UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA

TESIS DOCTORAL

Contribuciones a la planificación de tráfico con soporte de QoS y paquetes de longitud variable en conmutadores con colas reales de salida

Rubén Pascual Arteaga Mesa

Las Palmas de Gran Canaria, 2008

Resumen

Internet es una de las plataformas más accesible y extendida de la sociedad de la información. En ella se soportan las transferencias de información de servicios tan diversos como una página web, un vídeo musical o un correo electrónico. Con el objetivo de que el usuario obtenga una comunicación satisfactoria, cada uno de estos servicios requiere algún tipo de garantía en las prestaciones de alguno de sus parámetros, como pueden ser la protección del ancho de banda, el control del retardo y el *jitter*, o el ajuste de la tasa de pérdidas de paquetes.

La topología de la red de comunicaciones actual se caracteriza por la transmisión de la información en el dominio óptico y la conmutación de los paquetes en el dominio eléctrico. Este último punto se enfrenta a varios retos, entre los que destaca proporcionar diferentes calidades de servicio a los flujos de datos como consecuencia de la heterogeneidad del tráfico y la reducción de la eficiencia del conmutador causada por el uso de mecanismos de segmentación y reensamblado, requeridos por la transferencia de paquetes de longitud variable a partir de células de tamaño fijo, típicos de arquitecturas basadas en técnicas de almacenamiento con colas a la entrada y matrices de conmutación.

La primera contribución de la presenta Tesis Doctoral gira en torno a la propuesta de una arquitectura de conmutación, cuyas ventajas más destacables incluyen la conmutación de paquetes de longitud variable sin mecanismos de segmentación y reensamblado, la transmisión de paquetes *multicast* de forma eficiente, y la ausencia de matriz de conmutación. Estas características se han logrado desarrollando una topología basada en técnicas de almacenamiento con colas reales de salida, la cual se compone de tarjetas de línea

interconectadas mediante un *backplane multidrop*. Además, se acompaña de un demostrador basado en dispositivos programables y circuitos integrados discretos con el fin de validar el sistema.

El soporte de calidad de servicio de esta arquitectura se ha llevado a cabo gracias a la aportación de un algoritmo de planificación basado en el modelo de *Servicios Diferenciados*, donde se han parametrizado el peso y la prioridad de los flujos de datos a fin de garantizar el ancho de banda y controlar el retardo. Asimismo, se ha tenido en cuenta la generación de ráfagas de tráfico por parte del planificador.

Tanto la arquitectura como el algoritmo de planificación se han descrito y caracterizado a partir de un modelo de alto nivel en lenguaje C y un modelo de bajo nivel en lenguaje *Verilog HDL* a nivel *RTL*. Los resultados obtenidos muestran que las prestaciones del conmutador y del algoritmo de planificación propuesto con tráfico uniforme son ligeramente mejores a una arquitectura *CICQ* y muy cercanas a una arquitectura con colas de salida ideal. Además se han realizado diversas simulaciones en diferentes escenarios de funcionamiento con tráfico uniforme y a ráfagas que han permitido validar el soporte de la calidad de servicio. Finalmente, el algoritmo se ha implementado en un dispositivo programable con el objetivo de evaluar su coste *hardware*.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Acrónimos

ABR	*Available Bit Rate*
AF	*Assured Forwarding*
AMCC	*Applied Micro Circuits Corporation*
ATM	*Asynchronous Transfer Mode*
BE	*Best Effort*
CAC	*Call Admission Control*
CBR	*Constant Bit Rate*
CICQ	*Combined Input Crossbar Queued*
CIOQ	*Combined Input and Output Queue*
CIXB-1	*Combined Input-One-cell-Crosspoint Buffered Crossbar*
CLS	*Controlled Load Service*
CORR	*Carry-Over Round Robin*
CPRR	*Compensation Priority Round Robin*
CSIX	*Common Switch Interface Specification*
DDRR	*Dynamic Deficit Round Robin*
DDS	*Dynamic DiffServ Scheduling*
DiffServ	*Differentiated Services* (Servicios Diferenciados)
DLL	*Delay Locked Loop*
DRR	*Deficit Round Robin*
DS	*Distributed Scheduling*
DSI	Diseño de Sistemas Integrados
DSL	*Digital Subscriber Line*
EF	*Expedited Forwarding*

FAP	*Fabric Access Processor*
FE	*Fabric Element*
FIFO	*First-In, First-Out*
FPGA	*Field Programmable Gate Array*
GBSBF-LBF	*Guaranteed Bandwidth Smallest Buffer First - Largest Buffer First*
GMDS	*Gigabit MultiDrop Switch*
GPS	*General Processor Sharing*
GS	*Guaranteed Service*
HDL	*Hardware Description Language*
HDS	*Hierarchical DiffServ Scheduling*
HoL	*Head-of-Line*
HRR	*Hierarchical Round Robin*
IBP	*Interrupted Bernoulli Process*
IP	*Internet Protocol*
IPS	*Iterative Probabilistic Scheduling*
ISE	*Integrated Software Environment*
iSLIP	*Iterative Round Robin with Slip*
IUMA	Instituto Universitario de Microelectrónica Aplicada
MOQ	*Multiple Output Queue*
MT	Marco de Tiempo
OQ	*Output Queue*
PFQ	*Packet Fair Queuing*
PIFO	*Push-In, First-Out*
PIM	*Parallel Iterative Matching*
PLL	*Phase Locked Loop*
PQWRR	*Priority Queuing - Weighted Round Robin*
PWDRR	*Priority Weighted Double Round Robin*
PWRR	*Priority Weighted Round Robin*

QoS	*Quality of Service*
RAM	*Random Access Memory*
RED	*Random Early Detection*
RR	*Round Robin*
RTL	*Register Transfer Logic*
SDM	*Space-Division Multiplexing*
SDRR	*Shaped Deficit Round Robin*
SVC	*Shaped Virtual Clock*
TDM	*Time-Division Multiplexing*
ToS	*Type of Service*
UCE	Última Clase Enviada
VBR	*Variable Bit Rate*
VOQ	*Virtual Output Queue*
WDRR	*Weighted Deficit Round Robin*
WF^{2}Q	*Worst-Case Fair Weighted Fair Queueing*
WF^{2}Q+	*Worst-Case Fair Weighted Fair Queueing*
WFI	*Worst-case Fair Index*
WFQ	*Weighted Fair Queuing*
WM	*Weight Matrix*
WRED	*Weighted Random Early Detection*
WRR	*Weighted Round Robin*
WSS	*Weight Spread Sequence*
ZBT	*Zero Bus Turnaround*

Capítulo 1

Introducción

Actualmente, Internet es la plataforma de comunicación más extendida y accesible de la sociedad de la información a nivel mundial. En el seno de la red, la tasa de transferencia disponible se ha incrementado gracias a la evolución tecnológica de los enlaces, fundamentalmente. En paralelo, los elementos de conmutación han mejorado sus prestaciones y han ampliado el espectro de las aplicaciones soportadas, de forma que hoy en día es posible oír música, visualizar vídeos y realizar videoconferencias a través de Internet. Además, las tecnologías actuales de acceso de banda ancha, como *DSL* (*Digital Subscriber Line*) o el cable, se han popularizado y proporcionan una conexión asequible desde el hogar. En el entorno empresarial, la implantación progresiva de la tecnología *Gigabit Ethernet* ha facilitado el incremento de la velocidad de acceso. El despliegue de estas tecnologías de acceso de banda ancha y su gran acogida por parte de los usuarios han llevado consigo una alta demanda de recursos. En consecuencia, y con el objetivo de mantener este ritmo de crecimiento, las comunidades científica e industrial continúan explorando nuevas tecnologías de conmutación y transmisión de información.

No obstante, el explosivo crecimiento del número de usuarios[1] y la convergencia de los servicios de voz, datos y vídeo acentúan las debilidades de las redes de comunicaciones actuales [LK05a][CMY+07]. Las redes de datos

[1]Internet ha alcanzado la cifra de 1400 millones de usuarios a nivel mundial este año, lo que supone un crecimiento del 304% respecto al año 2000. Se estima que en el año 2012 esta cantidad alcanzará los 1800 millones de usuarios a nivel mundial. Fuentes: *Internet World Stats* y *Jupiter Research*, 2008.

tradicionales proporcionan únicamente servicios sin garantías. Sin embargo, existe la necesidad de priorizar el tráfico con el objetivo de diferenciar los flujos de datos de cada aplicación y, como resultado, obtener unas prestaciones satisfactorias. De esta manera, la situación actual se caracteriza por una elevada tasa de transferencia que presenta una eficiencia moderada debido a los limitados mecanismos existentes para soportar calidad de servicio (*Quality of Service - QoS*). Por lo tanto, Internet necesita evolucionar mediante nuevas tecnologías y estándares capaces de obtener el máximo rendimiento del ancho de banda disponible con garantías de calidad de servicio. Como resultado, aparecen multitud de soluciones capaces de plantear nuevas arquitecturas y modificar la gestión del control de las redes de comunicaciones.

A tenor de las cuestiones planteadas anteriormente, la presente Tesis Doctoral se centra en proponer soluciones a la conmutación del tráfico que mejoren las prestaciones de la actual generación de arquitecturas de conmutación. Además, la convergencia de las diferentes aplicaciones con diversos requisitos de funcionamiento impone que esta mejora incluya mecanismos flexibles y adaptables de soporte de calidad de servicio.

1.1 Planteamiento del problema

En los últimos años se ha producido una evolución progresiva en las redes de comunicación. El mercado, independientemente de que sea analizado desde el punto de vista del usuario o del proveedor, ha proporcionado nuevas oportunidades de negocio que se han traducido en ingentes ingresos para el sector. Como referencia, en la Figura 1.1 se representan los ingresos estimados de los prestadores de servicio a nivel mundial, destacando la diversidad de aplicaciones existentes y el notorio crecimiento de los servicios de datos vinculados a dispositivos móviles, la televisión digital y la banda ancha.

Este último servicio ha impulsado la penetración de Internet a nivel mundial hasta el punto de que en EE.UU. o Europa alcanza actualmente casi

al 75% y al 45% de la población, respectivamente[2]. Además, los servicios de banda ancha constituyen un foco de interés, puesto que su infraestructura es el vehículo de distribución de muchos de los nuevos servicios. Sin embargo, como estos servicios no fueron previstos en los orígenes de la concepción de la red, centrada principalmente en la transmisión de datos, los protocolos desarrollados originariamente deben actualizarse a fin de soportar la transmisión integrada de voz, vídeo y datos multimedia con diferentes requisitos.

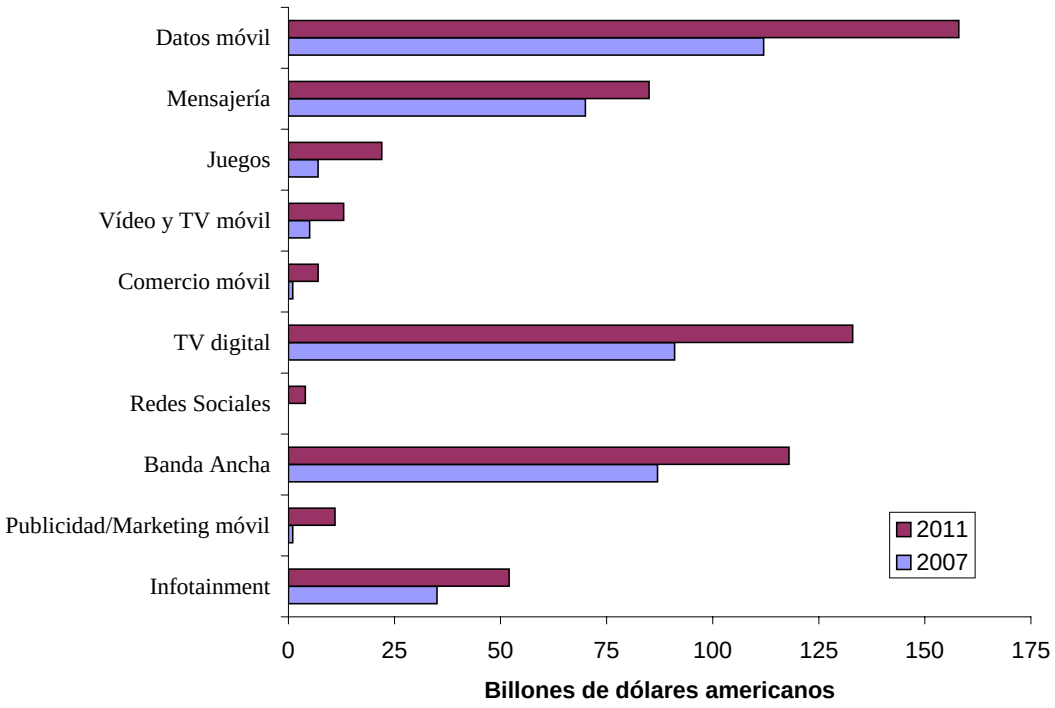


Figura 1.1: Estimación de los ingresos generados por diferentes servicios a nivel mundial[a]

[a]Fuente: *The Yankee Group*, 2007.

En la Tabla 1.1 se comparan, *grosso modo*, las características de los conmutadores de la red *IP* (*Internet Protocol*) original y actual según diferentes criterios. De acuerdo con las características heterogéneas del tráfico de la red *IP* actual, se pueden dividir las aplicaciones y, por lo tanto, sus flujos de datos, en varias categorías generales dependiendo de sus requisitos principales: an-

[2]Fuente: *Internet World Stats*, 2008.

cho de banda, retardo y tasa de pérdidas de paquetes. En consecuencia, durante la transmisión de una secuencia de vídeo, una videoconferencia o un fichero, la red debe asegurar un determinado ancho de banda mínimo, un retardo máximo o una tasa de pérdidas de paquetes en función del tipo de transmisión, a fin de obtener una conexión satisfactoria.

	Red *IP* original	Red *IP* actual
Paquetes	Ninguna distinción	Prioridades, pesos, sellos de tiempo, dependencias,...
Objetivo	Maximizar el número total de paquetes enviados	Maximizar el peso total de los paquetes enviados
Limitaciones	Sólo servicio *best effort*	Posible eliminación o retraso de los paquetes *best effort* sin calidad de servicio
Servicios	Correo electrónico, www, ftp, ...	Videoconferencia, vídeo sobre *IP*, juegos en red,...

Tabla 1.1: Comparación de las características de las redes *IP*

Asimismo, añadir garantías de calidad de servicio manteniendo una utilización eficiente de los recursos de la red aún representa un reto. Casi todas las aplicaciones multimedia generan tráfico periódico con tasas variables por un periodo de tiempo prolongado que requiere la disponibilidad de ciertos recursos de la red. A fin de satisfacer una determinada calidad de servicio, el tráfico reserva los recursos continuamente. Sin embargo, la asignación de recursos basada en las tasas de pico podría resultar en una pobre eficiencia de la red, provocando una reserva de recursos excesiva. Reconociendo y analizando este problema, los proveedores de servicios de Internet permiten cierta reserva adicional de recursos utilizando la técnica de *multiplexación estadística*. No obstante, su utilización tiende a crear problemas puntuales de exceso de carga que se resuelven a nivel de conmutación [LHD+04].

La *multiplexación estadística* es una técnica enfocada a la gestión de los flujos de datos en lugar de al soporte de diferentes calidades de servicio. Como

consecuencia de ello, se han propuesto modelos específicos con este fin entre los cuales cabe mencionar las tecnologías de red denominadas *Servicios Integrados* [BCS94] y *Servicios Diferenciados* [BBC+98]. Los Servicios Integrados se caracterizan por la reserva de recursos para cada flujo individual de datos. Dicha reserva abarca toda la ruta que sigue un paquete para proporcionar calidad de servicio asegurando el retardo y el ancho de banda para aplicaciones en tiempo real intolerantes al *jitter* y a la pérdida de paquetes. Básicamente, los Servicios Integrados aseguran el soporte de diferentes calidades de servicio de cada flujo específico de datos. Desafortunadamente, la información del estado de cada conexión se debe mantener individualmente en cada conmutador, lo que representa una enorme capacidad de almacenamiento y procesamiento. De aquí que esta solución plantee serias dudas sobre su escalabilidad en las redes troncales (*backbone networks*) de Internet.

En la otra opción, los Servicios Diferenciados, el tráfico se clasifica en función de la prioridad y se encamina usando mecanismos individuales por nodo. Esta aproximación permite que el tráfico con características de servicio similares atraviese múltiples nodos con un comportamiento parecido. Las ventajas que presenta la tecnología de Servicios Diferenciados sobre la de Servicios Integrados se pueden resumir en la eliminación de la señalización extremo-a-extremo y la mejora de la eficiencia, ya que la clasificación y la planificación de paquetes se basan en clases de tráfico en lugar de en flujos de datos. No obstante, esta solución no proporciona garantías de calidad de servicio extremo-a-extremo, sino un tratamiento preferencial de los paquetes en cada nodo de la red.

Los Servicios Diferenciados definen un campo en la cabecera de los paquetes, denominado *ToS* (*Type of Service*), que determina el tratamiento que recibe un paquete en cada uno de los nodos de la red de acuerdo con las diferentes calidades de servicio previamente establecidas. En la actualidad se proporciona el mejor servicio al tráfico de la clase *EF* (*Expedited Forwarding*) y se asegura el envío de los paquetes de la clase *AF* (*Assured Forwarding*) asignando diferentes prioridades de descarte. Los paquetes de la clase

BE (*Best Effort forwarding*) se siguen transmitiendo en la red *IP* actual sin añadir ningún soporte de calidad de servicio. Para garantizar los diferentes tratamientos a las clases de tráfico se disponen elementos independientes de almacenamiento en los diferentes nodos. Como referencia, en la Figura 1.2 se representa el funcionamiento de una red con soporte para diferentes calidades de servicio basada en el modelo de Servicios Diferenciados.

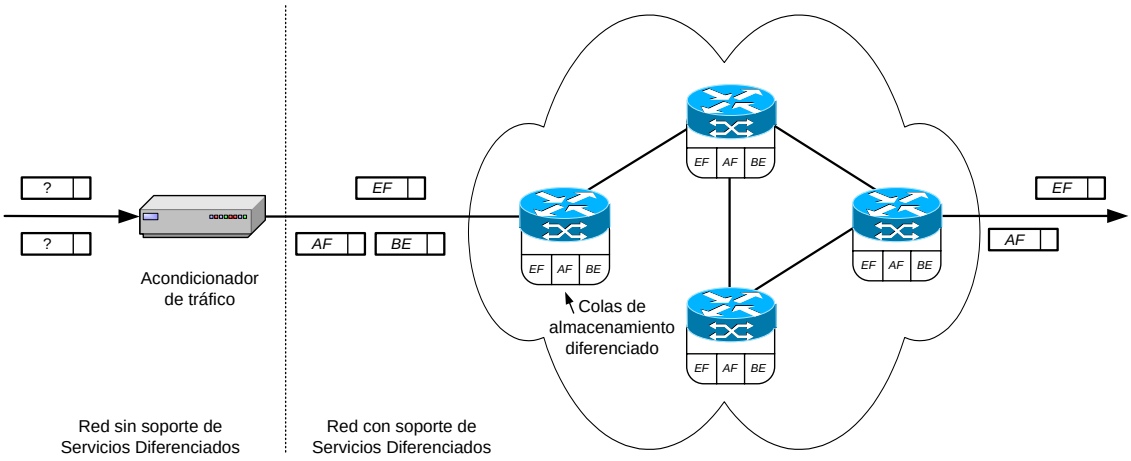


Figura 1.2: Funcionamiento de una red con soporte para diferentes calidades de servicio basada en el modelo de Servicios Diferenciados

Al mismo tiempo que se produjo la evolución técnica de los enlaces de fibra óptica y la convergencia de diferentes servicios en la misma red de comunicación, las arquitecturas de conmutación que constituían Internet fueron objeto de una intensa labor investigadora por parte de la comunidad científica con el objetivo de solventar ambos aspectos.

A finales de los años 80 y principios de los 90, el objetivo principal de los conmutadores era proporcionar el máximo *throughput*. La arquitectura más utilizada en ese momento se basaba en la ubicación de las colas de almacenamiento en los puertos de salida. Sin embargo, estos esquemas requerían una velocidad interna de transferencia de información superior a la de los enlaces externos. Por lo tanto, muchas propuestas se centraron en superar las limitaciones de la velocidad interna. Un ejemplo representativo es el conmutador *Knockout* [YHA87] basado en el empleo de un concentrador, elemento capaz

de reducir la aceleración interna del conmutador a costa de aumentar la tasa de pérdidas de paquetes.

El incremento de las necesidades de ancho de banda, junto al aumento de las velocidades de transferencia alcanzadas por los enlaces de fibra óptica, hizo inviables los conmutadores de alta velocidad con colas a la salida. Con el fin de superar esta limitación interna de la velocidad, los investigadores centraron sus esfuerzos en conmutadores con colas a la entrada a pesar de las pérdidas de prestaciones que presentaban frente a los conmutadores con colas a la salida debido al bloqueo de cabecera (*Head-Of-Line blocking - HOL blocking*) [KHM87].

