Transmisión de Datos Multimedia en Entornos LAN: Encaminador Multimedia MMR

María Blanca Caminero Herráez

Departamento de Informática
Escuela Politécnica Superior de Albacete
Universidad de Castilla-La Mancha
Campus Universitario
02071-ALBACETE
e-mail: blanca@info-ab.uclm.es

Resumen. El creciente incremento del rango de aplicaciones que hacen uso de transmisiones multimedia producido en los últimos años, ha impuesto unos nuevos requisitos de funcionamiento sobre los elementos de conmutación. El factor clave es la capacidad de garantizar la calidad de servicio (QoS) a los flujos multimedia. En entornos de área amplia, ATM es la solución más ampliamente empleada. Sin embargo, para entornos más reducidos (redes de área local sobre todo) tal solución no existe. El proyecto del *Encaminador Multimedia MMR* tiene como objetivo el diseño de un elemento de conmutación y encaminamiento, capaz de satisfacer las garantías de calidad de servicio requeridas por las aplicaciones multimedia, sin dejar de lado al tráfico tradicional de tipo ***best-effort*** que sigue conviviendo con estas nuevas aplicaciones. Además, se persigue obtener una implementación rápida y compacta, realizable en un único chip, capaz de garantizar la calidad de servicio adecuada, y que proporcione latencias del orden de las obtenidas con los encaminadores ***cut-through***, empleados en multicomputadores y redes de sistema y locales (SAN y LAN).

1 Introducción

En los últimos años, el precio de los elementos de computación e interconexión ha disminuido rápidamente. Esto provoca una explosión tanto en el uso de aplicaciones digitales electrónicas, como en su variedad. Los sistemas de bajo coste soportan no solo las aplicaciones tradicionales de procesadores de texto, hojas de cálculo, correo electrónico y bases de datos, sino también nuevas aplicaciones en red, como videoconferencia, correo electrónico multimedia, VRML y otras muchas aplicaciones basadas en multimedia.

Existen otras aplicaciones, como la generación en tiempo real de escenarios virtuales, o la reconstrucción de imágenes 3-D en tiempo real, que necesitan una potencia de cálculo mayor de la disponible en sistemas monoprocesador. Por tanto, necesitan emplear computadores paralelos. Sin embargo, aunque existiese la potencia de cálculo necesaria para estas aplicaciones dentro de un sistema monoprocesador, a

menudo los propios datos son distribuidos por naturaleza. Esto sucede en las bases de datos distribuidas, videoconferencia, acceso a servidores web, y aplicaciones de vídeo bajo demanda. Además, todas estas aplicaciones necesitan elementos de interconexión de alta velocidad, puesto que su número de usuarios crece continuamente.

La necesidad de un mayor ancho de banda es especialmente importante en entornos locales. Es en estos entornos donde se suelen ejecutar aplicaciones como reuniones virtuales, acceso a bases de datos de imágenes, y juegos 3-D con varios usuarios. Estas aplicaciones necesitan un gran ancho de banda, para conseguir ser interactivas, así como otras restricciones relativas a la temporización. Y ese ancho de banda debe ser proporcionado a través de una red compartida por gran número de aplicaciones.

Tradicionalmente, los avances en la tecnología de interconexión de redes de multicomputadores se han centrado en la obtención de baja latencia y alta productividad para el tráfico *best-effort*. Por tanto, estas redes no están preparadas para ofrecer garantías de servicio a múltiples aplicaciones simultáneamente. Lo mismo puede decirse sobre las redes de altas prestaciones aparecidas en los últimos años [1,21], que emplean la misma tecnología de las redes de multicomputadores.

En resumen, el abanico de nuevas aplicaciones surgidas recientemente, exhibe diferentes mezclas de cálculo, comunicaciones y coordinación distribuida, de manera que imponen tanto una mayor demanda de prestaciones de la red (un cambio cuantitativo), como unos determinados requerimientos de servicio (un cambio cualitativo) [3].

Los requisitos de calidad de servicio (*Quality Of Service*, QoS) impuestos por la mayoría de las aplicaciones multimedia necesitan de mecanismos específicos para proporcionarles un soporte adecuado. La mayoría de encaminadores empleados en multicomputadores y redes de estaciones de trabajo (NOWs) emplean conmutación *wormhole* o *virtual cut-through* (VCT). Estas técnicas son capaces de proporcionar bajas latencias a los mensajes en media, pero no proporcionan ningún soporte a la QoS.

Existen algunas propuestas para soportar comunicaciones en tiempo real en redes de multicomputadores, a la vez que proporcionan comunicación con baja latencia para tráfico *best-effort* [20,15]. Sin embargo, en estas propuestas se asume que los mensajes de tiempo real son muy cortos y solo consumen una pequeña parte del ancho de banda. Por tanto, no son apropiados para proporcionar QoS a largos flujos de datos que pueden solicitar más ancho de banda del disponible en un cierto enlace.

La única técnica de conmutación diseñada para proporcionar QoS, soportar un gran número de conexiones, y alcanzar una latencia relativamente baja es ATM [19]. Sin embargo, ATM es fruto de un compromiso entre muchos intereses enfrentados, y está adaptada principalmente para redes de área amplia (WANs). Las restricciones físicas de las redes WAN son muy diferentes de las que imponen las redes LAN. Por tanto, podemos encontrar distintas soluciones arquitectónicas, más eficientes, para encaminadores LAN. El objetivo clave consiste en poder proporcionar garantías de calidad de servicio (QoS) con latencias comparables a las de los encaminadores *cut-through* empleados en multiprocesadores.

En conclusión, no parece existir ninguna arquitectura de encaminador que proporcione un soporte eficiente para los requerimientos de las aplicaciones multimedia en entornos de área local.

2 Requerimientos de las aplicaciones

El tráfico generado por las aplicaciones multimedia exhibe un comportamiento diferente del de otras aplicaciones tradicionales, como las de computación paralela, de tiempo real, servidores remotos, etc. A continuación resumimos sus características más distintivas:

- *Flujos de datos muy largos.* Las transmisiones de audio y vídeo suelen estar compuestas de flujos de datos muy largos, que duran desde unos pocos segundos hasta horas.
- *Amplio rango de requisitos de ancho de banda.* Por ejemplo, las transmisiones de vídeo requieren mucho más ancho de banda que las de audio. Además, estos requisitos pueden no ser constantes a lo largo de toda la duración de la conexión.
- *Gran número de conexiones.* En cada nodo de la red, pueden estar ejecutándose simultáneamente varias aplicaciones multimedia. Además, cada una de estas aplicaciones puede requerir la transmisión de varios flujos de datos. Como los requisitos de QoS pueden satisfacerse mejor si se emplea un esquema orientado a la conexión, tenemos que se debería ser capaz de soportar un gran número de conexiones de forma concurrente. Para un entorno LAN, este número estará en el rango de entre unos cientos o unos pocos miles de conexiones.
- *Sensibilidad al jitter.* El tráfico multimedia es muy sensible a la variación de la latencia o *jitter*. Por ejemplo, las tramas de vídeo deben llegar a intervalos regulares. Si una trama llega demasiado pronto debe ser almacenada en el nodo destino, incrementando por tanto la necesidad de espacio de almacenamiento. Por el contrario, si llega demasiado tarde, no puede ser visualizada y debe descartarse. En este caso, se muestra de nuevo la trama anterior, y se reduce la calidad del vídeo.
- *Tolerancia a la latencia.* Sobre todo, en el establecimiento de la conexión.
- *Mensajes de control cortos.* Muchas aplicaciones multimedia requieren transmitir cortos mensajes de control además de los largos flujos de datos. Este tráfico no es sensible al *jitter*, pero las prestaciones de las aplicaciones multimedia pueden ser sensibles a la latencia de los mensajes de control. En particular, estos mensajes no deberían verse bloqueados por los flujos de datos.

Algunas aplicaciones generan tráfico CBR, pues el proceso de compresión y descompresión de la información puede reducir inaceptablemente la calidad de la señal. Por otro lado, también circulará por la red tráfico *best-effort* generado por aplicaciones tradicionales. En este caso, el tráfico *best-effort* no debe interferir en las garantías de QoS del tráfico multimedia. Sin embargo, el ancho de banda debe adjudicarse con cuidado para evitar que el tráfico *best-effort* se quede sin recursos.

Con la arquitectura de conmutador propuesta en [8] y descrita a continuación, se pretenden conseguir los siguientes objetivos:

1. Soportar un gran número de conexiones multimedia, satisfaciendo sus necesidades de QoS.

2. Adjudicar el ancho de banda sobrante a tráfico ***best-effort***, de manera que su latencia media sea mínima.
3. Maximizar la utilización de los enlaces cuando la red alcanza la saturación.

Para lograr esos objetivos, hay que seleccionar cuidadosamente varios parámetros de diseño de la red, tanto cualitativos como cuantitativos, que se muestran en la Tabla 1. La topología de la red viene determinada por la distribución física de los nodos. Por tanto, asumimos que tanto el tamaño de la red como el patrón de interconexión de los nodos vienen definidos por el usuario. En el resto del capítulo, analizaremos la selección de los parámetros cualitativos, para que sean capaces de satisfacer los requerimientos de las aplicaciones, a la vez que maximizando la utilización de los enlaces, y minimizando la complejidad del encaminador. Para cada parámetro, veremos qué opciones existen y cuál es la elegida para el encaminador.

