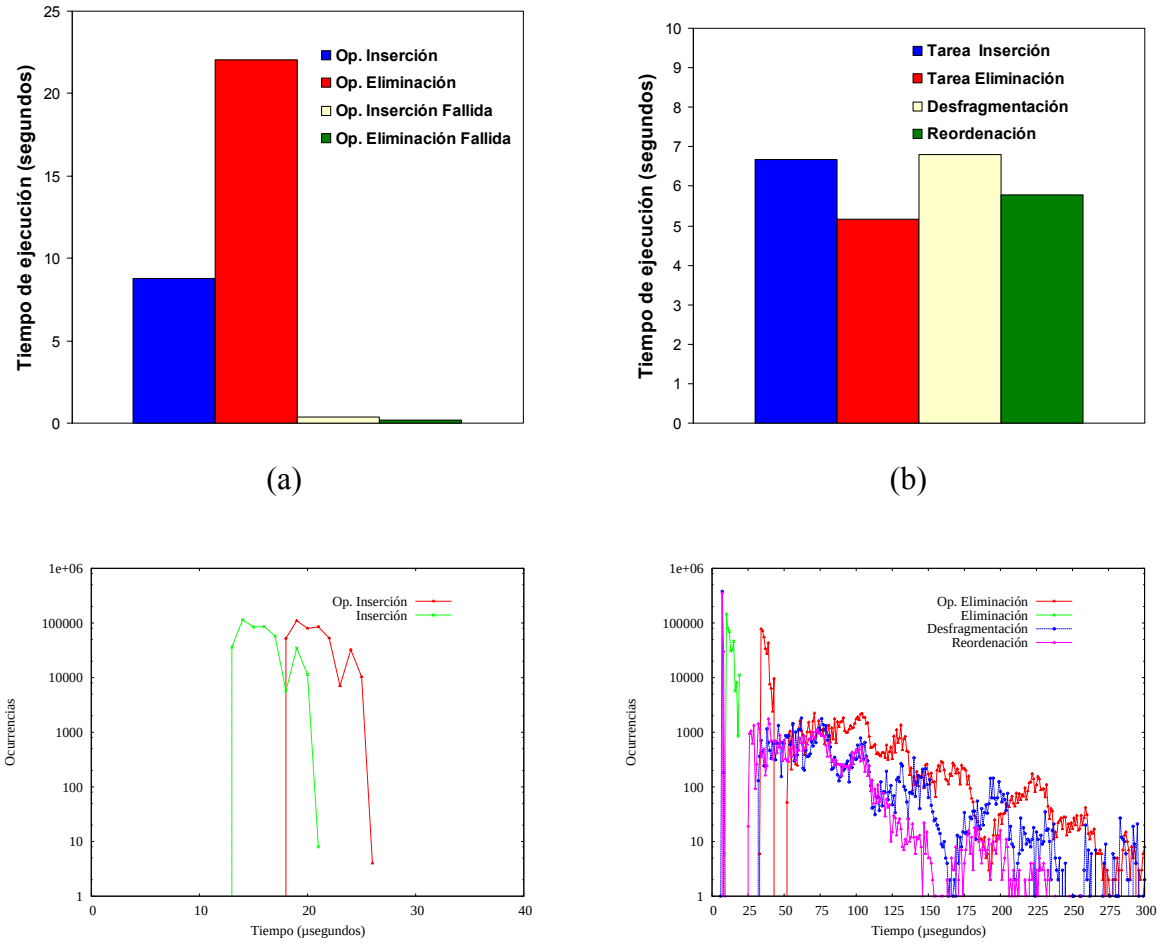


Figura 6.9: Tiempos de operaciones y tareas y su distribución para el Modelo 2.

Para confirmar la composición de cada uno de los dos bloques que se acaban de proponer, se van a utilizar las mismas pautas que se utilizaron en el Modelo 1, añadiendo otra nueva.

- El primer bloque tiene un alto porcentaje de las operaciones de eliminación que se aplicaron durante la simulación y el porcentaje de estas operaciones sin intercambio también es muy alto.

- El comportamiento de *Operación de Eliminación* es parecido al de la tarea *Desfragmentación*, puesto que existen dos partes diferenciadas.

- El comportamiento de la tarea *Reordenación* es similar al de *Operación de Eliminación* porque también tienen dos bloques diferenciados.

Como muestra la Tabla 6.10, durante la simulación efectuada, el 78,83% de las ocurrencias de *Operación de Eliminación* no han efectuado ningún intercambio entre dos conjuntos de la tabla de arbitraje. Por tanto, si las ocurrencias pertenecientes al primer bloque de *Operación de Eliminación* representaran al 78,83% del total, sería una evidencia muy importante para afirmar el porqué de los dos bloques que se ha supuesto.

	Número de casos	Porcentaje
Operaciones de eliminación sin intercambio	334875	78,83%
Desfragmentaciones sin intercambio	377737	87,80%
Reordenaciones sin intercambio	382131	88,82%

Tabla 6.10: Número de operaciones de eliminación, tareas *Desfragmentación* y *Reordenación* sin intercambio en el Modelo 2.

En la Figura 6.10 se observa como en el intervalo de tiempo que va de los 33 µs a los 43 µs se aglomera un porcentaje de ocurrencias igual al de operaciones de eliminación en las que no se han llevado a cabo ningún intercambio durante la simulación. A partir de los 52 µs en adelante se encuentran el resto de ocurrencias, cuyo porcentaje es igual al de operaciones de eliminación con intercambio.

Tras averiguar que el número de ocurrencias de la primera división de *Operación de Eliminación* es igual al de ocurrencias de la misma operación sin intercambios, ahora se va a estudiar la relación entre el comportamiento de la tarea *Desfragmentación* y la tarea *Reordenación* con el de *Operación de Eliminación*. La relación que existe es que tanto la operación como las dos tareas se dividen en dos partes como se puede observar en la Figura 6.9(d), e igual que se ha supuesto que el primer bloque de ocurrencias de *Operación*

de Eliminación está formado por aquellas en las que no ha habido intercambios, se vuelve a suponer lo mismo con ambas tareas. Si se comprueba esta suposición, se podrá dar un paso más para confirmar la hipótesis planteada respecto al comportamiento de *Operación de Eliminación*.

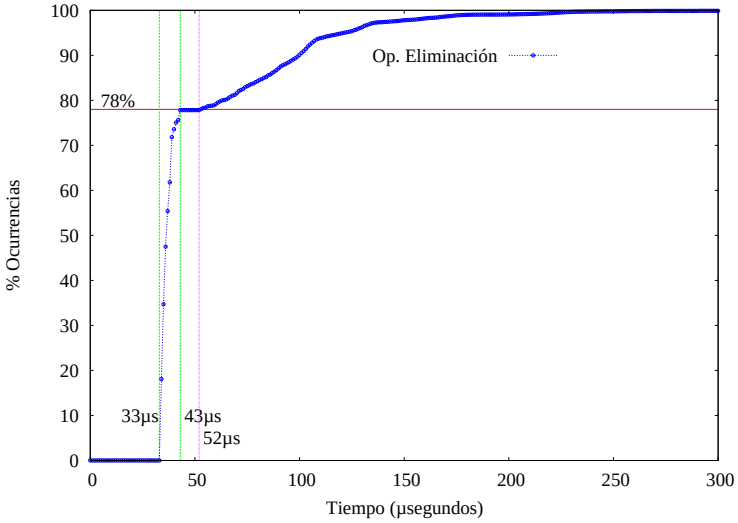


Figura 6.10: Porcentajes acumulados de las ocurrencias de *Operación de Eliminación* en el Modelo 2.

La Figura 6.11 muestra el comportamiento de la tarea *Desfragmentación* y de las dos tareas que están incluidas dentro de ella. Los dos bloques de ocurrencias que se han mencionado están formados por las que han invertido de entre 6 µs y 8 µs uno, y de 32 µs en adelante el otro.

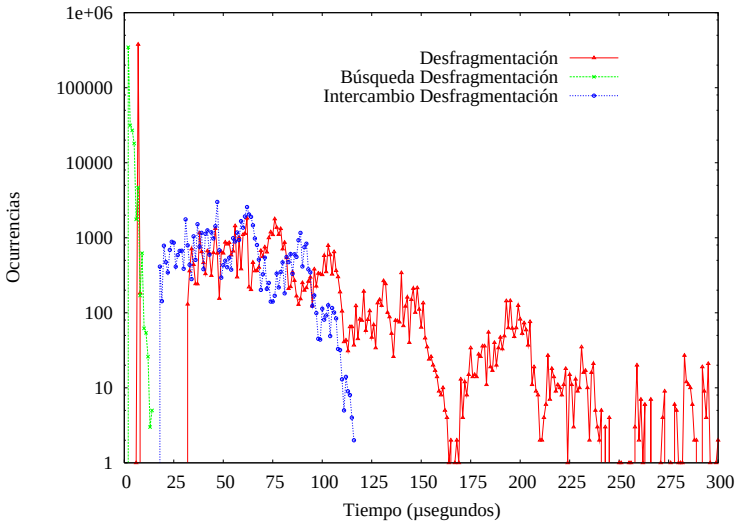


Figura 6.11: Distribución de los tiempos de ejecución de la tarea *Desfragmentación* y de las subtareas que la componen en el Modelo 2.

Ninguna ocurrencia del primer bloque ha invertido más tiempo que ninguno de los intercambios realizados dentro de la tarea, lo cual ya confirma de por sí la imposibilidad de que se haya producido intercambio alguno en esas desfragmentaciones. Por tanto, es seguro que ninguna *Desfragmentación* que haya aplicado al menos un intercambio entre dos conjuntos de la tabla de arbitraje esté representada en el primer bloque.

Aún así, para corroborar todo lo expuesto en el párrafo anterior, se va a mostrar que el primer bloque representa al mismo número de desfragmentaciones sin intercambio que ha habido. La Tabla 6.8 indica que el 87,80% de las tareas *Desfragmentación* aplicadas sobre la tabla de arbitraje, no han intercambiado dos conjuntos entre sí. La Figura 6.12 muestra los porcentajes acumulados de las desfragmentaciones realizadas, y muestra como en el intervalo de 6 a 8 microsegundos, se concentran aproximadamente el 88% de las ocurrencias de la tarea *Desfragmentación*. Así pues, queda descartada la posibilidad de que alguna *Desfragmentación* en la que no se haya aplicado ningún intercambio esté en el segundo bloque.

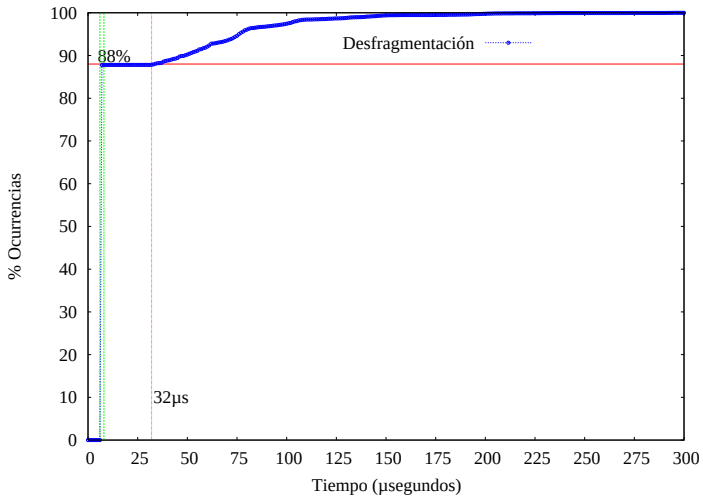


Figura 6.12: Porcentajes acumulados de la tarea *Desfragmentación* en el Modelo 2.

Tras estudiar el comportamiento de la tarea *Desfragmentación*, ahora corresponde el estudio de *Reordenación*. La Figura 6.13 muestra la distribución de las ocurrencias de esta tarea y de las subtareas que la componen, según el tiempo que han invertido. Al igual que *Operación de Eliminación* está dividido en dos bloques: El primero abarca desde los 7 a los 9 microsegundos y el segundo va de los 25 µs en adelante.

