Reducción del Número de Canales Virtuales para Soportar Calidad de Servicio en Clusters*

A. Martínez, F. J. Alfaro, J. L. Sánchez
Escuela Politécnica Superior
Univ. Castilla-La Mancha
{alejandro,falfaro,jsanchez}@info-ab.uclm.es

J. Duato
Facultad de Informática
Univ. Politécnica Valencia
jduato@disca.upv.es

Resumen

Los estándares de interconexión actuales que proporcionan soporte para calidad de servicio (QoS) incorporan en interfaces y conmutadores 16 o incluso más canales virtuales (CVs) para ello. Sin embargo, la mayoría de las implementaciones no ofrecen tantos CVs porque es demasiado costoso en términos de área de chip. Nosotros creemos que el número de CVs necesarios en los conmutadores para el soporte de QoS puede ser reducido significativamente eliminando ciertas redundancias en el procesamiento del tráfico. En particular, algunas de las decisiones de planificación hechas en las interfaces de red pueden ser fácilmente reutilizadas en los conmutadores sin alterar significativamente el comportamiento global de los planificadores.

1. Introducción

Muchas de las aplicaciones que se ejecutan hoy día en Internet normalmente presentan requisitos de ancho de banda y/o latencia [10]. A éstos se les conoce como requisitos de QoS. Dado que los clusters se están convirtiendo la forma más popular para proporcionar servicios hacia Internet, parece razonable que exista un cierto soporte de QoS en ellos.

La mayoría de las propuestas de conmutadores para clusters con soporte de QoS incorporan 16 o incluso más CVs, asignando un CV diferente a cada clase de tráfico. Esto incrementa la complejidad del conmutador y también impide el uso de esos CVs para otros propósitos.

Por otro lado, las implementaciones finales de conmutadores no suelen incorporar demasiados CVs. Además, estudios recientes [9] sugieren que cuando la tecnología lo permita, la tendencia será incrementar el número de puertos por conmutador, en lugar de incrementar los CVs por puerto. Ésta es, de hecho, la tendencia que están siguiendo los fabricantes en los nuevos productos que lanzan al mercado.

En este trabajo mostramos que son suficientes dos CVs en cada puerto del conmutador para proporcionar QoS. Los resultados de simulación muestran que las aplicaciones obtienen una QoS similar a la obtenida con las propuestas tradicionales, pero usando menos CVs y, por tanto, menos espacio de buffer y un menor tiempo de procesamiento. Esto se puede conseguir reutilizando en los conmutadores algunas de las decisiones de planificación realizadas en las interfaces de red. La forma de hacer esto es utilizar un CV para el tráfico con necesidad de garantías de QoS y otro para el resto del tráfico.

Este artículo se estructura de la siguiente forma: en la Sección 2 se revisan las propuestas existentes para QoS en clusters. La Sección 3 se centra en nuestra propuesta. Finalmente, la evaluación de las propuestas se realiza en la Sección 4 y la Sección 5 contiene las conclusiones y el trabajo futuro.

*Este trabajo ha sido parcialmente financiado por la CICYT con el proyecto TIC2003-08154-C06 y por el Ministerio de Educación y Ciencia con una beca FPU.

2. Trabajo relacionado

En los últimos años se han propuesto diversos diseños de clusters con soporte para QoS. Todos ellos incorporan CVs para proporcionar soporte de QoS. Entre los más recientes se encuentran los estándares InfiniBand y PCI Express Advanced Switching (AS). InfiniBand [6] soporta hasta 16 CVs. Por otro lado, la arquitectura AS [1] incorpora hasta 20 CVs. En el caso de implementar un gran número de CVs, esto requeriría una fracción significativa del chip y podría hacer del procesamiento de los paquetes una tarea demasiado compleja.

Por otro lado, ha habido propuestas que intentan ofrecer QoS con tan sólo dos CVs. Por ejemplo, el Avici TSR [3] es un caso bien conocido de esto. El Avici TSR capaz de segregar el tráfico en dos categorías, una preferente y otra normal. Sin embargo, está limitado a esta clasificación y no puede diferenciar más categorías. En los estándares recientes del IEEE se recomienda considerar hasta siete clases de tráfico [5]. Por lo tanto, aunque diferenciar dos categorías es un gran avance, puede no ser suficiente.