A principios de los años 90 se empezaron a imponer los conmutadores con colas virtuales de salida (*Virtual Output Queue - VOQ*) propuestos en [AN89] donde cada puerto de entrada dispone de una cola por cada puerto de salida. La principal ventaja que aportaron fue la reducción de la velocidad interna de transferencia de información frente a los conmutadores con colas a la salida y la mejora de las prestaciones frente a un conmutador con colas a la entrada. Sin embargo, su mayor inconveniente es que requiere de un proceso de planificación mucho más complejo que en un conmutador con colas a la salida, y que presenta una capacidad limitada de proporcionar calidad de servicio.

Otra cuestión que deben afrontar los conmutadores son los paquetes de longitud variable. Los conmutadores actuales, basados en colas a la entrada, requieren la segmentación de los paquetes en células de tamaño fijo en los puertos de entrada [CYR+04][KP05] y el reensamblado en los puertos de salida. Para que el funcionamiento de dichos mecanismos sea satisfactorio se deben añadir memorias de almacenamiento adicionales en ambas localizaciones. Además, la velocidad interna de la matriz de conmutación se debe incrementar para compensar la pérdida de eficiencia interna debida a la información de *overhead* añadida en las tramas de longitud fija y la pérdida de ancho de banda debida a la transmisión de células incompletas. Por último, el algoritmo de planificación debe considerar que la transferencia interna de células de tamaño fijo corresponde realmente a un paquete de longitud varia-

ble [MBG+01][GKS05]. La arquitectura genérica de un conmutador con estas características se muestra en la Figura 1.3.

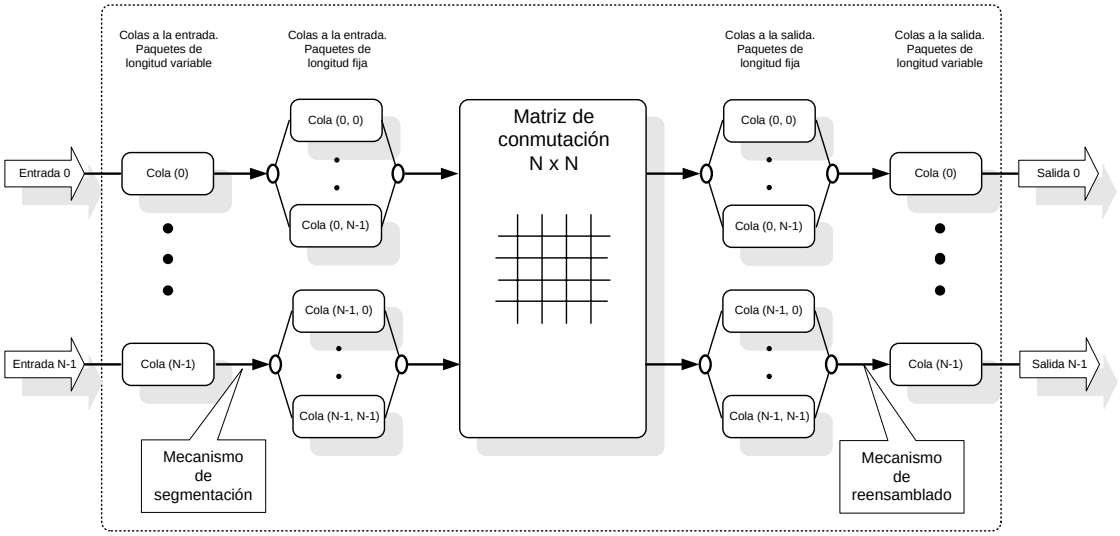


Figura 1.3: Arquitectura de un conmutador de alta velocidad con soporte para paquetes *IP*

En resumen, el modelo de Servicios Diferenciados ha alcanzado una gran relevancia debido a la necesidad de soportar diferentes calidades de servicio en una red de comunicaciones con servicios heterogéneos y en constante crecimiento. Asegurar estas garantías resulta una tarea compleja debido a que las arquitecturas de conmutación actuales se basan, mayoritariamente, en técnicas de almacenamiento con colas a la entrada. Además, los paquetes de longitud variable incrementan la complejidad del conmutador al introducir las etapas de segmentación y reensamblado. Todos estos aspectos han motivado un elevado interés por proponer soluciones que puedan solventar el soporte de calidad de servicio conjuntamente con la conmutación de paquetes de longitud variable.

1.2 Motivación de la tesis

La situación actual de las redes de comunicación se caracteriza por un crecimiento del número de usuarios debido al despliegue de las tecnologías de

acceso y la convergencia de diversas aplicaciones que demandan soporte de calidad de servicio [LK05a]. La generación actual de conmutadores y *routers* de tráfico, como los dispositivos *Ethernet Routing Switch 8600* [Nor1] de *Nortel* [Nor], *Big Iron Ethernet Switch* [FN1] de *Foundry Networks* [FN], *EX-8200 Ethernet Switch* [JN1] de *Juniper Networks* [JN] o *Catalyst 6500* [CS1] de *Cisco Systems* [CS], solventan estos aspectos con un elevado coste.

Sirva como ejemplo ilustrativo el dispositivo de la serie *Catalyst* de *Cisco*[3]. Este conmutador se basa en una matriz *crossbar* con un soporte limitado de calidad de servicio gracias a los algoritmos de prioridad estricta y *Weighted Round Robin* (*WRR*). Además, la solución adoptada con el fin de mejorar las prestaciones en este dispositivo consiste en incrementar la tasa de transferencia interna tres veces respecto a las tasas de los enlaces externos [CS2]. Esta situación se puede extender a los restantes conmutadores comerciales citados previamente. La aceleración interna dependerá, principalmente, de las tasas de los enlaces externos, de las interfaces de interconexión y de la implementación y arquitectura interna de la matriz de conmutación [DN1].

Profundizando en los detalles internos del ejemplo propuesto, su topología se fundamenta en la interconexión de las tarjetas de línea gracias a la utilización de una matriz de conmutación en un *backplane* con enlaces serie punto-a-punto. Esta arquitectura plantea ciertas dudas, como son la reducida capacidad de proporcionar calidad de servicio debido a los algoritmos de planificación implementados, la cantidad de colas de almacenamiento presentes en los puertos de entrada y salida necesarias para soportar la aceleración interna, y la transferencia de los paquetes *multicast* puesto que los enlaces serie punto-a-punto obligan a replicar los paquetes. Todos estos aspectos incrementan la complejidad y reducen las prestaciones de los conmutadores, por lo que originan un gran interés investigador en torno a las propuestas que aporten mejoras sobre estas cuestiones.

[3]En el capítulo 2 se puede consultar una explicación más detallada acerca de las topologías de las matrices de conmutación, los algoritmos de planificación y las técnicas de almacenamiento.

Otro aspecto a destacar es la utilización de matrices de conmutación *crossbar*. Las limitaciones tecnológicas impiden la implementación de conmutadores con colas a la salida, los cuales representan el mejor compromiso entre el retardo y el *throughput* [HK88]. Concretamente, la velocidad de acceso a la memoria en conmutadores con un gran número de puertos plantea dificultades al alcanzar tasas extremadamente elevadas. Una arquitectura con colas a la salida que fuera capaz de solventar la limitación de la velocidad de acceso a las memorias sin producir pérdidas de paquetes tendría un gran valor investigador y comercial.

Dentro de los tipos de tráfico que se generan y transmiten en Internet, los paquetes de longitud variable resultan de especial importancia debido al esfuerzo que requiere su procesado. Los conmutadores actuales introducen elementos de almacenamiento en sus puertos de entrada y salida con el objetivo de realizar las tareas de segmentación y reensamblado y la consiguiente pérdida de ancho de banda. La propuesta de una arquitectura sin la necesidad de aplicar estos mecanismos para conmutar el tráfico de longitud variable sería de mucho interés, ya que, por un lado, reduciría la complejidad del conmutador y, por otro lado, evitaría introducir aceleración en la matriz de conmutación a fin de compensar la pérdida de ancho de banda debida al *overhead*.

Independientemente de la arquitectura de la matriz de conmutación y de la eficiencia interna, todos los conmutadores deben incluir un criterio de planificación del tráfico. El algoritmo elegido debe proporcionar calidad de servicio a los diferentes flujos de datos. Sin embargo, los algoritmos implementados en los tres conmutadores comerciales previamente mencionados consisten en algoritmos de prioridad estricta con *Weighted Round Robin* (*WRR*), *Shaped Deficit Weight* o *Weighted Fair Queuing* (*WFQ*) basados en la técnica de Servicios Diferenciados. Por lo tanto, se hace patente la limitación en los conmutadores de alta velocidad de proporcionar calidad de servicio. Además, estas propuestas presentan limitaciones frente a la conmutación de los paquetes de longitud variable, que deben hacer uso de mecanismos de segmentación y

reensamblado. Debido a estas cuestiones, se plantea la planificación de tráfico como un tema central de esta Tesis Doctoral.

1.3 Objetivos de la tesis

La finalidad de esta Tesis Doctoral es proporcionar soluciones a la planificación de tráfico con requisitos de calidad de servicio, proponiendo una arquitectura de conmutación de altas prestaciones. A fin de alcanzar estos objetivos, la presente Tesis Doctoral se desglosa en los siguientes hitos:

- Estudio previo de las arquitecturas de conmutación, que incluirá un análisis detallado de las topologías de la matriz de conmutación y de las técnicas de almacenamiento de los paquetes. Este estudio se ampliará posteriormente al soporte de calidad de servicio, poniendo especial énfasis en propuestas arquitecturales basadas en el modelo de Servicios Diferenciados.

- Concepción y desarrollo de una arquitectura con colas a la salida. A fin de desarrollar este sistema, se deberá solventar, principalmente, el aspecto limitador de la velocidad de acceso a la memoria que redundará en la escalabilidad del sistema y la conmutación de los paquetes de longitud variable.

- Verificación de la arquitectura propuesta previamente. Dicha validación se llevará a cabo en primera instancia a partir de un modelo de alto nivel del sistema de conmutación que permitirá comparar las bondades del sistema propuesto con las investigaciones previas en esta área. Posteriormente se validará la arquitectura mediante un demostrador basado en dispositivos programables que incluirá toda la funcionalidad de la arquitectura propuesta.

- Propuesta de un algoritmo de planificación de tráfico capaz de soportar calidad de servicio para un conmutador con colas a la salida. Dicho planificador se estudiará e implementará conjuntamente con el demostrador y

deberá proporcionar garantías de *throughput*, ancho de banda y retardo de los paquetes maximizando el aprovechamiento de las ventajas de la arquitectura con colas a la salida. Además, se deberá tener en cuenta la transmisión de paquetes de longitud variable.

La consecución de estos objetivos es posible gracias a la experiencia obtenida en la División de *Diseño de Sistemas Integrados* (*DSI*) del *Instituto Universitario de Microelectrónica Aplicada* (*IUMA*) a partir de varios contratos con empresas dentro del área del diseño de arquitecturas de conmutación [Cross99][Giga00][Tera02] y diversas actividades de $I + D + i$ relacionadas con la planificación de tráfico [TIC2002-02998][Tob01].

1.4 Estructura de la tesis

Siguiendo el orden de los objetivos propuestos, este documento se organiza en cinco capítulos, de los cuales el capítulo actual lo constituye esta introducción.

En el segundo capítulo se revisan las diferentes arquitecturas de conmutación, así como sus técnicas de almacenamiento. Posteriormente, se presentan algunas técnicas de calidad de servicio desde una visión arquitectural enfocada a una posible implementación *hardware*. A raíz de las propuestas previas estudiadas, este capítulo concluye que la arquitectura con colas a la salida proporciona el mejor rendimiento frente a las restantes topologías de conmutación y sienta las bases algorítmicas para el soporte de calidad de servicio.

En el tercer capítulo se propone una nueva arquitectura de conmutación con colas a la salida compuesta por diversas tarjetas de línea y un *backplane* serie *multidrop* de alta velocidad. Ambos elementos se describen detalladamente a lo largo del capítulo desde el punto de vista arquitectural con el fin de realizar su implementación y mostrar las soluciones propuestas a los principales retos que presenta esta arquitectura.

En el cuarto capítulo, y dada su importancia, se estudian las prestaciones de la arquitectura propuesta con diferentes algoritmos de planificación. Se

proponen dos versiones diferentes donde, en primer lugar, se parte de un entorno sin soporte de calidad de servicio. El algoritmo propuesto para dicho entorno permite validar y comparar las prestaciones de esta arquitectura con otras propuestas significativas. En segundo lugar, se propone un algoritmo de planificación con mecanismos que permitan asegurar el soporte de la calidad de servicio. Los parámetros de configuración de dicho algoritmo permiten controlar el ancho de banda y el retardo gracias a la utilización de pesos y prioridades. En todos los casos se evalúan, junto con la arquitectura de conmutación y paquetes de longitud variable en diferentes escenarios y desde el punto de vista del modelo de alto nivel y de su implementación. La idoneidad de los algoritmos propuestos dependerá de las condiciones del tráfico de la red.

Por último, en el capítulo final, se presentan las conclusiones extraídas a partir del trabajo realizado y se plantean las líneas de investigación futuras.

Capítulo 2

Conmutadores de paquetes de alta velocidad

2.1 Introducción

En este capítulo se estudian diferentes arquitecturas de conmutación, así como diversas técnicas de almacenamiento, analizando especialmente aquellas que proporcionan un mejor rendimiento y facilitan el soporte de diversas aplicaciones con diferentes requisitos. La gestión de estas características vendrá dada, principalmente, por el algoritmo de planificación, aspecto que se aborda en profundidad en la segunda mitad de este capítulo.

2.2 Arquitecturas de conmutación

Los conmutadores se pueden clasificar en diferentes grupos atendiendo a diversas características como pueden ser, entre otras, el modo de multiplexación de la información, la situación de los elementos de almacenamiento, o la topología de la matriz de conmutación. Cada uno de estos aspectos desempeña un papel determinante en las prestaciones y en la complejidad del conmutador. No obstante, debido a la extensión que puede alcanzar un análisis profundo de todos los atributos de las arquitecturas de conmutación [Tob90][New92] [RCG94][AM95][CLO01][Tse05], se particulariza el estudio en uno de los fac-

tores más determinantes en la arquitectura resultante: las técnicas de multiplexación.

Concretamente, en esta sección se analizan las técnicas de multiplexación de la información en las que se basan los conmutadores: división en el tiempo (*Time-Division Multiplexing - TDM*) y división en el espacio (*Space-Division Multiplexing - SDM*). Posteriormente se presentan las diferentes arquitecturas de conmutación, así como sus ventajas y limitaciones.

2.2.1 Conmutadores por división en el tiempo

En los conmutadores por división en el tiempo se dispone de una única estructura de comunicación interna compartida por todos los paquetes que se transmiten desde los puertos de entrada a los puertos de salida [IM01]. La estructura interna puede estar constituida por un bus, un anillo, o una memoria, y permite que cualquier paquete entrante esté disponible en todos los puertos de salida, lo que facilita las transmisiones *multicast*. Sin embargo, la principal desventaja de esta arquitectura es que su escalabilidad está limitada por la capacidad de la estructura interna.

2.2.1.1 Conmutador con medio compartido

Dentro de los conmutadores por división en el tiempo se incluyen las arquitecturas de conmutación con medio compartido. Los paquetes entrantes se multiplexan por división en el tiempo en un medio compartido de alta velocidad de transferencia, como un bus o un anillo, cuyo ancho de banda es igual a N veces la tasa de los enlaces de entrada, siendo N el número de puertos de entrada. El *throughput* de este medio compartido determina la capacidad del conmutador. Como se muestra en la Figura 2.1, cada línea de salida se conecta al medio compartido de alta velocidad mediante una interfaz consistente en un filtro de direcciones y una memoria. Cada filtro examina la cabecera de todos los paquetes entrantes y acepta solamente aquellos destinados a su propia salida. Esta aproximación descentralizada tiene la ventaja de que cada puerto de salida es independiente y se puede implementar por

separado. Sin embargo, cuando un puerto de salida se congestiona temporalmente debido a una carga de tráfico elevada y comienza a descartar paquetes, los restantes puertos no pueden compartir sus recursos. En este caso, un conmutador con memoria compartida es capaz de agrupar todos los recursos de almacenamiento y mejorar su aprovechamiento.

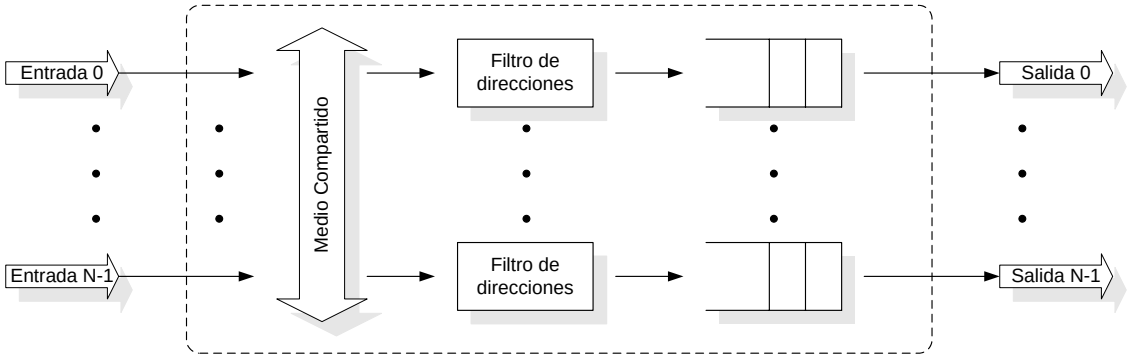


Figura 2.1: Arquitectura de un conmutador con medio compartido

2.2.1.2 Conmutador con memoria compartida

En un conmutador con memoria compartida los paquetes entrantes se multiplexan por división en el tiempo dentro de un flujo de datos único y se escriben ordenadamente en la memoria compartida. El encaminamiento de los paquetes se realiza extrayendo los paquetes almacenados a fin de formar un único flujo de datos de salida. Un módulo de control se encarga de proporcionar las direcciones de las posiciones de lectura de los paquetes salientes en función de la información de encaminamiento extraída de sus cabeceras.

2.2.2 Conmutadores por división en el espacio

En la multiplexación por división en el tiempo se comparte una única estructura de comunicaciones interna entre todos los puertos de entrada y salida, mientras que en la multiplexación por división en el espacio se proporcionan múltiples caminos físicos entre los puertos de entrada y salida. Las rutas entre los puertos de entrada y salida funcionan concurrentemente, por lo que

se pueden transmitir varios paquetes simultáneamente a través del conmutador. De esta manera, la capacidad total del conmutador es el producto del ancho de banda de cada ruta, por el número de paquetes que pueden transmitir simultáneamente. Así, la capacidad total del conmutador es teóricamente ilimitada. Sin embargo, en la práctica, las restricciones físicas inherentes a la implementación en aspectos como la sincronización o el número de pines, limita la capacidad total del conmutador.

Los conmutadores por multiplexación en el espacio se clasifican basándose en el número de rutas disponibles entre cualquier combinación de puertos de entrada y salida. En los conmutadores de ruta única sólo existe un camino entre cada combinación entre los puertos de entrada y salida, mientras que en los conmutadores de ruta múltiple existe más de un camino. Comparando ambas arquitecturas, en el primer caso se necesita un sistema de control de encaminamiento más simple que en el segundo. Sin embargo, en este último caso se mejora la tolerancia a fallos.

2.2.2.1 Conmutador con ruta única

Un conmutador de especial relevancia dentro de este grupo se caracteriza por presentar una topología *crossbar* $N \times N$, siendo N el número de puertos, como se observa en la Figura 2.2. Esta arquitectura consiste en una matriz cuadrada de N^2 elementos de cruce donde cada uno de ellos opera individualmente y corresponde a una combinación de puertos de entrada y salida. Cada elemento de cruce se activa independientemente con la información que proporcionan los paquetes entrantes sobre su puerto de salida. Esta propiedad de encaminamiento autónomo reduce la complejidad de la lógica de control y, al mismo tiempo, la distribuye entre todos los elementos de cruce. Además, se puede destacar la modularidad del sistema y la simplicidad de la arquitectura. Sin embargo, el número de elementos de cruce aumenta exponencialmente con el número de puertos, por lo que la escalabilidad del conmutador queda comprometida [RCG94].