Tabla 1. Parámetros a considerar en el diseño del encaminador multimedia

Parámetros cualitativos	Parámetros cuantitativos
Topología de la red	*Tamaño de la red*
Técnica de conmutación	Ancho de banda de los enlaces
Algoritmos de planificación del enlace	Número de puertos del encaminador
y del conmutador	Frecuencia de reloj del encaminador
Organización de los ***buffers***	Tamaño de los ***buffers***
Gestión de los ***buffers***	Número de canales virtuales
Organización del ***crossbar***	

3 Arquitectura del conmutador

3.1 Técnica de conmutación

Para poder garantizar valores de ***jitter*** acotados a las conexiones multimedia, es necesario emplear un esquema *orientado a la conexión*. Las técnicas de conmutación que reservan recursos sobre la marcha, como ***wormhole*** (WH) o ***virtual cut-through*** (VCT) [9], no son apropiadas, porque el tiempo que un mensaje puede permanecer bloqueado esperando la liberación de un recurso ocupado, no está acotado. Por otro lado, tanto los mensajes de control como el tráfico ***best-effort*** se pueden beneficiar de técnicas que proporcionan baja latencia, como la conmutación WH o VCT. En cambio, en este caso, un esquema orientado a la conexión no sería apropiado, pues la alta latencia que conlleva la fase de establecimiento de la conexión puede ser un orden de magnitud superior a la latencia obtenida con conmutación WH.

Por tanto, no existe ninguna técnica de conmutación capaz de satisfacer a la vez los requisitos de todas las clases de tráfico. Un buen compromiso consistiría en utilizar una técnica híbrida [7]. Los flujos de datos se transmitirían mediante alguna técnica orientada a la conexión (conmutación de circuitos, ATM, PCS), de manera que

primero se establece una conexión entre origen y destino, y por ella se transmiten los datos. Los mensajes de control y el tráfico *best-effort* emplearían WH o VCT.

En el diseño del encaminador multimedia se ha optado por el empleo de *pipelined circuit switching* (PCS) [6], por su simplicidad, porque puede combinarse fácilmente con WH o VCT, utiliza control de flujo para prevenir la pérdida de datos, requiere poco espacio de *buffers* asociado a cada canal virtual, y la sobrecarga introducida por la información de control es mucho menor que en ATM. Por estas razones, PCS parece más adecuado que ATM para integrar un encaminador en un solo chip, y proporcionar soporte para un entorno LAN.

En cuanto a la técnica de conmutación seleccionada para transmitir los mensajes de control, se ha preferido *virtual cut-through* sobre *wormhole* porque dichos mensajes de control suelen ser pequeños. WH y VCT se comportan de forma idéntica en ausencia de contención. Sin embargo, cuando aparece la congestión, VCT ofrece mejores prestaciones porque los paquetes bloqueados se extraen de la red. Esto beneficia al tráfico multimedia porque reducirá la interferencia con los flujos de datos. El principal inconveniente respecto a WH es la necesidad de *buffers* lo bastante grandes como para poder almacenar paquetes enteros. Como en nuestro caso los paquetes con mensajes de control serán en su mayoría pequeños, esto no supone un problema. Cuando haga falta transmitir mensajes largos de tráfico *best-effort*, dichos mensajes pueden dividirse en pequeños paquetes de tamaño fijo. Esto no introduce ninguna sobrecarga adicional, puesto que los modernos niveles software de mensajes, como FM [18], ya parten los mensajes en trozos de tamaño fijo, con el objeto de segmentar su transmisión a través del interfaz de red, e incrementar así las prestaciones obtenidas.

3.2 Organización de los *buffers*

Cuando se establece una conexión, se reserva un canal virtual en cada enlace perteneciente al camino entre origen y destino. Con el objeto de poder soportar un gran número de conexiones simultáneas, los *buffers* de cada enlace deben organizarse como un gran conjunto de canales virtuales. Cuando los canales físicos se multiplexan en un número elevado de canales virtuales, el retardo del encaminador se incrementa drásticamente. Esto se debe a los retardos introducidos por el multiplexor y el controlador de los canales virtuales. Además, el área de silicio dedicada a *buffers* se incrementa rápidamente con el número de canales virtuales. Más aún, los *crossbar* completamente demultiplexados llegan a ser prohibitivos al aumentar el número de canales virtuales. Por tanto, hace falta una organización distinta de los *buffers* si queremos soportar un gran número de canales virtuales.

Si consideramos sólo el espacio dedicado a *buffers*, los *buffers* centrales son la mejor opción, puesto que todos los enlaces comparten esos *buffers*. Pero esta organización tiene algunos inconvenientes:

1. Introduce cuellos de botella, y limita en gran medida el número de puertos del encaminador.
2. El control de flujo se hace más complejo.

3. Aunque sería posible combinar los ***buffers*** centrales con otros a la entrada, esto sólo es eficiente cuando la mayor parte de los datos puede ser transmitida de inmediato, sin necesidad de almacenamiento en los ***buffers*** centrales. En nuestro caso, esto no es posible, porque habrá ocasiones en que debamos retrasar la transmisión de algunos paquetes, en favor de otros datos de mayor prioridad.

Por tanto, una organización del encaminador basada en ***buffers*** centrales no nos sirve.

Los ***buffers*** a la salida son normalmente más eficientes que a la entrada porque eliminan el bloqueo de cabeza de línea (***HOL blocking***). Sin embargo, los paquetes entrantes deben ser procesados inmediatamente para decidir el ***buffers*** de salida donde deben almacenarse. El problema principal que tiene esta organización de ***buffers*** es que existe contención entre los paquetes entrantes por los ***buffers*** de salida. Cuando llegan varias ráfagas de datos por distintos enlaces de entrada, y van destinadas al mismo enlace de salida, aparece la contención, y deben descartarse algunos paquetes. Esto ocurre incluso si todos los datos son de alta prioridad. Este problema puede solucionarse si se emplea una organización de ***buffers*** multipuerto, muy cara y compleja, o bien utilizando control de flujo para evitar la pérdida de paquetes. Sin embargo, el control de flujo es muy complicado cuando solo se emplean ***buffers*** a la salida.

El enfoque tradicional utilizado en los encaminadores ***wormhole*** utiliza ***buffers*** *situados en los enlaces de entrada*. Esta organización simplifica considerablemente el control de flujo y es capaz de conseguir latencias bajas para cargas bajas y medias. Sin embargo, esta organización tiene dos problemas graves cuando se emplea en un entorno LAN:

1. Sufre de bloqueo de cabeza de línea cuando la red está fuertemente cargada. Este problema puede resolverse utilizando canales virtuales independientes para cada conexión, de manera que el tráfico de alta prioridad pueda adelantar al tráfico de baja prioridad.
2. El segundo problema es que el retardo de ida y vuelta de la información de control de flujo implica el uso de grandes ***buffers*** con el objeto de prevenir pérdidas de datos cuando los enlaces son largos [1]. Una posible solución consiste en emplear *control de flujo basado en créditos* [16] en lugar del comúnmente empleado ***stop-and-go***.

El *control de flujo basado en créditos* consiste en lo siguiente. El emisor dispone de un cierto número de créditos, que se corresponde con el número de ***buffers*** libres en el receptor. Este número de créditos se decrementa cada vez que se transmite un flit. El receptor envía un crédito al emisor cada vez que consigue enviar un flit. Esta técnica puede ser menos eficiente que ***stop-and-go*** porque por cada flit que se transmite, se requiere el envío de un crédito en la dirección opuesta. Para reducir esta sobrecarga de control, se emplean tamaños de flit grandes (~ 1024 bits). Además, el uso de flits grandes nos permite basar el diseño del encaminador en un ***crossbar*** multiplexado, así como utilizar algoritmos complejos de planificación de los enlaces. Esto se verá en las siguientes secciones.

Sin embargo, el uso de flits de gran tamaño también plantea nuevos problemas. Los flits grandes incrementan la latencia y los requerimientos de espacio de

almacenamiento. La latencia puede reducirse si se segmenta la transmisión de los flits con una granularidad menor, por ejemplo, a nivel de phit.[1]

Los canales virtuales se han organizado tradicionalmente como un conjunto de colas unidas por un multiplexor. Esta organización ya no es viable si se tiene un gran número de canales virtuales. En el encaminador multimedia, los canales virtuales se organizan como un conjunto de módulos de memoria RAM entrelazados según un esquema simple. Cada módulo de memoria tiene un puerto de entrada y otro de salida. Cada flit se almacena como un conjunto de palabras consecutivas en memoria. Los flits que pertenecen al mismo canal virtual se almacenan en conjuntos de posiciones de memoria adyacentes. Antes de transmitir cada flit, se transmite una palabra de control indicando el número de canal virtual al cuál pertenece el flit. Las palabras de control se distinguen de las de datos mediante un bit de control, como en [1]. El número de canal virtual se emplea para direccionar el *buffer* correspondiente dentro de la memoria RAM. Después de almacenar cada palabra, la dirección se incrementa automáticamente. El entrelazado se lleva a cabo mediante los bits menos significativos. El número de módulos de memoria y el tamaño de los flits se seleccionan de manera que concuerden con el tiempo de acceso a memoria en ciclos de reloj.