En contraste a esto, nuestra propuesta es novedosa porque, aunque utiliza tan sólo dos CVs en los conmutadores, el comportamiento global de la red es similar al de una red con conmutadores de muchos más CVs. Esto es posible porque los conmutadores reutilizan parte de las decisiones de planificación realizadas en las interfaces de red, que tienen tantos CVs como clases de tráfico (por ejemplo, 8 en la evaluación de prestaciones que vamos a ver). El resultado final es que la red proporciona un servicio diferenciado a todas las categorías de tráfico que se consideran.

Hasta donde nosotros sabemos, sólo Katevenis y su grupo [2] han propuesto algo similar antes. La idea fundamental de su arquitectura consiste en gestionar múltiples niveles de prioridad sobre dos colas en cada buffer. El funcionamiento del sistema es análogo al de una autopista de dos carriles, donde los coches avanzan por un carril y adelantan por el otro. Sin embargo, esta propuesta es compleja porque necesita hardware y señalización específicos. Además, es más limitada en sus objetivos que la nuestra porque está pensada para un encaminador de una sola etapa basado en un único crossbar. Este crossbar tiene pequeños buffers en los puntos de cruce que los autores dividen en dos CVs (pero los buffers de entrada siguen teniendo tantos CVs como categorías de tráfico). Por otro lado, nuestra propuesta es más simple y general, como veremos en la siguiente sección.

3. Reducción del número de CVs en los conmutadores

En esta sección presentamos nuestra propuesta para usar tan sólo dos CVs en los puertos de los conmutadores para proporcionar QoS. De este modo, se puede acelerar el procesamiento de los paquetes, reducir la complejidad de los conmutadores y/o utilizar el resto de los CVs para otros propósitos. Esta reducción de complejidad, y la aceleración que apareja, no implican una pérdida de funcionalidad porque están basadas en la eliminación de redundancias. La idea fundamental consiste en reutilizar en los conmutadores algunas de las decisiones de planificación llevadas a cabo en las interfaces de red.

Para que esta propuesta pueda ser efectiva debemos asumir dos hipótesis. La primera suposición que hacemos es que existe un criterio estático para ordenar los paquetes que se van a transmitir. De este modo, cada paquete sería etiquetado con un nivel de servicio que no cambiaría en ningún momento. Esto es necesario porque si vamos a mantener el orden de llegada, es necesario que este orden sea siempre correcto. De todos modos, dadas las latencias tan pequeñas que vamos a manejar para paquetes que necesitan QoS, este requisito no es muy restrictivo.

La segunda suposición que necesitamos es que haya un mecanismo de control de admisión de conexiones (CAC) para el tráfico con necesidades de QoS, de modo que ningún enlace sea sobreutilizado. En este caso, el requisito es necesario para resolver los problemas asociados a los sistemas con prioridades estrictas, que son la garantía de ancho de banda y la ina-

nición. La inclusión de un CAC en la red está plenamente justificada, más allá de nuestra propuesta, para evitar que ráfagas de tráfico provoquen congestión.

Así pues, nuestra propuesta consiste en utilizar dos CVs, diferenciando entre el tráfico que ha reservado ancho de banda (tráfico que necesita QoS) y el que no (tráfico best-effort) para poder ofrecer garantías de ancho de banda. En una situación en la que este tipo de garantías fuese innecesaria o irrealizable por el coste del CAC, sería posible prescindir de esta separación y utilizar tan sólo un CV. Como veremos en la siguiente sección, habrá una diferenciación del tráfico dentro de las dos categorías, aunque en la descripción que sigue nos centraremos en el tráfico con necesidades de QoS.

Ahora vamos a describir cómo funciona esta propuesta. Supongamos que varios paquetes llegan a un conmutador desde una interfaz de red. Si la interfaz implementa un arbitraje basado en prioridades, el primer paquete debe ser aquel con mayor prioridad. Así que, en lugar de separar los paquetes en varios CVs de acuerdo a su clase de servicio o prioridad, los ponemos todos en la misma cola en el orden en que llegan. Más tarde, cuando el conmutador va a ser planificado, se procede a inspeccionar las colas de entrada, pero sólo es necesario mirar el paquete en cabeza de cada cola. El motivo es que si está antes que los otros es porque tenía más prioridad cuando abandonó la interfaz de red.