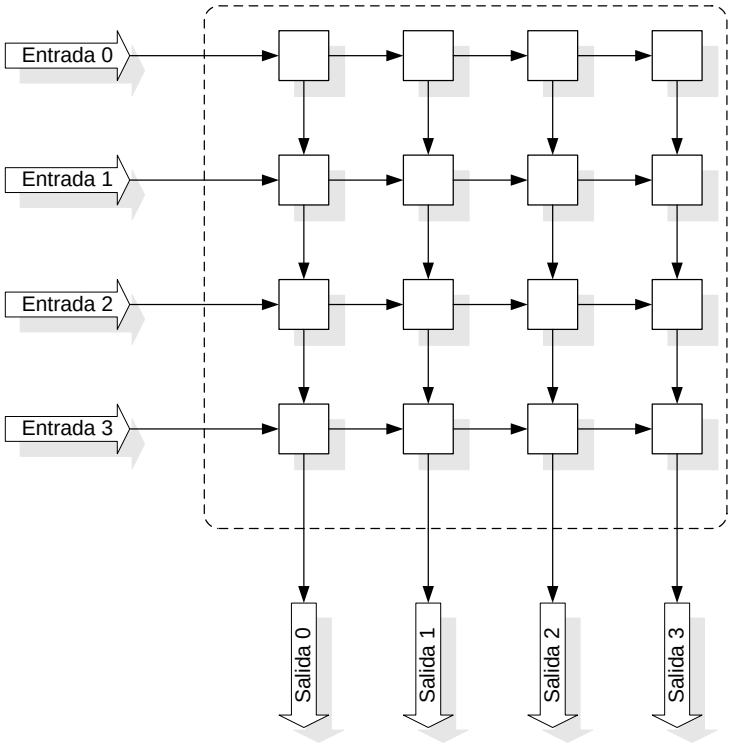


Figura 2.2: Arquitectura de un conmutador *crossbar* 4×4

Otra topología con ruta única es la del conmutador completamente interconectado, caracterizado por reemplazar el requisito del incremento de velocidad de las topologías con un medio compartido, por buses individuales entre todos los puertos de entrada y salida. La arquitectura obtenida es simple y no bloqueante aunque el coste *hardware* es alto.

Por último, los conmutadores *Banyan* se caracterizan por el encaminamiento automático y por componerse de elementos de conmutación 2×2, como se muestra en la Figura 2.3. Estos elementos se conectan en etapas sucesivas, siendo el número de elementos requeridos una de sus ventajas más importantes. Comparando con los conmutadores *crossbar* o completamente interconectados, esta familia incrementa el número de elementos de conmutación logarítmicamente en vez de exponencialmente, mejorando la escalabilidad. Sin embargo, pueden producirse pérdidas [Lea90] en la matriz de conmutación debido al acceso simultáneo de dos paquetes al mismo enlace interno entre los elementos de conmutación 2×2.

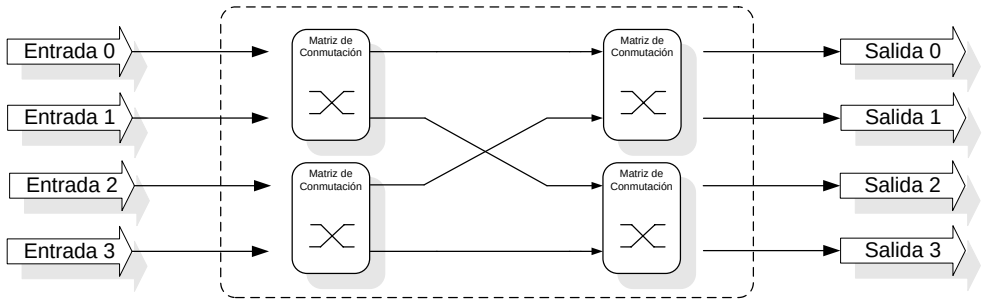


Figura 2.3: Arquitectura de un conmutador *Banyan* 4×4

2.3 Técnicas de almacenamiento

Los flujos de datos en una red de comunicaciones pueden tener el mismo destino. En esos casos, los paquetes pertenecientes a dichos flujos pueden colisionar al competir entre ellos por los mismos recursos, como por ejemplo, un enlace interno de la matriz de conmutación o un puerto de salida. Cuando existe la posibilidad de una colisión interna en la matriz de conmutación, se dice que ésta es bloqueante. En el caso del acceso simultáneo al mismo puerto de salida, se denomina contención de los puertos de salida. En ambos casos, las opciones son transmitir un paquete y descartar o almacenar los restantes.

Las primeras investigaciones sobre las técnicas de almacenamiento demostraron que los conmutadores con memorias independientes situadas en los puertos de salida alcanzaban un rendimiento óptimo entre el retardo y el *throughput* del sistema [HK88]. Sin embargo, debido al aumento de la demanda de conmutadores de alta capacidad, tanto la tasa de los enlaces como el número de puertos se incrementaron. En consecuencia, los requisitos de velocidad de las memorias y de la matriz de conmutación tuvieron que adaptarse y se convirtieron en el principal elemento limitador de la escalabilidad de los sistemas con colas a la salida [SZ98a][MRS03]. Sin embargo, a fin de diseñar conmutadores que alcanzaran una mayor velocidad y número de puertos, la actividad investigadora también se centró en conmutadores con memorias situadas en los puertos de entrada o salida, con memoria interna, o que combinaran las configuraciones previas de almacenamiento.

2.3.1 Técnica de almacenamiento con colas a la entrada

Cuando las memorias se localizan en los puertos de entrada del conmutador, como se presenta en la Figura 2.4, se dice que el conmutador está utilizando la técnica de almacenamiento con colas a la entrada. Usualmente, el funcionamiento de la memoria sigue una política *FIFO* (*First-In, First-Out*), de manera que el paquete recibido en primer lugar en la memoria se atiende antes que los recibidos posteriormente.

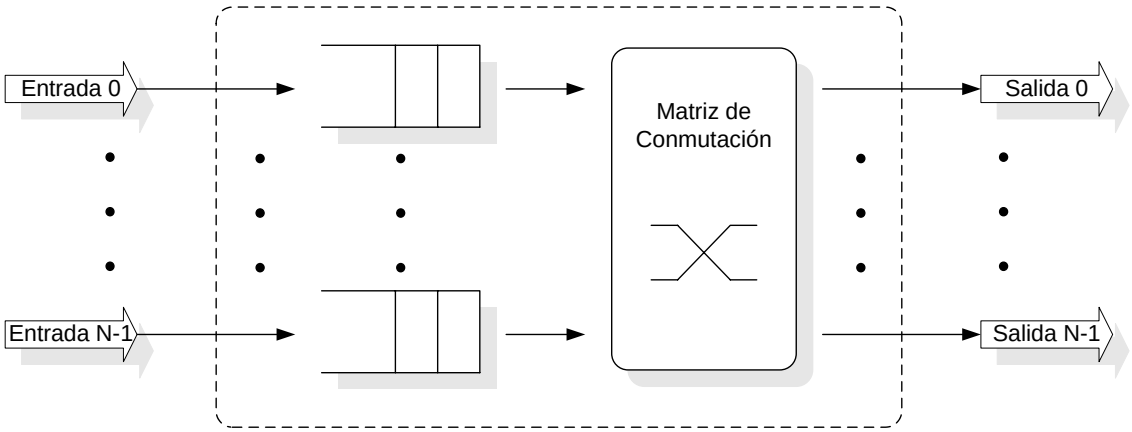


Figura 2.4: Arquitectura de un conmutador con colas a la entrada

Esta topología simplifica significativamente la implementación del conmutador, ya que la matriz de conmutación opera a la misma velocidad que los enlaces externos. Además, cada cola funciona en paralelo, por lo que el *throughput* total del conmutador es la suma del *throughput* de todas las memorias que componen las colas. Estas características proporcionan una gran escalabilidad a este tipo de arquitecturas.

En contraposición, un conmutador con colas a la entrada necesita almacenar los paquetes entrantes antes de la planificación y la transmisión, ya que si se produce una colisión en un puerto de salida se debe descartar el paquete. En caso de que haya una única cola por puerto y la política de servicio sea *FIFO*, sólo se conmutan aquellos paquetes que ocupen las cabeceras de sus correspondientes colas. En consecuencia, los restantes paquetes que no están situados en la cabecera deben esperar hasta alcanzar la primera posición. A

este fenómeno se le denomina bloqueo de cabecera (*Head-Of-Line blocking - HOL Blocking*) y limita el *throughput* máximo a un 58,6% bajo las premisas de tráfico uniforme aleatorio y un elevado número de puertos [KHM87]. Por lo tanto, el rendimiento obtenido de un conmutador con colas a la entrada con las características descritas es bastante bajo. Otra desventaja del conmutador con colas a la entrada es el requerimiento de un algoritmo de planificación complejo, el cual debe solventar, por un lado, los problemas del bloqueo de cabecera y, por otro lado, la contención de los puertos de salida cuando dos o más paquetes provenientes de diferentes puertos de entrada compartan el mismo puerto de destino.

2.3.1.1 Mejoras en la técnica de colas a la entrada

Existen un gran número de técnicas basadas en este tipo de conmutadores que mejoran notablemente el rendimiento de un conmutador con colas a la entrada basado en una política *FIFO*. Entre ellas cabe destacar la política de ventanas [KHM87], almacenamiento agrupado [TC94], expansión de los puertos de entrada [Mak98] o aceleración combinada con técnicas de almacenamiento en los puertos de entrada y salida [PM98][CGM+99], capaces de emular el rendimiento de un conmutador con colas a la salida. Un estudio más profundo en [XL03] analiza la distribución óptima de las memorias entre los puertos en función de la probabilidad de pérdida de paquetes.

Una atención especial requiere la política de colas virtuales de salida (*Virtual Output Queue - VOQ*) [AN89] debido a su relevancia e influencia. Una forma de eliminar el problema del bloqueo de cabecera en los conmutadores con colas a la entrada es disponer en cada puerto de entrada una cola por cada puerto de salida con una política de servicio *FIFO*, como se muestra en la Figura 2.5. De esta manera, cada paquete entrante se almacena en una cola individual sin bloquear ningún otro paquete dirigido a un puerto de salida diferente. Un ejemplo representativo de esta técnica de almacenamiento es el conmutador *Tiny Tera* [MIM+97] propuesto por McKeown *et al.* en 1997.

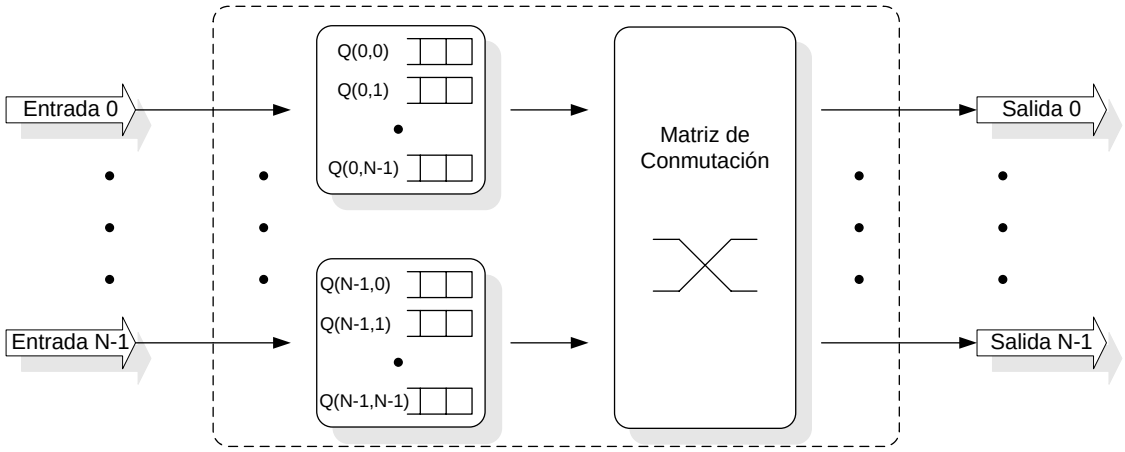


Figura 2.5: Arquitectura de un conmutador con colas virtuales de salida

Con la política de colas virtuales de salida, un puerto de entrada podría tener concedida la transferencia de los paquetes almacenados en éste, a más de un puerto de salida. Puesto que cada puerto de entrada sólo puede transmitir un paquete en un intervalo de tiempo, el resto de los paquetes tiene que esperar, por lo que sus correspondientes puertos de salida pueden estar ociosos. Esta ineficiencia puede ser mitigada ejecutando iterativamente el algoritmo de planificación. Este aspecto de los conmutadores con colas virtuales de salida ha concentrado un gran esfuerzo investigador para lograr una planificación óptima [AOS+93][McK95].

2.3.2 Técnica de almacenamiento interno

En la técnica de almacenamiento interno la memoria está localizada entre los puertos de entrada y salida. En cada intervalo de tiempo los puertos de entrada pueden almacenar todos los paquetes entrantes, mientras que los puertos de salida pueden leer los paquetes salientes. Esta arquitectura es comparable, tanto desde el punto de vista de la funcionalidad como del rendimiento del retardo y el *throughput*, a un conmutador con colas a la salida.

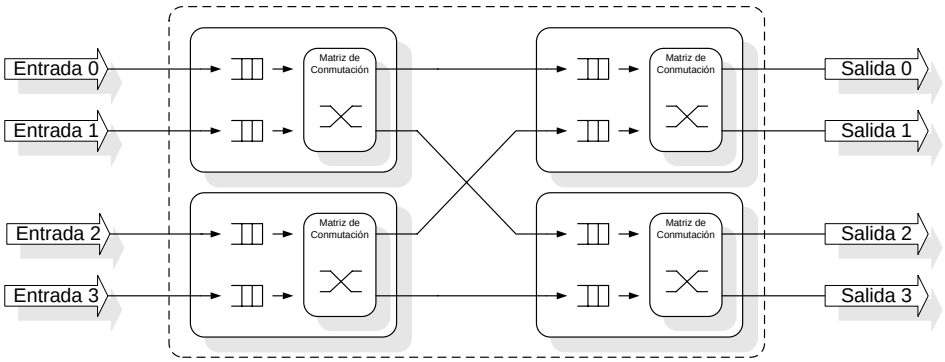
En el caso de disponer de una memoria interna compartida por todos los puertos, se maximiza el aprovechamiento de la capacidad de la memoria y se reduce la posibilidad de congestión de un puerto, como sucede en el experi-

mento *Prelude* [DCS88]. Sin embargo, la velocidad de acceso es extremadamente alta, puesto que debe permitir el acceso simultáneo de todos los puertos del conmutador. En consecuencia, las topologías con las memorias internas distribuidas aparecen como una solución a la limitación de la velocidad, como la arquitectura presentada en [NHR96]. La Figura 2.6 muestra dos casos donde las memorias están localizadas dentro de la matriz de conmutación de forma distribuida. La primera aproximación, representada en la Figura 2.6(a), se basa en la conmutación multietapa [GP94][CFF+97]. Mientras que esta arquitectura tiene el mismo requisito de ancho de banda de memoria mínimo que los conmutadores con colas a la entrada, un conmutador *crossbar* con memoria interna y N puertos basado en conmutación por división en el espacio [OYS+92], como el que se muestra en la Figura 2.6(b), requiere $O(N^2)$ elementos de almacenamiento, bastante más que los $O(N)$ elementos requeridos para un conmutador con colas a la entrada puro. Debido a que los elementos de memoria no son un recurso compartido, es crucial distribuir adecuadamente el tráfico entre los puertos de entrada para evitar el llenado de una cola específica.

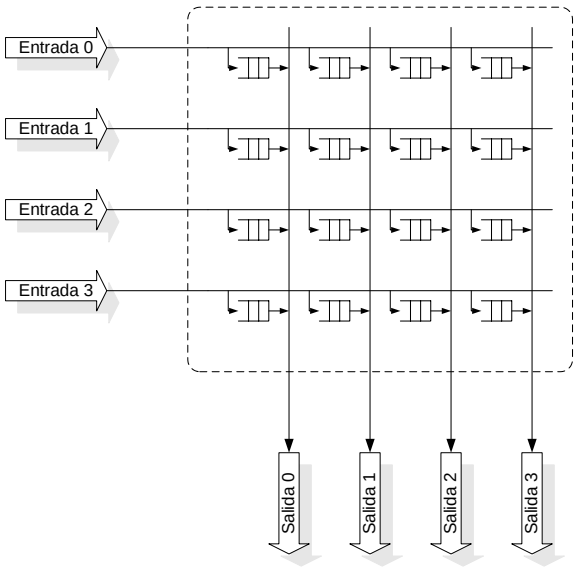
En otra opción de encolamiento interno se comparte una memoria no *FIFO* por todos los puertos de entrada y salida. Además, cada puerto de salida mantiene su propia cola lógica, de la cual lee las direcciones de los paquetes, es decir, existe una lista enlazada de los paquetes almacenados en memoria. Desde el punto de vista del rendimiento, esta aproximación de memoria compartida es equivalente a las técnicas de almacenamiento con colas a la salida. Además, ya que se comparte la memoria central entre todos los puertos de salida, se puede lograr un uso más eficiente que con el uso de técnicas basadas en colas a la salida [OSM+90][SIG+91].

Desde el punto de vista de la implementación del mecanismo de control de la memoria compartida, las colas lógicas se pueden organizar como una lista enlazada [CL07] o como colas *FIFO* dedicadas a cada puerto de salida [LKR+90]. En este último caso, la capacidad de almacenamiento se incrementa pero se reduce la complejidad de la implementación, ya que el manejo

de los punteros en una lista enlazada es una operación más compleja que las operaciones de lectura y escritura de una cola *FIFO*. Además, en caso de error, la fiabilidad de las colas *FIFO* es mejor, puesto que el número de paquetes perdidos en una lista enlazada puede ser mucho mayor que en una cola *FIFO*. En [SG94] se presentó una propuesta basada en asociar un identificador a cada paquete entrante y almacenar estos identificadores en una memoria direccionable por contenido evitando la necesidad del mantenimiento de las colas lógicas. En comparación con la propuesta de listas enlazadas se obtiene un coste *hardware* similar, pero con una menor velocidad y mayor coste debido a la utilización de este tipo de memoria.



(a) Arquitectura *banyan*



(b) Arquitectura *crossbar*

Figura 2.6: Arquitecturas de conmutación con almacenamiento interno

2.3.3 Técnica de almacenamiento con colas a la salida

Uno de los hitos a alcanzar en la presente Tesis Doctoral es la concepción y el desarrollo de un conmutador basado en colas a la salida. La arquitectura mostrada en la Figura 2.7 se corresponde con esta topología. Asumiendo que los paquetes entrantes se deben conmutar tan pronto como llegan a los puertos de entrada, esta situación puede provocar la llegada simultánea de múltiples paquetes destinados al mismo puerto. En este caso, todos los paquetes se transfieren a las colas en el puerto de salida dentro del intervalo de tiempo actual, produciendo un incremento de la tasa de transferencia de la matriz de conmutación. En el peor caso, un conmutador de dimensiones $N \times N$ debe transmitir N paquetes al mismo puerto de salida en un determinado momento, lo que requiere que el conmutador opere N veces más rápido que la velocidad del enlace externo. Por lo tanto, la aceleración interna puede limitar la escalabilidad de esta arquitectura de conmutación. Sin embargo, y pese a los requerimientos de velocidad que presenta esta arquitectura, esta técnica es óptima en términos de *throughput* y retardo [HK88].

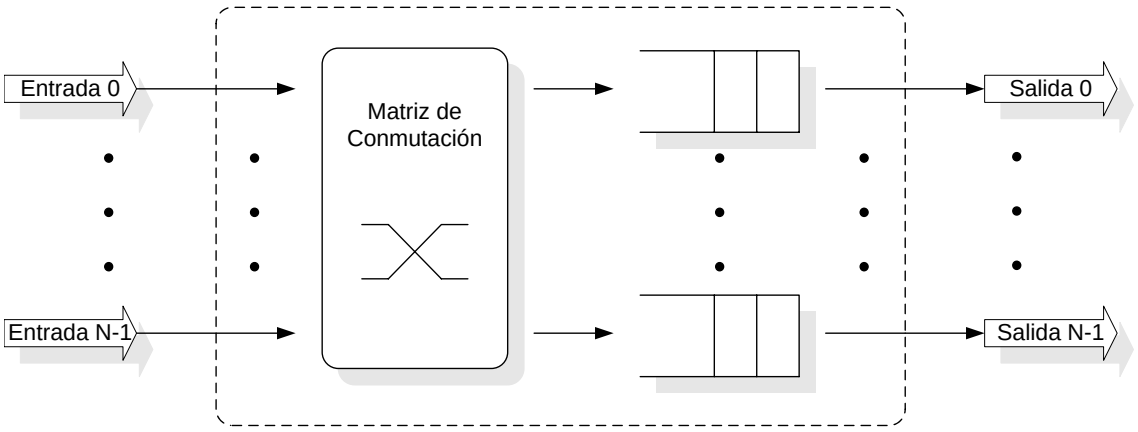


Figura 2.7: Arquitectura de un conmutador con colas a la salida

Algunas arquitecturas representativas de esta técnica de almacenamiento son los conmutadores *Knockout* [YHA87] y *Gauss* [Vri90], basados en un bus de comunicación *broadcast*, el conmutador *ATOM* [SNS+89] basado en un bus de comunicación compartido en el tiempo, o el conmutador *SCOQ* [CM93], que

se fundamenta en una configuración multietapa de elementos de conmutación *Banyan* más una red de ordenación *Batcher* [Bat68]. En los apartados siguientes se hará una revisión extensa de la arquitectura *Knockout* debido a su relevancia con los objetivos propuestos en la presente Tesis Doctoral.