La organización de *buffers* propuesta se muestra en la Figura 1. La dirección de lectura que se necesita para recuperar los flits la proporciona el planificador de enlaces. Debe señalarse que una palabra puede ser leída tan pronto como sea

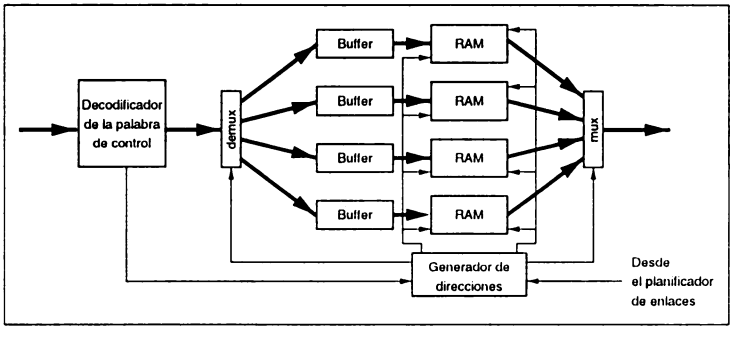

Figura 1. Organización de los *buffers* en el MMR

almacenada, sin tener que esperar a la recepción del flit completo. De esta forma, se puede segmentar la transmisión al nivel de palabra. El tiempo de acceso a memoria incrementa la latencia en unos pocos ciclos de reloj, pero las aplicaciones multimedia normalmente toleran bien la latencia.

La arquitectura del encaminador multimedia MMR se muestra en la Figura 2. Los *buffers* que soportan los canales virtuales se organizan de la forma descrita en párrafos anteriores. Hay pequeños *buffers* de tamaño de un phit que van almacenando los flits que llegan. Sin pérdida de generalidad, asumiremos que el tamaño de phit es igual al tamaño de palabra. En caso contrario, habría un deserializador encargado de empaquetar varios phits en una sola palabra. Los phits de control se decodifican,

[1] Un *phit* es la cantidad de información que puede transmitirse en paralelo a través de un enlace.

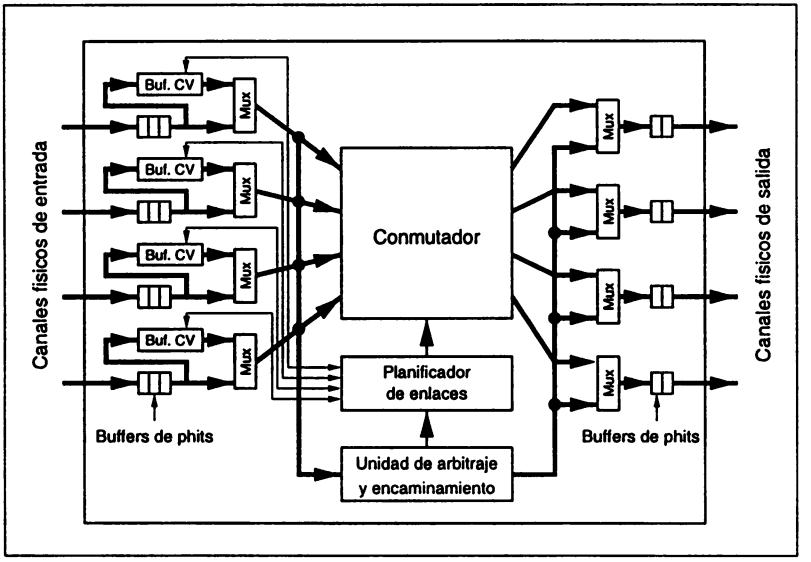

Figura 2. Arquitectura del encaminador multimedia MMR

ejecutando la acción correspondiente: incrementar un contador de créditos, seleccionar la dirección inicial de memoria para almacenar un flit que llega, etc. Los *buffers* de phits son lo bastante profundos como para ser capaces de almacenar todos los phits que puedan llegar durante la decodificación de un phit de control (esto es, durante el cálculo de la dirección de memoria donde deben de almacenarse esos phits).

Los *buffers* de phits también permiten el encaminamiento con baja latencia de mensajes cortos mediante VCT, en caso de que no exista contención (es decir, el enlace de salida solicitado está libre). De manera similar, los *buffers* de phit permiten un procesamiento rápido de las sondas y reconocimientos cuando se establece una conexión.

3.3 Organización del conmutador

La mayoría de conmutadores se implementan como *crossbars*. Existen varias organizaciones posibles: multiplexados, parcialmente multiplexados y completamente demultiplexados [5]. La organización basada en un *crossbar* completamente demultiplexado llega a ser prohibitiva cuando tenemos un número elevado de canales virtuales. Incluso para un número relativamente pequeño de canales virtuales, se suelen emplear *crossbars* multiplexados. Por tanto, el encaminador MMR utilizará un *crossbar multiplexado*. En esta organización, el conmutador va montado sobre un *crossbar* con tantos puertos como canales físicos.

El principal inconveniente de un *crossbar* multiplexado consiste en que se necesita arbitrar cada vez que un canal de entrada cambia de un canal virtual a otro. El arbitraje se requiere en el lado de la entrada para seleccionar un canal virtual de cada

canal físico de entrada. También es necesario un arbitraje en el lado de la salida porque varios canales virtuales pueden solicitar el mismo enlace de salida.

Los algoritmos de arbitraje de los enlaces de salida y del conmutador se describirán en las secciones 3.10 y 3.11, respectivamente. Debe hacerse notar que el arbitraje se oculta durante la transmisión del flit anterior, puesto que los flits son lo bastante grandes. La sobrecarga introducida por la reconfiguración del *crossbar* se amortiza sobre una mayor cantidad de datos, cuando los flits son grandes.

Una característica interesante de los *crossbar* multiplexados es que no son necesarios los *buffers* a la salida. Como los puertos de salida del conmutador van conectados directamente a los enlaces de salida, los flits se transmiten directamente a través del conmutador y del correspondiente enlace de salida. Sin embargo, pueden emplearse unos *buffers* con capacidad para unos pocos phits para segmentar la transmisión de información a través del conmutador y del enlace. Por último, será necesario un serializador en el caso de que el camino de datos interno sea más ancho que los enlaces físicos externos.

3.4 Transmisión de paquetes y flits

El arbitraje es difícil de realizar si las peticiones llegan de forma asíncrona. En ese caso, las peticiones deben almacenarse y sincronizarse de manera que se dé preferencia a las peticiones de mayor prioridad, aunque lleguen más tarde. Además, cuando un enlace de entrada dado gana el arbitraje para transmitir el siguiente flit, tiene que esperar hasta que el enlace de salida correspondiente quede libre, incluso si ya ha acabado de transmitir el flit anterior. En consecuencia, se desperdicia una parte del ancho de banda. Por consiguiente, con el objeto de aprovechar completamente el ancho de banda del conmutador y de los enlaces, a la vez que simplificar el diseño del encaminador, el encaminador MMR asignará síncronamente los puertos del conmutador y los canales de salida a los canales virtuales con peticiones pendientes.

La transmisión de flits se organiza como una secuencia de ciclos de flit. Durante cada uno de éstos, todos los enlaces de entrada con flits listos comienzan transmitiendo una palabra de control con el identificador del canal virtual al que pertenece el flit. Después, transmiten síncronamente un flit por el conmutador y los enlaces de salida. Al mismo tiempo, se lleva a cabo el arbitraje para asignar los puertos del conmutador y los enlaces de salida para el siguiente ciclo de flit.

Una vez que ha finalizado la transmisión del flit en curso, se reconfigura el *crossbar*. Esta operación requiere un sólo ciclo de reloj. Durante la reconfiguración, no se transmiten flits de datos. El MMR aprovecha este ciclo para las transmisiones pendientes de sondas de encaminamiento, sondas en retroceso *backtracking* y reconocimientos para el establecimiento de conexiones. Una vez que el *crossbar* se ha reconfigurado, comienza la transmisión del siguiente flit, es decir, empieza el siguiente ciclo de flit. Aunque dentro de cada encaminador la transmisión es síncrona, diferentes encaminadores trabajan asíncronamente entre sí.

La transmisión síncrona de flits es eficiente para los flujos de datos, pero no para los mensajes de control y de tráfico *best-effort*. Estos mensajes se transmiten empleando conmutación VCT. Cuando un mensaje es mayor que un flit, se divide en paquetes de tamaño fijo. Dado el gran tamaño de los flits, se asume que el tamaño de paquete es igual a un flit. La unidad de control de flujo en VCT es un paquete.

Por tanto, las unidades de control de flujo tienen el mismo tamaño en PCS y en VCT, con lo que el diseño del encaminador se simplifica. Las cabeceras de los paquetes se encaminan tan pronto como llegan a un encaminador, a través de la unidad de arbitraje y encaminamiento. Si tanto el puerto de entrada al conmutador solicitado como el enlace de salida están libres (es decir, no están transmitiendo ningún flit durante el ciclo actual) y hay canales virtuales libres en el enlace de salida solicitado, los paquetes de control se transmiten inmediatamente porque tienen mayor prioridad que los flujos de datos. Esta transmisión no se sincroniza con los ciclos de flit.