Obviamente, una interfaz de red no puede arbitrar sobre todos los paquetes, sino sólo sobre aquellos que tiene almacenados en un momento dado. Por lo tanto, cuando no hay más paquetes de alta prioridad para enviar, se transmite un paquete de baja prioridad. Es importante apreciar que como los paquetes best-effort tienen su propio CV, nunca interferirán con el tráfico que necesita QoS. Así que, de vuelta a nuestro ejemplo, el paquete de baja prioridad es también un paquete que necesita QoS y por ello se guarda en el CV correspondiente. Puede suceder que una vez que el paquete llegue al conmutador, tenga que esperar a que el planificador le permita cruzar. Si, entre tanto, otros paquetes de mayor prioridad son transmitidos desde la interfaz de red, éstos tendrían que ser almacenados en el mismo CV y detrás de nuestro paquete de baja prioridad. De este modo, el planificador penalizaría a los paquetes de alta prioridad porque éstos tendrían que esperar a que el paquete de baja prioridad que los precede fuera atendido. Esta situación, que de ahora en adelante denominaremos como *error de orden*, tiene poco impacto en el rendimiento global, como veremos en la siguiente sección. El motivo es que todo el tráfico que necesita QoS tiene ancho de banda garantizado, de modo que siempre podrá avanzar con retardos limitados. Así, los paquetes de alta prioridad tendrán garantizada una espera corta.

Por otro lado, el orden de llegada de los paquetes best-effort también se mantiene, al utilizarse un único CV para esta categoría. Esto permitirá que se produzca también una diferenciación entre los paquetes de este tipo. Por otro lado, si llega un punto en el que hay más tráfico best-effort que ancho de banda disponible, las interfaces de red inyectarán simpre los niveles más prioritarios antes, de modo que se saturarán más tarde. Este comportamiento se verá en detalle en la siguiente sección.

En resumen, nuestra propuesta consiste en ordenar el tráfico que necesita QoS en las interfaces de red y garantizar que no consumirá todo el ancho de banda. Además, se rellenan los huecos que surjan en el flujo de salida con tráfico best-effort, el cual no ofrece ni necesita ninguna garantía. Por ese motivo utilizamos dos CVs en los conmutadores, uno para tráfico que necesita QoS, manteniendo el orden de llegada, y otro para el tráfico best-effort de modo que éste no interfiere con el tráfico que necesita QoS.

4. Evaluación de Prestaciones

A lo largo de esta sección vamos a evaluar la propuesta presentada en la sección anterior, para lo que se ha desarrollado una detallada herramienta de simulación. En primer lugar, describiremos los parámetros de simulación y las consideraciones de modelado que hemos usado a lo largo de las simulaciones. En se-

gundo lugar, mostraremos y analizaremos los resultados que hemos obtenido.

4.1. Arquitectura simulada

La red que hemos utilizado para simular nuestra propuesta fue una red multietapa. Sin embargo, nuestra propuesta es válida para cualquier topología de red, incluyendo redes directas e indirectas. Los conmutadores que usamos utilizan una arquitectura de buffers a la entrada y la salida, con un crossbar para conectar los puertos. Se implementó Virtual Output Queueing (VOQ) para evitar el problema del head-of-line blocking a nivel de conmutador. Esto no incrementa la memoria de buffer necesaria, sólo el tiempo de planificación del crossbar. El motivo es que, hoy en día, es habitual implementar los CVs y las colas sobre un espacio de memoria común compartido. Es decir, nuevas colas implican más punteros y un árbitro más complejo, pero no necesariamente espacio de buffer adicional.

Los parámetros de los elementos de la red utilizados en este estudio de prestaciones se muestran en la Tabla 1. La red que utilizamos fue una multietapa *perfect-shuffle* de 64 puertos. El resto de parámetros son los típicos para un estudio de estas características. Hay que señalar que suponemos algo de aceleración en el crossbar interno con respecto a la velocidad de los enlaces (x1,5). Esto es habitual en la mayoría de los conmutadores comerciales. Se usa control de flujo basado en créditos y, por tanto, no se descartan paquetes.