2.3.3.1 El conmutador *Knockout*

Los conmutadores con colas a la salida proporcionan el mejor compromiso entre el retardo y el *throughput*. El principal problema de esta arquitectura es el alto requerimiento de velocidad de la memoria. Sin embargo, desde un punto de vista práctico, la probabilidad de que todos los puertos de entrada reciban tráfico destinado al mismo puerto de salida es muy baja, por lo que no será necesario escalar la velocidad de la memoria en función del número de puertos de entrada. En su lugar, se establece un número máximo de paquetes que se pueden almacenar simultáneamente en la memoria de un puerto de salida, cuyo valor es menor que el número de puertos de entrada. En el caso de recibir más paquetes de los que se pueden almacenar en la memoria en un intervalo de tiempo, el exceso de paquetes se descarta. Este concepto se denomina principio *knockout* [YHA87]. Evidentemente, la cuestión más importante en el diseño de un conmutador *Knockout* es determinar la cantidad máxima de paquetes que pueden transferirse a un mismo puerto de salida, ya que la elección de un valor elevado supone encontrar nuevamente el cuello de botella de la velocidad de la memoria. En el caso de elegir un valor demasiado bajo, se incrementa la probabilidad de pérdida de paquetes.

El planteamiento del principio *knockout* es bastante esperanzador, ya que reduce la velocidad de la memoria necesaria para realizar una implementación de esta arquitectura. Sin embargo, no existen implementaciones comerciales basadas en la filosofía *knockout*, debido a que en los estudios teóricos se asume que la distribución del tráfico en los puertos de entrada es incorrelado, aspecto que en la práctica no tiene por qué ser cierto. Además, la idea de descartar paquetes se rechaza exceptuando los casos en los cuales las colas o el conmutador estén colapsados.

Aunque el principio *knockout* no se utiliza en los sistemas de conmutación reales, es cierto que ha alcanzado un gran atractivo para muchos investigadores, puesto que el retardo y el *throughput* de esta topología son similares a los obtenidos de las arquitecturas con colas a la salida. Por este motivo, a continuación se profundiza en las características del conmutador *Knockout*.

Arquitectura básica

En la Figura 2.8 se muestra la arquitectura del conmutador *Knockout* a nivel de bloques. Se compone de una matriz de interconexión de N interfaces, siendo N el número de puertos de entrada. Esta matriz de interconexión se caracteriza por conectar cada puerto de entrada con todos los puertos de salida, por lo que se denomina conexión *broadcast*.

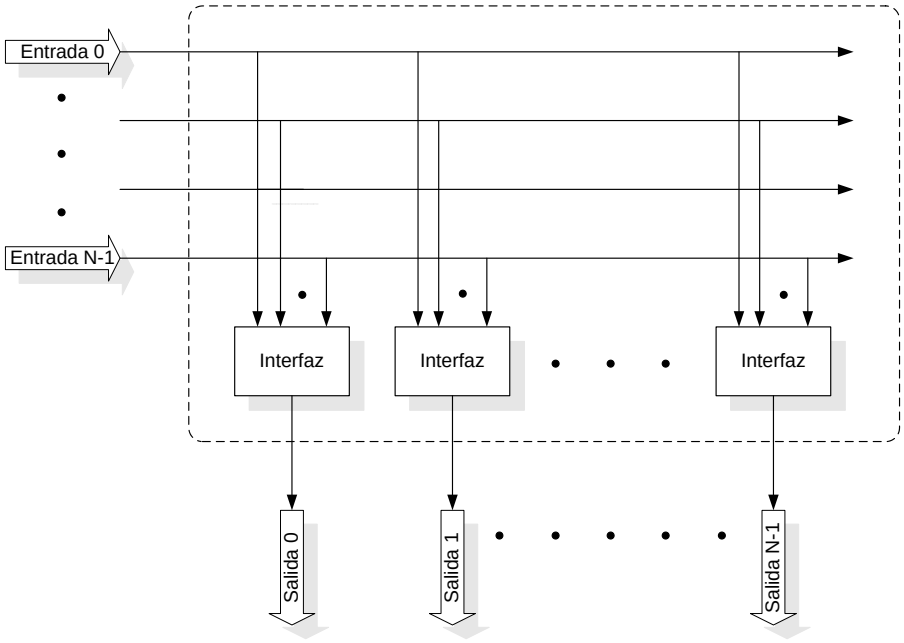


Figura 2.8: Arquitectura de un conmutador *Knockout*

Como cada puerto de salida tiene acceso a todos los puertos de entrada, no existe ningún tipo de bloqueo en la matriz de interconexión. La congestión se produce en las interfaces de los puertos de salida, puesto que los paquetes pueden llegar simultáneamente de los diferentes puertos de entrada al mismo puerto de salida. Otro punto destacable de la arquitectura es la modularidad que presenta, ya que la matriz de interconexión puede residir en un *backplane*

donde se puedan insertar y extraer tarjetas individuales con la electrónica de los puertos de entrada y salida.

La Figura 2.9 ilustra la arquitectura interna de las interfaces asociadas con cada puerto de salida del conmutador. La interfaz se compone de tres elementos: un filtro de direcciones, un concentrador y un *buffer* compartido. El filtro recibe todos los paquetes procedentes de todos los puertos de entrada y debe descartar aquellos que no tengan por destino ese puerto de salida. El concentrador transforma las N líneas de entrada correspondientes a los N puertos de entrada en las L líneas de salida, siendo $L < N$. Estas L líneas de salida del concentrador pasan al *buffer* compartido que se compone de un desplazador y L colas *FIFO* independientes. El mecanismo de memoria compartida permite utilizar completamente todos los *buffers*, como si se tratara de una única memoria de L entradas y una salida con política *FIFO*.

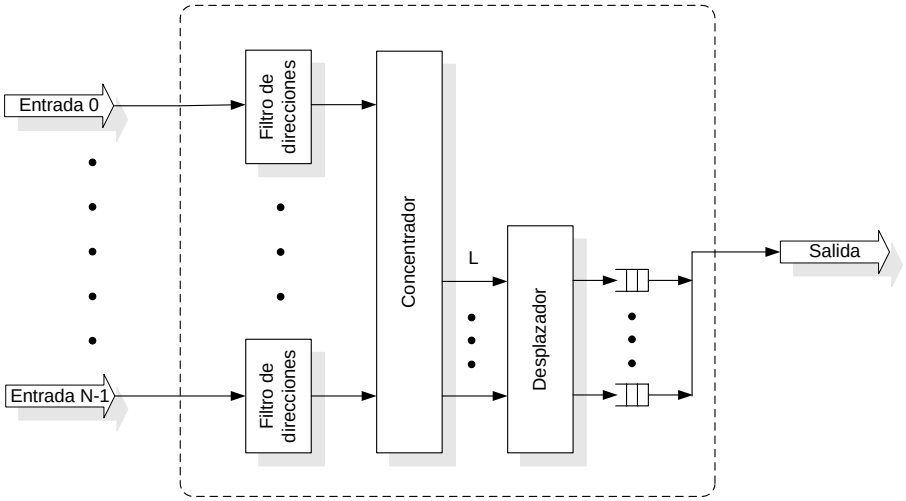


Figura 2.9: Arquitectura interna de una interfaz

El mejor compromiso entre el *throughput* y el retardo del conmutador *Knockout* frente a otras propuestas fomentaron el estudio en profundidad de esta arquitectura y su interfaz. En [EHY87] se propone un conmutador con capacidad de conmutar tráfico de longitud variable segmentando los paquetes entrantes en células de tamaño fijo y evitando que el concentrador elimine alguna de las células que componen un paquete.

Además de la conmutación de los paquetes de longitud variable, resulta de especial interés el tráfico *multicast*. Desde un punto de vista general, este tipo de tráfico incrementa la velocidad interna de la matriz de conmutación con el fin de obtener las mismas prestaciones que en el caso de tráfico *unicast* e introduce más *hardware* en el conmutador para gestionar los múltiples destinos de los paquetes. Si se estudia el caso concreto del tráfico *multicast* en un conmutador *Knockout*, se descubre que aumentan las pérdidas de paquetes en el concentrador debido al incremento de la probabilidad de generar contención en los puertos de salida.

Una de las soluciones adoptadas para el tráfico *multicast* consiste en diseñar un elemento de conmutación multietapa [KL95] con una red, previa al elemento de conmutación, encargada de replicar los paquetes [OR97][Mir98]. Otra topología capaz de conmutar tráfico *multicast* se compone de una estructura en forma de árbol y un concentrador [LL97][NS99]. Finalmente, las redes *broadcast* utilizan un medio interno compartido, como un bus o un anillo, para copiar y encaminar los paquetes, seguido por concentradores conectados a las redes de salida [EHY88][CC95].

El concentrador es un elemento especialmente crítico y complejo dentro de la arquitectura *Knockout* debido a la cantidad de recursos *hardware* que necesita y a la dependencia de dichos recursos con el número de puertos del conmutador. En [LL95] se presentan varios algoritmos que reducen su complejidad y su coste. Siguiendo esta misma línea de investigación, en [LSC98] se introducen *buffers* en el concentrador y se agrupan las líneas de salida. Debido a la complejidad que puede alcanzar este elemento, en [LT01] se propone el diseño y verificación del concentrador *Knockout* enfatizando este último aspecto, puesto que se desarrollan modelos a nivel *RTL* (*Register Transfer Logic*) y de comportamiento con el fin de comprobar la equivalencia entre ambos.

Los mecanismos para diferenciar el tráfico en un conmutador *Knockout* podrían situarse en la matriz de conmutación [NS99], utilizando una matriz multietapa, o en los concentradores o *buffers* de salida [EDW93], utilizando un mecanismo de descarte con aquellas células de baja prioridad.

Otra filosofía de diseño e implementación del conmutador *Knockout* se basa en el principio de agrupación de canales (*Channel Grouping Principle*) [Pat88] en el que la técnica de conmutación de paquetes se divide en dos fases, como se observa en la Figura 2.10. En la primera, los puertos de salida se agrupan de manera que un paquete procedente de cualquier puerto de entrada se encamina a cualquier puerto intermedio del grupo en el que se encuentra el puerto de salida. En la segunda fase, se conmuta el paquete desde el puerto intermedio a la salida correspondiente. Esta topología suaviza el problema de la contención de los puertos de salida, de aquí que se alcance un mejor compromiso entre el retardo y la complejidad de la primera etapa y se reduzcan las pérdidas de paquetes. En [KR00] se presenta la arquitectura *multicast MSXmin*, basada en este principio, como una mejora de la arquitectura *MOBAS* [CC95]. En [Ma96] se estudia la probabilidad de pérdidas de ambas configuraciones, *Knockout* original y esta propuesta, añadiéndose dos prioridades para diferenciar el tráfico, con lo que se alcanza una baja probabilidad de pérdidas en el caso de paquetes de alta prioridad.

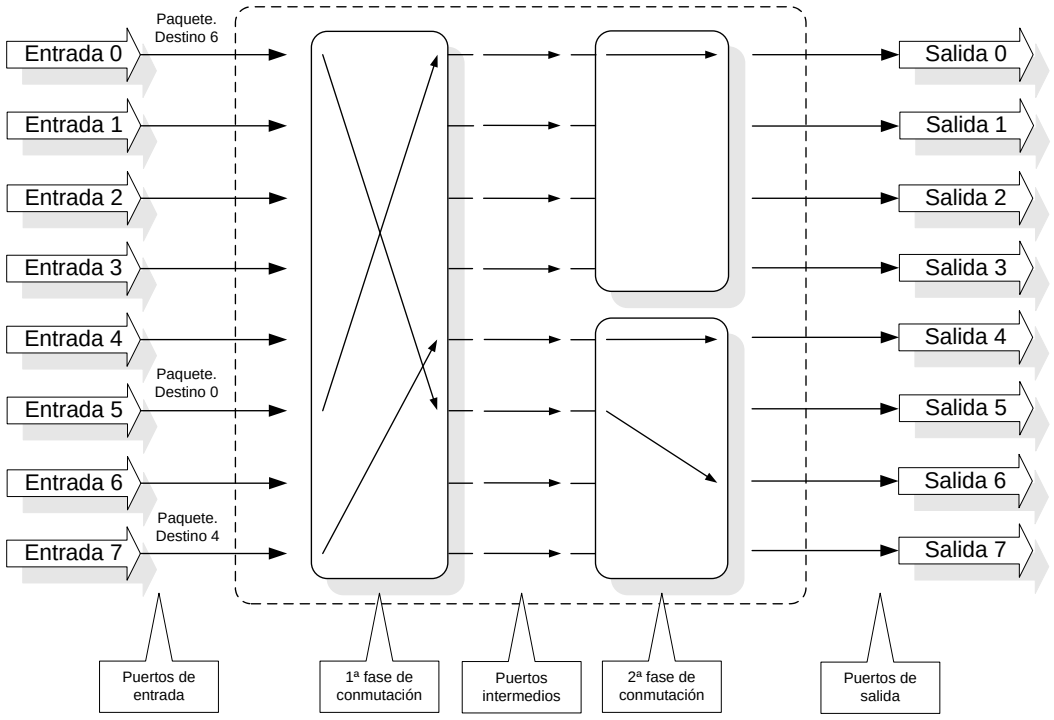


Figura 2.10: Principio de agrupación de canales

2.3.3.2 Técnica de almacenamiento con múltiples colas a la salida

Recientemente, Danilewicz [DGK+04] propuso una arquitectura con prestaciones similares a un conmutador con técnicas de almacenamiento basadas en colas a la salida. La Figura 2.11 muestra el conmutador con múltiples colas a la salida (*Multiple Output Queuing - MOQ*). En esta arquitectura, los *buffers* están localizados en los puertos de salida, disponiendo de N colas independientes cada uno. Cada cola almacena paquetes provenientes de un único puerto de entrada. En consecuencia, se requiere una matriz de conmutación de $N \times N^2$ puertos, siendo N el número de puertos de entrada y N^2 el número de colas en los puertos de salida.

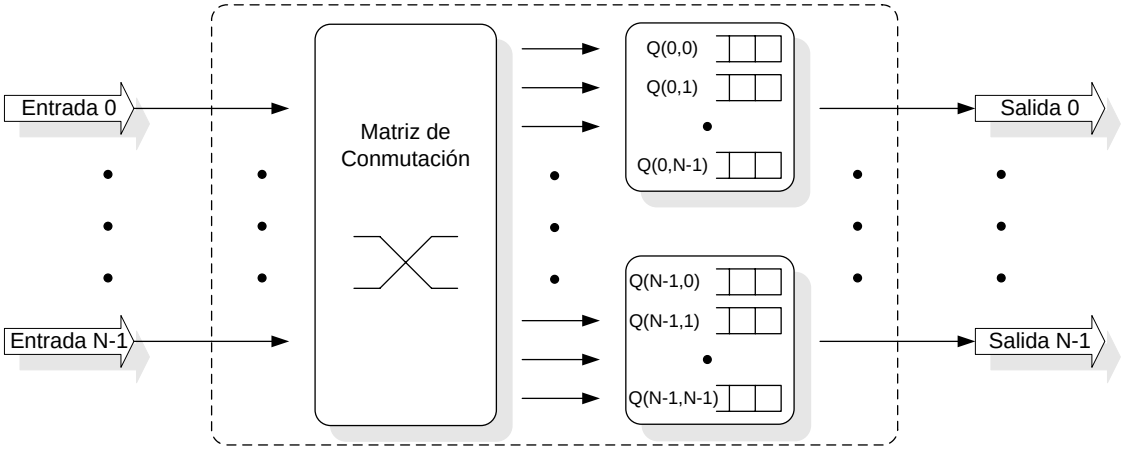


Figura 2.11: Arquitectura de un conmutador con múltiples colas a la salida

Una de las principales características de esta arquitectura es la utilización del mismo número de colas que un conmutador con colas virtuales de salida, lo que representa que la matriz de conmutación y las memorias de almacenamiento sólo necesitan alcanzar la velocidad de los enlaces externos. En consecuencia, la topología con múltiples colas a la salida reduce la exigencia de la velocidad de la arquitectura clásica con colas a la salida. Otra característica es la reducción de la complejidad del algoritmo de planificación frente a la arquitectura con colas virtuales de salida, debido a que cada puerto de salida dispone de un planificador independiente en lugar de un planificador centrali-

zado, típico de una arquitectura con colas virtuales de salida. Por último, hay que destacar la facilidad de conmutar tráfico *multicast*, gracias a la matriz de conmutación de $N \times N^2$ puertos que evita las colisiones en los puertos de salida.

Sin embargo, esta arquitectura incrementa el coste *hardware* y la complejidad de la matriz de conmutación, ya que, con el objetivo de proporcionar prestaciones similares a una arquitectura con colas a la salida, necesita los recursos propios de una arquitectura con colas virtuales de salida. Además, la expansión de los puertos de salida limita la escalabilidad del sistema.

2.3.4 Técnicas de almacenamiento combinadas

A lo largo del capítulo se han presentado diferentes técnicas de almacenamiento y sus correspondientes arquitecturas. Cada una de ellas presenta un compromiso entre rendimiento, complejidad y recursos necesarios. Sin embargo, en los últimos años se ha realizado un gran esfuerzo investigador en la mejora de las prestaciones y de la escalabilidad, y en la reducción de la complejidad de las topologías combinando varias técnicas de almacenamiento.

2.3.4.1 Técnica de almacenamiento con colas a la entrada y a la salida

Como se ha mencionado en las secciones anteriores, la implementación de las técnicas de almacenamiento con colas a la entrada resulta más sencilla que aquellas con colas a la salida, aunque su rendimiento es, por lo general, limitado. Por otro lado, las técnicas de almacenamiento con colas a la salida representan la arquitectura ideal para un conmutador de paquetes que proporciona el máximo rendimiento alcanzable. Sin embargo, resulta muy costoso implementar este conmutador debido a la elevada tasa de transferencia de la matriz de conmutación y de la memoria. Además, la comparación entre ambas arquitecturas supone evaluar el coste *hardware* frente al rendimiento.

Como alternativa a ambas soluciones, se presentan arquitecturas, como la representada en la Figura 2.12, que combinan la utilización de encolado

en los puertos de entrada y salida (*Combined Input and Output Queued - CIOQ*). En este tipo de arquitecturas híbridas, además de disponer de *buffers* en todos los puertos, se requiere una pequeña aceleración interna en la matriz de conmutación, como en la arquitectura de conmutación de paquetes *Saturn* propuesta por Chao en [Cha00].

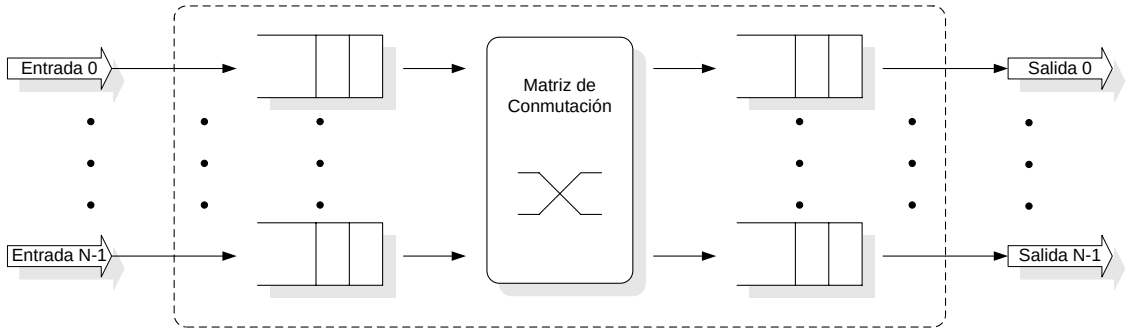


Figura 2.12: Arquitectura de un conmutador con colas a la entrada y a la salida

A finales de los años 80 se realizaron los primeros estudios que demostraron que con este tipo de arquitecturas se obtenían mejoras en el *throughput* máximo respecto a conmutadores con técnicas de almacenamiento con colas puras a la entrada [OMK+89]. Posteriormente, se analizó que el factor limitante del rendimiento era el bloqueo de cabecera para aceleraciones de los puertos de salida mayores de tres [GG91]. La llegada de la técnica de colas virtuales de salida eliminó el bloqueo de cabecera de las técnicas basadas en colas a la entrada. Se ha demostrado que un conmutador con colas virtuales de salida y colas a la salida con una aceleración de $N/2$ es suficiente para emular perfectamente un conmutador con colas a la salida puro [MPZ97]. Estudios posteriores han demostrado que con incrementar la aceleración al doble es suficiente [SZ98a][KPC+99][CGM+99].