Puesto que la transmisión de un paquete de control puede tardar más de lo que queda del ciclo de flit actual, el puerto de entrada y el enlace de salida correspondientes se consideran como ocupados durante el arbitraje para el siguiente ciclo de flit. Si el puerto de entrada y/o el enlace de salida solicitados están ocupados, pero hay canales virtuales libres en el siguiente encaminador, se reserva un canal virtual y el paquete se almacena en el ***buffer*** correspondiente del encaminador actual. De esta forma, se planificará síncronamente junto con el resto de flits correspondientes a flujos de datos. Si no hubiese canales virtuales libres en el siguiente encaminador, el paquete se bloquea y se almacena en el ***buffer*** correspondiente del encaminador actual.

Los paquetes ***best-effort*** se encaminan también tan pronto como su cabecera llega a la unidad de arbitraje y encaminamiento. Sin embargo, estos paquetes tienen menor prioridad que los flujos de datos. Si el puerto de salida solicitado tiene canales virtuales libres en el siguiente encaminador, se reserva un canal virtual. En otro caso, el paquete se bloquea. En ambos casos, el paquete se almacena en el ***buffer*** correspondiente del encaminador actual. Los paquetes ***best-effort*** se planifican síncronamente junto con los flits de los flujos de datos. Cuando se ha transmitido completamente un paquete ***best-effort*** o de control, el canal virtual que ocupaban se libera.

3.5 Unidad de arbitraje y encaminamiento

La unidad de arbitraje y encaminamiento ejecuta el *algoritmo de encaminamiento*. Este algoritmo determina el camino seguido por las sondas cuando se establece una conexión, y por los paquetes ***best-effort***. Un algoritmo completamente adaptativo puede ayudar considerablemente a encontrar un camino libre cuando la red se congestiona o cuando algunas regiones están muy cargadas. Sin embargo, el encaminamiento completamente adaptativo puede producir bloqueos. Este problema es difícil de resolver, especialmente cuando la topología de la red es irregular, y no tiene más restricciones que la de ser conexa. Sin embargo, ya existe una propuesta de un algoritmo completamente adaptativo para redes WH con topología irregular [23,24].

Este algoritmo necesita dos canales virtuales por enlace, y también es válido para conmutación VCT y PCS. Por tanto, lo emplearemos para encaminar paquetes mediante conmutación VCT.

En cambio, para PCS podemos incrementar la flexibilidad de encaminamiento si empleamos algoritmos con backtracking. Estos algoritmos evitan los bloqueos liberando los recursos que fueron previamente reservados cuando la cabecera se

bloquea debido a que no hay recursos disponibles. Al establecer las conexiones, emplearemos ***exhaustive profitable backtracking*** (EPB) [12]. Este algoritmo lleva a cabo una búsqueda exhaustiva sobre los caminos mínimos de la red hasta que encuentra un camino válido, o hasta que la sonda retrocede hasta el nodo fuente. Para evitar explorar dos veces los mismos caminos, cada canal virtual tiene asociado un registro histórico con los enlaces de salida que ya se han explorado.

El proceso de establecimiento de conexiones de PCS se verá ligeramente modificado en el encaminador MMR. En PCS una conexión se establece encaminando una sonda desde el origen al destino. Esta sonda contiene información de control así como la dirección destino [11]. En el MMR, además, la sonda lleva información acerca de los requisitos de ancho de banda. Cada vez que la sonda se encamina con éxito, se reservan algunos recursos en el encaminador correspondiente. En particular, se reservan un canal virtual y sus ***buffers*** asociados. Una vez que la sonda alcanza el nodo destino, se devuelve un reconocimiento a la fuente, siguiendo el mismo camino que la sonda, pero en sentido contrario. Cuando la fuente recibe este reconocimiento, la conexión está establecida. Si la sonda no puede reservar ningún camino, puede llegar a retroceder hasta el nodo fuente.

La unidad de arbitraje y encaminamiento mantiene las *correspondencias de canales* (***channel mappings***) entre los canales virtuales de entrada y salida para las conexiones establecidas. Los canales virtuales se especifican mediante el canal físico al que pertenecen, y el número de canal virtual dentro del mismo. Hace falta mantener correspondencias directas e inversas entre canales. Las directas se necesitan para hacer avanzar los flits de datos, mientras que las inversas son utilizadas por las sondas en retroceso y por los reconocimientos. Estas correspondencias también se emplean para propagar información de estado.

3.6 Mecanismos de reserva y reparto del ancho de banda

La red debe de proporcionar mecanismos para garantizar los requerimientos de QoS de distintas aplicaciones. El encaminador MMR soporta una abstracción de una conexión virtual llevada a cabo mediante canales virtuales. Se emplea un protocolo flexible de reserva de recursos, junto con cierto soporte para modificar las características de conexiones ya existentes, mediante palabras de control.

Los distintos tipos de QoS se garantizan mediante dos tipos de mecanismos. Por un lado, mediante el *control de admisión de conexiones*; por otro lado, existen *mecanismos de vigilancia durante la transmisión de los datos*, situados en los encaminadores, en los interfaces de red o en ambos.

El soporte para ofrecer garantías de QoS dentro del encaminador MMR se implementa como las soluciones a tres problemas básicos:

1. Reserva del ancho de banda
2. Planificación de los enlaces
3. Planificación del conmutador

El *esquema de reserva de ancho de banda* adjudica una porción adecuada del mismo para cada conexión al tiempo que ésta se establece, mientras que el algoritmo

de planificación del enlace garantiza que el ancho de banda reservado, y solo el reservado, está disponible durante la transmisión de los datos.

La *planificación de los enlaces* debe ser capaz de atender las necesidades tanto del tráfico multimedia como del *best-effort*, y tiene tres objetivos:

1. Satisfacer los requisitos de QoS del tráfico multimedia
2. Adjudicar el resto del ancho de banda a los paquetes *best-effort*
3. Maximizar la utilización del enlace

La *planificación del conmutador* trata de minimizar los conflictos internos por causa de los puertos del encaminador, entre las conexiones planificadas. Un algoritmo de planificación del conmutador pobre ofrece una baja utilización del ancho de banda a través del encaminador, y por tanto, provoca una baja utilización de los enlaces.

Las estrategias de planificación del conmutador y de los enlaces deben actuar de una manera estrechamente acoplada para utilizar el ancho de banda de la manera más efectiva posible. El mayor desafío en un encaminador MMR de un solo chip es que estas estrategias deben tener implementaciones rápidas y compactas. Existen varias opciones para la reserva de ancho de banda, y la planificación tanto del conmutador como de los enlaces. Antes de discutirlas, analizaremos los requerimientos de ancho de banda que presentan distintas aplicaciones.

3.7 Características de los flujos de datos

Los flujos de datos se clasifican normalmente como de *caudal constante* (CBR, *constant bit rate*) o *caudal variable* (VBR, *variable bit rate*). Las conexiones CBR son relativamente fáciles de manejar. Los requisitos de ancho de banda pueden ser satisfechos si se reserva el ancho de banda necesario al establecer la conexión. Las conexiones VBR son más difíciles de manejar. Cuando se establece una conexión VBR, una aplicación puede tener un conocimiento exacto o aproximado de los requerimientos de ancho de banda medios y máximos. Por ejemplo, es posible calcular el ancho de banda medio y máximo que necesita una película comprimida que está almacenada, pero no lo es si se trata de una transmisión en tiempo real de vídeo comprimido.

Los flujos de datos que necesitan QoS pueden experimentar un *jitter* excesivo durante su transmisión. Los fragmentos que llegan a su destino demasiado tarde, normalmente son descartados. Pero en algunas aplicaciones, no toda la información es igual de importante. Por ejemplo, en el caso del vídeo MPEG-2 [13], las tramas de tipo I son más importantes que las de tipos B o P, porque estas últimas necesitan información codificada en las primeras. En este caso, es posible establecer prioridades distintas para tramas distintas.

Otra posibilidad consiste en asignar prioridades distintas a diferentes trozos de información dentro de cada trama [4]. Las prioridades pueden emplearse a la hora de reservar ancho de banda o de planificar los enlaces.

Por último, hay conexiones "fiables" y "no fiables". Las *conexiones fiables* son aquellas que no intentan utilizar más ancho de banda del que han reservado. En otro caso, se trata de una *conexión no fiable*. Entonces, se necesitan comprobaciones sobre la marcha (*mecanismos de vigilancia*) para garantizar que una conexión no supera su

parte de ancho de banda de la red, de manera que la red pueda seguir ofreciendo sus niveles de QoS al resto de conexiones.

3.8 Detalles de implementación

El ancho de banda de los enlaces y de los puertos del conmutador se divide en *ciclos de flit*. Estos ciclos se agrupan en *"vueltas"* o ***frames***. El número de ciclos de flit que hay en una vuelta es un múltiplo entero K ($K > 1$) del número de canales virtuales que tiene cada enlace físico. El ancho de banda de una conexión se reserva como un número entero de ciclos de flit. Estos ciclos de flit serán asignados a la conexión durante cada vuelta, según los vaya pidiendo. Por tanto, un mayor valor de K proporciona una mayor flexibilidad para reservar ancho de banda. Sin embargo, puede incrementar el ***jitter***, porque las vueltas tardan más en acabar. Por tanto, el valor que se selecciona para K es un compromiso entre flexibilidad y ***jitter***. Otro factor a tener en cuenta es que a mayor K hará falta mayor espacio para almacenar los contadores de ciclos ocupados por cada conexión, lo que puede dificultar la implementación monochip del MMR.