Es importante señalar que nuestro objetivo es evaluar el rendimiento de nuestra propuesta bajo unas condiciones justas. En otras palabras, no tratamos de conseguir el mejor rendimiento o de proponer el diseño completo de un nuevo conmutador. Por ese motivo, no se ha pretendido usar las más actuales técnicas de encaminamiento, control de congestión, etc.

En la Tabla 1 también mostramos la cantidad de memoria necesaria para implementar cada CV. Proponemos utilizar 4 Kbytes porque ese espacio permite almacenar dos paquetes completos. Esto es necesario para asegurar el flujo continuo de paquetes, ya que usamos control de flujo basado en créditos y *Virtual Cut-Through*. Nótese que los conmutadores que utilicen nuestra propuesta ahorran memoria (y, por lo tanto, espacio de chip) en los puertos en un factor de 4 comparado con el uso típico de 8 CVs. Para dar una idea del impacto final en el diseño del chip, en el caso del conmutador ATLAS I [8], los buffers y la lógica de control de flujo (que depende en gran medida del número de CVs) necesitó un 21 % del área del chip.

Parámetro	**Valor**
Puertos por conmutador	8 puertos
Tamaño de paquete	64 a 2048 bytes
Tamaño de cabecera	8 bytes
Tamaño de msj. de control	8 bytes
Tamaño de CV	4 Kbytes
BW de los canales	2 Gb/s
BW de los crossbars	3 Gb/s
Interfaces de red	64
Retraso planif. crossbar	20/10 ns
Retraso arbitraje salida	10/5 ns

Tabla 1: Parámetros de simulación.

Los retrasos de arbitraje han sido escogidos para reflejar las latencias reales en implementaciones finales, así como el uso de menos CVs en nuestra propuesta. Hemos tenido en cuenta la investigación realizada en el área, como el trabajo de Peh [11], que indica que el tiempo de arbitraje es logarítmico en el número de CVs (y colas de VOQ). En la Tabla 1, el número a la izquierda de la barra es el retraso para conmutadores de 8 CVs, mientras que el siguiente es para conmutadores de 2 CVs. Por lo tanto, la aceleración que esperamos es de 2,0.

Hemos implementado un sencillo CAC basado en el ancho de banda medio. Aunque existen otras propuestas más complejas y potentes, ésta basta para probar nuestra propuesta.

4.2. Modelo de tráfico

Hemos considerado los niveles de servicio definidos en el estándar IEEE 802.1D-2004 [5] en el Anexo G, los cuales son muy apropiados para este estudio. En la Tabla 2 se incluyen las

Nivel	Nombre	% BW	Tam. paquete	Notas
7	Control de red	1	64 bytes	auto-similar
6	Audio	15,625	128 bytes	Conexiones CBR de 64 Kb/s
5	Vídeo	15,625	$\leq$ 2 Kbytes	Trazas MPEG-4 de 750 Kb/s
4	Carga controlada	15,625	2 Kbytes	Conexiones CBR de 1 Mb/s
3	Excellent-effort	13,031	2 Kbytes	auto-similar
2	Pref. Best-effort	13,031	2 Kbytes	auto-similar
1	Best-effort	13,031	2 Kbytes	auto-similar
0	Background	13,031	2 Kbytes	auto-similar

Tabla 2: Tráfico inyectado por host.

características del tráfico inyectado en la red. Sin embargo, hemos añadido un octavo nivel de servicio, *Preferential Best-effort*, con una prioridad entre el *Excellent-effort* y el *Best-effort*. De este modo, la carga de la red está compuesta de 8 niveles de servicio diferentes, cada uno con una prioridad descendente, de modo que el nivel 7 tiene la mayor prioridad y el 0 la menor. La proporción de cada categoría ha sido elegida para proporcionar resultados significativos.

El patrón de destinos se basa en la ley de Zipf [12], como se recomienda en [4], con un $k = 1$. De este modo, el tráfico no está uniformemente distribuido, sino que, por cada nivel de servicio y origen de datos, se establece un *ranking* u ordenación entre todos los destinos. Así, habrá destinos más probables para un grupo de conexiones, donde la probabilidad se obtiene con la mencionada ley de Zipf.