2.3.4.2 Técnica de almacenamiento con colas a la entrada y memoria interna

Tanto los conmutadores con colas a la entrada como los conmutadores que combinan colas virtuales de salida con colas a la salida presentan una alta escalabilidad y han sido objeto de un intenso esfuerzo investigador a lo largo de la pasada década. Estas topologías, basadas en colas virtuales de salida, necesitan un algoritmo de planificación centralizado para conectar los puertos de entrada con los puertos de salida. Actualmente, estos algoritmos son uno de los cuellos de botella en este tipo de arquitecturas. La arquitectura *crossbar* con almacenamiento interno en los elementos de cruce se basa en planificadores distribuidos, los cuales reducen la complejidad algorítmica de la planificación y permiten una mayor escalabilidad, tanto en el número de puertos como en la velocidad de los enlaces. Un ejemplo de esta arquitectura se encuentra en una patente de 1982 [BD82]. Posteriormente, en 1987, Nojima [NTF+87] implementó una matriz de conmutación basada en *crossbar* con memoria interna.

Un conmutador *crossbar* con memoria interna infinita evita las colisiones en los elementos de cruce, puesto que equivale a un conmutador con colas a la salida con memoria dedicada para los paquetes de cada entrada [RTH88]. Limitando el tamaño de la cola en cada elemento de cruce a una célula de 53 bytes, una célula *ATM*, y disponiendo de un tamaño de cola a la entrada suficiente, un conmutador con técnicas de almacenamiento combinadas en los puertos de entrada y *crossbar* con memoria interna (*Combined Input Crossbar Queued - CICQ*) reduce el tamaño total de la memoria requerida [GBG91]. Además, se ha probado que el *throughput* de un conmutador *crossbar* con memoria interna *FIFO* y política de selección aleatoria se puede aproximar al 100% de *throughput* [RF93].

En [SZ98b] se presenta y estudia un conmutador con soporte de calidad de servicio y paquetes de longitud variable. Dicho conmutador dispone de colas a la entrada, colas internas en la matriz de conmutación *crossbar* y colas a

la salida. El elevado número de elementos que componen esta arquitectura supone una desventaja si se utiliza un algoritmo centralizado, puesto que su implementación es muy compleja. Por lo tanto, Stephens y Zhang proponen un algoritmo distribuido entre los distintos puertos de la arquitectura de conmutación con el fin de afrontar este problema. Además, se necesita una velocidad interna superior a la tasa de los enlaces externos para transferir los paquetes de longitud variable.

2.3.4.3 Técnica de almacenamiento con colas virtuales de salida y memoria interna

En el año 2000 Nabeshima [Nab00] propuso el primer conmutador con colas virtuales de salida y colas en los elementos de cruce de una matriz de conmutación *crossbar*, reduciendo significativamente la cantidad de memoria interna en la matriz de conmutación. El algoritmo de planificación selecciona el paquete situado en la cabecera con el mayor tiempo de encolado, tanto en los puertos de entrada como en las colas de los elementos de cruce, siendo el resultado una completa eliminación del bloqueo de cabecera y una reducción del retardo medio por paquete respecto a un conmutador con colas a la entrada.

El conmutador con colas virtuales de entrada y matriz de conmutación *crossbar* propuesto en [YC01] soporta nativamente paquetes de longitud variable sin requerir mecanismos complejos de segmentación y reensamblado. Se demuestra que esta arquitectura tiene un retardo menor que una arquitectura con colas virtuales de salida y política de planificación *iSLIP* (*iterative round robin with SLIP*) [McK95] para la conmutación de células y paquetes de longitud fija.

Otro arquitectura similar se presenta en [ROJ+01] y se caracteriza por almacenar únicamente un paquete en la cola de cada elemento de cruce (*Combined Input-One-cell-Crosspoint Buffered Crossbar - CIXB-1*) alcanzando un *throughput* del 100% bajo tráfico uniforme. El retardo medio es proporcional a la longitud de la ráfaga y muy cercano al retardo de un conmutador con colas a la salida. En [ROC01] se presentó una generalización donde se añadieron K

células por elemento de cruce, además de memoria en los puertos de entrada y salida, alcanzando el máximo *throughput* bajo tráfico uniforme y no uniforme.

2.4 Gestión del tráfico

A lo largo del presente capítulo se han estudiado y analizado en profundidad las arquitecturas de conmutación y las técnicas de almacenamiento más significativas. Sin embargo, una vez que se han almacenado los paquetes en las colas de un conmutador, aún queda pendiente la tarea de la planificación, es decir, elegir un paquete y transmitirlo. En los objetivos de la presente Tesis Doctoral se impone, como requisito adicional al de soportar la transferencia efectiva de paquetes de longitud variable, proporcionar calidad de servicio. Por tanto, los paquetes no se pueden transmitir arbitrariamente y requieren la aplicación de un algoritmo parametrizado, aspecto en torno al cual gira esta sección.

2.4.1 Tecnologías de red

La red ha ido evolucionando a fin de proporcionar garantías de calidad de servicio a los usuarios en función de los servicios demandados. Por ejemplo, *ATM*, ampliamente adoptada por las redes troncales, puede reservar ancho de banda y recursos de memoria en función de las conexiones virtuales. Del mismo modo, los Servicios Integrados pueden proporcionar calidad de servicio a cada flujo de datos en la red *IP*. Otro enfoque, los Servicios Diferenciados, proporciona tratamientos a los paquetes de las distintas clases de tráfico, sin basarse en flujos de datos, sino en identificadores internos de cada paquete. Una visión más amplia sobre la calidad de servicio se puede encontrar en [Wan01][CG02].

2.4.1.1 Servicios Integrados

El concepto de flujo de datos se introduce como un caudal simple y distinguible de información procedente del mismo usuario y que requiere la misma calidad

de servicio. El soporte de diferentes clases de servicio requiere que la red, específicamente los conmutadores, gestionen explícitamente el ancho de banda y la memoria para proporcionar calidad de servicio a cada flujo. Por lo tanto, la reserva de recursos, el control de admisión, la planificación del tráfico y la gestión de la memoria son también claves en los Servicios Integrados [BCS94].

Asimismo, se requiere el mantenimiento del estado de los flujos de información en los conmutadores, lo que supone un cambio importante en el modelo de conexión. Internet no está orientada a mantener información sobre las conexiones y se necesita mantener la información del estado de los diferentes flujos, refrescándola periódicamente. En el caso de *ATM*, una vez que se configura una conexión, todos los recursos solicitados permanecen activos y asociados hasta que finalice la transmisión. Dentro de las conexiones se pueden distinguir aquellas con servicio garantizado (*Guaranteed Service - GS*) con un retardo fiable y acotado, y aquellas con un control de la carga de tráfico (*Controlled Load Service - CLS*) con un retardo variable que pueden soportar pequeñas pérdidas de paquetes.

2.4.1.2 Servicios Diferenciados

La diferenciación de servicios acomoda en pocos flujos de datos las necesidades de aplicaciones heterogéneas y las expectativas de los usuarios. El objetivo principal de esta filosofía es proporcionar una discriminación de servicios escalable sin la necesidad de mantener información de cada flujo de datos ni mecanismos de señalización, como ocurre en los Servicios Integrados. Los Servicios Diferenciados [BBC+98] emplean un pequeño y bien definido conjunto de bloques con los cuales se puede atender a una gran cantidad de servicios.

El mecanismo de Servicios Diferenciados establece varios bits con información sobre el tipo de servicio en la cabecera de los paquetes. Estos bits se utilizan para determinar cómo se tratan los paquetes en los conmutadores dentro de la red. Por consiguiente, y de acuerdo con este modelo, el tráfico de la red se clasifica y se agrupa en comportamientos con similares necesidades con los mismos bits del tipo de servicio. Diferentes valores en los bits del tipo

de servicio se traducen en diferentes servicios en los conmutadores. La ventaja de este esquema es que muchos flujos de datos se pueden agrupar en uno solo y procesar en conjunto, logrando la simplificación del procesamiento y del almacenamiento. Puesto que cada paquete se procesa individualmente, no existe una fase de señalización, ni de reserva de recursos, ni ningún otro proceso en el seno de la red.

2.4.2 Calidad de Servicio

La principal tarea de una red de comunicaciones es el transporte de la información de un punto a otro asegurando una determinada calidad de servicio (*Quality of Service - QoS*). Actualmente, llegar a garantizar una mínima calidad resulta realmente complicado, debido al incremento del número de usuarios conectados a la red, a la mayor exigencia de ancho de banda para las nuevas aplicaciones, al aumento del número de restricciones y tipos de calidades de servicio, y al uso más extendido de comunicaciones con múltiples destinatarios. Proporcionar alguna de estas garantías resulta una tarea mucho más ardua en una red no orientada a conexión como es Internet. Aunque existen otros mecanismos, como el control de flujo proporcionado por la capa de transporte o algunos protocolos de encaminamiento dinámico, resultan inadecuados para asegurar calidad de servicio. Actualmente, el gran número de aplicaciones se traduce en un amplio abanico de requisitos aplicables a cualquier transferencia de información a través de la red. Algunas aplicaciones requieren una transmisión con una baja tasa de pérdidas mientras que otras necesitan un retardo acotado. En cualquier caso, los recursos de la red se deben gestionar apropiadamente para transmitir cualquier tipo de tráfico y maximizar el número de comunicaciones sin perder flexibilidad.

Con el fin de optimizar la utilización de los recursos de la red mientras se satisfacen los requisitos individuales de los usuarios, los mecanismos de la gestión del tráfico deben proporcionar acceso a los recursos de la red introduciendo diferentes prioridades y combinando varias técnicas, entre las que cabe citar la política de control y conformación de los flujos de datos, la pla-

nificación de los paquetes, y la gestión de los recursos de memoria como las más importantes. En la presente Tesis Doctoral se estudian y se plantean soluciones dentro del ámbito de la planificación del tráfico a fin de soportar calidad de servicio siguiendo una filosofía de Servicios Diferenciados.

2.4.3 Planificación de paquetes

Las redes de paquetes permiten a los usuarios compartir recursos, como la memoria o el ancho de banda de los enlaces. Sin embargo, el problema de la contención de los puertos de salida se presenta inevitablemente. Por lo tanto, dado un número de usuarios que comparten el mismo enlace, se necesita un planificador de paquetes a fin de determinar el orden en el que se transfieren los paquetes. En otras palabras, se requiere un algoritmo de planificación que, al mismo tiempo que solvente la contención de los puertos de salida, priorice el tráfico con el objetivo de proporcionar calidad de servicio.

Desde un punto de vista general, se espera que un planificador de paquetes reúna varias propiedades. En primer lugar, el tiempo utilizado en las operaciones de selección y encaminamiento de paquetes debe ser reducido. Asimismo, se desea que la complejidad y la variación temporal de este proceso sean bajas. En segundo lugar, dicho planificador debe proporcionar un tratamiento justo a los diferentes flujos de datos, independientemente de su número y de los requisitos de calidad de servicio de cada uno. En tercer lugar, y en el peor caso de funcionamiento, el retardo asociado a la clase de tráfico más desfavorecida no debería ser excesivo. Por último, y de especial relevancia en esta Tesis Doctoral, se espera que la complejidad *hardware* sea asumible para permitir una implementación eficiente.

La simplicidad y la complejidad temporal colisionan siempre con la justicia y el interés por establecer un límite perfectamente acotado para el retardo. Por lo general, los planificadores justos en un corto periodo de tiempo y con un retardo acotado estrictamente, resultan muy complejos desde el punto de vista temporal y difíciles de implementar. Los esquemas con baja complejidad temporal son fáciles de implementar pero, habitualmente, no pueden pro-

porcionar justicia en un corto periodo de tiempo ni acotar adecuadamente el retardo.

En los planificadores basados en sellos de tiempo, uno de los dos tipos de algoritmos de planificación conservadores[1] más extendidos, se mantiene un reloj de tiempo virtual para emular el algoritmo ideal *Generalized Processor Sharing* (*GPS*) [PG93] y atender a los flujos utilizando como unidad mínima los paquetes (*Packet Fair Queuing - PFQ*) en lugar de flujos infinitesimales. Entre los algoritmos *PFQ* que tratan de emular *GPS* se puede citar al algoritmo *Weighted Fair Queuing* (*WFQ*) [DKS89][PG93]. Dicho algoritmo se caracteriza por un límite de retardo bajo y una buena justicia, pero su complejidad temporal es $O(N)$, donde N es el número de flujos activos. Las variantes de este algoritmo, como *Virtual Clock* [Zha90] o *Worst-Case Fair Weighted Fair Queueing* (*WF*^{2}*Q*) [BZ96], usan diferentes métodos para calcular el sello de tiempo, pero todavía presentan, al menos, una complejidad temporal de $O(logN)$. Puesto que el proceso de planificación se compone del cálculo del sello y la ordenación de los paquetes en función del sello, y teniendo en cuenta que el mejor algoritmo conocido para insertar un número en una matriz ordenada presenta una complejidad $O(logN)$, es improbable que se encuentre un planificador basado en sellos de tiempo que mejore este límite [Guo04].

Por otro lado, existe un segundo tipo de planificadores conservadores basados en la filosofía *Round Robin* (*RR*), los cuales resultan simples de implementar y presentan una complejidad temporal $O(1)$. Sin embargo, muestran una gran tendencia a generar ráfagas de tráfico y mostrar injusticias en un corto periodo de tiempo. *Deficit Round Robin* (*DRR*) [SV96] y *Carry-Over Round Robin* (*CORR*) [SMT98] son dos ejemplos típicos de esta filosofía. En este tipo de planificadores se sirve un flujo de datos durante un periodo de tiempo continuo en proporción al peso de dicho flujo, resultando una salida con una alta probabilidad de generación de ráfagas para cada flujo. Por estos motivos, no se consideran adecuados los planificadores basados en la filosofía *Round Robin*

[1]Los algoritmos de planificación conservadores siempre transmiten paquetes mientras haya alguno disponible en cualquier cola [Zha95].

para proporcionar calidad de servicio en las redes de paquetes, a no ser que se solventen los problemas de la generación de ráfagas e injusticias.

2.4.4 Políticas de planificación con calidad de servicio

El cumplimiento de la calidad de servicio de un flujo de datos puede ser analizado desde un punto de vista cualitativo por un usuario. Sin embargo, dicho análisis no es válido para el control de la calidad de servicio debido a los términos en que se expresa. Una política de planificación requiere términos cuantitativos para poder asegurar los parámetros del flujo de datos. Normalmente, la calidad de servicio puede expresarse con varios parámetros como el *throughput*, retardo, ancho de banda, variación del retardo (*jitter*) o la probabilidad de pérdidas de paquetes.

2.4.4.1 *Throughput*

El *throughput* cobra especial interés en las arquitecturas de conmutación con colas a la entrada y todas sus variantes. La política de planificación, conjuntamente con la técnica de almacenamiento, realiza la conexión entre los puertos de entrada y salida, por lo que determina el *throughput* del conmutador. La solución más eficiente pasa por utilizar una técnica de almacenamiento con colas a la salida. Además si dicha técnica se combina con una política de planificación conservadora, se obtiene el máximo *throughput* [KHM87][HK88][OMK+89].

2.4.4.2 Control del ancho de banda. Paquetes de longitud fija y variable

Antes de profundizar en las políticas de planificación de paquetes de longitud variable, se debe enunciar la problemática que rodea a la conmutación de este tipo de paquetes. Las arquitecturas de los conmutadores actuales están basadas en las técnicas de almacenamiento con colas virtuales de salida, puesto que mejoran el rendimiento de los conmutadores con colas a la entrada al

eliminar el bloqueo de cabecera, y no requieren aceleración interna. Sin embargo, estas arquitecturas necesitan sincronizar los puertos de entrada y salida con el fin de reducir la complejidad del planificador [GKS05] y alcanzar un *throughput* elevado [MMA+99]. En consecuencia, la división de los paquetes de longitud variable en células de tamaño fijo facilita la sincronización de los puertos y permite la toma de decisiones, por parte del planificador, en el mismo instante de tiempo en todos los puertos.

En contraposición, la conmutación directa de los paquetes de longitud variable conlleva una mayor complejidad en la política de planificación y un bajo *throughput* [MS01]. Por lo tanto, la forma más habitual de conmutar paquetes de longitud variable es la segmentación de células de longitud fija que, posteriormente, se almacenan y ensamblan en los puertos de salida. En este proceso se debe garantizar un retardo constante en toda la transmisión del paquete, ya que si una sola célula se paraliza en los puertos de entrada, las restantes células tendrán que esperar en los puertos de salida hasta que ésta se reciba [MS01][DY02]. Por otro lado, la pérdida de una célula supone la pérdida de un paquete completo. Este aspecto se debe tener en cuenta en la política de planificación incrementando la complejidad del algoritmo.

Un segundo aspecto sobre el proceso de segmentación es el tamaño de la célula frente al tamaño de los paquetes [CYR+04][KP05]. Una célula puede contener parte de un paquete, un paquete completo o más de un paquete. En todos estos casos, la política de planificación debe decidir sobre dos aspectos claves: el tamaño adecuado de la célula y el instante del envío de la última célula [MBG+01][GKS05], habitualmente incompleta. Ambos puntos son determinantes en la utilización del ancho de banda y en el retardo del conmutador.

Si sólo se consideran paquetes de longitud fija, el algoritmo de planificación *Weighted Round Robin* (*WRR*) [KSC91] es una excelente opción. Dicho algoritmo almacena los paquetes entrantes en sus correspondientes colas y, a cada una de ellas, le asocia un peso individual. El planificador realiza una consulta *Round Robin* por todas las colas pero, en vez de enviar un paquete, envía

tantos como indique el peso antes de pasar a atender la siguiente cola. Este esquema es bastante flexible, ya que se pueden asignar los pesos en función de la cantidad de ancho de banda necesitada y sus requisitos *hardware* son asumibles.

En el caso de paquetes de longitud variable, el algoritmo *Deficit Round Robin* (*DRR*) usa una disciplina *Round Robin* donde se le asigna una pequeña parte de servicio a cada cola (*Quantum*). La única diferencia respecto al algoritmo *Round Robin* tradicional es que si un paquete es demasiado largo comparado con el servicio que tiene asignado, en primer lugar, no se envía y, en segundo lugar, el servicio asignado se acumula para la siguiente iteración en la cual se vuelva a intentar el envío del paquete. De esta manera, el déficit de cada cola se almacena y las injusticias que se cometan en un corto plazo de tiempo por el algoritmo *DRR* se compensan a lo largo del tiempo.

El algoritmo *DRR* se caracteriza, entre otras cuestiones, por introducir un banco de registros en el que se almacenan los déficits de las colas. La granularidad constante del algoritmo conlleva que no sea capaz de proporcionar simultáneamente retardos bajos para los paquetes de pequeña o mediana longitud conjuntamente con un alto *throughput*. El algoritmo *Dynamic Deficit Round Robin* (*DDRR*) [YNO+02] solventa este problema cambiando dinámicamente la granularidad en función de la longitud de los paquetes almacenados en las cabeceras de las colas.

En el caso de combinar paquetes de longitud variable con diferentes pesos en las colas de almacenamiento, la solución más adecuada parece *Weighted Deficit Round Robin* (*WDRR*) [WL02]. Sin embargo, dicha solución presenta injusticias en el reparto del ancho de banda sin ser capaz de controlar el retardo de los paquetes a fin de soportar diferentes calidades de servicio.

2.4.4.3 Control del retardo. Prioridad de los paquetes

En 1990, Kalmanek *et al.* [KKK90] propusieron un planificador denominado *Round Robin Jerárquico* (*Hierarchical Round Robin - HRR*) enfocado a paquetes de longitud fija. Dicho algoritmo divide el tiempo en marcos de longi-

tud constante, los cuales se componen de ranuras de transmisión y de otros marcos más breves. Cuando se recibe un paquete, este algoritmo puede asignarlo, o bien a una ranura, o bien a un marco de menor longitud. Con una filosofía similar se presentó el algoritmo *Stop and Go* [Gol91] en 1991. En ambos casos, la longitud del marco de tiempo acota el retardo de los paquetes. No obstante, ambas soluciones presentan un problema de acoplamiento entre el retardo y la granuralidad del ancho de banda asignado [CG02] y están orientadas a conexión. En consecuencia, se requieren marcos de tiempo de gran longitud para asignar flujos con bajas exigencias de ancho de banda y marcos de tiempo breves para proporcionar un retardo reducido. La utilización de una jerarquía permite alcanzar un compromiso entre ambos servicios, ya que combinan marcos de tiempo de diferentes tamaños.

No obstante, los marcos de tiempo no son la única técnica de control existente del retardo. La prioridad de un paquete es una medida cuantitativa de la importancia del flujo de datos al que pertenece. Esta medida se traduce en el tiempo que permanece almacenado en el sistema. De esta manera, un flujo de datos de máxima prioridad, o un paquete perteneciente a dicho flujo, se envía antes que cualquier otro al considerarse de mayor importancia. Por lo tanto, en primer lugar, los paquetes entrantes se deben clasificar en función de su importancia para, posteriormente, proceder a su transmisión. Una solución sencilla y eficiente para planificar este tráfico es un planificador jerárquico. Puesto que sólo compiten los paquetes de una misma prioridad por el ancho de banda, automáticamente se simplifica el proceso de planificación.