Se da el mismo caso que en la tarea *Desfragmentación*, ninguna ocurrencia del primer bloque tiene un tiempo mayor o igual al menor tiempo invertido por un intercambio. Este hecho constata que el mencionado primer bloque está formado solamente por ocurrencias

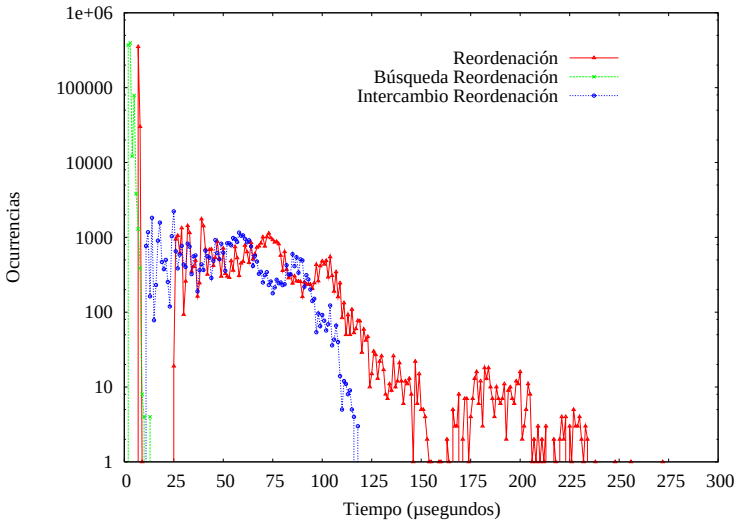


Figura 6.13: Distribución de las ocurrencias de la tarea *Reordenación* y de las subtareas que la componen en el Modelo 2.

de la tarea estudiada en las que no se ha aplicado ningún intercambio entre dos conjuntos de la tabla de arbitraje.

De todas maneras, se va a comprobar en este caso también que el porcentaje de casos representados en el primer bloque es igual al de casos de *Reordenación* sin haber aplicado intercambio alguno. La Tabla 6.10 muestra que dicho porcentaje es igual al 88,82% de casos. En la Figura 6.14 se observa que dicho 88,82% de ocurrencias está representado en el intervalo que va desde los 7 a los 9 microsegundos.

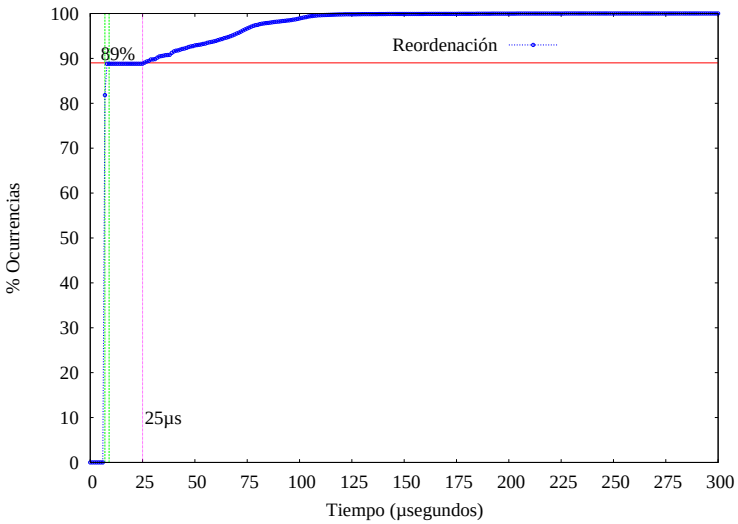


Figura 6.14: Porcentajes acumulados de la tarea *Reordenación* en el Modelo 2.

Hasta ahora se puede hablar con total seguridad de que el primer bloque de operaciones de eliminación está formado por el mismo porcentaje de casos en los que no se ha dado ningún intercambio y que además, la primera parte de ocurrencias de las tareas *Desfragmentación* y *Reordenación* están formadas por aquellos casos en los que tampoco se han producido intercambios.

Un caso de *Operación de Eliminación* en el que no se aplique ningún intercambio, consiste en: la tarea *Eliminación*, después se aplica *Reordenación* seguida de *Desfragmentación*. Como no se ha dado ningún intercambio, con todo lo que se ha averiguado hasta ahora, esta ocurrencia estaría representada en la Figura 6.9(d) dentro de los primeros bloques de casos de las tareas *Reordenación* y *Desfragmentación*. Si se suma el máximo tiempo de cada uno de estos bloques, con el máximo tiempo empleado en una tarea *Eliminación*, el valor obtenido nunca sería mayor o igual que el menor valor del segundo bloque de casos de *Operación de Eliminación*.

La Tabla 6.11 contiene los tiempos que se han indicado en el párrafo anterior. Al sumar los tiempos mencionados, el valor obtenido es de 36 µs. Éste es inferior a 52 µs, que es el menor tiempo invertido en llevar a cabo una *Operación de Eliminación* perteneciente al segundo bloque de ocurrencias de la Figura 6.9(d). Con esto, se asegura que ninguna ocurrencia en la que no se haya dado un intercambio puede pertenecer al segundo bloque. Además, como el primer bloque está formado por la misma proporción de casos en las que no se ha dado ningún intercambio, queda claro que no puede haber dentro de ese bloque ninguna ocurrencia de *Operación de Eliminación* en la que se haya dado algún intercambio.

	Tiempos (µs)
Tiempo máximo del primer bloque de *Desfragmentación*	8
Tiempo máximo del primer bloque de *Reordenación*	9
Tiempo máximo de la tarea *Eliminación*	19
Tiempo mínimo del segundo bloque de la *Operación de Eliminación*	52

Tabla 6.11: Tiempos utilizados para comprobar el comportamiento de *Operación de Eliminación* en el Modelo 2.

Así pues, se puede afirmar que el primer bloque está formado por ocurrencias en las que no se han dado intercambios y el segundo por las que sí han aplicado alguno sobre la tabla de arbitraje.

6.4.3 Resultados obtenidos para el Modelo 4

Operación/Tarea	Tpo. total	Tpo. medio	Desviación típica
Operación de Inserción	**8450203 µs**	**19,642 µs**	**1,744**
Inserción	6411594 µs	14,904 µs	1,660
Búsqueda Inserción	879855 µs	2,045 µs	0,208
Act. Singulares Inserción	1306613 µs	3,037 µs	1,177
Operación de Eliminación	**15837771 µs**	**36,814 µs**	**30,467**
Eliminación	5036812 µs	11,708 µs	2,111
Act. singulares Eliminación	1217741 µs	2,831 µs	0,582
Desfragmentación	7816900 µs	18,170 µs	30,834
Búsqueda Desfragmentación	1028461 µs	2,391 µs	0,972
Intercambio Desfragmentación	4302223 µs	57,110 µs	20,802
Operación Inserción Fallida	**323770 µs**	**4,694 µs**	**0,460**
Operación Eliminación Fallida	**151591 µs**	**2,147 µs**	**0,354**

Tabla 6.12: Tiempos de operaciones y tareas y su distribución para el Modelo 4.

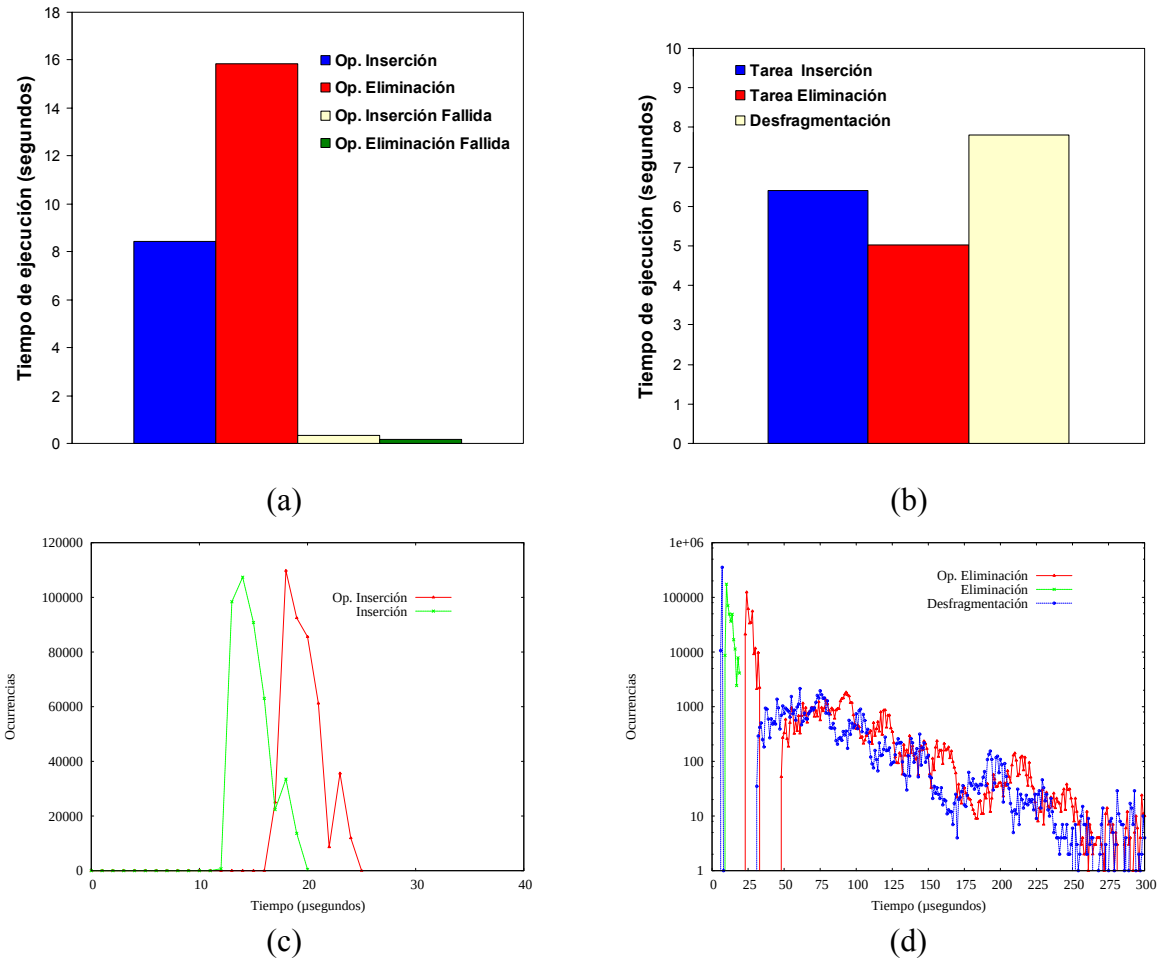


Figura 6.15: Tiempos de operaciones y tareas y su distribución para el Modelo 4.

La Figura 6.15(a) y la Tabla 6.12 muestran que la mayor parte del tiempo de este modelo se ha empleado en realizar *Operación de Eliminación*, puesto que esta operación está compuesta de las tareas *Eliminación* y *Desfragmentación*, y *Operación de Inserción* solamente está formada por la tarea de *Inserción*.

En la Figura 6.15(c) se observa que el comportamiento de *Operación de Inserción* está totalmente ligado al de la tarea *Inserción*, puesto que como se ha dicho antes, es la única tarea que se aplica dentro de la operación.

La Figura 6.15(d) muestra el comportamiento de *Operación de Eliminación*. La distribución de las ocurrencias de esta operación es similar a la que se daba en el Modelo 1, puesto que en ambos casos la operación está formada por las mismas tareas (Figura 6.1(d)). Sigue habiendo una división en dos conjuntos de ocurrencias. El primer conjunto va desde 23 µs hasta 33 µs, mientras que el segundo va de 48 µs en adelante. Se vuelve a suponer que el primer conjunto está formado por las ocurrencias en las que no se ha aplicado ningún intercambio sobre la tabla de arbitraje, y los puntos en los que se va a basar la comprobación de este supuesto son:

- Este bloque tiene un alto porcentaje de las operaciones de eliminación que se aplicaron durante la simulación, al igual que el porcentaje de operaciones en las que no se han dado intercambios.

