Diseño de un Protocolo COMA en Bus Común Distribuido Con Reemplazo

Diego R. Llanos Ferraris, Benjamín Sahelices Fernández, Agustín De Dios Hernández

Resumen— **La utilización de sistemas multicomputador débilmente acoplados permite la construcción de máquinas de memoria compartida distribuida con una buena relación precio/rendimiento. Se presenta aquí un protocolo que permite la utilización de un bus común para la comunicación entre los nodos de proceso de un sistema COMA. El protocolo presentado (denominado VSR-COMA) supone una evolución del protocolo COMA-BC, que ha demostrado su utilidad en la ejecución de aplicaciones paralelas sobre sistemas multicomputador. VSR-COMA incorpora una serie de mejoras que posibilitan una utilización más cercana a las condiciones de trabajo reales de un sistema COMA: asociatividad por conjuntos, mecanismos avanzados de reemplazo y primitivas de sincronización. Se describe el protocolo VSR-COMA en detalle, indicando además la estrategia de reemplazo propuesta.**

Palabras clave— **Sistemas COMA en bus común, COMA-BC, VSR-COMA, reemplazo, desalojo.**

I. Introducción

EL protocolo VSR-COMA supone una evolución del protocolo COMA-BC [1], [2], [3]. COMA-BC es un protocolo COMA en bus común que busca obtener altas prestaciones mediante la utilización de múltiples computadores de propósito general. Para ello, COMA-BC utiliza un esquema de conexión no jerárquico, junto con un mecanismo híbrido snoopy-directorios para el control y mantenimiento de la coherencia. Cada uno de los bloques lleva asociado un único propietario, que es el nodo encargado de responder las solicitudes de lectura o escritura provenientes de otros nodos. El propietario de un bloque no es fijo: cuando un nodo solicita un bloque para escribir sobre él, dicho nodo pasa a ser el nuevo propietario del bloque.

La estructura de un nodo de proceso en COMA-BC puede verse en la figura 1. Cada nodo dispone de un controlador de coherencia que se encarga de gestionar la información presente en la *memoria de atracción* (AM) [4]. Para ello, el controlador de coherencia del nodo dispone de una tabla con la información de estado referente a cada marco de bloque cache de la memoria de atracción local. Además, dicho controlador mantiene de forma local una tabla con la información de directorio, en donde se almacena la identidad del propietario de cada bloque presente en el sistema.

Departamento de Informática, Universidad de Valladolid, España. E-mail: {*diego, benja, agustin*}*@infor.uva.es*.

COMA-BC se ha diseñado al objeto de que el número de eventos que circulan por el bus se mantenga lo más bajo posible, al ser el bus el cuello de botella de esta clase de sistemas. Para ello, se aprovecha la capacidad de difusión que poseen los sistemas basados en bus común, mediante la cual cada mensaje enviado por un nodo es recibido por todos los demás. COMA-BC utiliza esta característica para mantener actualizada la información de directorio y estado de cada nodo.

En COMA-BC existen diferentes estados posibles para cada bloque: *Inválido*, que indica que el contenido del marco cache no es válido; *Limpio*, que indica que puede accederse al bloque para lectura; y *Sucio*, que indica que puede accederse al bloque tanto para lectura como para escritura. Por otra parte, existen dos estados transitorios: *Inv-Esp-RRB* e *Inv-Esp-RRI*. Estos estados permiten indicar que el contenido del marco correspondiente es inválido, y que dicho marco está pendiente de la resolución de una transacción de lectura o escritura, respectivamente. La necesidad de estados transitorios es una consecuencia de la utilización de un bus común para la interconexión de los nodos. En efecto, al no existir la posibilidad de reservar el bus hasta que se complete una transaccion, dichas transacciones pueden solaparse, iniciándose una segunda antes de que se complete la primera. Esta característica obliga a la utilización de estados transitorios, para indicar la existencia de una transacción en curso sobre el bloque correspondiente.