La estructura de datos empleada para soportar decisiones rápidas de planificación consiste en un conjunto de *vectores de bits de estado*, donde cada bit está asociado con un solo canal virtual. Estos vectores de bit proporcionan información sobre diferentes condiciones para todos los canales virtuales del encaminador. Por ejemplo, podemos considerar los siguientes vectores de estado: flits disponibles, ***buffer*** de entrada lleno, créditos disponibles, petición de servicio CBR, ancho de banda CBR servido, petición de servicio VBR, ancho de banda VBR servido, conexión VBR de prioridad alta, conexión VBR de prioridad media y conexión VBR de prioridad baja (aparecen subrayados los que utilizamos por el momento). Los bits de estos vectores se van actualizando cuando cambia el estado del canal virtual correspondiente. Por ejemplo, si en un canal virtual dado no hay créditos, y llega uno procedente del siguiente encaminador, el bit correspondiente del vector "créditos disponibles" se actualiza.

Los bits de estado pueden ir asociados con los canales virtuales a la entrada o a la salida, dependiendo de la implementación del algoritmo de planificación del conmutador (sección 3.11). Puede que hagan falta las correspondencias ente canales para poder actualizar los bits de estado. Por ejemplo, en el caso de que llegue un nuevo flit y si los bits de estado están asociados a los canales de salida, hay que buscar el canal virtual de salida que corresponde con el canal virtual de entrada por el que ha llegado el flit.

Estos vectores de bits de estado pueden simplificar de forma considerable la ejecución concurrente del algoritmo de planificación de los enlaces a través de los puertos. Por ejemplo, suponiendo que los vectores de bits de estado están asociados con los canales virtuales de entrada, cada enlace físico de entrada puede determinar rápidamente el conjunto de canales virtuales de entrada que hay en ese enlace con flits y créditos disponibles, servicio CBR pedido y no servido del todo en la vuelta actual, simplemente calculando el "Y" lógico de los vectores correspondientes. De manera análoga, se puede calcular rápidamente el conjunto de canales que satisfacen otras condiciones.

3.9 Reserva del ancho de banda

Cuando se está estableciendo una conexión en el encaminador MMR, el nodo fuente genera una sonda de encaminamiento que lleva información sobre los requerimientos de ancho de banda. La sonda intenta establecer una conexión construyendo un camino desde la fuente hacia el destino, reservando a su paso el ancho de banda en los enlaces, y los *buffers* correspondientes. Si la reserva de recursos se lleva a cabo con éxito, la conexión queda establecida, y se admite la petición. Si los recursos necesarios no están disponibles en algún punto del camino, la conexión falla, y se liberan todos los recursos reservados hasta ese momento. Si se utiliza una búsqueda con vuelta atrás *backtrack*, se pueden seguir otros caminos a través de la red.

Para *conexiones CBR*, este mecanismo es sencillo de implementar. La sonda debe transportar información sobre el ancho de banda que necesita la conexión c, medido en ciclos de flit por vuelta: lo llamaremos BW_m^c. Cada enlace de salida L necesita tener asociado un registro $resBW_L$ que lleve la cuenta del número total de ciclos de flit por vuelta que han sido reservados hasta el momento. Este registro se incrementa en BW_m^c cuando se admite una conexión, y se decrementa en esa misma cantidad cuando dicha conexión se libera.

Una conexión CBR c sólo puede ser admitida si se cumple la siguiente condición:

$$resBW_L + BW_m^c < numero\ de\ ciclos\ de\ flit\ en\ una\ vuelta$$

Es posible reservar algo de ancho de banda para el tráfico *best-effort* con el fin de evitar que se quede sin atender.

Para las *conexiones VBR*, el problema es más difícil de resolver. Algunas conexiones VBR envían un flujo de datos continuo, pero la cantidad de datos por unidad de tiempo varía con el tiempo. Este podría ser el caso de flujos de audio comprimido. Otras conexiones VBR envían ráfagas de datos de tamaño variable, pero no envían datos entre ráfagas. Este sería el caso de las tramas de vídeo comprimido. Para poder tratar con los diferentes requisitos de las conexiones, una sonda que establezca una conexión VBR v llevará los anchos de banda permanente BW_m^v y pico BW_p^v para esa conexión. En el caso de que se trate de un flujo continuo de datos, el ancho de banda permanente puede hacerse igual al ancho de banda medio que requiere esa conexión. En el caso del tráfico a ráfagas, el ancho de banda permanente puede ponerse a cero. El significado del ancho de banda pico es evidente en ambos casos. Estos valores pueden ser estimados, según el conocimiento de que se disponga del comportamiento de la conexión. Para soportar la reserva de ancho de banda para conexiones VBR, cada enlace de salida necesita dos registros asociados: $resBW_L$ mantiene el número de ciclos de flit por vuelta que han sido reservados (esto es, igual que en el caso de conexiones CBR), y $resBWp_L$ almacena el ancho de banda pico total requerido por todas las conexiones que utilizan el enlace L. Estos contadores se incrementan y decrementan respectivamente con los valores de BW_m^v y BWp_m^v, cuando las conexiones se establecen y se eliminan.

Una conexión VBR v solo se aceptará si se cumplen dos condiciones:

$$resBW_L + BW_m^v < numero\ de\ ciclos\ de\ flit\ en\ una\ vuelta$$
$$resBWp_L + BW_p^v < CF \times numero\ de\ ciclos\ de\ flit\ en\ una\ vuelta$$

El parámetro *CF* se denomina *factor de concurrencia*, se almacena en un registro separado, y su valor se establece durante la inicialización del encaminador. Hay que hacer notar que el mecanismo de reserva propuesto no garantiza que la conexión obtendrá el ancho de banda pico en caso de necesidad. Si se proporcionase tal garantía, se podría desperdiciar una gran parte del ancho de banda. Sin embargo, ese ancho de banda será asignado a esa conexión con una probabilidad elevada. Esa probabilidad depende del factor de concurrencia *CF*. Un factor de concurrencia mayor significa que el ancho de banda del enlace será compartido entre más conexiones VBR, decrementando así las garantías de QoS. El factor de concurrencia es un compromiso entre las garantías de QoS y el número de conexiones que pueden ser atendidas concurrentemente.

Durante la transmisión de los datos, el protocolo de vigilancia actúa limitando la inyección de flits nuevos en la red, de tal forma que cada conexión no utilice más ancho de banda que el que reservó al establecerse. Este mecanismo solo es necesario en la transmisión de flujos con requerimientos de QoS. El tráfico ***best-effort*** no necesita establecer conexiones, y se transmite mediante VCT. La inyección de flits para paquetes ***best-effort*** se limita automáticamente puesto que solo utiliza el ancho de banda disponible después de satisfacer los requerimientos de conexiones a las que se les garantiza algún nivel mínimo de QoS.

El protocolo de control de admisión en el MMR es similar al empleado en redes ATM. La diferencia principal consiste en que en el MMR, el control de flujo evita que los flits sean descartados. Además, los ***buffers*** de flit son relativamente pequeños, provocando una rápida propagación de la información de control de flujo. Finalmente, el control de flujo puede propagarse hasta el interfaz de red del nodo fuente, limitando la inyección de nuevos flits. La vigilancia dentro del MMR solamente es necesaria si no hay control en el interfaz, y se permiten conexiones no fiables.

3.10 Planificación de los enlaces

Como se describe en la discusión anterior, el mecanismo básico para soportar QoS en el MMR consiste en reservar ancho de banda para cada conexión cuando ésta se establece, y en garantizar que ese ancho de banda reservado estará disponible durante la transmisión de datos.

Para *conexiones CBR* no fiables, cada canal virtual requiere mantener una variable de estado que almacene el ancho de banda adjudicado a ese canal. El algoritmo de planificación del enlace opera sobre la base de una vuelta. Lleva la cuenta del número de ciclos de flit asignados a cada canal virtual durante cada vuelta. Este algoritmo asegura que ningún canal virtual consume más ancho de banda del reservado.

Si se efectúa una vigilancia en el interfaz de red, solo hay que almacenar el ancho de banda adjudicado a cada conexión en el interfaz de red del nodo fuente. Si el interfaz de red limita el número de ciclos de flit por vuelta asignados a cada conexión, esa conexión no puede consumir más ciclos en cualquier encaminador del camino, si se le garantizan el mismo número de ciclos en cada vuelta.

Para *conexiones VBR*, la planificación el enlace es algo más compleja. Para conexiones no fiables, cada canal virtual necesita información de estado para almacenar el ancho de banda permanente y pico de esa conexión. Además, cada canal virtual almacena la prioridad de los datos que se están transmitiendo. Esa prioridad

puede modificarse dinámicamente enviando palabras de control desde el interfaz de red. Como se mencionó antes, el algoritmo de planificación del enlace funciona sobre la base de una vuelta, y lleva la cuenta del número de ciclos de flit asignados a cada canal virtual durante cada vuelta. El algoritmo asigna primero los ciclos de flit de las conexiones CBR. Después, asigna el ancho de banda permanente a cada conexión VBR. El mecanismo de reserva de ancho de banda permite asegurar que todas las conexiones VBR obtendrán su ancho de banda permanente. Después, el algoritmo de planificación considera las prioridades. Empezando por las conexiones de mayor prioridad, el algoritmo asigna ciclos a una conexión tras otra, asegurando que ningún canal virtual consume más ancho de banda del que reservó como pico.