- El comportamiento de *Operación de Eliminación* es parecido al de la tarea de *Desfragmentación*, puesto que existen dos partes diferenciadas.

La Tabla 6.13 muestra que el 15,07% de las operaciones de eliminación ha llevado a cabo algún intercambio. Observando la Figura 6.16, el primer conjunto de ocurrencias representa aproximadamente al 85% de las ocurrencias, o lo que es lo mismo, tiene un porcentaje igual al de operaciones de eliminación sin intercambio.

	Num. Ocurrencias	% Ocurrencias
Operaciones de eliminación con intercambio	64859	15,07%

Tabla 6.13: Operaciones de eliminación con intercambio en el Modelo 4.

En la Figura 6.17 se muestra el comportamiento de la tarea *Desfragmentación*, y vuelve a darse el caso de que hay dos grupos, el primero va de los 6 a los 7 microsegundos y el segundo de los 31 microsegundos en adelante. La hipótesis sigue siendo que el primer grupo está formado por ocurrencias de la tarea *Desfragmentación* que no han realizado

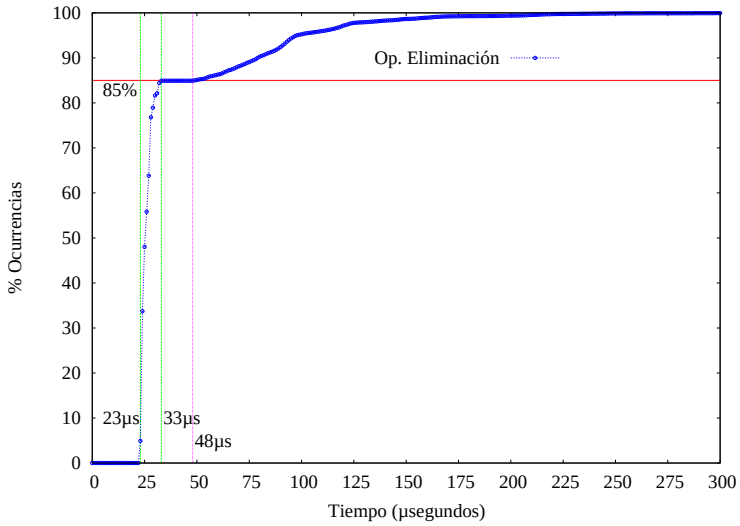


Figura 6.16: Porcentajes acumulados de *Operación de Eliminación* en el Modelo 4.

intercambio. En esa Figura se ve que ninguna ocurrencia del primer conjunto ha invertido un tiempo superior a 17 µs, que es el tiempo mínimo empleado por un intercambio durante la simulación, lo cual demuestra que en ninguna de ellas se ha realizado intercambio alguno.

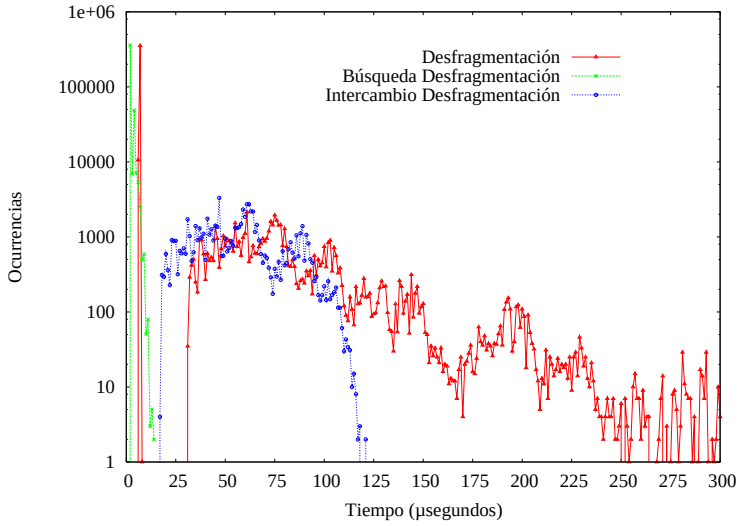


Figura 6.17: Distribución de las ocurrencias de la tarea *Desfragmentación* y de las subtareas que la componen en el Modelo 4.

Además, el porcentaje de ocurrencias representadas en el primer bloque, según la Figura 6.18 es igual al de ocurrencias de operaciones de eliminación sin intercambio. Una vez más, se puede afirmar que las desfragmentaciones sin intercambio tienen una duración de entre 6 y 7 microsegundos,

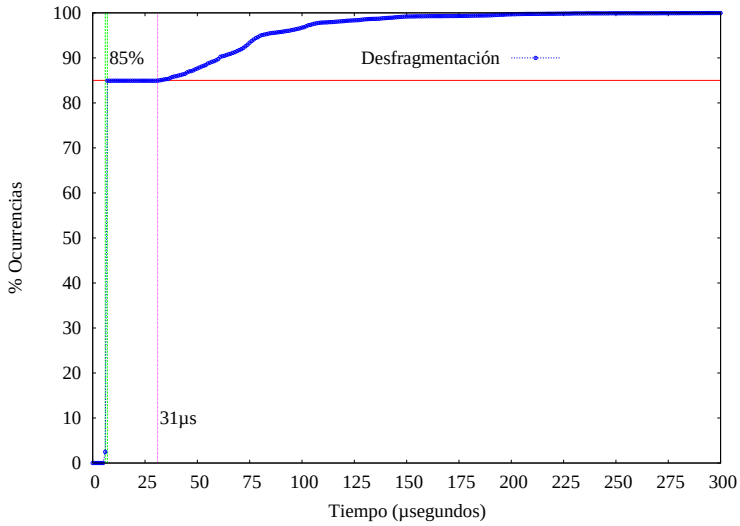


Figura 6.18: Porcentajes acumulados de las ocurrencias de la tarea *Desfragmentación* en el Modelo 4.

Ahora se va a comprobar que todas las ocurrencias de *Operación de Eliminación* que han invertido entre 23 y 33 microsegundos son aquellas en las que no se han dado ningún intercambio. En cualquiera de ellas, se ha aplicado una tarea *Eliminación* y otra *Desfragmentación* sin intercambio, que como se ha afirmado en el párrafo anterior, tiene una duración máxima de 7 microsegundos. Si se suman los tiempos máximos de cada tarea, el valor obtenido no será mayor o igual que el menor tiempo invertido por cualquier ocurrencia perteneciente al segundo bloque de operaciones de eliminación. La Tabla 6.14 contiene los valores necesarios para realizar esta suma.

	Tiempos (µs)
Tiempo máximo del primer bloque de *Desfragmentación*	7
Tiempo máximo de la tarea *Eliminación*	19
Tiempo mínimo del segundo bloque de *Operación de Eliminación*	48

Tabla 6.14: Tiempos utilizados para comprobar el comportamiento de *Operación de Eliminación* en el Modelo 4.

El resultado de la suma es igual a 26 µs, valor muy inferior al mínimo del segundo bloque, por lo que es imposible que haya alguna ocurrencia de *Operación de Eliminación* sin intercambios representada en el segundo bloque. Además, como en el primer bloque se representan además el mismo porcentaje de ocurrencias sin intercambios, es imposible que represente algún caso de *Operación de Eliminación* con intercambio.

Así pues, ya se puede decir con certeza que el comportamiento de *Operación de Eliminación* depende de si ha habido o no algún intercambio. Si no lo ha habido, el tiempo de duración está relacionado con el tiempo que se ha empleado en la tarea *Eliminación*, mientras que si lo ha habido, el tiempo dedicado a ello en la tarea *Desfragmentación* es el que influye sobre *Operación de Eliminación*.

6.4.4 Comparativa de modelos

Tras estudiar el comportamiento de los 4 modelos propuestos, se van a comparar entre sí para ver qué propuesta es la que mejores resultados ofrece. En esta comparativa se va a volver a tomar al Modelo 2 como representante de los modelos 2 y 3.

Operación/Tarea	**Modelo 1**	**Modelo 2**	**Modelo 4**
Operación de Inserción	**13063258 µs**	**8782460 µs**	**8450203 µs**
Inserción	6372569 µs	6678711 µs	6411594 µs
Búsqueda Inserción	883823 µs	986310 µs	879855 µs
Act. Singulares Inserción	1305578 µs	1370408 µs	1306613 µs
Desfragmentación	3849510 µs		
Búsqueda Desfragmentación	896048 µs		
Intercambio Desfragmentación	840208 µs		
Operación de Eliminación	**15428036 µs**	**22065482 µs**	**15837771 µs**
Eliminación	4970904 µs	5172454 µs	5036812 µs
Act. singulares Eliminación	1173646 µs	1279360 µs	1217741 µs
Desfragmentación	7504323 µs	6815659 µs	7816900 µs
Búsqueda Desfragmentación	1014093 µs	1037383 µs	1028461 µs
Intercambio Desfragmentación	4089010 µs	3356594 µs	4302223 µs
Reordenación		5790483 µs	
Búsqueda Reordenación		1199973 µs	
Intercambio Reordenación		2373373 µs	
Tiempo total del modelo	**28491294** µs	**30847942** µs	**24287974** µs

Tabla 6.15: Tiempos totales de las operaciones y tareas para los tres modelos.

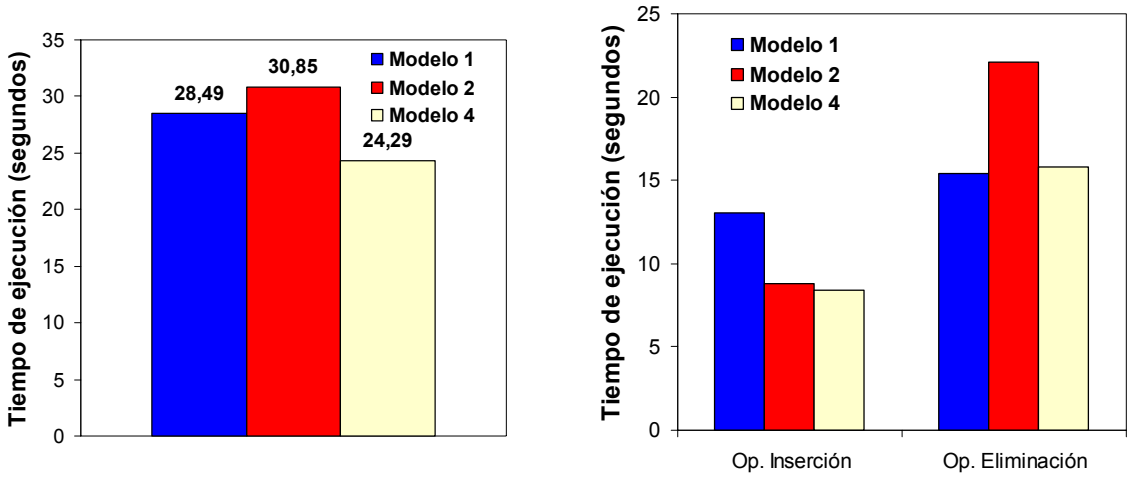
La Tabla 6.15 muestra los tiempos totales de cada operación y sus tareas asociadas en los tres modelos que se han evaluado. Se puede observar que el Modelo 4 ha sido el que menos tiempo ha necesitado para llevar a cabo el mismo caso. En concreto, la Figura 6.19 muestra la mejora que se obtiene con el Modelo 4 respecto a los otros dos y cómo se ha calculado dicha mejora.

$$\text{Mejora del Modelo 4 respecto al Modelo 1} = \left(1 - \frac{24287974\,\mu s}{28491294\,\mu s}\right) \times 100 = 14{,}75\%$$

$$\text{Mejora del Modelo 4 respecto al Modelo 2} = \left(1 - \frac{24287974\,\mu s}{30847942\,\mu s}\right) \times 100 = 21{,}26\%$$

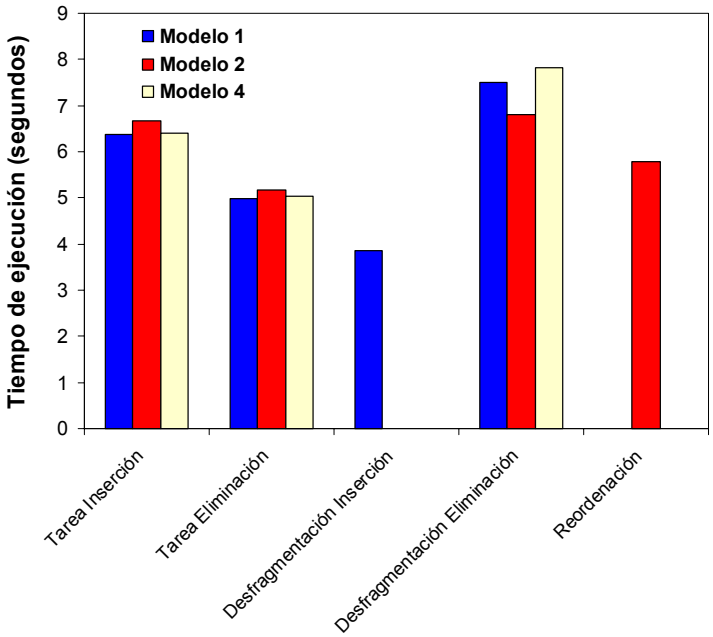
Figura 6.19: Mejora obtenida con el Modelo 4 respecto al resto de modelos.