COMA-BC ha sido probado con aplicaciones pertenecientes al conjunto Splash-2 [5], obteniéndose índices de rendimiento comparables a otras arquitecturas COMA [1], lo que ha demostrado la viabilidad de un protocolo COMA distribuido sobre bus común. Sin embargo, COMA-BC tiene algunas limitaciones. La principal atañe a la función de mapa de la memoria de atracción local, que posee correspondencia directa. Por otra parte, la ausencia de un mecanismo de desalojo de bloques obliga a que el tamaño de la memoria de atracción local en cada nodo sea igual al tamaño de la memoria global. Este hecho, aunque favorece en gran medida la replicación de bloques en el sistema, limita la escalabilidad del conjunto de trabajo (*working set*) de las aplicaciones. En tercer lugar, sería deseable que el protocolo incorporara una operación de memoria

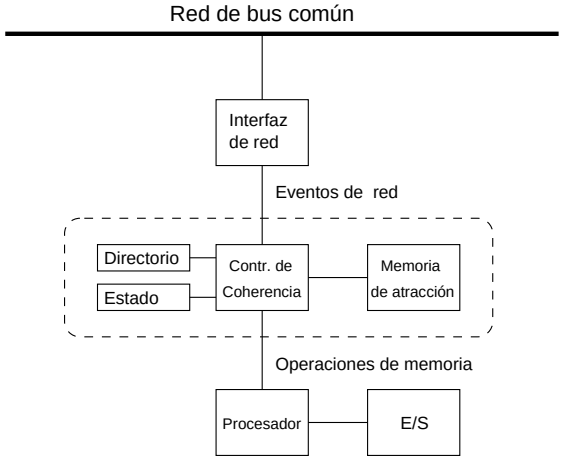

Fig. 1. Estructura de un nodo de proceso en COMA-BC.

del tipo "Test and Set" (TAS), para implementar mecanismos de sincronización entre procesos.

Los buenos resultados obtenidos en la ejecución de aplicaciones paralelas utilizando COMA-BC han animado a sus autores a continuar el desarrollo de protocolos COMA sobre bus común. El trabajo se ha centrado en el desarrollo de un protocolo que elimine las limitaciones de COMA-BC anteriormente descritas. Este trabajo ha permitido desarrollar VSR-COMA, un protocolo COMA en bus común que incorpora un elaborado mecanismo de reemplazo de bloques.

El resto del documento está organizado como sigue. La sección II especifica los objetivos de diseño establecidos para VSR-COMA. La sección III describe la estructura de un nodo en VSR-COMA, así como la filosofía y el funcionamiento general del protocolo. La sección IV describe el protocolo de coherencia en detalle, a través de su diagrama de estados. La sección V especifica la estrategia utilizada en VSR-COMA para la selección del nodo destino de una operación de reemplazo. Finalmente, la sección VI enumera las conclusiones de nuestro trabajo.

II. Objetivos de diseño

Los objetivos de diseño de VSR-COMA son una extensión de los propuestos para COMA-BC, y se basan en la obtención de un protocolo de coherencia tipo COMA para un sistema multicomputador de bus común. La eliminación de las limitaciones de COMA-BC descritas en el apartado anterior se traducen en los siguientes requisitos:

- Utilización en la memoria de atracción de una función de mapa de tipo asociativo por conjuntos.
- Desarrollo de un mecanismo de reemplazo que utilice la característica de difusión del bus común para seleccionar el nodo destino más apropiado para enviar el bloque.
- Implementación de una operación de memoria de tipo "Test And Set" (TAS), al objeto de facilitar la utilización de mecanismos de sincronización entre procesos.