Para las conexiones VBR fiables, solo es necesario mantener la información de estado referente al ancho de banda permanente y pico requerido por cada conexión, en el interfaz de red del nodo fuente. En cada nodo intermedio, el planificador del enlace asignará ciclos de flit a los flits que vayan llegando durante cada vuelta, dando prioridad a las conexiones CBR, luego a las VBR de mayor prioridad y así.

No todos los flits que llegan al encaminador durante una vuelta podrán ser transmitidos dentro de la misma vuelta, porque las ráfagas que llegan por enlaces de entrada diferentes pueden necesitar el mismo enlace de salida.

Como el esquema de reserva no garantiza que el ancho de banda pico de una conexión estará disponible en un momento dado, puede ocurrir que las conexiones VBR de baja prioridad no entreguen todos los flits a tiempo. Estos flits pueden ser descartados en el nodo destino. Sin embargo, puede ocurrir también que el control de flujo propague la congestión hacia el nodo fuente. Este nodo puede detectar la situación y dejar de transmitir información porque no llegará a tiempo. Así se puede disminuir el desperdicio de ancho de banda que supone la transmisión de información de será descartada en destino por llegar tarde. El ancho de banda en exceso (BW_p - BW_m) requerido por las conexiones VBR se sirve en turno, intentando atender del todo una conexión antes de pasar a la siguiente. La idea aquí consiste en que es preferible atender la mayoría de las conexiones completamente a riesgo de no servir algunas de ellas, que servir la mayoría pero solo en parte. Ciertamente, existen muchas otras posibilidades.

3.11 Planificación del conmutador

Además de la reserva de ancho de banda y de la planificación de los enlaces, también hemos de considerar la reserva de ancho de banda dentro del conmutador, y la planificación del mismo. Para cada puerto de entrada, el planificador de enlaces proporciona un canal virtual de entre muchos, para transmitir un flit durante el ciclo actual. La *planificación del conmutador* se refiere al proceso de determinar que puertos de entrada se conectan con qué puertos de salida durante un ciclo de flit.

La planificación del conmutador es necesaria porque conexiones que discurren por distintos puertos de entrada pueden compartir el mismo enlace de salida, y viceversa. En lo que sigue, hablaremos indistintamente de puertos y de enlaces, puesto que cada puerto de entrada/salida, está asociado con un único enlace de entrada/salida.

La planificación del conmutador debe llevarse a cabo junto con la planificación de los enlaces. En el caso ideal, los canales virtuales se deben seleccionar de manera que no existan conflictos en el uso de los puertos del conmutador y de los enlaces de

salida. Existen varios esquemas de planificación posibles. Podemos clasificarlos en primer lugar como ***input-driven*** o ***output-driven***.

Los *esquemas* ***input-driven*** empiezan considerando el conjunto de canales virtuales de cada enlace de entrada. Para cada conjunto, el algoritmo de planificación de los enlaces determina el canal virtual que debe transmitir un flit durante el siguiente ciclo de flit. La correspondencia directa entre canales indica el enlace de salida que emplea este canal virtual. Puesto que las peticiones de distintos puertos de entrada pueden competir por el mismo puerto de salida, debe existir algún tipo de arbitraje en los puertos de salida.

Por otro lado, los *esquemas* ***output-driven*** consideran el conjunto de canales virtuales de entrada que solicitan un enlace de salida dado. Para cada conjunto, el algoritmo de planificación de enlaces determina el canal virtual que debería transmitir un flit durante el siguiente ciclo. Como puede existir contención por el uso de los puertos de entrada, debe existir algún tipo de arbitraje para asignarlos.

Aunque los esquemas ***output-driven*** son aparentemente más complejos, la complejidad es similar a la de los *esquemas* ***input-driven*** si los vectores de bits de estado se almacenan en el lado de salida del encaminador. Debe señalarse que los esquemas ***output-driven*** proporcionan una perspectiva global de todos los canales virtuales que solicitan un enlace de salida dado. Por otro lado, los esquemas ***input-driven*** proporcionan una vista global de todos los canales virtuales que atraviesan un canal de entrada dado. Por tanto, no está claro cual es el esquema que funcionará mejor.

La situación sería distinta en el caso de utilizar un conmutador completamente demultiplexado. En ese caso, los esquemas ***output-driven*** proporcionan mejores prestaciones [10]. Sin embargo, para un número elevado de canales virtuales, el uso de un ***crossbar*** completamente demultiplexado es inviable.

Vamos a considerar ahora un esquema ***input-driven***. El planificador de enlaces selecciona un canal virtual de un puerto de entrada para transmisión, mientras que el arbitraje en los puertos de salida selecciona uno de entre los canales virtuales de los distintos puertos que compiten por los enlaces de salida. En lugar de llevar a cabo un arbitraje independiente en cada puerto de salida, los algoritmos de planificación del conmutador intentan planificar todos los puertos concurrentemente, y establecer el conmutador síncronamente. Estas decisiones de planificación pueden clasificarse de acuerdo con el mecanismo empleado para arbitrar entre múltiples peticiones dirigidas hacia un puerto de salida. Este arbitraje puede llevarse a cabo mediante prioridades estáticas o dinámicas, selección arbitraria, ***deadlines***, etc.

En [26], [25] y [17] aparecen resúmenes de algunos de los distintos algoritmos existentes en la literatura para éste propósito.

La combinación del planificador de enlaces y el arbitraje de puertos de salida es lo que denominamos *planificación del conmutador* en el encaminador MMR.

Por último los algoritmos de planificación del conmutador pueden clasificarse de acuerdo con el número de candidatos ofrecidos por el planificador de enlaces sobre cada grupo de canales virtuales pertenecientes a un puerto. Normalmente, se calcula un único candidato. Sin embargo, con el objeto de maximizar el probabilidad de asignar con éxito un puerto de salida dentro de un ciclo de flit, se puede calcular un conjunto de candidatos por cada enlace. De esta forma, si un canal virtual determinado no puede ser atendido debido a conflictos, puede suceder que haya otro canal virtual dentro del conjunto de candidatos que pueda ser servido.

Por ejemplo, suponiendo de nuevo que el encaminador utiliza un esquema ***input-driven***, en lugar de seleccionar un solo canal virtual por cada puerto de entrada, el planificador puede elegir un conjunto de candidatos. Este conjunto se obtiene como resultado de ejecutar varias operaciones con los vectores de bits de estado. Asumiremos en este ejemplo que el encaminador utiliza un arbitraje basado en la selección aleatoria.

En primer lugar, se selecciona aleatoriamente un enlace físico de entrada. De ahí se escoge un canal virtual de entre los candidatos, según el planificador de enlaces. Después, se selecciona aleatoriamente el siguiente puerto de entrada. De su conjunto de candidatos, se eliminan aquellos que utilizan los puertos de salida anteriormente planificados. Se escoge de entre los restantes un canal virtual, y así hasta recorrer todos los puertos de entrada. Hay que señalar que el proceso de eliminar canales virtuales conectados a puertos ya asignados puede realizarse fácilmente mediante vectores de bits. Cada vector de bits (uno por enlace de salida) indicaría los canales virtuales de entrada conectados a ese enlace. Esta información es redundante con la correspondencia entre canales, pero puede acelerar considerablemente la ejecución del algoritmo de planificación.

El algoritmo de planificación se ejecuta completamente en cada vuelta, atendiendo todas las conexiones activas, así como el tráfico ***best-effort***. Para cada vuelta, el algoritmo de planificación se invoca en cada ciclo de flit. Durante cada ciclo de flit, los enlaces de salida y los puertos de entrada se asignan síncronamente a un conjunto de canales virtuales libre de conflictos, para transmitir un solo flit de cada canal virtual. A la vez, el algoritmo de planificación calcula el conjunto de canales virtuales que transmitirán un flit durante el siguiente ciclo de flit. Entonces, el encaminador espera a que finalice la transmisión del flit en curso. El conmutador se reconfigura según la planificación calculada, empieza el siguiente ciclo de flit, y el algoritmo de planificación del conmutador se invoca de nuevo, y así hasta que se completa la vuelta.

El esquema descrito intenta maximizar la probabilidad de asignar canales virtuales a cada uno de los enlaces de salida durante cada ciclo de flit, empleando un conjunto de candidatos en cada puerto de entrada en lugar de un solo candidato. Sin embargo, cuando hay solo unos pocos canales virtuales por atender para una clase de servicio dada, puede ocurrir que todos los puertos de salida no se emparejen en un ciclo de flit. Es posible considerar el número de candidatos ofrecidos por cada puerto de entrada y adjudicar los enlaces de salida en orden creciente de número de candidatos. Sin embargo, este tipo de soluciones podría incrementar la complejidad del conmutador, con lo que debe buscarse una implementación eficiente.

Recordemos que nuestro objetivo de diseño consiste en maximizar la utilización del ancho de banda del conmutador, pero intentando minimizar la complejidad de los algoritmos de planificación, para poder conseguir un encaminador compacto y rápido.