La Figura 6.20(a) muestra gráficamente qué tiempo empleado por el Modelo 4 es el menor de todos. Para buscar las causas de por qué se ha dado esta situación, es necesario consultar qué tiempos han empleado las operaciones y las tareas. La Figura 6.20(b) muestra los tiempos empleados en cada una de las operaciones en cada modelo y la Figura 6.20(c) los invertidos para llevar a cabo las tareas.



(a)

(b)



(c)

Figura 6.20: (a) Tiempos empleados por cada modelo para ejecutar la misma simulación.
(b) Tiempos empleados por las operaciones para cada modelo.
(c) Tiempos empleados por las tareas para cada modelo.

A continuación, se van a mostrar las diferencias entre el mejor modelo, el Modelo 4, respecto a los otros dos para encontrar el motivo de esa mejora.

La Figura 6.20(b) muestra cómo el Modelo 4 ha invertido respecto al Modelo 1 mucho menos tiempo en *Operación de Inserción*, mientras que en *Operación de Eliminación* ha invertido un poco más que el Modelo 1. Así pues, la causa de por qué obtiene mejor resultado el Modelo 4, se encuentra en la aplicación de *Operación de Inserción* en el Modelo 1.

Cuando se explicó el Modelo 1, se comentó que durante *Operación de Inserción* se iba a aplicar sobre la tabla de arbitraje la tarea *Desfragmentación*. La Figura 6.20(c) muestra cómo el tiempo invertido en la tarea *Inserción* es prácticamente el mismo en ambos modelos, por lo que el tiempo que se ha necesitado para aplicar *Desfragmentación* ha sido el causante de que este modelo haya obtenido peores resultados.

Ahora es el turno de comentar las diferencias entre el Modelo 2 y el 4 que hacen al segundo más rápido. La Figura 6.20(b) muestra cómo el Modelo 4 es más rápido que el 2 tanto al aplicar *Operación de Inserción* como *Operación de Eliminación*.

Como se comentó anteriormente, el Modelo 2 aplica sobre la tabla de arbitraje la tarea *Reordenación* mientras que el Modelo 4 no la lleva a cabo. Para obtener más información sobre si la tarea *Reordenación* es la causante de esta diferencia, se observa la Figura 6.20(c). En ella se ve que el tiempo invertido por el Modelo 2 para aplicar dicha tarea es bastante grande, mucho más que la diferencia existente a su favor al aplicar la tarea *Desfragmentación*, dejando claro que dicha tarea es la causante del peor funcionamiento.

Básicamente éstas han sido las causas que han hecho posible que el Modelo 4 obtenga mejores resultados. Pero durante el estudio que se realizó, también se encontraron otros datos que sería bastante importante citarlos para después poder sacar conclusiones.

La Tabla 6.16 muestra los tiempos medios de las operaciones y de las tareas en cada uno de los modelos. Se puede observar que los menores tiempos medios obtenidos para las tareas más costosas, a excepción de *Desfragmentación Eliminación*, son los que se han dado para el Modelo 1. Es más, uniendo las desfragmentaciones realizadas durante *Operación de Inserción* y *Operación de Eliminación* para el Modelo 1, y comparando las medias obtenidas en los tres modelos, también es menor la media resultante en el primer modelo (Tabla 6.17).

Para explicar por qué el Modelo 1 no ha sido el más rápido, teniendo mejores medias en las tareas más costosas, hay que ver el número de intercambios que se han llevado a cabo a lo largo de cada modelo.

Operación/Tarea	**Modelo 1**	**Modelo 2**	**Modelo 4**
Operación de Inserción	**30,365 µs**	**20,415 µs**	**19,642 µs**
Inserción	14,813 µs	15,525 µs	14,904 µs
Búsqueda Inserción	2,054 µs	2,293 µs	2,045 µs
Act. Singulares Inserción	3,035 µs	3,186 µs	3,037 µs
Desfragmentación	8,948 µs		
Búsqueda Desfragmentación	2,083 µs		
Intercambio Desfragmentación	51,629 µs		
Operación de Eliminación	**35,862 µs**	**51,290 µs**	**36,814 µs**
Eliminación	11,555 µs	12,023 µs	11,708 µs
Act. singulares Eliminación	2,728 µs	2,974 µs	2,831 µs
Desfragmentación	17,443 µs	15,843 µs	18,170 µs
Búsqueda Desfragmentación	2,357 µs	2,411 µs	2,391 µs
Intercambio Desfragmentación	55,060 µs	55,223 µs	57,110 µs
Reordenación		13,460 µs	
Búsqueda Reordenación		2,789 µs	
Intercambio Reordenación		48,467 µs	

Tabla 6.16: Tiempos medios de las operaciones y tareas en cada uno de los modelos.

Operación/Tarea	**Modelo 1**	**Modelo 2**	**Modelo 4**
Desfragmentación	13,196 µs	15,843 µs	18,170 µs
Búsqueda Desfragmentación	2,220 µs	2,411 µs	2,391 µs
Intercambio Desfragmentación	54,444 µs	55,223 µs	57,110 µs

Tabla 6.17: Tiempos medios generales de la tarea *Desfragmentación* y las subtareas que la componen.

Viendo la Tabla 6.18, queda bien claro por qué el Modelo 1 ha tardado más tiempo que el Modelo 4. Ha llevado a cabo mayor número de intercambios, y aunque el tiempo medio que tienen sus intercambios respecto a los del Modelo 4 es menor, su coste temporal sigue siendo muy alto, y hace que invierta más tiempo en intercambios.

	Intercambios *Desfragmentación*	Intercambios *Reordenación*	Total
Modelo 1	16273 (*Inserción*)/74263 (*Eliminación*)	0	90536
Modelo 2	60781	48968	109749
Modelo 4	75331	0	75331

Tabla 6.18: Número de intercambios llevados a cabo en cada modelo.

Por otro lado, lo menos esperado ha sido el menor número de intercambios en el Modelo 1 respecto al Modelo 2, ya que el propósito por el que se utiliza la tarea

Reordenación es el de evitar tener que llevar a cabo un mayor número de intercambios para facilitar el funcionamiento a la tarea *Inserción*.

Tras todo lo expuesto, puede pensarse que los resultados obtenidos sean pura casualidad, puesto que sólo se presentan datos de una simulación consistente en un millón de operaciones. Por ello, se han simulado 9 casos "distintos" siguiendo el mismo procedimiento, pero sin realizar un estudio de su comportamiento. Simplemente se ha comparado:

- Tiempo total empleado para cada modelo.

- Relación entre el tiempo total empleado para los modelos 1 y 2 respecto el 4.

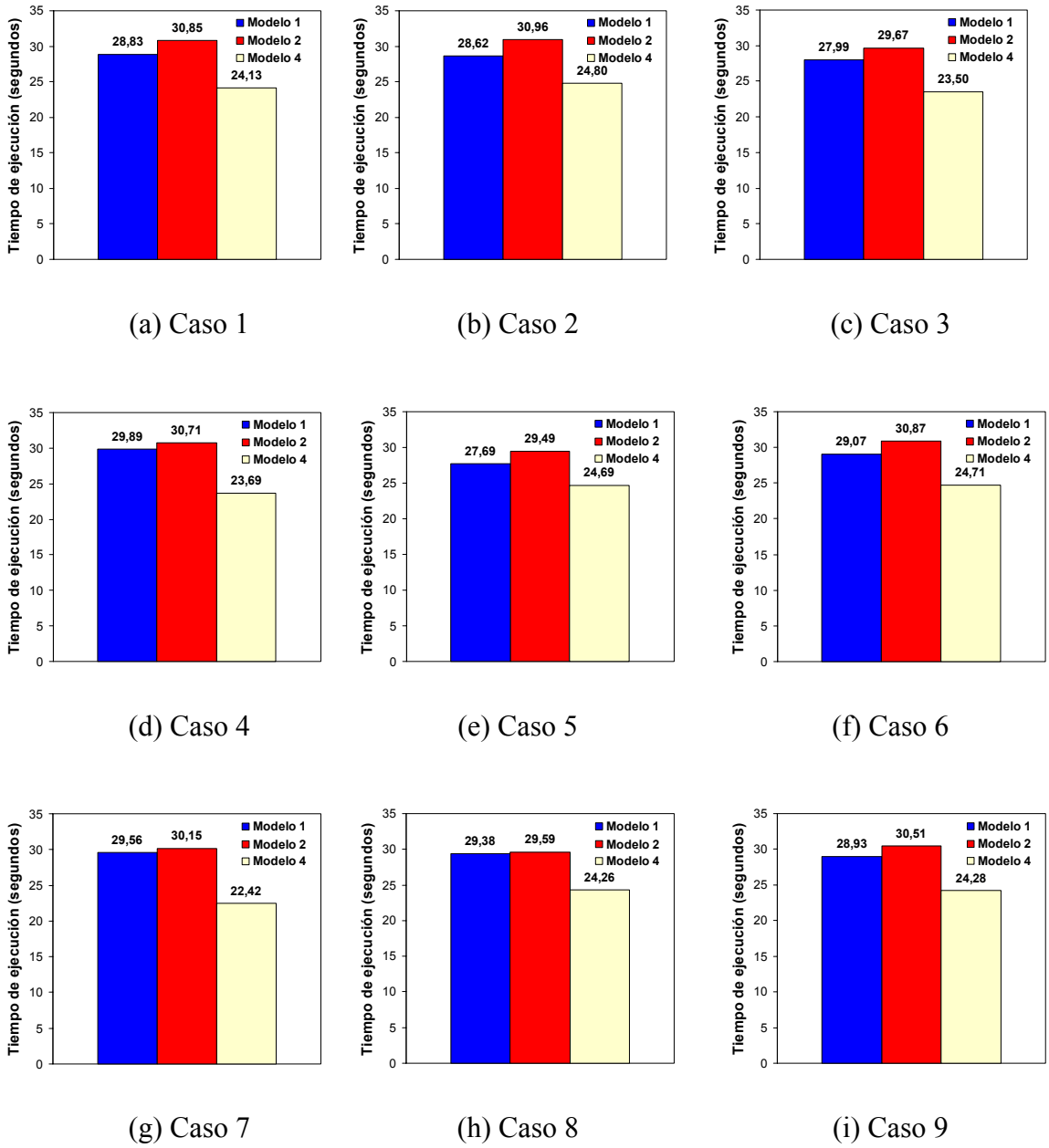
La Figura 6.21 se compone de gráficas que muestran los tiempos totales para cada modelo en cada uno de los 9 casos que se han simulado.