Siguiendo esta filosofía, en [KKL00] se propone una política para planificar el tráfico con calidad de servicio consistente en una arquitectura con múltiples colas a la entrada basada en las conexiones virtuales de *ATM*. Un primer planificador estático selecciona una conexión por cola basándose en la prioridad. Posteriormente, se utiliza un algoritmo dinámico, *Parallel Iterative Matching* (*PIM*) [AOS+93], a fin de combinar las entradas con las salidas.

Al igual que se desarrollaron algoritmos de planificación basados en el algoritmo *Round Robin*, en el caso de la política de planificación ideal *GPS*

también se han producido varias aportaciones. Cabe destacar la implementación *Worst-case Fair Weighted Fair Queueing* (*WF*^{2}*Q*+) [BZ97] por proporcionar un límite del retardo muy ajustado con un excelente índice de justicia (*Worst-Case Fair Index - WFI*).

Una solución empleada para soportar paquetes de longitud variable con calidad de servicio consiste en emplear el algoritmo *DDRR*. En una primera versión [WWL05], se propone un planificador donde se asocia un *token* a cada paquete en función de su longitud y peso para aplicar un algoritmo *DDRR* sobre el valor de este *token*. En la segunda versión [WWM+06], se utilizan dos instancias jerárquicas de este algoritmo. En una primera discriminación, se clasifican los paquetes del mismo origen y, posteriormente, se realiza un segundo filtrado entre todos los paquetes elegidos por origen.

Otra implementación de un planificador jerárquico [JKM05] consiste en aplicar, en primer lugar, un algoritmo *Shaped Deficit Round Robin* (*SDRR*) [JK04] y, en segundo lugar, un algoritmo basado en sellos de tiempo denominado *Shaped Virtual Clock* (*SVC*) [SV97]. El resultado de la combinación de ambos algoritmos conforma el tráfico, soporta paquetes de longitud variable y emplea pesos para distinguir entre los flujos de datos y limitar las tasas de pico.

En el caso de no requerirse la transmisión de paquetes de longitud variable, se puede acudir a la combinación de un algoritmo *WRR* con un algoritmo de prioridad estricta para proporcionar calidad de servicio, *Priority Queuing - Weighted Round Robin* (*PQWRR*) [MMW01]. En [ADF04], los paquetes se clasifican en tres clases de servicio. La clase *EF* (*Expedited Forwarding*) recibe la máxima prioridad y ancho de banda ilimitado. El algoritmo procesa varias clases *AF* (*Assured Forwarding*) que están ponderadas por el algoritmo *WRR* recibiendo mayor prioridad que la clase *BE* (*Best Effort forwarding*). Esta última clase recibe la menor prioridad y no se pondera con el algoritmo *WRR*. Por lo tanto, no se garantiza el ancho de banda asignado a la clase *BE*. En [ZH07], el algoritmo *WRR* pondera las clases de tráfico *AF* y *BE* a fin de asegurar un ancho de banda mínimo a cada una. La Figura 2.13 presenta esta

última versión del algoritmo jerárquico denominada *Priority Weighted Round Robin* (*PWRR*).

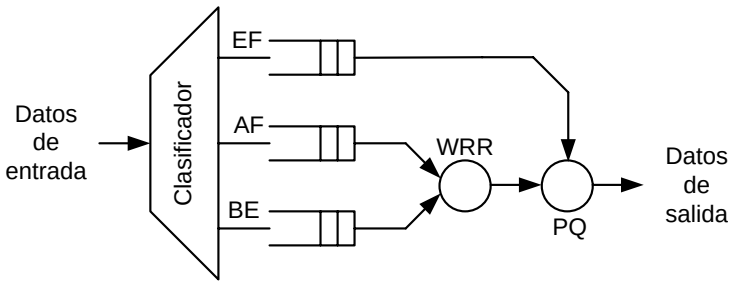


Figura 2.13: Planificador jerárquico *PWRR*

2.5 Arquitecturas de conmutación con QoS

A pesar de que ha habido un gran esfuerzo investigador en desarrollar políticas de planificación con garantías de calidad de servicio, muy pocas de ellas están enfocadas a su implementación en arquitecturas escalables, como por ejemplo, colas virtuales de salida o colas combinadas. La mayor parte de la investigación se ha centrado en proporcionar soporte a diferentes calidades de servicio en conmutadores con colas a la salida o con memoria compartida. Dada la limitada escalabilidad de estas arquitecturas, los siguientes apartados revisan varios algoritmos con soporte de calidad de servicio conjuntamente con su arquitectura de conmutación, dando una mayor importancia a las propuestas implementables físicamente.

2.5.1 Colas virtuales de salida

En [SPS99] se presenta un algoritmo basado en una arquitectura con colas virtuales de salida con el fin de proporcionar calidad de servicio a tráfico *ATM*. Dicho algoritmo permite configurar pesos y prioridades a fin de soportar tráfico con tasa constante (*Constant Bit Rate - CBR*), variable (*Variable Bit Rate - VBR*) y sin restricción (*Available Bit Rate - ABR*), aunque sólo es capaz de distinguir entre dos prioridades.

Chiusi y Francini [CF99] presentan una arquitectura jerárquica para resolver el soporte de la calidad de servicio. El tráfico se clasifica en tres colas diferentes en función de su principal característica: retardo garantizado, ancho de banda garantizado, o *best effort*. Cada cola se procesa con un algoritmo apropiado para el tráfico almacenado como pueden ser WF^2Q, *WRR* o *RR*. Posteriormente, se aplica un algoritmo de prioridad estricta para atender correctamente los diferentes tipos de tráfico. La ventaja más importante de esta arquitectura de conmutación es su flexibilidad a expensas de su alto coste *hardware* debido a la implementación de varios algoritmos más sencillos. Una publicación posterior [FC02] extiende los resultados previos para el tráfico *multicast*.

Otro planteamiento para proporcionar calidad de servicio en un conmutador con colas virtuales de salida consiste en dividir el mecanismo de planificación [TEA+03]. Por un lado, un planificador local situado en las tarjetas de los puertos de entrada y salida se encarga de diferenciar las colas virtuales. Por otro lado, un planificador central situado en la matriz de conmutación se encarga de diferenciar dos prioridades. Esta división de la lógica mejora las posibilidades de escalabilidad de la planificación sin empeorar el rendimiento del algoritmo.

Otras propuestas han surgido a partir de las evoluciones del algoritmo *iSLIP* [McK95]. En [McK99], McKeown presenta tres variantes basadas en las prioridades, los umbrales y los pesos para mejorar la funcionalidad del algoritmo original. En [GL02] se muestran mejoras en el control de las prioridades y en [YLZ03] se presenta un algoritmo con control dinámico del ancho de banda (*Dynamic DiffServ Scheduling - DDS*). El algoritmo de planificación *Hierarchical DiffServ Scheduling* (*HDS*) [YWL+04], una mejora del algoritmo *DDS*, se divide en dos fases. En primer lugar, se seleccionan los paquetes en las colas virtuales de salida teniendo en cuenta las diferentes calidades de servicio. En segundo lugar, un algoritmo combina los puertos de entrada seleccionados previamente con los puertos de salida a fin de maximizar el *throughput*. Este algoritmo garantiza un ancho de banda mínimo para tráfico

EF y *AF* y asegura un reparto justo del ancho de banda para el tráfico *BE*, siendo su complejidad menor que la del algoritmo *DDS*.

Otra propuesta, que se desmarca de la filosofía determinista de la planificación jerárquica, se presenta en [SCD07]. Salami *et al.* investigan sobre el soporte de diferentes calidades de servicio con una política de planificación basada en la probabilidad (*Iterative Probabilistic Scheduling - IPS*). Los resultados muestran cómo es posible mantener la justicia y garantizar la calidad de servicio sin utilizar prioridades estrictas en un conmutador con colas virtuales de entrada y colas a la salida. Los resultados presentados se han obtenido utilizando una distribución de Poisson pero, dada la característica principal de este algoritmo, no se puede garantizar su rendimiento al depender de la distribución del tráfico entrante.

Hay que destacar un aspecto que reduce el rendimiento de una arquitectura basada en colas virtuales de salida. La diferencia entre garantizar la calidad de servicio en una arquitectura con colas virtuales de salida y en una arquitectura con colas a la salida es, principalmente, la capacidad de conmutación limitada que tiene la matriz de conmutación en la arquitectura con colas virtuales de salida. En un conmutador con colas a la salida, cualquier paquete está disponible inmediatamente para ser elegido en el proceso de planificación debido a que no existe contención en los puertos de salida. Como resultado, se puede garantizar la calidad de servicio en un conmutador con colas a la salida utilizando un algoritmo de planificación adecuado en cada puerto de salida individualmente. Sin embargo, en un conmutador con colas virtuales de salida algunos paquetes no son transmitidos puntualmente entre los puertos. En consecuencia, dicho paquete pierde su oportunidad de recibir servicio en los puertos de salida y, en el peor caso, podría no cumplir algunos de los parámetros de calidad de servicio. En conclusión, el punto clave para proporcionar garantías de calidad de servicio en un conmutador con colas virtuales de salida es el diseño de un algoritmo que garantice que los paquetes almacenados se transmitan puntualmente entre los puertos. Este punto se puede extender a arquitecturas con técnicas combinadas, ya que, en menor

medida, sufren el mismo problema de conmutación que una arquitectura con colas virtuales de salida.

2.5.2 Colas a la entrada y a la salida

Al igual que en el caso de los conmutadores con colas virtuales de salida, las arquitecturas de conmutación con colas a la entrada y a la salida también son fuente de investigación. En [BDE+04] Balakrishnan *et al.* presentan un estudio sobre un algoritmo enfocado a mantener las tasas de bits de los flujos de datos en un conmutador repartiendo proporcionalmente el ancho de banda. Para alcanzar este objetivo, dicho conmutador requiere duplicar su velocidad interna. Con el mismo requisito de aceleración, pero con el objetivo de soportar diferentes clases de tráfico, Lee y Kuo [LK05b] presentan un algoritmo capaz de emular las prestaciones de un conmutador con colas reales de salida. En ambos casos, el requisito de la aceleración en el entorno de conmutadores de alta velocidad se considera muy exigente.

2.5.3 Colas a la entrada y memoria interna

Dentro del interés investigador en el campo de los conmutadores, las arquitecturas con colas a la entrada y memoria interna han sido objeto de un esfuerzo más intenso debido a las mejores prestaciones de la matriz de conmutación *crossbar* con memoria interna. En primer lugar, el hecho de proporcionar diferentes calidades de servicio implica la utilización de varias colas, tanto en los puertos de entrada del conmutador como en los elementos de conmutación de la matriz *crossbar*. Sin embargo, resulta especialmente interesante poder proporcionar servicio a diferentes clases sin tener que utilizar un gran número de colas. En [CK04] Chrysos y Katevenis proponen un método de ajuste dinámico de las prioridades de dos colas de almacenamiento por elemento de conmutación de manera que proporcionan servicio a múltiples clases. Sin embargo, esta reducción de la cantidad de memoria de almacenamiento se traduce en una menor diferenciación entre las clases de servicio.

Otro enfoque para soportar calidad de servicio consiste en incrementar la complejidad del algoritmo en las colas a la entrada manteniendo un algoritmo más sencillo para los elementos de conmutación y utilizar una única cola por elemento de conmutación. Estos algoritmos pueden estar basados en una combinación de sellos de tiempo y prioridades, como sucede en el algoritmo *Distributed Scheduling* (*DS*) [HYG06], o en mecanismos *Round Robin* modificados, como se presenta en [YQW06]. Cabe destacar de las aportaciones publicadas en [YQW06], la combinación de un algoritmo denominado *PWDRR* (*Priority Weighted Double Round Robin*) en las colas virtuales de salida con un segundo algoritmo denominado *CPRR* (*Compensation Priority Round Robin*) en las colas internas de la matriz de conmutación *crossbar*. El rendimiento obtenido es similar a un conmutador con colas reales de salida y planificación *Weighted Fair Queuing* (*WFQ*).

En otras investigaciones, el objetivo principal de una arquitectura y sus correspondientes algoritmos es asegurar el máximo *throughput*. En [OMW06], se presenta el algoritmo *GBSBF-LBF* (*Guaranteed Bandwidth Smallest Buffer First - Largest Buffer First*). Este algoritmo se basa en un mecanismo de créditos para asegurar el reparto del ancho de banda, y en el control del nivel de ocupación de las colas internas de la matriz de conmutación para mejorar la utilización de los recursos y las prestaciones del conmutador. Sin embargo, a fin de alcanzar un rendimiento elevado, el ancho de banda no reservado se reparte posteriormente utilizando un algoritmo que maximice el *throughput*. Esta propuesta aúna la garantía de mínimo ancho de banda reservado con un máximo *throughput*.

2.5.4 Colas a la salida

Una de las propuestas más recientes para ofrecer calidad de servicio en un conmutador con colas a la salida son las memorias *PIFO* (*Push-In, First-Out*). Asumiendo que se conmutan los paquetes y no existen limitaciones de velocidad en las memorias, cada paquete recibido se inserta ordenadamente en una memoria y sólo se lee el paquete de la cabecera de la cola. Su posición en dicha

memoria depende exclusivamente de la calidad de servicio definida. El principal inconveniente de esta arquitectura procede de la operación de ordenación. En el mejor caso, una operación de inserción tiene una complejidad temporal $O(logN)$. Este valor resulta elevado y no se puede implementar desde un punto de vista práctico para un conmutador de alta velocidad.

En [WH07] se estudia este tipo de almacenamiento y se propone una cola *PIFO* indexada (*iPIFO*) para soportar calidad de servicio. En un intento de emulación de esta memoria, Qiu *et al.* [QLY+06] proponen un conmutador con colas virtuales de entrada con prioridades, una matriz de conmutación *crossbar* con memoria interna, y un incremento de velocidad del sistema al doble para emular el rendimiento de las colas *PIFO*. Aunque la propuesta original de memorias *PIFO* es inabordable desde el punto de vista *hardware*, la emulación sí es posible, si bien con un alto coste *hardware*.

2.6 Arquitecturas comerciales

En las secciones anteriores se han descrito las arquitecturas de conmutación más significativas, así como algunas líneas de investigación centradas en el soporte de diferentes calidades de servicio desde una visión arquitectural. En esta sección se hace hincapié en los conmutadores comerciales, describiendo, a grandes rasgos, sus arquitecturas internas y sus prestaciones.

Actualmente, una gran parte de las topologías de conmutación comerciales adoptan arquitecturas basadas en *switched backplanes*, caracterizadas por la interconexión de varias tarjetas de línea, físicamente conectadas a un *backplane*, a través de una matriz de conmutación. En esta topología, se utilizan normalmente arquitecturas de conmutación basadas en colas virtuales de salida para diferenciar el tráfico en los puertos de entrada con el fin de soportar diferentes calidades de servicio, y colas en los puertos de salida a fin de poder agrupar los paquetes y solventar el problema de la aceleración de la matriz de conmutación.

La planificación del tráfico se puede llevar a cabo en los puertos de entrada, en la matriz de conmutación y en los puertos de salida. Sin embargo, la funcionalidad en cada ubicación es distinta. En los puertos de entrada se planifica el tráfico con el objetivo de proporcionar calidad de servicio. El planificador de la matriz de conmutación pretende maximizar la tasa de transferencia de paquetes. En el puerto de salida se implementan algoritmos muy sencillos con el objetivo de transmitir los paquetes al procesador de red.

Compañías como *Applied Micro Circuits Corporation*, *Broadcom Corporation*, *Dune Networks*, *Enigma Semiconductor*, *Erlang Technology*, *Fujitsu*, *Fulcrum*, *Integrated Device Technology*, *Marvell*, *Mercury Computer Systems*, *PMC-Sierra*, *Tundra* o *Vitesse Semiconductor Corporation* comercializan dispositivos que permiten la implementación de topologías de conmutación basadas en arquitecturas *switched backplanes*. En esta sección se profundizará en dispositivos representativos de las compañías *Vitesse Semiconductor Corporation*, *Applied Micro Circuits Corporation* y *Dune Networks* con el objetivo de ilustrar los diferentes conceptos comentados a lo largo del presente capítulo.

2.6.1 *Vitesse Semiconductor Corporation*

Los productos comerciales de la empresa *Vitesse Semiconductor Corporation* [VSC] se pueden clasificar en tres generaciones: *CrossStream* [VSC1][VSC2], *GigaStream* [VSC3] y *TeraStream* [VSC4]. *CrossStream* se compone de una matriz de conmutación (*VSC880*), y la lógica de gestión y almacenamiento de los paquetes (*VSC870*), siendo capaces de conmutar tráfico de longitud variable. *GigaStream* se compone de una matriz de conmutación (*VSC882*), y de un motor de colas inteligente (*VSC872*). Finalmente, *TeraStream* se enfoca al mercado de las redes de comunicación de alta velocidad y se compone de una matriz de conmutación *crossbar* con memoria interna (*VSC881*), y, de forma similar al *GigaStream*, de un motor de colas inteligente (*VSC871*).

El diseño de estos dispositivos de comunicaciones contó con la colaboración de la División de Diseño de Sistemas Integrados (*DSI*) del Instituto Universitario de Microelectrónica Aplicada (*IUMA*). A través de varios contratos con

la empresa *Vitesse Semiconductor Corporation* [Cross99][Giga00][Tera02], los investigadores de esta división diseñaron varias partes de estos dispositivos, concretamente los puertos de entrada y salida de *CrossStream*, los puertos de entrada de *GigaStream*, que incluyeron un algoritmo de planificación de tráfico con soporte de calidad de servicio, y los puertos de entrada de *TeraStream*. Estos trabajos consolidaron una línea de investigación y proporcionaron una inestimable experiencia en el diseño de arquitecturas de conmutación de altas prestaciones.

2.6.1.1 *CrossStream*

La arquitectura *CrossStream* se basa en enlaces serie de alta velocidad entre la matriz de conmutación y la tarjeta de línea. La Figura 2.14 muestra las conexiones entre ambos circuitos integrados. Los enlaces multiplexan y transfieren la información relativa a la petición y confirmación de los paquetes de datos a transmitir, la información de control de flujo y los paquetes de datos.

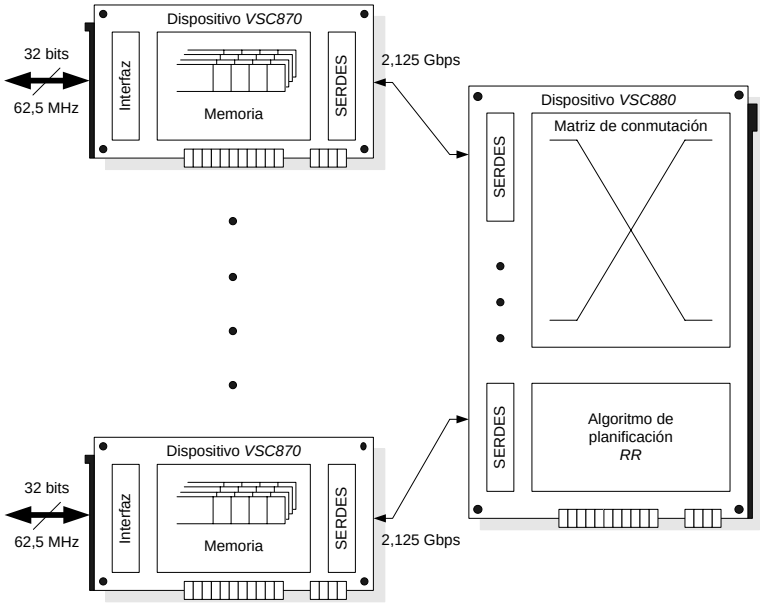


Figura 2.14: Arquitectura *CrossStream*

El circuito integrado *VSC880* se compone de una matriz de conmutación *crossbar* 16×16 con enlaces serie a 2,125 Gbps (2 Gbps para la transmisión

de los paquetes). Los puertos realizan las funciones lógicas necesarias previas a la conmutación, como por ejemplo, separar las peticiones de los datos multiplexados en el mismo enlace, o la inserción de las confirmaciones de los puertos de salida concedidos.

La planificación de los paquetes de longitud variable entre el *VSC870* y el *VSC880* durante la conmutación se realiza en tres pasos:

1. Cada tarjeta de línea (*VSC870*) envía múltiples peticiones a la matriz de conmutación (*VSC880*).

2. La planificación se realiza en el dispositivo *VSC880*. Cuando las peticiones alcanzan la matriz de conmutación, el planificador determina los puertos de salida libres y selecciona las peticiones sobre la base de un algoritmo *RR*.

3. La matriz comunica los puertos de salida concedidos a las tarjetas de línea a fin de comenzar la transmisión de los paquetes.

La arquitectura *CrossStream* puede soportar también la conmutación de células, pero este modo de funcionamiento no está integrado en la matriz de conmutación, aspecto que hace infrecuente a esta arquitectura. La siguiente generación de dispositivos de *Vitesse Semiconductor Corporation*, *GigaStream*, representa la arquitectura típica de conmutación de paquetes de longitud fija.