Al igual que en COMA-BC, se ha considerado de especial importancia la reducción al mínimo del tráfico generado en la red. Esto ha llevado a la necesidad de establecer un mecanismo de reemplazo que seleccione el nodo destino más apropiado para recibir el bloque. En otros protocolos COMA con reemplazo, la selección del nodo destino se realiza de forma aleatoria (como es el caso de COMA-F [6]) o utilizando mecanismos de consulta para establecer el nodo con mejores posibilidades de aceptar el bloque, solucion adoptada en DICE [7], [8]. Ambas soluciones consumen un gran ancho de banda. En el caso de la selección aleatoria del nodo destino, la solicitud de reemplazo utiliza sólo un evento de red, lo que es el mínimo imprescindible. Sin embargo, las probabilidades de que ese nodo no disponga de espacio libre hace que la solicitud de reemplazo pueda ser rechazada, generándose una nueva solicitud e incrementando el tráfico de red. Por otra parte, aunque la solicitud sea aceptada, existe la posibilidad de que haya otro nodo que haya utilizado el bloque en el pasado y que esté en mejores condiciones de admitirlo, disminuyéndose así la tasa de fallos en el futuro.

En el caso de la selección de nodo por consulta, el número de eventos a intercambiar, dados n nodos, es de $n+1$, divididos como sigue: una solicitud, $n-1$ respuestas y el envío del bloque. Para seleccionar el nodo destino, DICE utiliza un mecanismo basado en prioridades, en donde cada nodo evalúa su capacidad para recibir el bloque y responde a la solicitud de desalojo indicando su prioridad. Este mecanismo es claramente superior al de la selección aleatoria, ya que utiliza la información de estado de cada nodo para seleccionar el destino más adecuado. Sin embargo, esto se realiza a expensas de un tráfico de red mucho mayor, tráfico que además crece linealmente con el número de nodos.

El protocolo VSR-COMA realiza la selección del nodo destino de una operación de reemplazo en función del estado de las memorias de atracción de los nodos remotos. Sin embargo, contrariamente a lo que sucede en DICE, esto no supone ningún incremento del tráfico de red, ya que cada nodo en VSR-COMA utiliza la información intercambiada a través del bus para mantener actualizada una tabla que contiene el estado de cada una de las memorias de atracción remotas. De esta manera, cuando

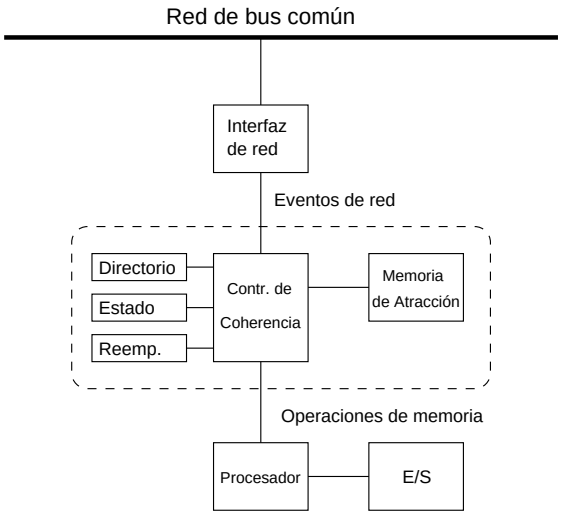

Fig. 2. Estructura de un nodo de proceso en VSR-COMA.

se necesita desalojar un bloque, el nodo local puede seleccionar el nodo remoto más apropiado para recibir dicho bloque. Por otra parte, la presencia en cada nodo de toda la información de reemplazo disponible permite utilizar estrategias de reemplazo más elaboradas, sin que esto suponga un aumento de tráfico en la red.

III. Gestión distribuida de la coherencia

La figura 2 muestra la estructura de un nodo de proceso en VSR-COMA. El controlador de coherencia se diferencia del utilizado en COMA-BC en que incorpora una tabla con información de reemplazo. Dicha tabla contiene el estado de cada marco de bloque para cada una de las memorias de atracción remotas.

Tanto la actualización de directorios como el mantenimiento de la información de reemplazo en VSR-COMA se realiza de forma distribuida. Todos los eventos intercambiados en la red llevan un identificador del bloque al que hacen referencia y un indicador del marco de bloque que ocupan en el conjunto de origen. Estos identificadores son suficientes para que todos los nodos puedan actualizar la información de reemplazo que mantienen de forma local.