Se puede observar que en todos los casos simulados, el Modelo 4 sigue siendo el más rápido, seguido del 1 y por último, el Modelo 2. Incluso, las diferencias entre los modelos son similares en la mayoría de casos. Aún así, la Tabla 6.19 muestra la mejora obtenida por el Modelo 4 respecto al Modelo 1 (En la Tabla 6.19 denominada *Mejora 1*) y al Modelo 2 (En la Tabla 6.19 denominada *Mejora 2*).

Casos	Mejora 1	Mejora 2
1	16,31%	21,78%
2	13,34%	19,87%
3	16,02%	20,78%
4	20,75%	22,86%
5	10,85%	16,29%
6	14,97%	19,94%
7	24,17%	25,65%
8	17,44%	18,01%
9	16,09%	20,42%

Tabla 6.19: Mejoras obtenidas por el Modelo 4 respecto a los modelos 1 y 2 en los casos llevados a cabo.

En la Tabla 6.19 se observa que prácticamente todas las mejoras del Modelo 4 respecto a los modelos 1 y 2 tienen un porcentaje parecido a los obtenidos en las pruebas realizadas para evaluar los métodos implementados para gestionar la tabla de arbitraje. En



(a) Caso 1 (b) Caso 2 (c) Caso 3

(d) Caso 4 (e) Caso 5 (f) Caso 6

(g) Caso 7 (h) Caso 8 (i) Caso 9

Figura 6.21: Tiempos empleados para los 3 modelos en cada uno de los 9 casos extraordinarios que se han simulado.

concreto, la mejora del Modelo 4 respecto al Modelo 1 oscila en el rango [10,85%-24,17%], mientras que la mejora del Modelo 4 respecto al Modelo 2 oscila en el rango [16,29%-25,65%], mientras que los porcentajes que se obtuvieron en las pruebas fueron de 14,76% respecto al Modelo 1 y de 21,26% al Modelo 2. La causa de que algunos casos no tengan el mismo porcentaje puede estar influenciada por el número de intercambios que se hayan realizado durante su ejecución.

Así pues, tras ver todas las comparaciones realizadas, parece que el Modelo 4 es el que mejores resultados ha obtenido, y por ello, es conveniente realizar un estudio más a fondo de este modelo.

6.4.5 Estudio exhaustivo del comportamiento del Modelo 4

Hasta ahora, del Modelo 4 se conoce cómo se distribuyen las ocurrencias de *Operación de Eliminación* según se haya dado un intercambio o no durante *Desfragmentación*, y que el tiempo invertido para cada *Operación de Inserción* depende directamente del tiempo utilizado por la tarea *Inserción* que la compone.

Por tanto, para comenzar este estudio hay que referirse otra vez a las Figuras 6.15 (c) y (d), que muestran el comportamiento de *Operación de Inserción* y de *Operación de Eliminación*, respectivamente. En ellas se pueden observar bastantes picos en el comportamiento de ambas operaciones. Este estudio sólo se centrará en explicar los motivos que producen los picos en *Operación Inserción* y en las ocurrencias de *Operación de Eliminación* en las que no se haya llevado a cabo ningún intercambio sobre la tabla de arbitraje. Se desestimará el estudio del comportamiento de las operaciones de eliminación en las que se haya dado algún intercambio.

El hecho de desestimar el estudio de los picos producidos en las operaciones de eliminación con intercambio es debido a que el tiempo de ejecución de un intercambio sobre la tabla de arbitraje depende de muchos factores, siendo la mayoría de ellos cuestiones de implementación: nivel de la tabla de arbitraje donde se va a aplicar el intercambio, actualización de estructura de datos que contiene las peticiones atendidas en la tabla de arbitraje, número de peticiones atendidas, número de conjuntos singulares afectados por el intercambio, ubicación tanto de los conjuntos que albergan peticiones, como de los conjuntos singulares, etc.

Así pues, se comienza el estudio comentando los picos dados en *Operación de Inserción.* Como punto de partida, se sabe que su comportamiento depende directamente del de la tarea *Inserción*. La Figura 6.22 muestra la distribución de los tiempos empleados en *Operación de Inserción* y en todas las tareas que la componen. Además, en ella se puede observar la dependencia de la operación respecto a la tarea de *Inserción* al ser el comportamiento de ambos casi idénticos. Entonces, esos picos son causados por dicha tarea a la hora de ocupar un conjunto para poder atender una petición recién solicitada. En concreto hay dos picos significativos en *Operación Inserción*, uno para el valor 17 µs y el segundo para el valor 23 µs. Para averiguar qué es lo que provoca dichos picos, es necesario conocer el comportamiento de la operación y sus tareas en cada nivel.

En la Figura 6.23 se muestra cómo se comportan las operaciones de inserción y las tareas que contiene según el nivel del que se ha solicitado un conjunto. Se puede ver que las *Operaciones de Inserción* de nivel 1 son los causantes del segundo pico. Esto es debido

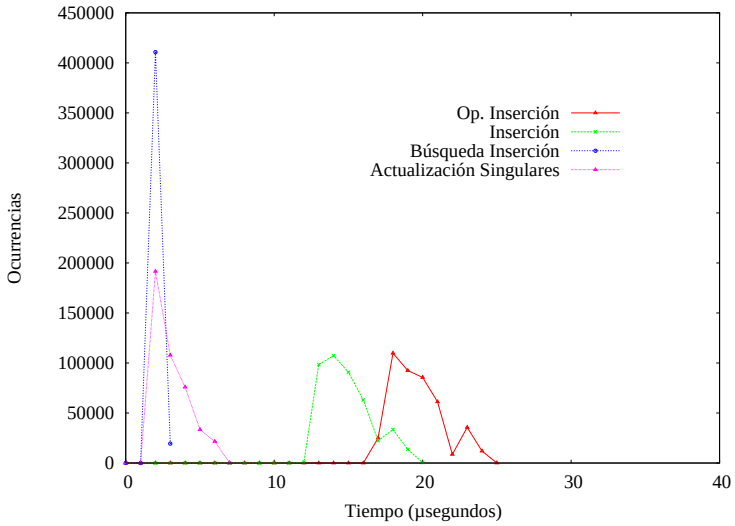
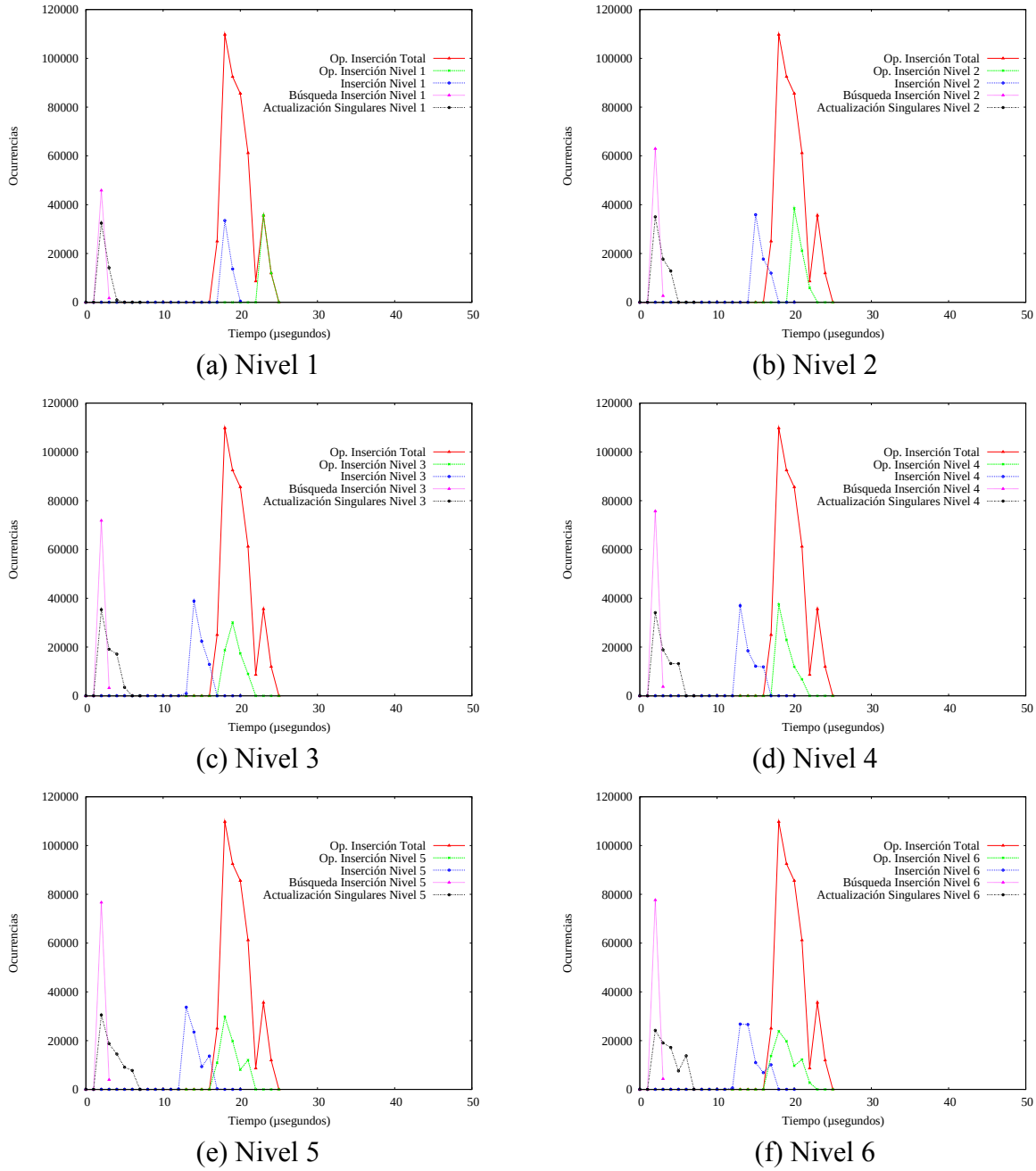


Figura 6.22: Ocurrencias de *Operación de Inserción* y de las tareas que la componen para el Modelo 4.

a que en estos casos, el número de conjuntos cuyo estado hay que cambiar es bastante mayor que en el resto de niveles. Es más, la peor ocurrencia que se ha dado de esta operación durante la simulación consiste en ocupar un conjunto de nivel 1 cuando la tabla de arbitraje está vacía. Para ello se debe cambiar el estado de 64 conjuntos: poner como "no libre" el conjunto $E_{0,0}$, ocupar el conjunto $E_{1,0}$ que es el que va a atender la petición, y todos sus hijos. Además, hay que añadir el tiempo necesario para la actualización de la estructura Singulares.

Por otro lado, el primer pico es consecuencia del resto de niveles, aunque en mayor proporción de las operaciones de inserción de los niveles 3, 4, 5 y 6, porque se distribuyen por el mismo intervalo de tiempo donde se encuentra el pico. Los tiempos mínimos de las operaciones de inserción de niveles altos son mayores porque el número de conjuntos cuyo estado hay que cambiar es mayor. Sin embargo, los tiempos máximos de las operaciones de inserción de estos niveles son prácticamente los mismos. Como se ve en la Figura 6.23, según es menor el nivel, los tiempos máximos empleados en actualizar la estructura *Singulares* son mayores. Los siguientes ejemplos mostrarán de qué manera afecta el nivel a la actualización de la estructura de datos *Singulares*.

En los siguientes ejemplos, se partirá de una situación en la cual la tabla de arbitraje tiene 64 entradas y está totalmente vacía, ya que este caso es el que peores resultados ofrece.



(a) Nivel 1 (b) Nivel 2

(c) Nivel 3 (d) Nivel 4

(e) Nivel 5 (f) Nivel 6

Figura 6.23: Distribución por niveles de los tiempos de *Operación de Eliminación* y las tareas que la componen en el Modelo 4.