2.6.1.2 GigaStream

Descripción general

La Figura 2.15 muestra un conmutador *CIOQ* formado por los circuitos integrados de *Vitesse Semiconductor Corporation*. Los puertos de entrada y salida se encuentran situados en el dispositivo *VSC872*. Cada circuito *VSC872* proporciona dos interfaces independientes con un ancho de banda de hasta 5,312 Gbps (5 Gbps para la transmisión de los paquetes). Los paquetes recibidos se clasifican en función de su tipo y clase de tráfico, y se almacenan en la cola correspondiente.

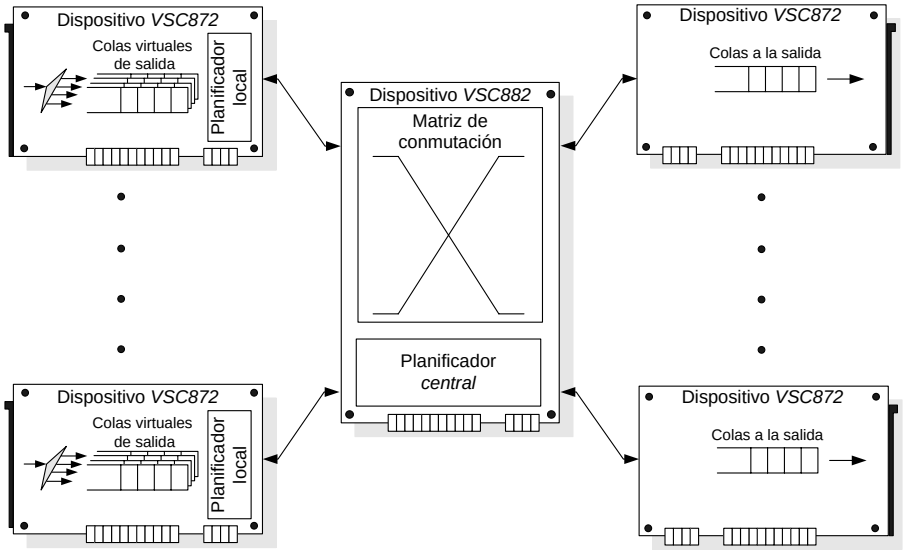


Figura 2.15: Arquitectura *GigaStream*

Debido a los problemas de bloqueo de cabecera, el dispositivo *VSC882* implementa una topología de colas virtuales de salida y proporciona 8 enlaces serie de conexión con la matriz *crossbar* que alcanzan una velocidad de 2,643 Gbps (2,5 Gbps para la transmisión de células). Como se dispone de 8 enlaces serie, se pueden transmitir hasta 8 paquetes simultáneamente en el mismo ciclo a 8 circuitos integrados *VSC882* diferentes. En consecuencia, el máximo ancho de banda de transmisión de un dispositivo *VSC872* es de 20 Gbps, mientras que las interfaces de los puertos de entrada reciben los paquetes con una tasa aproximada de 10 Gbps, lo que representa la utilización de aceleración interna y memoria para acomodar el tráfico en los puertos de salida.

Arquitectura de colas

La Figura 2.16 muestra el sistema de colas del dispositivo *VSC872*. El circuito soporta tanto tráfico *unicast* como *multicast*. En el caso de tráfico *unicast*, se establece una cola de máxima prioridad y siete colas de baja prioridad para cada uno de los puertos de salida. Por lo tanto, esta estructura se replica tantas veces como puertos de salida, es decir, 16. En el tráfico *multicast*, la topología de las colas de almacenamiento sigue la misma estructura que las colas de tráfico *unicast*. Cuando se recibe un paquete *multicast* se almacena

una única copia con múltiples punteros, transfiriéndose a cada destino independientemente y en sucesivos ciclos. Cuando se transmite a todos los destinos, se elimina el paquete. La realización física del dispositivo *VSC872* se presenta en la Figura 2.17.

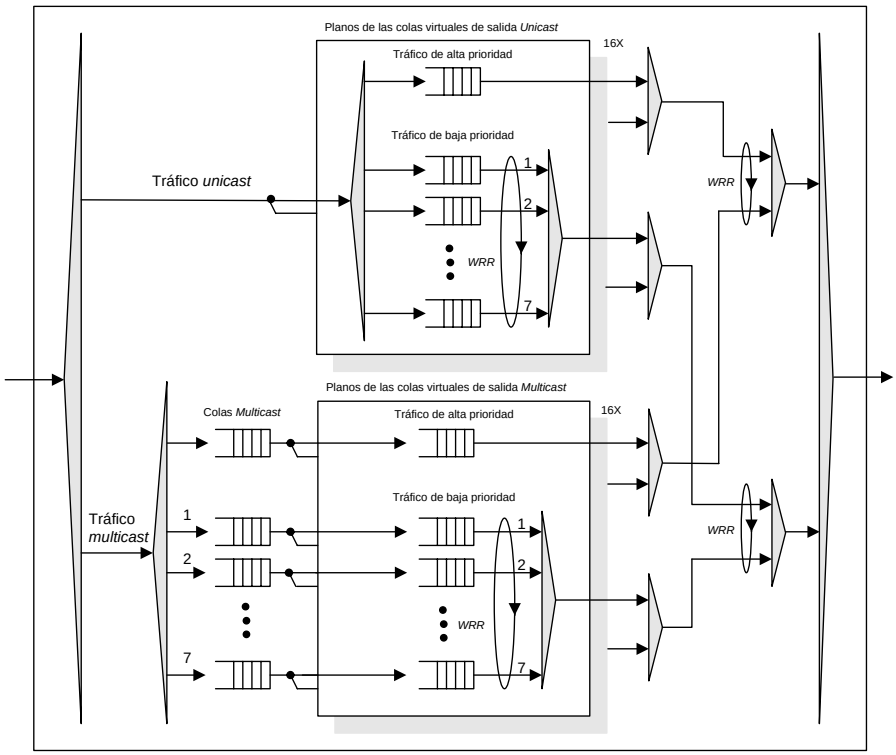


Figura 2.16: Estructura de las colas virtuales de salida del dispositivo *VSC872*

Algoritmo de planificación

La arquitectura *GigaStream* implementa tres fases de planificación que se llevan a cabo en ambos dispositivos. La primera fase genera las peticiones en el dispositivo local *VSC872* eligiendo todas las colas de alta prioridad y aquellas de baja prioridad que acumulen crédito suficiente. Dicho crédito se calcula sobre la base del peso asignado a cada clase de tráfico. En la segunda fase, el planificador central localizado en el dispositivo *VSC882* concede, en primer lugar, las peticiones de alta prioridad sobre la base de un algoritmo *iSLIP* modificado y, en segundo lugar, asigna el ancho de banda restante a las peticiones de baja prioridad. Finalmente, los puertos de entrada reciben los puertos de salida asignados y se selecciona el paquete a enviar.

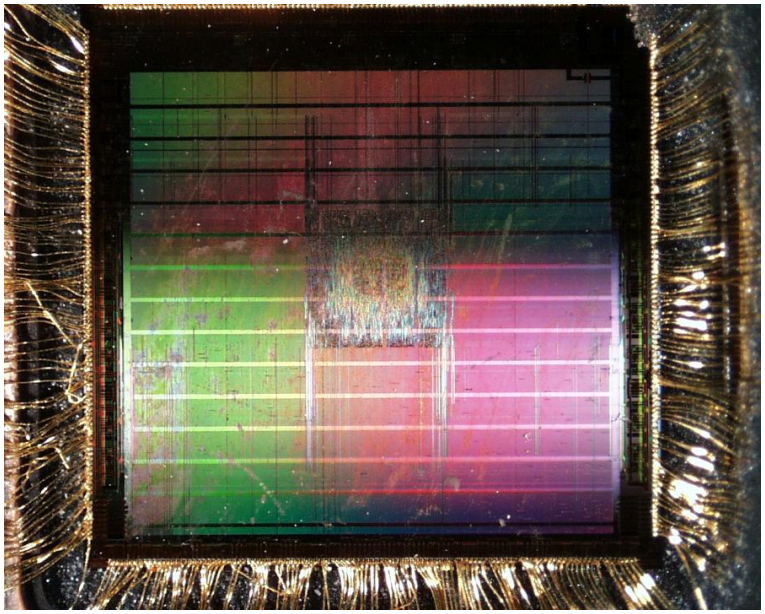


Figura 2.17: Realización física del dispositivo *VSC872*

Este algoritmo de planificación basado en una arquitectura *VOQ* con calidad de servicio fue concebido por Tobajas y presentado en [Tob01] como parte de su Tesis Doctoral en el año 2001.

2.6.1.3 TeraStream

La arquitectura *TeraStream* representa la tercera generación de dispositivos de *Vitesse Semiconductor Corporation*. Se caracteriza por una tarjeta de línea compuesta por el dispositivo *VSC871* y la matriz de conmutación *VSC881*, basada en un *crossbar* con memoria interna.

Esta arquitectura, comparada con los productos *CrossStream* y *GigaStream*, propone una topología capaz de alcanzar tasas de transferencia de 1 Tbps permitiendo la conexión de una gran cantidad de puertos de entrada en configuración multietapa, como se observa en la Figura 2.18. Además, facilita la combinación del tráfico de varios enlaces en uno único. Finalmente, implementa una matriz de conmutación *crossbar* con memora interna. Cada punto de interconexión dispone de dos colas de diferente prioridad, evitando la contención de los puertos de salida y proporcionando calidad de servicio.

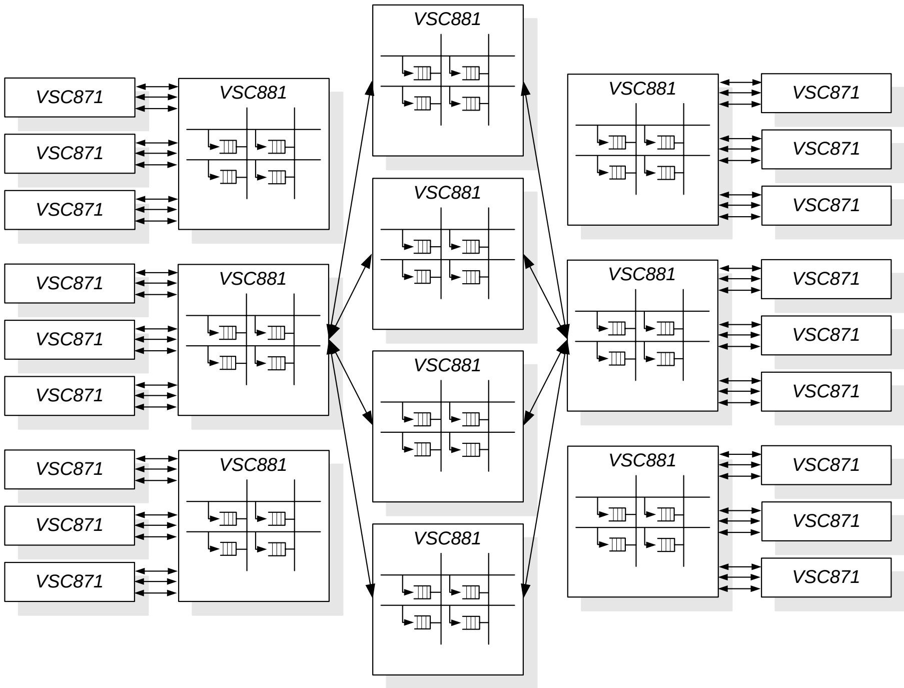


Figura 2.18: Configuración multietapa en el *chipset TeraStream*

2.6.2 *Applied Micro Circuits Corporation*

La empresa *Applied Micro Circuits Corporation* (*AMCC*) [AMCC] propone una arquitectura basada en una matriz de conmutación *crossbar* compuesta de múltiples circuitos integrados. Cada uno de estos dispositivos se encarga de un función específica: un subsistema de memoria (*S8905*), un mecanismo de control de colas con prioridades que dispone de varios algoritmos de planificación con diferentes calidades de servicio (*S8505*), y un árbitro central asociado a la propia matriz de conmutación *crossbar* (*S8605*). Es importante remarcar que *AMCC* se desmarca de la filosofía de utilizar únicamente dos circuitos integrados, típicamente la matriz de conmutación y el sistema de colas, alcanzando una gran flexibilidad y un alto *throughput* agregado. Esta plataforma se denomina *Cyclone*. La Figura 2.19 muestra un conmutador basado en esta arquitectura.

Al mismo tiempo, también se comercializa la arquitectura basada en dos circuitos integrados: un núcleo de conmutación (*PRS 80G* [AMCC1] o *PRS Q-*

80G [AMCC2]) y una interfaz en una tarjeta de línea (*PRS C48*/*C48X* o *PRS C192*/*C192X* [AMCC1][AMCC2]). La matriz de conmutación se caracteriza por la utilización de técnicas de memoria compartida, velocidad interna incrementada al doble y topologías 16×16 y 32×32. Las interfaces *PRS C48*/*C48X* se conectan a la matriz de conmutación a través de enlaces serie de 2,5 Gbps, en su versión original, y 3,2 Gbps, en su versión mejorada *X*, y tasas de transferencia de datos de 2,5 Gbps. En el caso de los dispositivos *PRS C192*/*C192X* las tasas de transferencia alcanzan los 10 Gbps. En la Figura 2.20 se observa una arquitectura de conmutación de 32 puertos, donde se aprecia la variedad de formatos posibles de los flujos de datos. Cabe destacar el soporte de diferentes calidades de servicio con 4 prioridades que permite elegir entre tres algoritmos de planificación: prioridad estricta, *WRR* y búsqueda exhaustiva para las clases de tráfico de alta prioridad.

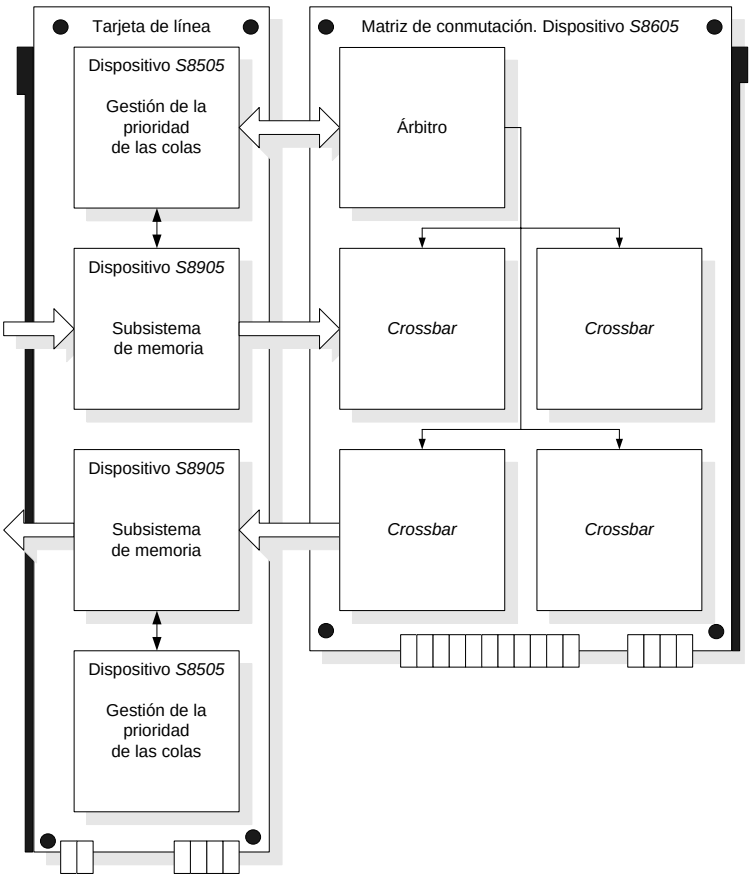


Figura 2.19: Arquitectura *Cyclone*

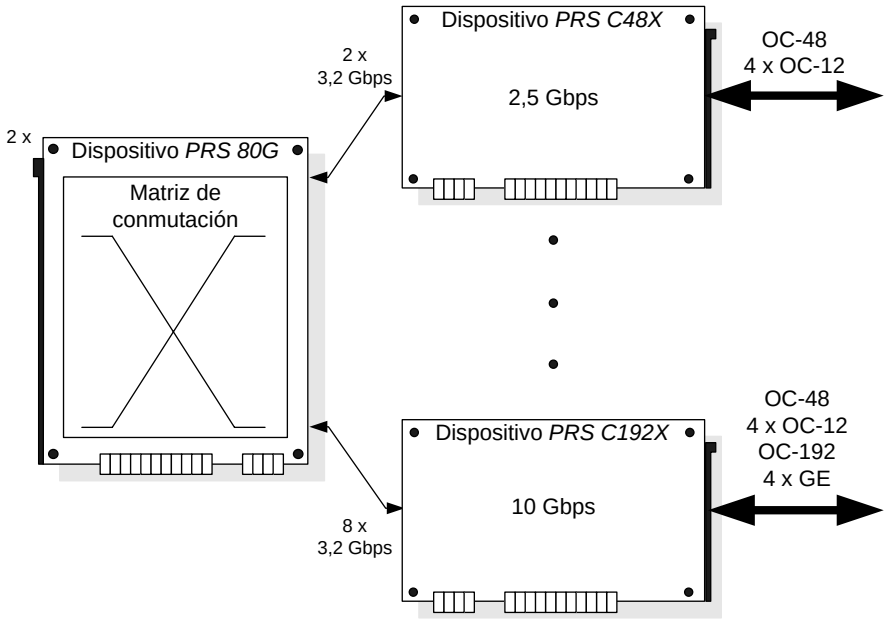


Figura 2.20: Arquitectura basada en la familia de dispositivos *PRS*

2.6.3 *Dune Networks*

La empresa *Dune Networks* [DN] proporciona una solución completa a la conmutación y gestión del tráfico en los puertos. Se compone de una matriz de conmutación con memoria compartida (*Fabric Element - FE200* [DN2]) y un número indeterminado de tarjetas de línea dependiendo del modelo del dispositivo elegido (*Fabric Access Processor - FAP10M* [DN3], *FAP10* y *FAP20V* [DN4]). La Figura 2.21 muestra la configuración básica de esta arquitectura.

Una de las principales características de estos dispositivos es su escalabilidad, debido a que admiten configuraciones multietapa y memoria externa. Entre los restantes detalles se puede mencionar el soporte de diferentes calidades de servicio garantizando un ancho de banda mínimo a cada flujo de datos y los algoritmos *Random Early Detection* (*RED*) y *Weighted Random Early Detection* (*WRED*) a fin de evitar la congestión de las colas.

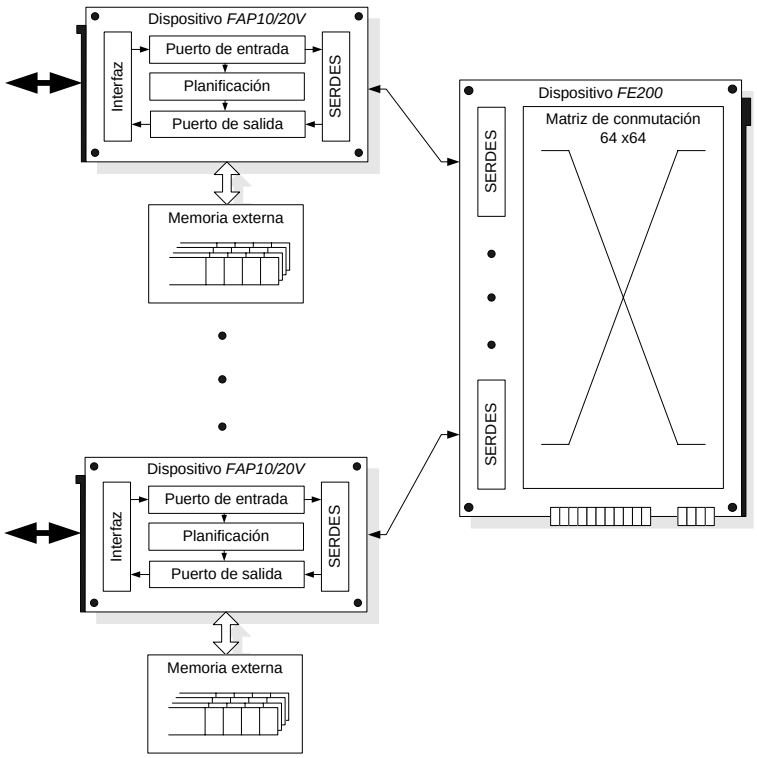


Figura 2.21: Arquitectura básica *FE200*

2.7 Resumen

A lo largo de este capítulo se han analizado las técnicas de almacenamiento, las arquitecturas de los conmutadores de paquetes y los algoritmos de planificación de tráfico con soporte de calidad de servicio más relevantes.

Desde el punto de vista del rendimiento, la arquitectura con colas a la salida proporciona el menor retardo con el mayor *throughput*. El principal inconveniente es la velocidad de la memoria, aspecto que se ha intentado paliar con la arquitectura *Knockout*. Sin embargo, dicha arquitectura introduce pérdidas de paquetes debido a la utilización de un concentrador. Por lo tanto, la implementación de un conmutador con colas a la salida sin pérdidas de paquetes es muy interesante, ya que proporciona las mejores prestaciones.