Al igual que en COMA-BC, VSR-COMA utiliza el concepto de propiedad para cada uno de los bloques presentes en el sistema. Todos los bloques tienen un propietario, que es el encargado de responder a las solicitudes de lectura o escritura que el resto de nodos realicen sobre ese bloque. La propiedad de un bloque cambia dinámicamente, a medida que los nodos realizan operaciones sobre ese bloque.

A grandes rasgos, el funcionamiento del protocolo puede describirse como sigue. Si se produce una solicitud de lectura, el propietario responde enviando una copia del bloque al nodo que lo ha solicitado. En caso de que la solicitud sea de escritura, el nodo propietario también envía una copia del bloque, pero el bloque se invalida en el nodo propietario y el nodo *local* -el que ha realizado la solicitud- pasa a ser el nuevo propietario.

En el caso de que un nodo desee desalojar un bloque del que es propietario, dicho nodo deberá seleccionar un nodo destino y enviarle una solicitud de desalojo. El nodo destino decidirá si acepta el bloque en función del estado de su memoria de atracción. Si dispone de suficiente espacio como para admitir el bloque, el nodo destino pasa a ser el nuevo propietario. En caso contrario, se envía una respuesta negativa y el nodo local deberá seleccionar un nuevo nodo destino.

El protocolo VSR-COMA ha sido validado a través de un modelo creado con Redes de Petri Coloreadas, lo que ha permitido verificar su funcionamiento bajo diferentes condiciones de trabajo.

IV. Protocolo de coherencia

Se denomina *operación de memoria* a cada solicitud de lectura, escritura o TAS realizada por el procesador y dirigida a un bloque de la memoria global. En VSR-COMA, el modelo de consistencia utilizado es el de consistencia secuencial, no considerándose necesaria la implementación de modelos de consistencia más relajados [9]. Por lo tanto, hasta que el controlador de coherencia no devuelve el resultado de una operación de memoria, el procesador no puede iniciar la siguiente. Las operaciones de memoria implementadas son PrRd (lectura), PrWr (escritura) y PrTAS ("test and set"). Por simplicidad en la descripción, sólo nos ocuparemos de las operaciones de lectura y escritura: la operación TAS produce en el protocolo los mismos efectos que la de escritura, al requerir la modificación del bloque.

Los *estados* en los que puede hallarse un bloque en VSR-COMA pueden dividirse en estados estables y transitorios. Al igual que en COMA-BC, la posibilidad de que se solapen transacciones en el bus obliga a utilizar estados transitorios, al objeto de indicar que un bloque determinado está pendiente de la finalización de una transacción y no puede ser accedido.

Los estados en los que puede encontrarse un bloque son los siguientes:

- Inv : La información presente en el marco de bloque correspondiente es inválida.
- Shared : Puede accederse al contenido del bloque a través de una operación de lectura, pero no de escritura ni de tipo TAS. El nodo no es el

propietario del bloque.

SharOwn : Puede accederse al contenido del bloque a través de una operación de lectura, pero no de escritura ni de tipo TAS. El nodo es el propietario del bloque, y *posiblemente* existan otras copias en otros nodos.

Excl : Puede accederse al contenido del bloque a través de una operación de lectura, escritura o TAS. No existen otras copias en el sistema. El nodo es el propietario del bloque.

InvWExcl : Es un estado transitorio que señala que el contenido del bloque es inválido, pero que el nodo está pendiente de completar una transacción para escribir sobre él. El nodo no es aún el propietario del bloque.

InvWShar : Es un estado transitorio que indica que el contenido del bloque es inválido, pero que el nodo está pendiente de completar una transacción para poder leer de él.

WExport : Es un estado transitorio que indica que el contenido del bloque es *válido*, pero el nodo está pendiente de desalojar el bloque. No pueden atenderse solicitudes de lectura o escritura sobre él, ya que, pese a que el bloque es válido, el nodo seleccionado para el desalojo puede haberlo aceptado, haber invalidado las restantes copias y haber escrito sobre él antes de que el actual propietario haya procesado el evento de respuesta positiva al reemplazo.