Ejemplo 26 *Partiendo de una tabla de arbitraje vacía, llega una petición de inserción que solicita un conjunto de nivel 6. Al ocupar el conjunto $E_{6,0}$, que es el que corresponde como ya se vio al describir el método de búsqueda de conjuntos candidatos. Entonces, aparecen seis nuevos conjuntos singulares ($E_{6,1}$, $E_{5,2}$, $E_{4,4}$, $E_{3,8}$, $E_{2,16}$ y $E_{1,32}$), tal como muestra la Figura 6.24. Esos conjuntos hay que insertarlos en la estructura Singulares. Estos conjuntos son el hermano de $E_{6,0}$, y desde el nivel 5 hasta el 1, el conjunto hermano del ancestro de $E_{6,0}$ en dicho nivel. Esto implica que hay que calcular 6 veces el índice del conjunto hermano de uno dado, y cinco veces el índice de un conjunto ancestro de $E_{6,0}$.*

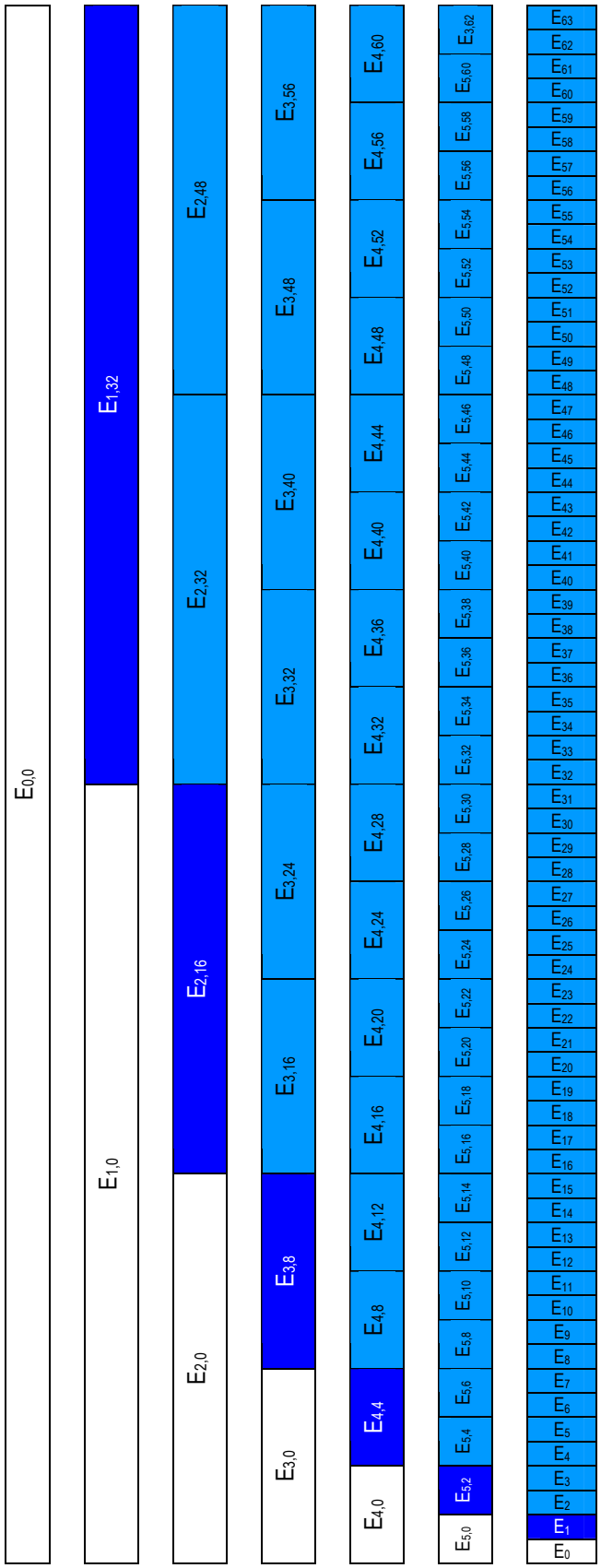


Figura 6.24: Tabla de arbitraje del Ejemplo 26 tras eliminar el conjunto $E_{6,0}$ y situación inicial del Ejemplo 28.

Ejemplo 27 *En este ejemplo, como ya se comentó, también se toma como punto de partida una tabla de arbitraje de 64 entradas totalmente vacía. En este caso, llega una petición de inserción que solicita un conjunto de nivel 3. Tras esto, se ocupa el conjunto* $E_{3,0}$*, pero aparecen dos nuevos conjuntos singulares, pasando la tabla de arbitraje a estar tal y como muestra la Figura 6.26. Además, también hay que introducir en la estructura de datos Singulares los índices de los conjuntos de los nuevos conjuntos singulares. Siguiendo los mismos pasos que en el ejemplo anterior, en esta ocasión solo es necesario calcular 3 veces el índice del conjunto hermano de uno dado y dos veces el índice del conjunto ancestro de* $E_{3,0}$*.*

Viendo los dos ejemplos anteriores, la conclusión que se obtiene es que en los peores casos, el tiempo de más que invierte una *Operación de Inserción* de tipo 3 respecto a la de tipo 6 a la hora de ocupar un mayor número de conjuntos, lo recupera en estos casos a la hora de actualizar la estructura singulares.

Así pues, al tener el mismo tiempo máximo las operaciones de estos niveles, siempre habrá un intervalo de tiempo en el que sus ocurrencias coincidirán, y esto es lo que provoca el primer pico del comportamiento de *Operación Inserción.*

Una vez se ha comentado el comportamiento de *Operación de Inserción*, toca el turno de describir el comportamiento de las operaciones de eliminación sin intercambio. Como se vio en el Apartado 6.4.3, estas ocurrencias están representadas en el primer bloque que se puede diferenciar de esta operación. Tal como muestra la Figura 6.25, su comportamiento está vinculado al de la tarea *Eliminación*, puesto que para la tarea *Desfragmentación* se invierte prácticamente el mismo tiempo en todas estas ocurrencias.

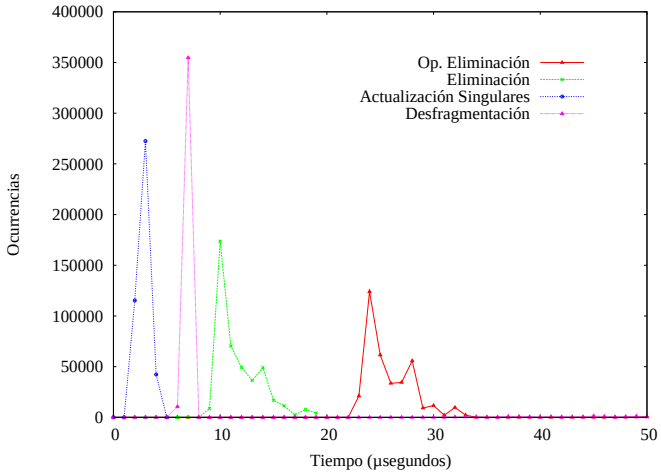


Figura 6.25: Distribución de los tiempos de *Operación de Eliminación* y sus tareas en el Modelo 4.

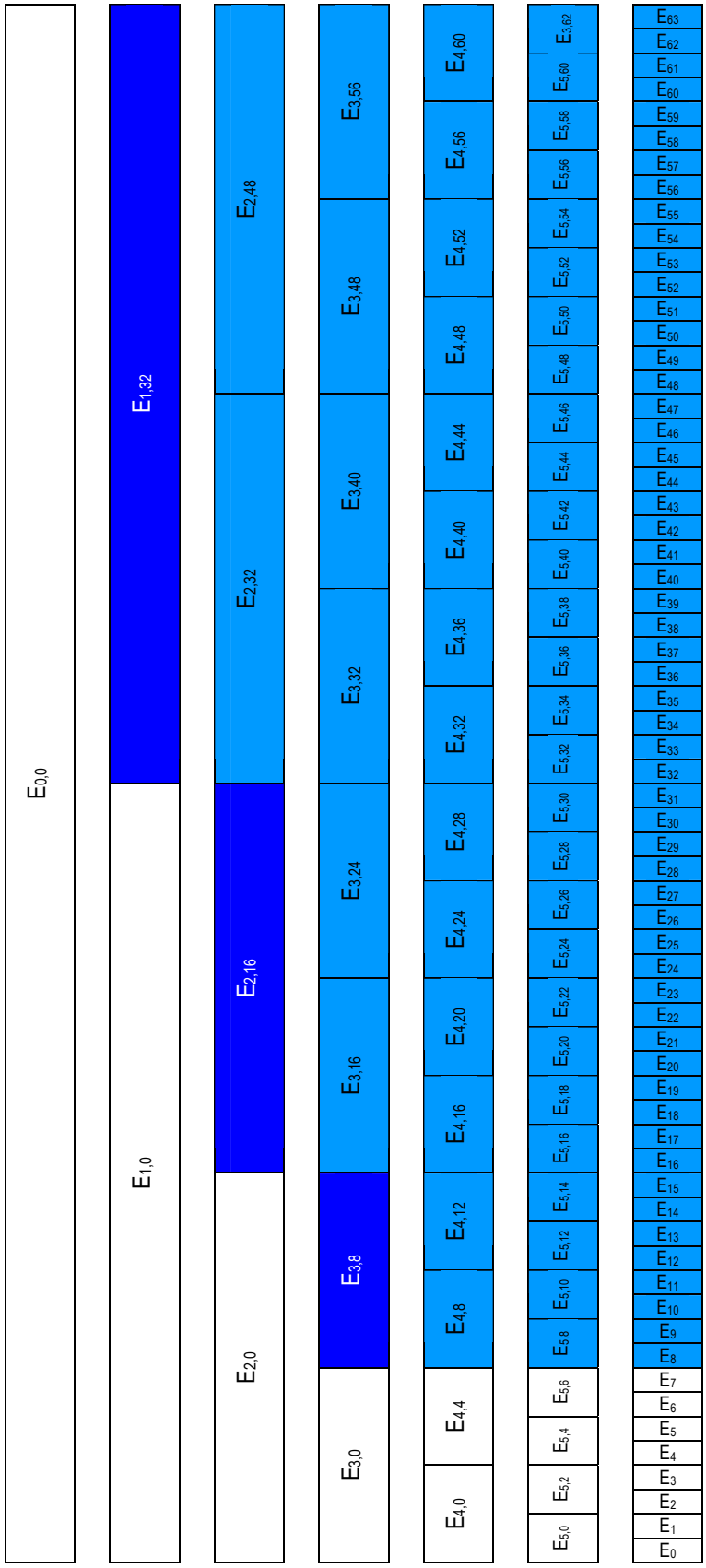


Figura 6.26: Tabla de arbitraje del Ejemplo 27 tras eliminar el conjunto $E_{3,0}$.

Volviendo a la Figura 6.25, se observan 4 picos en *Operación de Eliminación*, en concreto, están localizados para los valores 24 µs, 28 µs, 30 µs y 32 µs. Al igual que se hizo con *Operación de Inserción*, es necesario conocer el comportamiento de esta operación en cada uno de los niveles.

La Figura 6.27 muestra el comportamiento desglosado en niveles de *Operación de Eliminación* y de varias de las tareas que la componen. Se puede observar en dicha Figura que:

- El primer pico está formado por ocurrencias de nivel 3, 4, 5 y 6.

- El segundo pico está formado prácticamente en su totalidad por todas las ocurrencias de *Operación de Eliminación* de nivel 1.

- El tercer pico está formado casi totalmente por ocurrencias de nivel 5.