4 Algunos resultados prácticos

A continuación, y a modo ilustrativo, presentaremos algunos de los resultados obtenidos hasta la fecha, en cuanto a la evaluación del encaminador MMR con tráfico

multimedia. En concreto, se utilizó tráfico mixto CBR y VBR, este último de tipo MPEG-2. Estos resultados, junto con algunos más, aparecen en [2].

4.1 Condiciones de la simulación y carga empleada

La herramienta de simulación está escrita en C++, y realiza una simulación conducida por eventos. El arbitraje de los enlaces se realiza como se describió en la sección 3.10. El arbitraje del conmutador es *input-driven*, y asigna los puertos del conmutador de forma cíclica. Por cada enlace de entrada, el planificador de enlaces proporciona varios canales virtuales candidatos.

Hemos simulado el comportamiento de un encaminador aislado para observar cómo afecta el algoritmo de planificación a las prestaciones de los flujos que atraviesan el encaminador. Éste consta de 4 puertos de entrada y otros tantos de salida, con 256 canales virtuales por enlace físico. Los enlaces tienen 16 bits de ancho (esto es, el tamaño de phit es 16), y el ancho de banda de los enlaces es de 1.24 Gbps. Por tanto, el tiempo de ciclo del encaminador (el tiempo que tarda en transmitir un phit) es de 12.9 nanosegundos. No limitamos el tamaño de los *buffers*, para reflejar el efecto de emplear control de flujo. En todas las pruebas se han simulado 200 vueltas del planificador.

La carga se compone de una mezcla de conexiones CBR y VBR. Todavía no incluimos ni tráfico *best-effort* ni mensajes de control. El tráfico CBR se genera de forma totalmente sintética. Se compone de conexiones escogidas aleatoriamente del siguiente conjunto de requerimientos de ancho de banda medios: {64 Kbps, 1.54 Mbps, 55 Mbps}. El tráfico VBR es semi-sintético. Lo modelamos de forma similar al modelo de los trenes de paquetes que aparece en [14], pero con el comportamiento temporal del vídeo MPEG-2.

El estándar de codificación de vídeo MPEG-2 [13] codifica los flujos de vídeo como una secuencia de diferentes tipos de trama, I, P y B, ordenadas según un patrón fijo y repetitivo, denominado GOP (*Group Of Pictures*). EL GOP que hemos empleado es IBBPBBPBBPBBPBB. Cada trama requiere una distinta cantidad de ancho de banda. Las que más ancho de banda consumen son las de tipo I, pues llevan más información. En la Figura 3 mostramos la apariencia típica de una secuencia de vídeo MPEG-2.

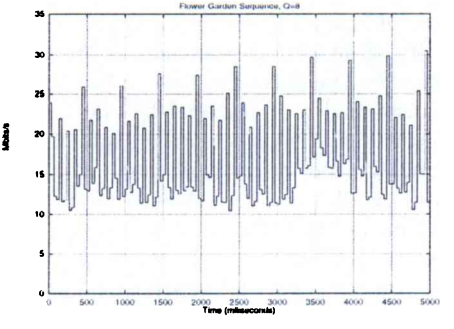

Figura 3. Ejemplo de secuencia de vídeo MPEG-2

Hemos modelado el flujo de vídeo MPEG-2 enviando una trama cada 33 milisegundos, esto es, con una tasa de transmisión de 30 tramas/seg. Los tamaños de las tramas se leen de ficheros de traza, obtenidos a partir de secuencias reales de vídeo MPEG-2. Los flits que componen una trama se transmiten de manera uniforme dentro de esos 33 milisegundos. Las conexiones se escogieron aleatoriamente de entre un conjunto de conexiones típicas, cuyas características aparecen en la Tabla 2. Consideramos un factor de concurrencia de 16, de manera que no afecta a la admisión de conexiones VBR.

Tabla 2. Características de las secuencias de vídeo MPEG-2

	Tamaño de Imagen (bits)		
Secuencias	Máx.	Mín.	Media
Ayersroc	535030	148755	232976
Hook	454560	159622	272738
Martin	444588	116094	199880
Flower Garden	900139	308411	497126
Mobile Calendar	970205	412845	600742
Table Tennis	933043	260002	440547
Football	590532	340246	441459

Las peticiones de conexión se generan aleatoriamente, para distintos niveles de carga. Cuando comienza la simulación, el control de admisión de conexiones puede rechazar algunas, si no existe suficiente ancho de banda en el correspondiente enlace de salida para atenderlas. El resto de conexiones se mantienen activas durante toda la simulación. En cada punto de carga, hay un 50 % del ancho de banda ocupado por tráfico CBR, y otro 50 % por tráfico VBR (MPEG-2).

4.2 Evaluación de prestaciones

En la Figura 4 se muestra la utilización media del *crossbar* considerando dos tamaños de flit: flits pequeños (128 bits) y flits grandes (1024 bits). En esta gráfica se representa únicamente la utilización debida a datos, eliminando por tanto la parte de utilización debida a la sobrecarga de control. Para niveles de carga bajos (entorno al 60 %) observamos que se obtienen niveles de utilización comparables tanto para flits pequeños, como para flits grandes. Al aumentar el nivel de carga, la utilización cuando los flits son pequeños apenas crece, quedándose en torno al 70 %. Es decir, con flits pequeños, el encaminador ofrece un 70 % de utilización cuando se satura. Por el contrario, al emplear flits grandes, se puede explotar alrededor de un 10 % más del ancho de banda. El encaminador es capaz de llegar al 80 % de utilización, antes de saturarse. En conclusión, se observa que el empleo de flits grandes es la mejor elección, considerando únicamente utilizar el enlace de forma eficiente. Esto es debido a que la información de control se inyecta con menor frecuencia, y por tanto, hay más ancho de banda disponible para la transmisión de datos.

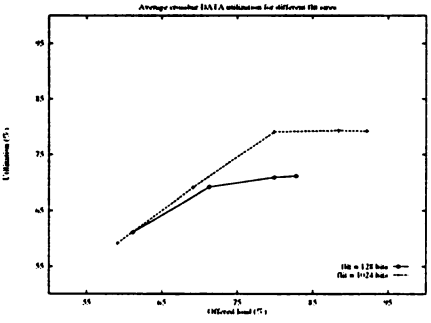

Figura 4. Efecto del tamaño de flit en la utilización del *crossbar* por datos.

Acabamos de ver cómo nuestro simple esquema de planificación es capaz de alcanzar utilizaciones del *crossbar* bastante elevadas. Sin embargo, y si queremos proporcionar garantías de calidad de servicio, necesitamos conocer cómo se transmite la carga.

Para ello, hemos medido la contribución de cada encaminador al retardo global experimentado por los flits. Es decir, para cada flit, se ha medido el tiempo transcurrido desde que se almacena en el buffer de entrada correspondiente, hasta que deja el encaminador a través de su enlace de salida. Hemos obtenido los retardos máximo y medio, y la desviación típica. Las resultados se muestran en la Tabla 3 , y han sido obtenidos para $K = 16$ y flits de 1024 bits. Los resultados vienen expresados en ciclos de reloj del encaminador (1 ciclo = 12.9 nanosegundos).

Podemos comprobar que el retardo que experimentan los flits CBR no crece de forma significativa cuando se incrementa la carga. Esto es debido a la mayor prioridad que se le da al tráfico CBR respecto al VBR cuando se compite por recursos. Por otro lado, el tráfico VBR sufre retardos extremadamente elevados para niveles de cargas de alrededor del 80 %. Por tanto, podemos concluir que, aunque nuestro encaminador es capaz de alcanzar utilizaciones tan elevadas, los retardos obtenidos para los flits pertenecientes a flujos VBR son intolerables.

Respecto a los valores de la desviación típica, podemos observar que se mantienen estables, excepto para el tráfico VBR a cargas elevadas. Esto significa que los retardos que experimentarán los flits son bastante predecibles. Más aún, esto sugiere que las prestaciones en términos de *jitter* puede ser buena. Esto lo veremos más adelante.

Tabla 3. Estadísticas del retardo sufrido por los flits.

	Carga (%)	Máximo	Medio	Desv. Típica
CBR	59.98	3299	127.50	288.00
	70.20	1913	135.26	288.00
	81.13	2969	148.99	289.00
VBR	59.98	29358	253.76	288.00
	70.20	200073	516.50	288.00
	81.13	26915769	521313.74	1074500784.00

Para comprobar cómo se distribuyen los retardos, hemos establecido seis umbrales (TH1 .. TH6), y hemos contado el número de flits cuyos retardos son mayores que cada uno de esos umbrales. TH1 es 10 veces el tiempo que necesita el *crossbar* para transmitir un flit (65 ciclos del encaminador = 838.5 nanosegundos). TH2 es el doble de TH1, TH3 es el doble de TH2, etc. Por tanto, TH6 es 32 x TH1 = 268.32 microsegundos. Los resultados se presentan en la Tabla 4 como porcentajes del total de flits transmitidos, para los dos niveles de carga considerados.