Se denomina *evento de protocolo* a cada uno de los mensajes intercambiados a través del bus común por los controladores de coherencia de los respectivos nodos. En VSR-COMA existe un total de nueve eventos de protocolo diferentes. Dichos eventos son los siguientes:

BusRreq : Solicitud de lectura de un bloque. Va dirigido al propietario del bloque, e incluye la etiqueta del bloque y el número del marco donde será almacenado en el nodo local.

BusRack : Respuesta al evento anterior. Incluye la etiqueta del bloque, una copia del mismo y el número del marco donde el bloque está almacenado en el nodo propietario.

BusWreq : Solicitud de escritura de un bloque. Va dirigido al propietario del bloque, e incluye la etiqueta del bloque y el número del marco donde será almacenado en el nodo local.

BusWack : Respuesta al evento anterior. Incluye la etiqueta del bloque, el propio bloque y el número del marco donde el bloque estaba almacenado en el nodo propietario (tras enviar este evento, el bloque queda invalidado y el nodo local es el nuevo propietario del bloque).

BusFInv : Evento utilizado por el nodo propietario de un bloque en estado SharOwn para invalidar el resto de copias que puedan existir de ese bloque. Incluye la etiqueta del bloque y el número del marco en donde está almacenado en el nodo propietario. No requiere respuesta.

BusEXreq : Solicitud de desalojo de un bloque. Este evento es generado por el propietario de un bloque, y va dirigido al nodo seleccionado por él como más adecuado para almacenar el bloque. Incluye la etiqueta del bloque, el marco que ocupa en el actual propietario, y una copia del bloque.

BusEXack : Respuesta positiva al evento anterior. El nodo que la genera pasa a ser el nuevo propietario del bloque. El evento incluye la etiqueta del bloque y el número del marco que el bloque ocupará en el nuevo propietario.

BusEXnak : Respuesta negativa al evento BusEXreq. El nodo que la genera no acepta el bloque, por lo que el nodo propietario deberá buscar otro candidato. Incluye la etiqueta del bloque rechazado.

BusRACE : Evento de resolución de condiciones de carrera. Se utiliza cuando un nodo recibe una solicitud respecto de un bloque del que ya no es el propietario, o bien cuando el bloque no puede ser accedido por estar pendiente de otra transacción. Contiene la etiqueta del bloque al que hacía referencia la solicitud.

La figura 3 muestra el diagrama de transiciones de estados del protocolo VSR-COMA. Se ha añadido un pseudo-estado, denominado NotPres, para indicar que el bloque no está presente en la memoria de atracción. Esta situación no es equivalente al estado Inv, porque no existe un marco de bloque vacío con la etiqueta de *ese* bloque.

Cuando la etiqueta de un bloque no está presente en la memoria de atracción y el controlador de coherencia recibe del procesador una operación de memoria sobre él, las acciones a realizar depende de la situación en el conjunto de atracción correspondiente al nodo local. Pueden darse cuatro situaciones diferentes:

1. Existe al menos un marco de bloque en estado Inv.
2. En caso contrario, existe al menos un bloque en estado Shared.
3. En caso contrario, existe al menos un bloque en estado SharOwn.
4. Todos los bloques son Excl.

Las acciones a realizar en cada una de estas cuatro situaciones aparecen reflejadas en la figura 4. En las dos primeras, el marco de bloque correspondiente cambia al estado InvWShar o InvWExcl, según la operación de memoria solicitada. En los