Los paquetes se generan de acuerdo a distintas distribuciones, como se puede ver en la Tabla 2. El tráfico de audio, vídeo y carga controlada se compone de conexiones punto a punto del ancho de banda indicado. El tráfico auto-similar es tráfico con ráfagas generado por fuentes on/off, gobernadas por distribuciones de Pareto, como se recomienda en [7].

4.3. Resultados de simulación

De los cuatro índices tradicionales de QoS, se han utilizado tres, que describimos a continuación. El primero de estos índices, la latencia, es el tiempo consumido por un paquete para alcanzar su destino. El ancho de banda es la cantidad de información transmitida por unidad de tiempo. Finalmente, también es habitual medir el jitter, que es la diferencia entre los retrasos de dos paquetes consecutivos pertenecientes a la misma conexión [4]. Esta métrica sólo tiene sentido entre los paquetes que pertenecen a una conexión y, por lo tanto, sólo se mide para el tráfico de audio y vídeo. Como no pueden perderse paquetes porque utilizamos un control de flujo basado en créditos, el índice pérdida de paquetes no ha sido considerado aquí.

El jitter máximo determina el espacio de aplicación que debe ser reservado por el receptor para aplicaciones de audio y vídeo. Resultados inadecuados de latencia o jitter pueden conducir a la pérdida de paquetes a nivel de aplicación. Por este motivo, en las gráficas también mostramos la función de distribución acumulativa de la latencia y el jitter, que ofrece la probabilidad de que un paquete obtuviese una latencia o jitter menor o igual que un valor dado.

Para esta evaluación, hemos variado el número de CVs en los conmutadores, aunque en todos los casos, el arbitraje está basado en prioridades. El diseño más complejo (etiquetado en las figuras como *Tradicional 8 CVs*), que representa la aproximación tradicional, utiliza 8 CVs. Como este número coincide con el número de clases de servicio que hemos considerado, no es posible que un paquete de baja prioridad se almacene en una misma cola delante de un paquete de mayor prioridad. Los otros dos modelos utilizan 2 CVs, pero en un

caso se trata de la aproximación tradicional (*Tradicional 2 CVs*) y en el otro de nuestra propuesta (*Nuevo 2 CVs*). Estos diseños se benefician de un arbitraje más sencillo que necesita menos tiempo para operar. Por otro lado, el espacio de buffer es más reducido al tener menos CVs, como se comentó en la sección anterior. Por último, con la propuesta tradicional de 2 CVs, las interfaces de red también utilizan 2 CVs.

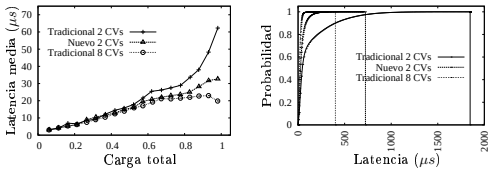


Figura 1: Rendimiento del tráfico de control de red.

La Figura 1 muestra la latencia media para el nivel de servicio 7, que se corresponde con el tráfico de control de red. La productividad para este nivel de servicio es la óptima en los tres casos. Podemos ver que los tres árbitros tienen éxito al producir una latencia media aceptable, aunque el caso *Tradicional 2 CVs* se dispara un poco en carga alta. Es más, la Figura 1 también muestra la función de distribución acumulativa de latencia, que indica la probabilidad de que la latencia de un paquete esté por debajo de un cierto valor. Los resultados de las propuestas *Tradicional 8 CVs* y *Nuevo 2 CVs* son aceptables, pero en el caso de *Tradicional 2 CVs* ya no lo son tanto, porque muchos paquetes se desvían de la media. En esta figura, los valores máximos se representan con líneas verticales.