Tabla 4. Distribución de los retardos de flit, para tráfico CBR y VBR

	Carga (%)	TH1	TH2	TH3	TH4	TH5	TH6
CBR	59.98	0.034	0.00056	0.00011	0	0	0
	70.20	0.063	0.00075	0	0	0	0
VBR	59.98	6.35	1.17	0.29	0.11	0.045	0.004
	70.20	19.50	6.81	2.09	0.64	0.20	0.075

Podemos comprobar que casi todos los flits CBR experimentan retardos inferiores al primer umbral (8.385 microsegundo) para ambos niveles de carga. Para tráfico VBR y el 70 % de carga, casi todos los flits sufren retardos menores que el sexto umbral. Para cargas menores, la mayor parte de los flits están dentro de cotas de retardo más estrictas, como la impuesta por TH5 (134.16 microsegundos). El retardo máximo para los flits VBR es de 200073 ciclos del encaminador, esto es, alrededor de 2.6 milisegundos (ver la Tabla 3). Estos resultados son bastante esperanzadores, porque un límite típico para el retardo en una transmisión de vídeo MPEG-2 es de 1 segundo entre extremos. Este es el valor para el CTD (*Cell Transfer Delay*) recomendado por el ATM Forum para servicios de distribución de vídeo mediante MPEG-2 [22].

La unidad de información para las aplicaciones de vídeo MPEG-2 y desde el punto de vista del receptor, es la *trama*. Por tanto, también hemos medido prestaciones del encaminador respecto a tramas de vídeo MPEG-2. El *retardo de una trama* es el tiempo transcurrido desde que el primer flit de la misma se almacena en un buffer de entrada del encaminador, hasta que el último flit de la trama sale por de algún enlace de salida. Hay que hacer notar que enviamos todos los flits que componen una trama de manera uniforme dentro de los 33 milisegundos de separación entre tramas adyacentes, por tanto cada trama tardará al menos 33 milisegundos en atravesar el encaminador. Este tiempo fijo se representa en la Figura 5 (izquierda) con una línea recta. Podemos comprobar en esa figura que el retardo introducido por el encaminador es bastante reducido: para el nivel de carga más alto (75 %) este retardo adicional está por debajo de los 200 microsegundos.

También hemos medido el *jitter* que sufren las tramas cuando atraviesan el encaminador. En transmisión de vídeo MPEG-2, cada trama debe llegar a su destino 33 milisegundos después que la trama anterior, para poder ser visualizada adecuadamente. Por tanto, definimos el *jitter entre tramas*, como la desviación con respecto a estos 33 milisegundos de separación requeridos. Los resultados se muestran en la Figura 5 (derecha). Podemos ver que para cargas menores del 70 % , el *jitter* se mantiene por debajo de 1 microsegundo. Para carga más elevada, el *jitter* se incrementa, pero no pasa de los 3.5 microsegundos. Estos son resultados son bastante esperanzadores, porque el *jitter* permitido en transmisiones de vídeo MPEG-2 se halla alrededor de varios milisegundos, esto es, el *jitter* debe ser lo bastante bajo como para

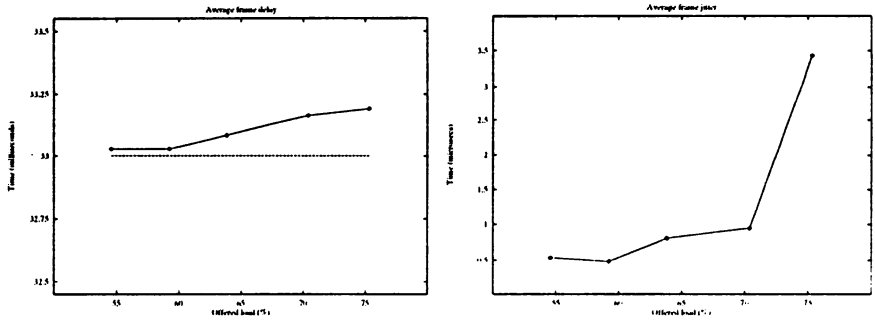

Figura 5. Retardo y *jitter* medios de las tramas MPEG-2

que una persona pueda ver la secuencia de vídeo de manera fluida, con una tasa regular.

5 Conclusiones

Hoy en día, las aplicaciones multimedia son cada vez más comunes. En este artículo hemos presentado una nueva arquitectura de encaminador diseñada para afrontar los requerimientos de estas aplicaciones, así como de las tradicionales aplicaciones de datos: *el encaminador multimedia MMR*. Hemos discutido cada uno de los parámetros involucrados en el diseño del encaminador, justificando la elección hecha para cada uno de ellos. Por último, y a modo ilustrativo, hemos mostrado algunos de los resultados obtenidos hasta la fecha en cuanto a prestaciones ofrecidas por el encaminador multimedia MMR.

Señalar por último que en el proyecto de desarrollo del encaminador multimedia MMR intervienen grupos de investigación pertenecientes a la Universidad Politécnica de Valencia y al Georgia Institute of Technology (EE.UU.), además de la Universidad de Castilla-La Mancha.

Referencias

1. Boden, N. y otros: Myrinet: A gigabit per second LAN. IEEE Micro, Febrero (1995)
2. Caminero, M.B.; Quiles, F.J.; Duato, J.; Love, D.; Yalamanchili, S.: Performance Evaluation of the Multimedia Router with MPEG-2 Video Traffic. Proceedings Workshop on Communications and Architectural Support for Network-based Parallel Computing. Enero (1999)
3. Chien, A.A.; Kim, J.H.: Approaches to quality of service in high-performance networks. Proceedings del Parallel Computer Routing and Communications Workshop. Julio (1997)
4. Cuenca, P.; Orozco-Barbosa, L.; Wang, L.; Garrido, A. y Quiles, F:: Packing scheme for layered coding MPEG-2 video transmission over ATM based networks. Proceedings IEEE ATM'97 Workshop. Mayo (1997)

5. Dally, W.J.: Virtual-channel flow control. IEEE Transactions on Parallel and Distributed Systems. Marzo (1992)
6. Dao, B.V.; Duato, J.; Yalamanchili, S.: Configurable flow control mechanisms for fault-tolerant routing. Proceedings 22nd International Symposium on Computer Architecture. Junio (1995)
7. Duato, J.; López, P.; Silla, F.; Yalamanchili, S.: A high performance router architecture for interconnection networks. Proceedings de International Conference on Parallel Processing. Agosto (1996)
8. Duato, J.; Yalamanchili, S.; Caminero, M.B.; Love, D.; Quiles, F.J.: MMR: A high-performance multimedia router. Architecture and design trade-offs. Proceedings del Fifth International Symposium on High Performance Computer Architecture (HPCA-5). Enero (1999)
9. Duato, J.; Yalamanchili, S.; Ni, L. : Interconnection Networks: An Engineering Approach. IEEE Computer Society. Agosto (1997)
10. Fulgham, M.L.; Snyder, L.: A comparison of input and output driven routers. Proceedings Euro-Par'96. Agosto (1996)
11. Gaughan, P.T.; Yalamanchili, S.: A family of fault-tolerant routing protocols for direct multiprocessor networks. IEEE Transactions on Parallel and Distributed Systems. Mayo (1995)
12. Gaughan, P.T.; Yalamanchili, S.: Adaptive routing protocols for hypercube interconnection networks. IEEE Computer. Mayo (1993)
13. Generic coding of moving pictures and associated audio. Recomendación H.262.Draft International Standard ISO/IEC 13818-2. Marzo (1994)
14. Jain, R.; Routhier, S.: Packet trains -- Measurement and a new model for computer network traffic. IEEE Journal on Selected Areas in Comunications. Septiembre (1986)
15. Jonsson, J.; Vasell, J.: A comparative study of methods for time-deterministic message delivery in a multiprocessor architecture. Proceedings IEEE International Parallel Processing Symposium. (1996)
16. Katevenis, M.G.H. y otros: ATLAS I: A single-chip ATM switch for NOWs. Proceedings del Workshop on Communications and Architectural Support for Network-based Parallel Computing. Febrero (1997)
17. Kim, J.H.: Bandwidth and latency guarantees in low-cost, high-performance networks. Tesis Doctoral. Universidad de Illinois en Urbana-Champaign. (1997)
18. Pakin,S.; Lauria, M.; Chien, A.: High performance messaging on workstations: Illinois Fast Messages on Myrinet. Proceedings Supercomputing 95. Noviembre (1995)
19. Prycker, M.: Asynchronous transfer mode: solution for broadband ISDN. Ellis Horwood Limited (1991)
20. Rexford, J.; Hall, J. y Shin, K.G.: A Router Architecture for Real-Time Point-to-Point Networks. Proceedings International Symposium on Computer Architecture. Mayo (1996)
21. Schroeder, M.D. y otros: Autonet: A high-speed, self-configuring local area network using point-to-point links. IEEE Journal on Selected Areas in Communications. Octubre (1991)
22. Schwartz, M.; Beaumont, D. : Quality of service requirements for audio-visual multimedia services. ATM Forum, ATM94-0640. Julio (1994)
23. Silla, F. y Duato, J.: Improving the efficiency of adaptive routing in networks with irregular topology. Proceedings de Conference on High Performance Computing. Diciembre (1997)

24. Silla, F. y otros: Efficient adaptive routing in networks of workstations with irregular topology. Proceedings Workshop on Communications and Architectural Support for Network-based Parallel Computing. Febrero (1997)
25. Stiliadis, D.: Traffic scheduling in packet-switched networks: Analysis, design and implementation. Tesis Doctoral. Universidad de California en Santa Cruz. Junio (1996)
26. Zhang, H.: Service disciplines for Guaranteed Performance Service in Packet-Switching Networks. Proceedings of the IEEE. Octubre (1995)