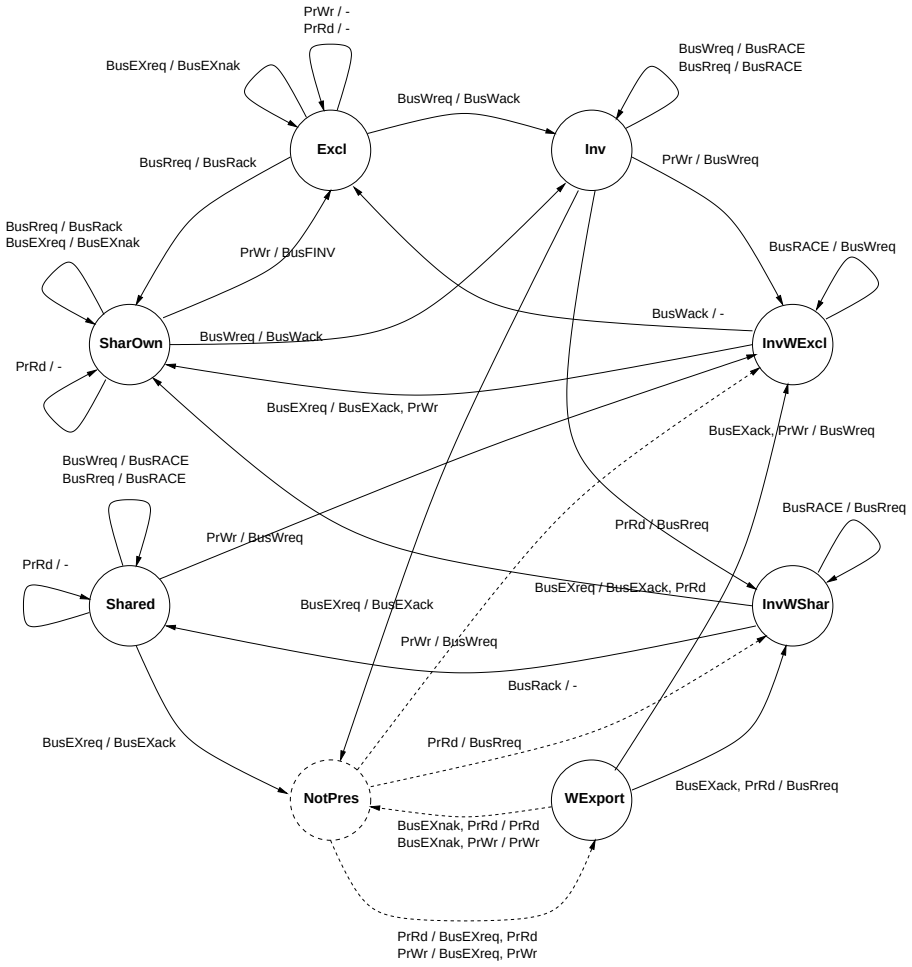

Fig. 3. Diagrama de estados de VSR-COMA. Las transiciones están etiquetadas con el formato Operacion-o-Evento-Recibido / Evento-Generado.

dos últimos casos, se hace necesario desalojar el bloque, ya que el nodo es el propietario de todos los bloques presentes en ese conjunto de la memoria de atracción. Por lo tanto, se genera un evento BusEXReq y la operación de memoria solicitada se vuelve a dejar en la cola de operaciones pendientes, a la espera de que se resuelva el desalojo antes de iniciar su procesamiento. Dicha cola tiene una longitud máxima de 1, ya que el modelo de consistencia utilizado es el modelo secuencial.

Cuando el bloque correspondiente haya sido reemplazado, el marco de bloque respectivo pasará al estado InvWShar o InvWExcl, según corresponda. Si la solicitud de reemplazo ha sido rechazada, se vuelve a examinar el estado de la memoria de atracción, ya que pudo haberse intercalado una transaccion que haya modificado dicho estado.

V. Estrategias de reemplazo

Una característica importante de VSR-COMA que no aparece reflejada en el diagrama de transición de estados es la selección del nodo destino de una operación de desalojo. Dado que cada nodo dispone de toda la información existente para poder elegir el nodo destino, la estrategia de reemplazo puede desligarse del protocolo en sí. Esto posibilita la utilización de diferentes algoritmos de selección de nodo destino sin necesidad de modificar el protocolo de red.