En la Figura 2 mostramos el rendimiento del tráfico de audio. De acuerdo a las directrices del IEEE, este nivel de servicio debería alcanzar una latencia y jitter menores de 10 ms. Aunque esto se consigue en los tres casos, la propuesta *Tradicional 2 CVs* produce un rendimiento inadecuado a carga alta. Hay que señalar que nuestra propuesta no consigue reducir los valores máximos de latencia y jitter hasta el nivel de la aproximación *Tradicional 8 CVs*. Sin embargo, los resultados son todavía

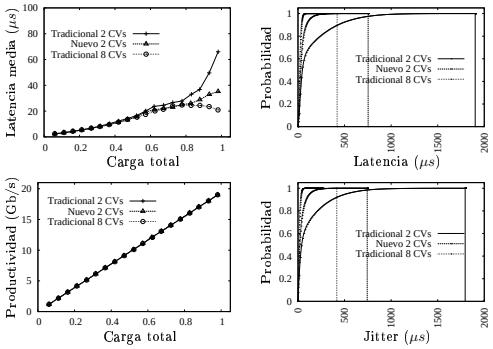


Figura 2: Rendimiento del tráfico de audio.

aceptables y suponen un significativo avance sobre el caso *Tradicional 2 CVs*.

Los resultados del tráfico de vídeo, que se muestran en la Figura 3, son similares a los de audio, aunque la latencia y el jitter son algo mayores. Sin embargo, dado el amplio margen que da el IEEE para este tipo de tráfico (para latencia y jitter, un máximo de 100 ms), las tres propuestas serían perfectamente válidas.

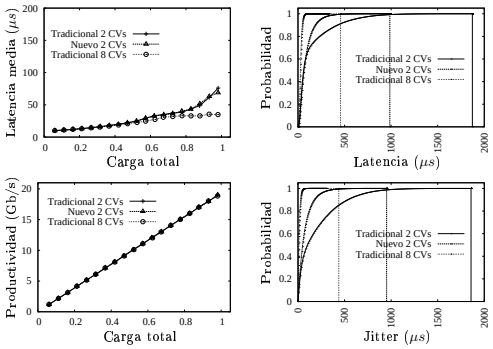


Figura 3: Rendimiento del tráfico de vídeo.

Para terminar con el tráfico con necesidad de QoS, los resultados del tráfico de carga controlada se presentan en la Figura 4. En este caso, la única garantía que es necesaria es la de productividad y en la figura queda claro que esto se consigue en los tres casos. Esto no es sorprendente, dado que el CAC se encargó de garantizar que hubiera suficiente ancho de

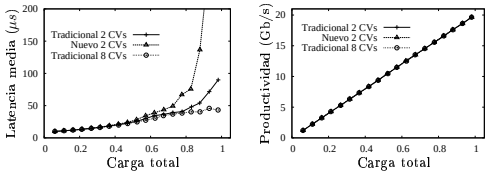
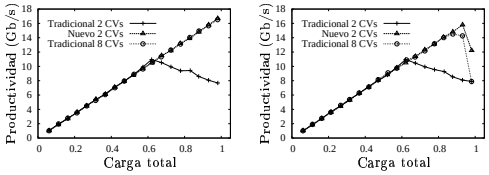


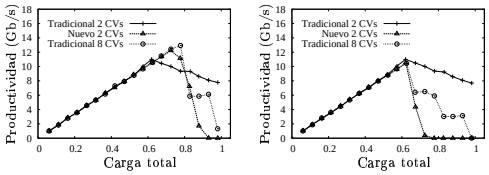
Figura 4: Rendimiento del tráfico de carga controlada.

banda para todos los flujos con necesidades de QoS.

En este punto, es posible concluir que, en lo que respecta al tráfico con necesidades de QoS, nuestra propuesta de tan sólo 2 CVs puede producir un rendimiento aceptable.



(a) Excellent-effort (b) Pref. Best-effort



(c) Best-effort (d) Background

Figura 5: Rendimiento del tráfico Excellent-effort.

Una vez estudiado detenidamente el comportamiento de los flujos con necesidad de QoS, debemos estudiar también el comportamiento del tráfico que no tiene estos requisitos. La Figura 5 muestra la productividad para los niveles de servicio best-effort. Para este tipo de tráfico los resultados de jitter no tienen sentido. Sería razonable que el tráfico *Excelent-effort* obtuviera un buen rendimiento, dado que está pensado para aplicaciones con la máxima prioridad dentro del best-effort. Como podemos ver en la figura, los modelos *Tradicional 8 CVs* y *Nuevo 2 CVs* ofrecen la productividad máxima. Sin embargo, el caso *Tradicional 2 CVs* no da tan buen resultado. El motivo es que, para este tipo de conmutadores, todo el tráfico best-effort es igual y, por lo tanto, no ofrecerá ninguna diferenciación en el rendimiento. Para el tráfico *Preferential Best-effort* sucede algo similar.