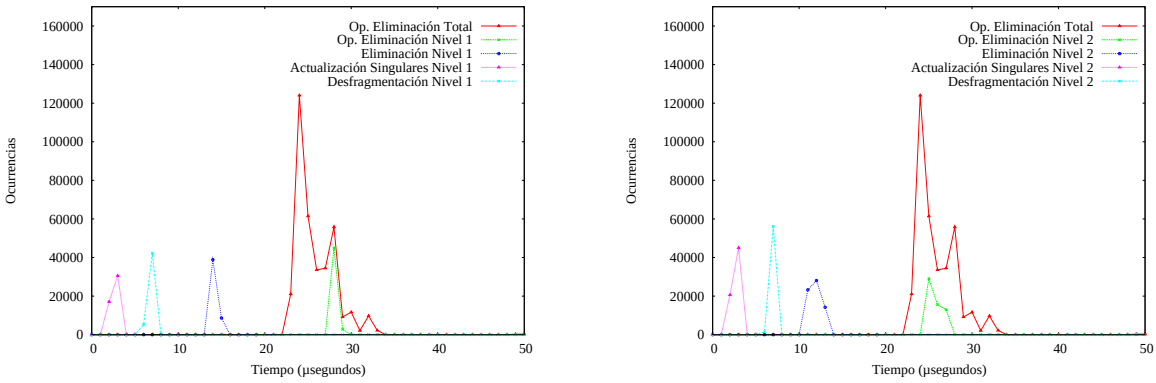
- El último pico está formado en su totalidad por ocurrencias de nivel 6.

- Las ocurrencias de nivel 3 y 4 también poseen un pico justo antes de su máximo valor, aunque no quede reflejado en el comportamiento total de esta operación.

A la hora de liberar un conjunto, una situación que puede surgir es que el conjunto recién liberado pueda unirse con su hermano en caso de estar éste último también libre, y formar juntos un conjunto libre de mayor tamaño. Para llevar a cabo todo esto, tras liberar el conjunto habrá que ir calculando índices de conjuntos hermanos y ancestros para ver si se puede dar esa situación. Por tanto, como se vio en *Operación Inserción*, a menor tamaño del conjunto, más niveles podría subir buscando conjuntos con los que unirse tras haber sido liberado, y en esos cálculos se invierte mucho tiempo. En el siguiente ejemplo se va a citar el peor caso de este bloque.

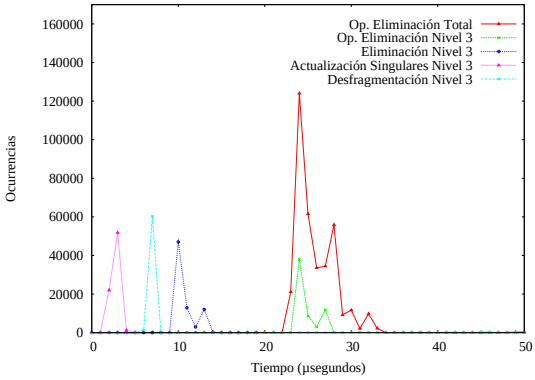
Ejemplo 28 *En este ejemplo, la situación inicial consiste en una tabla de arbitraje de 64 entradas con un solo conjunto ocupado, en concreto, el conjunto* $E_{6,0}$*. Por tanto, la tabla de arbitraje contiene 6 conjuntos singulares (*$E_{6,1}$*,* $E_{5,2}$*,* $E_{4,4}$*,* $E_{3,8}$*,* $E_{2,16}$ *y* $E_{1,32}$*), tal y como muestra la Figura 6.24. Cuando* $E_{6,0}$ *se libera, se debería calcular para liberar los conjuntos ancestros, seis veces el índice de un conjunto hermano y cinco veces el índice de un conjunto ancestro. Además, esos mismos cálculos se han de realizar a la hora de actualizar la estructura de datos Singulares.*

Por tanto, todo lo que se ha mencionado en el Ejemplo 28, hace que los intervalos de

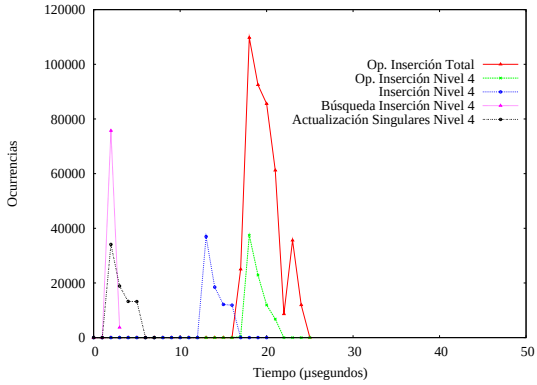


(a) Nivel 1

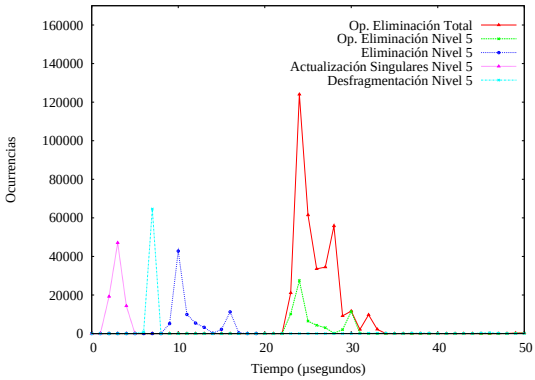
(b) Nivel 2



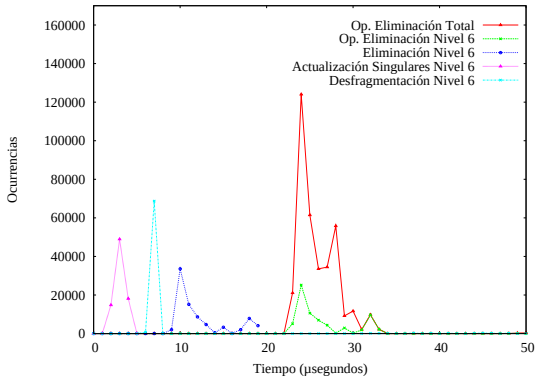
(c) Nivel 3



(d) Nivel 4



(e) Nivel 5



(f) Nivel 6

Figura 6.27: Distribución por niveles de los tiempos de *Operación de Eliminación* y las tareas que la componen en el Modelo 4.

las ocurrencias de menor nivel sean mayores. Esta situación en concreto no se dará muchas veces, pero otras de coste muy parecido sí, y esas son las que forman los dos últimos picos del comportamiento de la *Operación de Eliminación*.

El primer pico, como se ha dicho, se forma por las ocurrencias de nivel 3, 4, 5 y 6. Las ocurrencias de estos niveles, a la vez que pueden ser muy complejas, pueden ser tan simples que hacen que sus menores tiempos sean los menores de entre todos los niveles. En el Ejemplo 28 se muestra un caso muy simple de *Operación de Eliminación.*

Ejemplo 29 *En este ejemplo, la situación inicial consiste en una tabla de arbitraje de 64 entradas en la que existe un conjunto de nivel 5 que está ocupado, y a la vez, su hermano también está siendo ocupado (Figura 6.28). En un cierto momento, deja de ser necesario que dicho conjunto siga estando ocupado, y se ha de liberar. Su liberación constará de liberar ese mismo conjunto y a sus dos hijos, seguido del cálculo del índice de su conjunto hermano para comprobar si está libre o no, siendo el resultado negativo, y con ello, termina la tarea de Eliminación y a la vez, la Operación de Eliminación.*

Por tanto, la simplicidad de algunos casos en las operaciones de eliminación de conjuntos de menor tamaño, unida a la complejidad de otros, hace que sus ocurrencias abarquen un amplio intervalo de tiempo, y con ello, que formen entre todas el primer pico.

Por último para explicar por qué el tercer pico está formado casi en su totalidad por ocurrencias de nivel 1, es tan simple como se explicó el pico formado en la *Operación de Inserción* por las ocurrencias de este mismo nivel. Cuanto mayor sea el conjunto a eliminar, mayor será el número de conjuntos cuyo estado ha de cambiar. Por ello, la eliminación de estos conjuntos de la tabla de arbitraje puede afectar bien a 63 o a 64 conjuntos, según esté o no libre su conjunto hermano. Como además tampoco se han de realizar muchos cálculos de índices de conjunto ancestro o hermano, ya que al estar un nivel más abajo que el nivel 0, tienen solamente un ancestro, y por ello, todas sus ocurrencias son muy parecidas, tales que invierten aproximadamente el mismo tiempo en llevar a cabo la eliminación.

6.5 RESUMEN

El objetivo de este capítulo era el de evaluar los métodos cuya implementación se desarrolló en el capítulo anterior. Pero antes de ello, era necesario seguir unos pasos.

Primero había que decidir qué método de evaluación utilizar que estuviera acorde con los recursos que había entonces y pudiera proporcionar unos resultados con un nivel de detalle del sistema lo suficientemente bueno. De todos los métodos conocidos, el que mejor cumplía estas premisas era el consistente en *simulaciones*. Para llevar a cabo las simulaciones, era necesaria una herramienta para poder realizar las pruebas de las que obtener unos resultados que permitiesen posteriormente realizar la evaluación que se pretendía llevar a cabo.

Dicha herramienta consiste en un simulador de la tabla de arbitraje de los puertos de salida de conmutadores, encaminadores y terminales de una red InfiniBand. A lo largo del

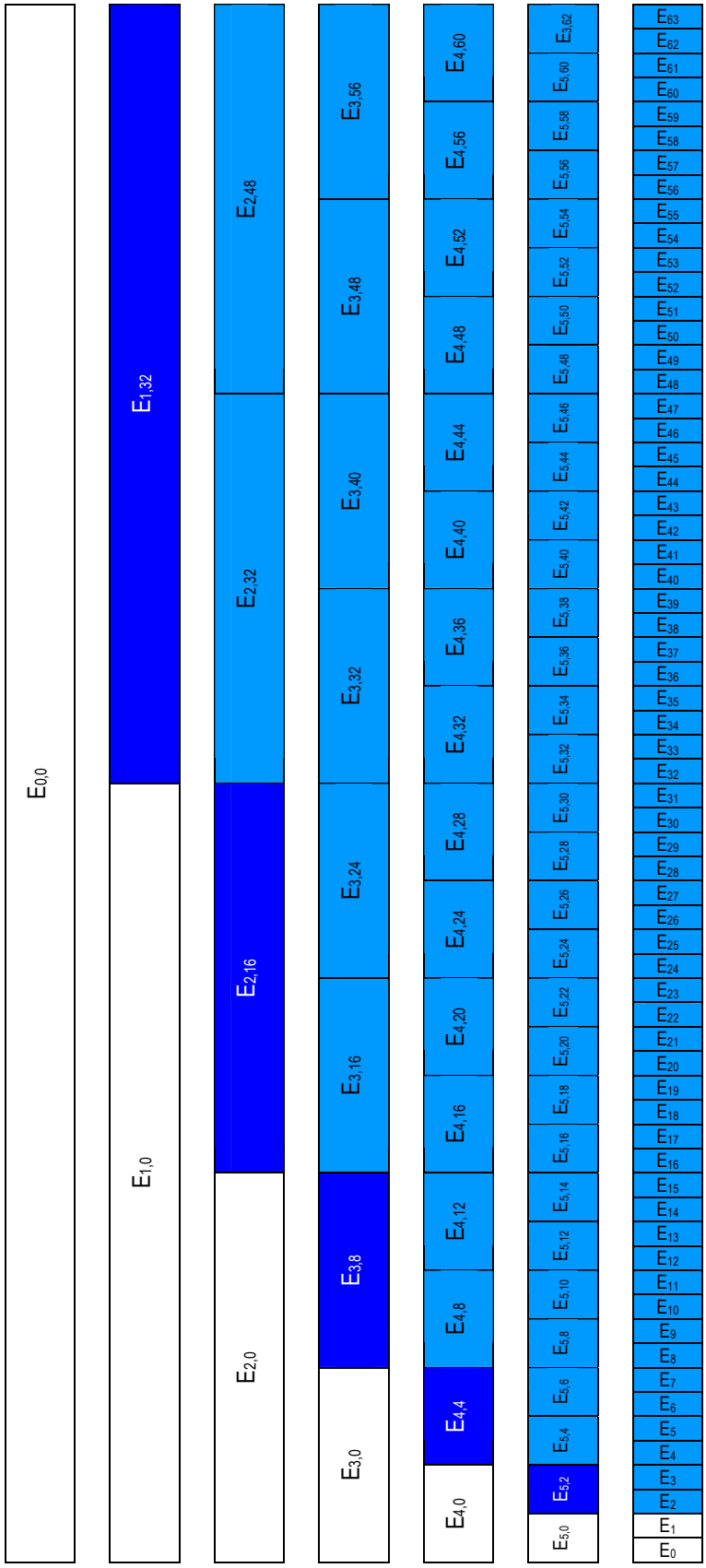


Figura 6.28: Situación inicial de la tabla de arbitraje del Ejemplo 29.

capítulo, se han comentado prácticamente todas las características de dicho simulador, pudiendo dividir dicha descripción en dos grandes partes: la primera consistía en las características generales del simulador y la segunda en la descripción de los resultados que éste proporciona.