A continuación describiremos brevemente la estrategia que utiliza VSR-COMA para la selección del nodo destino. El algoritmo es el siguiente:

1. Si el bloque a desalojar se encuentra en estado SharOwn, se busca un nodo que posea el bloque a desalojar en estado Shared. Si existen varios, se selecciona el nodo que posea un menor número de bloques en propiedad en ese conjunto.
2. En caso contrario, se busca un nodo que posea el bloque a desalojar en estado Inv. Si existen varios, se selecciona el nodo que posea un menor número de bloques en propiedad en ese

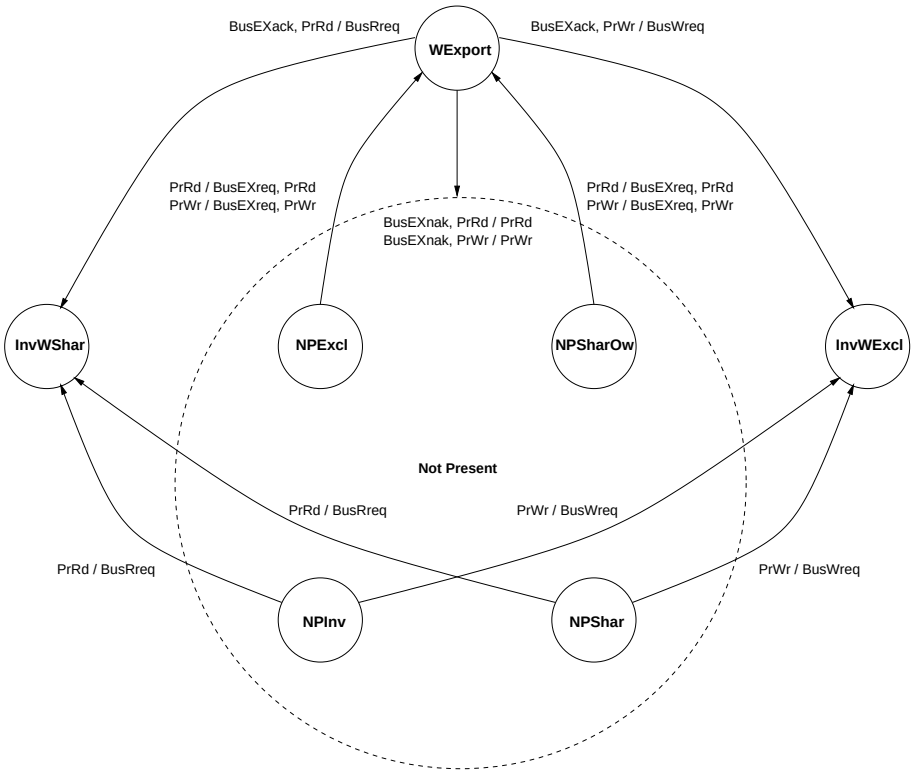

Fig. 4. Detalle del diagrama de estados de la figura 3, para el caso de que el bloque no esté presente en la memoria de atracción.

conjunto.

3. En caso contrario, se busca un nodo que posea *cualquier* bloque en estado Inv en ese conjunto. Si existen varios, se selecciona el que posea un menor número de bloques en propiedad en ese conjunto.
4. En caso contrario, se busca un nodo que posea algún bloque en estado Shared en ese conjunto. Si existen varios, se selecciona el que posea un menor número de bloques en propiedad en ese conjunto.

El paso 1 del algoritmo es la solución más lógica si el bloque está en estado SharOwn, ya que existe la posibilidad de que el bloque esté siendo utilizado en otros nodos. El paso 2 permite desalojar el bloque hacia un nodo que lo haya utilizado en un pasado reciente, lo que es de esperar que disminuya el número de faltas caché en el futuro. El paso 3 permite desalojar el bloque hacia un nodo que no necesite invalidar ningun bloque para recibirlo, al objeto también de reducir el número de faltas futuras, cuando el nodo intente recuperar una copia del bloque que tuvo que invalidar para recibir al bloque desalojado. Finalmente, el paso 4 obliga a enviar el bloque a un nodo con bloques en estado Shared, es decir, copias limpias de las cuales el nodo destino no es propietario, y que por lo tanto pueden eliminarse sin generar tráfico en la red. Dado que la presión de memoria [10] en los sistemas COMA es menor que el 100 % (para permitir la replicación de bloques), se garantiza que en alguna memoria de atracción existe al menos un bloque en estado Shared capaz de admitir el bloque.