El tráfico *Best-effort* y el tráfico *Background* tienen la menor prioridad del grupo best-effort. Como se puede ver, el caso *Tradicional 2 CVs* ofrece buenos resultados, justo donde menos se necesitan. En cualquier caso, estos resultados de la propuesta *Tradicional 2 CVs* son los mismos que obtienen los demás niveles de servicio best-effort.

5. Conclusiones y Trabajo Futuro

En la sección anterior hemos evaluado nuestra técnica bajo diversas condiciones. El modelo de referencia para el rendimiento de nuestra técnica es el que tiene tantos CVs como niveles de servicio, que en este caso eran 8. Las prestaciones que se obtenían respecto al tráfico con requisitos de QoS eran muy similares en términos de latencia y jitter, incluyendo los valores máximos. Respecto al tráfico best-effort, de nuevo los resultados eran similares respecto a la productividad. De este modo, se muestra que nuestra propuesta tiene el rendimiento esperado.

Si bien el uso de 2 CVs para ofrecer QoS no es nuevo, nuestra propuesta consigue ofrecer mucho mejores prestaciones respecto a la aproximación tradicional. En cuanto al tráfico que necesita garantías de QoS, el modelo tradicional de 2 CVs no tiene un mal rendimiento, pero con nuestra técnica hay una mejor diferenciación entre las categorías. Sin embargo, respecto al tráfico best-effort, el modelo tradicional ofrece un rendimiento muy pobre, porque no diferencia entre el tráfico más prioritario y el menos relevante. Por otro lado, nuestra propuesta sí que consigue esa diferenciación.

Estamos considerando diversas líneas de

trabajo para dar continuidad a este trabajo. En primer lugar, estamos preparando un estudio sobre modelos de conmutador más complejos que pueden beneficiarse de nuestra propuesta. Además, queremos portar la técnica a otros entornos como encaminadores de Internet y redes en el interior de chips. Por último, también estamos considerando implementar los conmutadores e interfaces de red en un lenguaje de descripción de hardware para examinar las reducciones reales en área y tiempo de arbitraje.

Referencias

[1] Advanced switching core architecture specification. Technical report, available at `http://www.asi-sig.org/specifications` for ASI SIG members.

[2] N. Chrysos and M. Katevenis. Multiple priorities in a two-lane buffered crossbar. In *Proceedings of the IEEE Globecom 2004 Conference*, November 2004.

[3] W. Dally, P. Carvey, and L. Dennison. Architecture of the avici terabit switch/router. In *Proceedings of the 6th Symposium on Hot Interconnects*, 1998.

[4] I. Elhanany, D. Chiou, V. Tabatabaee, R. Noro, and A. Poursepanj. The network processing forum switch fabric benchmark specifications: An overview. *IEEE Network*, March 2005.

[5] IEEE. 802.1D-2004: Standard for local and metropolitan area networks. `http://grouper.ieee.org/groups/802/1/`, 2004.

[6] InfiniBand Trade Association. *InfiniBand architecture specification volume 1. Release 1.0*, October 2000.

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

[8] G. Kornaros, D. Pnevmatikatos, P. Vatsolaki, G. Kalokerinos, C. Xanthaki, D. Mavroidis, D. Serpanos, and M. Katevenis. Implementation of ATLAS I: a single-chip ATM switch with backpressure. In *Proceedings of the 6th Symposium on Hot Interconnects*, 1998.

[9] C. Minkenberg, F. Abel, M. Gusat, R. P. Luijten, and W. Denzel. Current issues in packet switch design. In *ACM SIGCOMM Computer Communication Review*, pages 119–124, January 2003.

[10] D. Miras. A survey on network QoS needs of advanced internet applications. Technical report, Internet2 - QoS Working Group, 2002.

[11] L. Peh and W. Dally. A delay model and speculative architecture for pipelined routers. In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture*, 2001.

[12] G. K. Zipf. *The Psycho-biology of Languages.* Houghton-Miffin, MIT, 1965.