En lo que a las características se refiere, se comentó qué arquitectura simulaba, exponiendo hasta qué nivel de detalle es fiel este simulador a una tabla de arbitraje real.

Se realizó una descripción de las características de este simulador. En concreto, se comentó qué es un simulador dirigido por eventos, explicando los dos tipos de eventos que existen, *Petición de Inserción* y *Petición de Eliminación*. En ambos casos se explicó en qué consistían y qué representaban en un sistema real. También se comentó que la carga utilizada sobre el simulador se genera aleatoriamente, tal que para cada tipo de evento, la probabilidad de que se produzca es del 50%. Por último, se comentó qué parámetros se le proporcionan al simulador para poder configurar las pruebas que se han de realizar con él.

A la hora de describir los resultados obtenidos por el simulador, se volvió a realizar una división en varios grupos, según el tipo de información que proporcionara. Los tres grupos eran: resultados relacionados con las peticiones, relacionados con los intercambios y relacionados con los tiempos de ejecución.

En el primer conjunto se mencionaban resultados tales como el número de peticiones solicitadas, y de ellas, cuántas son atendidas y cuántas no atendidas.

En el caso del segundo grupo, toda la información tiene que ver con los intercambios. Por ejemplo, número de intercambios realizados durante la desfragmentación, pudiendo incluso obtener una distribución por niveles de dichos intercambios. También obtenía resultados relacionados con el número de reordenaciones o desfragmentaciones en las que no se han llevado a cabo intercambio alguno.

El último grupo es el más importante, ya que permite evaluar directamente los métodos que fueron implementados. Además, fue en este apartado donde se definieron dos términos muy importantes en este capítulo, los términos *tarea* y *operación*. Del término tarea se dijo que coincide con alguno de los métodos implementados en el simulador o parte de ellos, coincidiendo estos métodos, la mayoría de las veces, con alguno de los que se propusieron en los capítulos 4 y 5. A continuación, se definió operación como el total de tareas que realiza el simulador como respuesta a un evento que ha recibido.

Después, se describió como se iban a realizar las pruebas, introduciendo de nuevo los modelos que en capítulos anteriores se habían propuesto para combinar los métodos que se

han implementado para este Proyecto Fin de Carrera. Además, durante la explicación de cómo se iban a llevar a cabo las pruebas, se diferenciaron dos términos que se han utilizado mucho a lo largo del capítulo: *simulación* y *ejecución*. El primero se define como un conjunto de operaciones realizadas por el simulador, mientras que al segundo como un conjunto de simulaciones efectuadas durante una misma llamada al simulador.

Tras esto, se pasó a comentar los resultados obtenidos, describiendo primero el comportamiento de cada uno de los modelos que se han ejecutado en el simulador, aunque dos de ellos, los modelos 2 y 3, se agruparon en uno, en el Modelo 2, puesto que su comportamiento era muy similar.

Una vez ya se había descrito el comportamiento de los modelos, se compararon sus resultados entre sí para decidir qué modelo aprovechaba mejor los métodos implementados en este trabajo. El resultado de la comparativa fue que el Modelo 4 obtuvo los mejores resultados ya que era el modelo que menos intercambios llevaba a cabo, la cual es la tarea más costosa de todas.

Por tanto, sabiendo que el Modelo 4 era el que mejores resultados obtenía, se procedió a realizar un estudio exhaustivo de su comportamiento, estudiando en concreto el comportamiento de *Operación de Inserción* y de las ocurrencias de *Operación de Eliminación*. Este estudio se centró prácticamente en interpretar ciertos picos que se daban en el comportamiento de ambas operaciones.

Capítulo 7

CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se presentan las principales conclusiones que se han obtenido durante el desarrollo de este Proyecto Fin de Carrera y se proponen varios trabajos que den continuidad a este proyecto.

7.1 CONCLUSIONES

Cuando se propuso este Proyecto Fin de Carrera, el objetivo era poder evaluar un conjunto de algoritmos propuestos en la tesis doctoral realizada por Francisco José Alfaro Cortés, que es también tutor de este trabajo. El propósito de dichos algoritmos es la gestión de las tablas de arbitraje de los puertos de salida de conmutadores, encaminadores y terminales de una subred InfiniBand. La combinación de estos algoritmos da lugar a varios modelos de gestión, siendo tres de ellos propuestos en la mencionada tesis doctoral.

Para llevar a cabo el proceso de evaluación, en este proyecto se ha realizado previamente la implementación de los algoritmos, así como la propuesta de un nuevo modelo que surge durante el desarrollo del proyecto. Este nuevo modelo de gestión propuesto difiere de los propuestos en la tesis en la forma de buscar el conjunto de entradas que tienen que asignarse a una nueva petición. En concreto, en *Operación de Inserción* se usa un método que busca el conjunto singular de menor tamaño capaz de atender la petición que ha recibido la tabla de arbitraje, mientras que los modelos propuestos en la

tesis usan un método que realiza la búsqueda de izquierda a derecha de un conjunto libre capaz de atender la petición.

Tras realizar una serie de pruebas con los diferentes modelos, se analizaron los resultados que se obtuvieron, los cuales daban al Modelo 4 (modelo propuesto en este proyecto) como el que mejor rendimiento obtenía. Las razones más importantes que han producido este hecho han sido:

- El método *búsquedaInteligenteDeConjunto* ha sido el máximo responsable del resultado obtenido por el Modelo 4. Al proponer la estructura de datos *Singulares*, se buscaba un nuevo método que mejorara el tiempo de búsqueda de un conjunto de entradas para después asignarlo a una petición dada. Cuando se propuso inicialmente este método, esa era la única finalidad que se buscaba de él, pero tras estudiarlo, se vio que ofrecía más ventajas que la de rebajar el tiempo de búsqueda, como no tener que aplicar *Reordenación* tras una *Eliminación*, ni *Desfragmentación* tras una *Inserción*. Cuando se evaluaron los resultados, se pudo comprobar que no aplicar esas tareas gracias a la forma como actúa ahora el método de búsqueda, hacía que los tiempos obtenidos por el Modelo 4 fueran mucho menores que los del resto, mientras que la mejora que se obtuvo en la búsqueda de conjuntos resultó ser insignificante.

- Durante la evaluación, se pudo observar que el Modelo 4 aplicaba un número menor de intercambios sobre la tabla de arbitraje que el resto. Como las tareas consistentes en realizar intercambios son bastante importantes, dado su alto coste de temporal, unido a una alta utilización de ellas, se produce un incremento importante de los tiempos de ejecución de las operaciones en las que se realizan.

A parte de estas conclusiones referentes al mejor rendimiento del Modelo 4 respecto a los demás, se han obtenido otras conclusiones a resaltar sobre otros aspectos que son:

- La tarea *Reordenación* se propuso para que el método de búsqueda anterior encontrara un conjunto tal que, cuando el conjunto seleccionado fuese ocupado, la tabla no pasase a estar no normalizada, y así, evitar tener que realizar una *Desfragmentación* sobre la tabla de arbitraje. Sin embargo, el número de intercambios que se han realizado en los modelos que utilizaban esta tarea ha sido superior al del Modelo 1, que no la utiliza. Por tanto, su utilización puede considerarse en estos momentos poco útil.

- La estructura de datos *Singulares* permite manejar la información de la tabla de arbitraje de forma sencilla.

Así pues, a partir de los resultados obtenidos a lo largo de este Proyecto Fin de Carrera, parece lógico darle continuidad a este trabajo, ya sea para completar algunos estudios, realizar otro tipo de pruebas, así como para implementar los algoritmos en sistemas reales.

7.2 TRABAJO FUTURO

Como se ha dicho en el apartado anterior, los resultados obtenidos son lo suficientemente interesantes como para dar continuidad a este trabajo, realizando otros estudios complementarios. Se plantean varías líneas de trabajo que den continuidad a este Proyecto Fin de Carrera. Éstas son:

- Repetir el estudio utilizando una función de distribución distinta para generar los eventos con el fin de estudiar posibles diferencias entre los resultados obtenidos en este trabajo con los que obtenga el otro.

- Estudiar más exhaustivamente el comportamiento de los intercambios, ya que no se realizó ahora porque su complejidad era demasiado alta y se salía del objetivo de este proyecto. En caso de realizarse, se podría proponer alguna mejora en el funcionamiento de dicha tarea, lo cual sería bastante beneficioso ya que es la tarea más costosa de todas.

- El año pasado, Juan Antonio Villar Ortiz presentó el Proyecto Fin de Carrera *"Estudio y puesta en marcha de una subred InfiniBand"* [Vil04]. Junto con el proyecto que ahora se presenta, se podría dar un paso más en el estudio de los algoritmos propuestos en este trabajo, puesto que proporciona suficiente documentación para poder iniciar la implementación de los métodos aquí propuestos en una subred InfiniBand real.

- En el trabajo desarrollado en [Alf03] se partía de un algoritmo de inserción basado en recorrer los conjuntos de izquierda a derecha en el árbol para seleccionar el primer conjunto libre. A partir de este algoritmo se desarrollaban una serie de proposiciones y teoremas para demostrar que la metodología propuesta conseguía hacer una gestión eficiente de la tabla de arbitraje. Con el Modelo 4 propuesto en este Proyecto Fin de Carrera se varía el algoritmo de inserción y se adapta para que realice la búsqueda en base a la estructura *Singulares*. Así pues, se podría retomar la teoría formal desarrollada en [Alf03] para demostrar que la

metodología propuesta es válida con el nuevo algoritmo de *inserción* sin necesidad de usar el algoritmo de *reordenación*.

- La evaluación realizada aquí se ha hecho sólo sobre una máquina concreta. Se podría realizar la misma evaluación en otras máquinas y tratar de obtener unas expresiones que permitan calcular el tiempo de cada operación/tarea en función de la frecuencia del procesador y de los parámetros de la operación.

Bibliografía

[Alf03] Francisco J. Alfaro. *Una sencilla metodología para garantizar calidad de servicio en subredes InfiniBand*. Número 188 en Colección Tesis Doctorales. Servicio de publicaciones de la Universidad de Castilla-La Mancha, 2004. ISBN: 84-8427-320-2.

[IBA99] *InfiniBand™ Trade Association*, 1999. http://infinibandta.com.

[Inf00] InfiniBand Trade Association. *InfiniBand Architecture Specification. Volume 1. Release 1.0*, Octubre 2000.

[Inf01] InfiniBand Trade Association. *InfiniBand Architecture Specification. Volume 1. Release 1.0.a*, Junio 2001.

[Inf02] InfiniBand Trade Association. *InfiniBand Architecture Specification. Volume 1. Release 1.1*, Noviembre 2002.

[Jai91] Raj Jain. *The art of computer system performance analysis: techniques for experimental design, measurement, simulation and modeling*. John Wiley and Sons, Inc., 1991.

[LVAS04] Joaquín Laborda, Juan A. Villar, Francisco J. Alfaro, José L. Sánchez. *Implementación y evaluación de un modelo de gestión de la tabla de arbitraje de InfiniBand*. Technical Report #DIAB-04-12-1, Diciembre 2004.

[Pel00] Joe Pelissier. *Providing Quality of Service over InfiniBand Architecture Fabrics*. In Proceedings of the 8th Symposium on Hot Interconnects, Agosto 2000.

[Shan02] Tom Shanley, *InfiniBand Network Architecture*. Addison-Wesley, Octubre 2002.

[Vil04] Juan A. Villar. *Estudio y puesta en marcha de una subred InfiniBand*. Proyecto Fin de Carrera, Julio 2004.