Dado que la información de reemplazo se actualiza a través del procesamiento de los mensajes intercambiados por el bus, puede darse la circunstancia de que esa información no esté actualizada en el momento de tomar la decisión sobre el nodo destino, al existir eventos pendientes de tratamiento. Por lo tanto, cabe la posibilidad de que el bloque sea enviado a un nodo que no esté en disposición de aceptarlo, ya que para ello debería desalojar a su vez otro bloque. En este caso, el nodo destino responde a la solicitud con un evento BusEXnak. El nodo propietario deberá entonces reiniciar el proceso de selección. Es interesante destacar que el bus permite definir una secuencia única de eventos, visible por todos los nodos. Por lo tanto, cuando el nodo propietario reciba la respuesta negativa de reemplazo, ya dispondrá de la información de reemplazo actualizada, al haber tenido que procesar todos los eventos anteriores que han modificado la situación, pudiendo así volver a intentar el desalojo.

VI. Conclusiones

El desarrollo de VSR-COMA ha permitido confirmar la viabilidad de un protocolo COMA de gestión distribuida para sistemas multicomputador unidos por bus común. Este protocolo incorpora todas las características necesarias para la gestión de memoria en un sistema COMA: asociatividad por conjuntos, mecanismos avanzados de reemplazo y operaciones de memoria de tipo "Test And Set", que facilitan la sincronización entre procesos concurrentes.

La separación entre la gestión del desalojo de bloques y la estrategia de reemplazo otorga al VSR-COMA una gran flexibilidad, permitiendo la implementación y evaluación de diferentes estrategias de reemplazo sin necesidad de modificar el protocolo, lo que no es posible con los protocolos COMA actuales.

Agradecimientos

Los autores desean agradecer al Dr. Josep Torrellas, de la Universidad de Illinois en Urbana-Champaign (USA), así como a los miembros del grupo de investigación por él dirigido (IA-COMA), sus comentarios y sugerencias acerca del presente trabajo.

Referencias

[1] B. Sahelices, *COMA-BC: una Arquitectura de Memoria Slo Cache en Bus Comn no Jerrquica*. PhD thesis, Departamento de Informática, Universidad de Valladolid, 1998.

[2] B. Sahelices Fernández, J. Illescas, and L. Alonso Romero, "Coma-bc: A cache only memory architecture multicomputer for non-hierarchical common bus networks," in *Proceedings of the 6th Euromicro Workshop on Parallel and Distributed Processing*, pp. 502–508, 1998.

[3] B. Sahelices Fernández, A. De Dios Hernández, and J. J. Cabana, "A coma multicomputer system for one-line common bus," in *Proceedings of the 1st IASTED International Conference on Parallel and Distributed Systems, Euro-PDS'97*, pp. 38–41, 1997.

[4] E. Hagersten, A. Landin, and S. Haridi, "Ddm - a cache-only memory architecture," *IEEE Computer*, pp. 44–54, September 1992.

[5] S. Cameron Woo, M. Ohara, E. Torrie, J. Pal Singh, and A. Gupta, "The splash-2 programs: Characterization and metodological considerations," *Proceedings of the 22nd Anual International Symposium on Computer Architecture*, pp. 24–36, June 1995.

[6] T. Joe, *COMA-F: A Non-hierarchical Cache Only Memory Architecture*. PhD thesis, Department of Electrical Engineering, Stanford University, 1995.

[7] S. Cho, J. Kong, and G. Lee, "Coherence and replacement protocol of dice - a bus based coma multiproccesor," *Journal of Parallel and Distributed Computing*, pp. 14–32, April 1999.

[8] G. Lee, B. QuattleBaum, S. Cho, and L. Kinney, "Global bus design of a bus-based coma multiprocessor dice," in *Proceedings of International Conference on Computer Design*, October 1996.

[9] M. D. Hill, "Multiprocessors should support simple memory-consistency models," *IEEE Computer*, pp. 28–34, August 1998.

[10] A. Landin and F. Dahlgren, "Bus-based coma - reducing traffic in shared-bus multiprocessors," in *Second IEEE Conference on High Performance Computer Architectures*, February 1996.