Además, dicha arquitectura con colas a la salida debe soportar calidad de servicio para conmutar el tráfico de las diferentes aplicaciones. Entre los me-

canismos disponibles de calidad de servicio destacan los Servicios Diferenciados, que permiten agregar en pocos flujos de servicio múltiples flujos de datos. Cada uno de estos flujos se procesa individualmente gracias al algoritmo de planificación. Este elemento, de gran relevancia, permite controlar varios parámetros de la transferencia de información, como el retardo o el ancho de banda, utilizando mecanismos basados en prioridades y créditos.

Los dos puntos anteriormente citados, la arquitectura de conmutación con colas a la salida y la planificación de tráfico con soporte de calidad de servicio, son los dos puntos en los que se engloban las principales aportaciones de esta Tesis Doctoral. En el siguiente capítulo se presentan las características, el modelo de alto nivel, la implementación, el coste *hardware* y el rendimiento de una nueva propuestas de arquitectura de conmutación con colas a la salida, denominada *Gigabit MultiDrop Switch* (*GMDS*). El algoritmo de planificación con soporte de calidad de servicio de este conmutador se explica detalladamente en el capítulo 4.

Capítulo 3

Arquitectura del conmutador *GMDS*

3.1 Introducción

Los avances tecnológicos han propiciado el crecimiento y el despliegue de las redes de comunicación basadas en la transmisión en el dominio óptico y en la conmutación de paquetes en el dominio eléctrico. Actualmente, aunque el esfuerzo de la comunidad científica se ha centrado en la conmutación en el dominio óptico [SS06] a fin de atender el continuo crecimiento de la demanda de ancho de banda y calidad de servicio, la rápida evolución de los circuitos integrados ha propiciado que los conmutadores de paquetes basados en *switched backplanes* sean la topología más habitual para la conmutación del tráfico de alta velocidad, como se describió en la sección 2.6 del capítulo anterior dedicado a las arquitecturas comercialmente disponibles. Dichas arquitecturas se basan en múltiples tarjetas de línea, cuyos interfaces pueden ser eléctricos y/u ópticos, conectadas con una matriz de conmutación a través de enlaces serie punto-a-punto pasivos [MLA+03]. Estos elementos se distribuyen físicamente en diferentes tarjetas en un *rack*, como se muestra de forma genérica en la Figura 3.1.

Sin embargo, con esta arquitectura se producen diversos efectos adversos que deben ser considerados, ya que limitan su aplicación práctica para los sistemas de conmutación de alta velocidad:

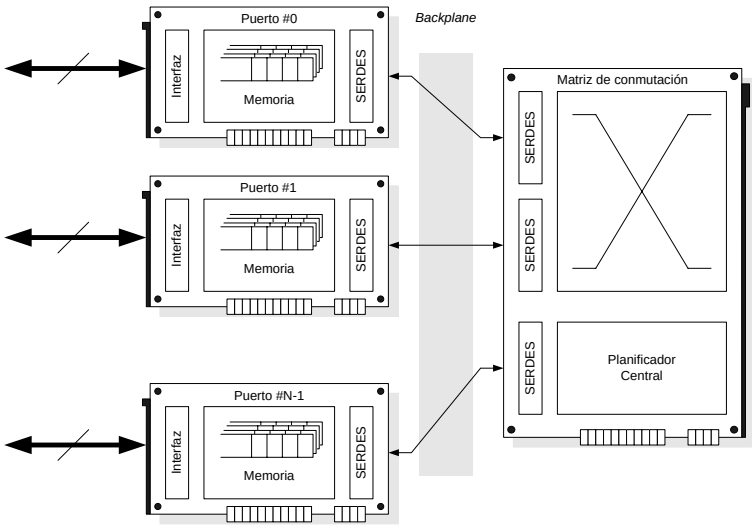


Figura 3.1: Arquitectura típica de un sistema de conmutación

1. El proceso de planificación muestra escasas posibilidades de proporcionar un amplio espectro de calidades de servicio, puesto que los paquetes no están siempre disponibles en los puertos de salida debido a la complejidad de la arquitectura y del algoritmo de planificación.

2. La transferencia de tráfico *multicast* reduce significativamente el rendimiento del sistema debido a la incapacidad de estas arquitecturas para conmutar tráfico con múltiples destinos de manera efectiva.

3. La sincronización de los puertos de entrada y salida, aspecto fundamental de los sistemas de transmisión punto-a-punto, presenta una importante desventaja desde el momento en el cual se incrementa la velocidad de la línea debido a los efectos del *skew* en la señal de reloj, entre otros.

4. La conmutación de paquetes de longitud variable reduce la eficiencia de la utilización del ancho de banda debido al *overhead* requerido por los mecanismos de segmentación y reensamblado.

5. La implementación de un mecanismo efectivo de control de flujo punto-a-punto en este tipo de arquitecturas resulta compleja debido al uso de una matriz de conmutación, que dificulta mantener actualizada la in-

formación del estado de las colas de almacenamiento en los puertos de destino.

A raíz de estas cuestiones quedan patentes las limitaciones que afectan a la escalabilidad y al soporte de calidad de servicio de las topologías actuales. Además, no son capaces de proporcionar soluciones adecuadas a la tendencia actual de mayor exigencia de ancho de banda para hacer frente a los nuevos servicios y, a su vez, la integración de todos ellos en la misma red de comunicación. Esta convivencia de diferentes servicios desemboca en la gestión simultánea de diferentes calidades de servicios. En consecuencia, la propuesta de una arquitectura de conmutación que proporcione una buena calidad de servicio, una transmisión eficiente de tráfico *multicast* y tramas de longitud variable, y que además simplifique la arquitectura de conmutación con el objetivo de evitar complejos mecanismos de planificación y gestión, resulta de gran interés.

En la presente Tesis Doctoral se propone la arquitectura de un conmutador que unifica las características anteriormente expuestas. Esta propuesta se basa en la utilización de un *backplane* serie *multidrop* que permite la comunicación entre un origen y múltiples destinos de forma innata y, a la vez, elimina la necesidad de disponer de una matriz de conmutación. Esta arquitectura se ha denominado *Gigabit MultiDrop Switch* (*GMDS*) [ATE+08] y se referenciará como conmutador *GMDS* en esta Tesis Doctoral. En la primera parte del capítulo se describe la arquitectura propuesta, haciendo especial hincapié en las ventajas que aporta frente a los conmutadores actuales. Posteriormente, se presenta un demostrador de la arquitectura.

3.2 Descripción general de la arquitectura de conmutación

La Figura 3.2 muestra la arquitectura del conmutador *GMDS* de tamaño 4×4. En la parte inferior de dicha figura se observan cuatro tarjetas de línea,

marcadas como #0, #1, #2 y #3, conectadas a través de un *backplane* serie *multidrop*.

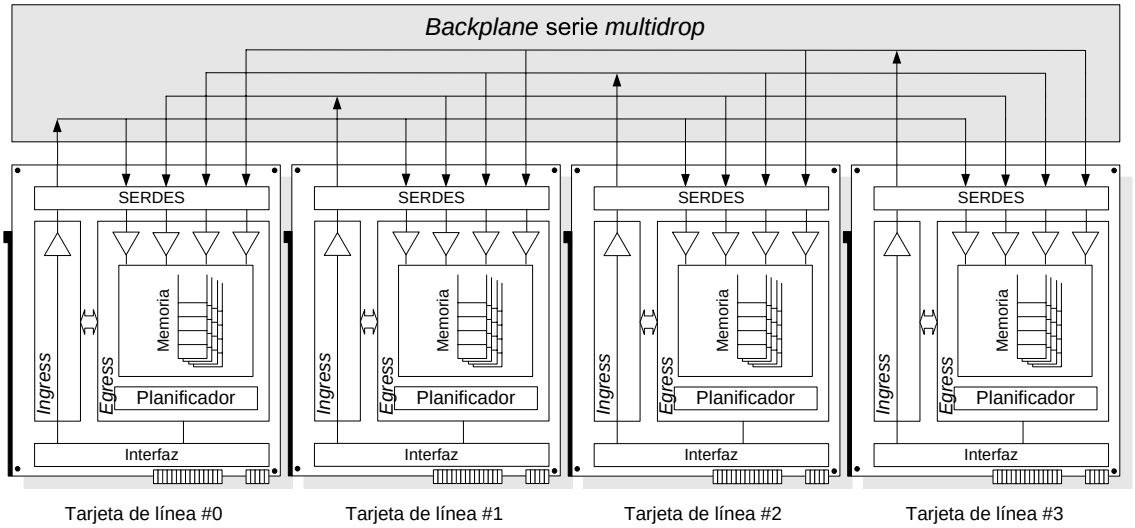


Figura 3.2: Arquitectura del conmutador *GMDS* 4×4

El *backplane* interconecta las tarjetas de línea a través de un conjunto de enlaces serie *multidrop* pasivos, los cuales se definen como la conexión de un punto con múltiples destinos en paralelo y en un único sentido. Por lo tanto, cada tarjeta de línea dispone de un enlace serie *multidrop* de salida que reciben las restantes tarjetas. Esta característica implica que cada *transmisor*, *puerto de entrada* o módulo *Ingress*, se conecta en paralelo a todos los *receptores*, *puerto de salida* o módulo *Egress*. Gracias a esta configuración, los puertos de salida se sincronizan localmente con todos los puertos de entrada, recibiendo los paquetes de cualquier fuente de tráfico y extrayendo únicamente la información dirigida a ellos.

3.2.1 Elementos principales

El conmutador *GMDS* se compone de dos elementos principales: una tarjeta de línea y un *backplane* serie *multidrop*. La tarjeta de línea se divide, a su vez, en un puerto de entrada, transparente a la comunicación, y un puerto de salida, donde se encuentran las colas de almacenamiento y el planificador

de tráfico. Esta configuración se corresponde con la topología de un conmutador con colas reales de salida. Además, este sistema se concibe para soportar diferentes calidades de servicio, por lo que cada puerto de salida se compone de múltiples colas *FIFO* con el objetivo de clasificar todos los paquetes en función de su origen y su clase de tráfico. Debido a las variaciones propuestas respecto a la topología original basada en colas a la salida, se considera importante presentar una descripción general del conmutador *GMDS* con el objetivo de clarificar su funcionamiento.

3.2.1.1 Modelo del puerto de entrada

El paquete entrante accede de forma casi transparente a los puertos de salida, como se muestra en la Figura 3.3. Los paquetes de datos procedentes de los puertos de entrada se almacenan en una cola antes de su transferencia a los puertos de salida. En esta cola no se produce ningún tipo de planificación. Su función es esperar a que se complete la recepción del paquete con el fin de verificar su integridad. Posteriormente, se multiplexan con la información de control de flujo procedente del puerto de salida de la misma tarjeta de línea. Al transmitirse esta información por el *backplane* serie *multidrop*, las restantes tarjetas reciben toda la información de control de flujo, es decir, cualquier tarjeta de línea conoce el estado de todas las colas de cualquier otro puerto de salida, permitiendo, de esta forma, la realización de un control de flujo individualizado.

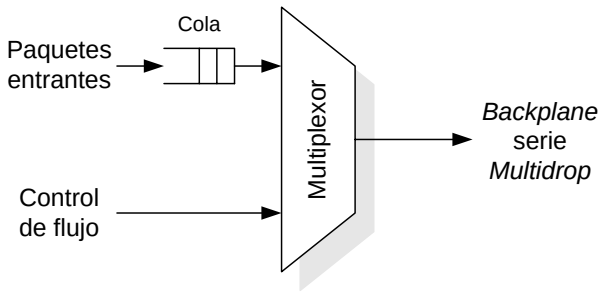


Figura 3.3: Modelo del puerto de entrada del conmutador *GMDS*

3.2.1.2 Modelo del puerto de salida

El puerto de salida es la parte más compleja del sistema. Su modelo se muestra en la Figura 3.4, en el que se representa las colas procedentes de todos los puertos de entrada. Debido a la característica *multidrop* del *backplane*, todos los puertos de salida reciben todos los paquetes. En consecuencia, se deben filtrar las direcciones de destino y descartar aquellos paquetes que no estén dirigidos al puerto de salida correspondiente. Posteriormente, se almacenan en función de su fuente y su clase de tráfico. La última parte mostrada en esta figura es el planificador de tráfico encargado de seleccionar los paquetes a transmitir. La información de flujo se procesa y se mantiene independientemente de los paquetes de datos.

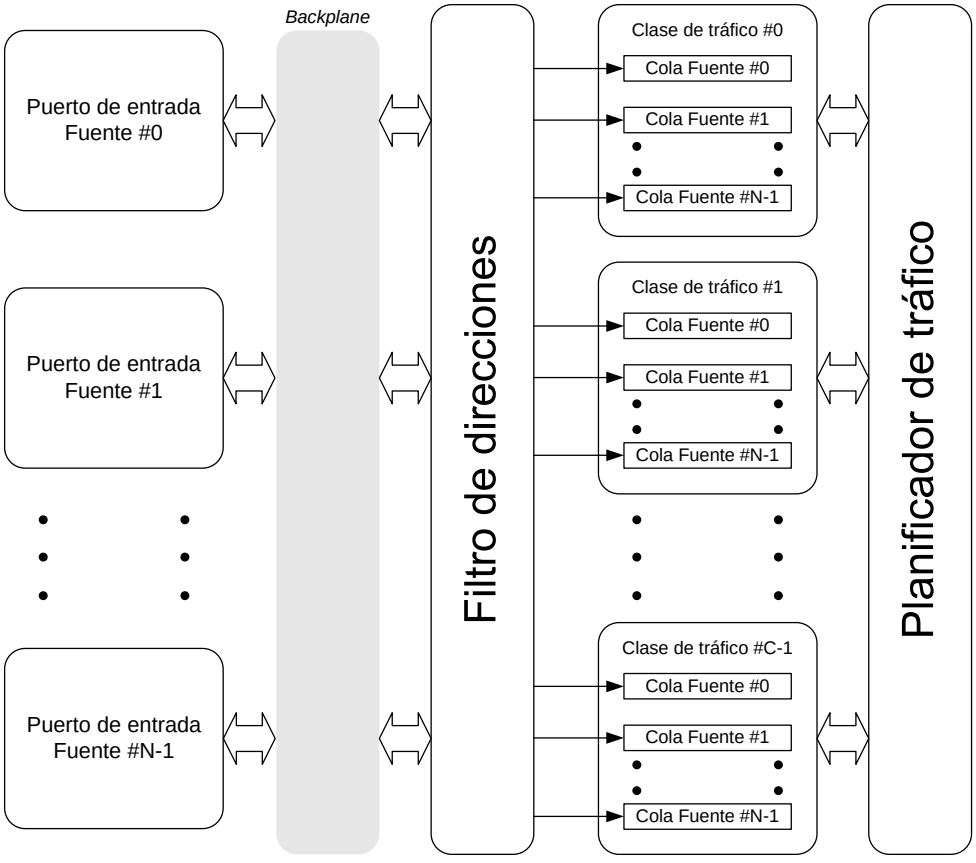


Figura 3.4: Modelo del puerto de salida del conmutador *GMDS*

3.2.2 Funcionamiento general

Los paquetes se reciben a través de una interfaz en el puerto de entrada y se multiplexan con la información de control de flujo del puerto de salida de la tarjeta de línea local. Este flujo de datos se transmite a través del *backplane* serie *multidrop* a todas las tarjetas de línea remotas, es decir, se produce una comunicación *broadcast* gracias a las características propias del *backplane* con el coste de una transmisión *unicast*. En consecuencia, los puertos de salida reciben simultáneamente los paquetes y la información de control de flujo de todos los puertos de entrada.

En cada tarjeta de línea, los puertos de salida deben, en primer lugar, filtrar los paquetes entrantes comprobando la dirección del destino. A continuación, un gestor de memoria se encarga de almacenar el paquete en su correspondiente cola, que se asocia a una fuente y clase de tráfico determinada. Al mismo tiempo, se actualiza la información del nivel de ocupación de esta cola. Finalmente, se transfiere al planificador la información del estado de todas las colas de la tarjeta de línea con el objetivo de que pueda seleccionar el paquete que se extrae de la memoria de almacenamiento.

La información de control de flujo realiza el mismo recorrido que los paquetes. Sin embargo, en el puerto de salida se mantienen registros de operación y mantenimiento con el objetivo de enviar información sobre el estado de las colas al puerto de entrada. En el caso de que la cola asociada a una fuente o clase de tráfico activara el mecanismo de control, el puerto de salida comunicaría al puerto de entrada la operación a realizar. Dicha información se multiplexaría con los paquetes entrantes y se transmitiría a través del *backplane*, lo que representa que todas las tarjetas de línea remotas conocerían el estado de la tarjeta de línea local. Por lo tanto, la tarjeta de línea remota podría adecuar su tasa de transferencia al estado de las colas locales, reduciendo o incrementado la tasa de transferencia. Al diferenciarse las colas de almacenamiento en función de sus fuentes y clases de tráfico, este mecanismo de control de

flujo extremo-a-extremo es capaz de incrementar el rendimiento del sistema debido a la posibilidad de gestionar cada cola individualmente.

3.2.3 Características generales

La configuración descrita anteriormente constituye un conmutador con colas reales de salida, ya que los paquetes se almacenan en los módulos *Egress* y, por lo tanto, se eliminan los problemas de contención y complejidad de los sistemas basados en colas a la entrada o con memoria compartida. Las principales ventajas que aporta la arquitectura propuesta son:

1. Se elimina la necesidad de la matriz de conmutación, elemento central en los conmutadores clásicos, que constituye el punto crítico en las prestaciones y la fiabilidad de un sistema de conmutación [MLA+03].

2. La ausencia de matriz de conmutación permite el acceso directo desde la entrada a la línea del *backplane* correspondiente a los puertos de salida, eliminando la necesidad de encolado y planificación en el módulo *Ingress*, y haciendo que el sistema sea muy simple en su entrada.

3. El uso de dominios de reloj independientes y no sincronizados facilita el desarrollo de los transmisores.

4. Dado que existe una línea dedicada para cada transmisor, no se necesita la sincronización entre ellos y, por lo tanto, se permite la transmisión de paquetes de longitud variable sin ningún tipo de aceleración ni relleno.

5. La transmisión de paquetes de longitud variable no introduce ningún mecanismo de segmentación y reensamblado, por lo que no se pierde ancho de banda en la transmisión y se simplifica el diseño de los puertos de entrada y salida. Esta característica supone que se aprovecha mejor el ancho de banda en la comunicación interna del conmutador y presenta una alta eficiencia.

6. La transmisión de paquetes *multicast*/*broadcast* es muy eficiente, ya que la característica *multidrop* del *backplane* permite la comunicación *broadcast* con todos los receptores con el mismo coste *unicast* de una comunicación punto-a-punto.

7. Posibilidad de soporte de calidad de servicio mejorada. La planificación se lleva a cabo en los puertos de salida frente a las políticas basadas en los puertos de entrada. En consecuencia, se simplifica el algoritmo de planificación y su política introduce una mayor componente determinista en la toma de decisiones en detrimento de la componente estadística, más común en la planificación en los puertos de entrada.

8. Soporte de control de flujo extremo-a-extremo basado en umbrales independientes para cada cola.

Sin embargo, la elección de esta arquitectura también presenta diversos desafíos que deben ser resueltos:

1. Acceso crítico a la memoria. La tasa de transferencia de la memoria es, en el peor de los casos, $n \times b$, donde n es el número de puertos y b es la tasa de transferencia de un puerto individual.

2. Adaptación de los dominios de reloj. Los puertos de salida reciben un flujo de datos con una tasa de bits de $n \times b$. Sin embargo, la tasa de salida es b. Esta adaptación puede resultar especialmente compleja.

3. Puerto de salida más complejo. En las arquitecturas con colas a la entrada o colas virtuales de salida, el *hardware* se concentra en el puerto de entrada frente al puerto de salida. En la arquitectura propuesta se invierte esta distribución y se introducen varios aspectos que no se han incluido en las arquitecturas del estado del arte de los conmutadores con colas a la salida, entre las que cabe citar el soporte de la calidad de servicio.

4. La escalabilidad del conmutador *GMDS* está condicionada por la capacidad de desarrollar un *backplane* que soporte un mayor número de puertos, un incremento de la velocidad de acceso a la memoria, y una tasa de transferencia de datos superior.

A lo largo de esta sección se ha planteado la arquitectura de conmutación con colas a la salida que se propone en esta Tesis Doctoral denominada conmutador *GMDS*. El aspecto central y más característico de dicha arquitectura es la topología basada en colas a la salida. Dicha propuesta ha sido posible gracias a la utilización de un *backplane* serie *multidrop* en lugar de una matriz de conmutación clásica. Sin embargo, esta solución requiere un análisis pormenorizado, puesto que plantea varios retos. En la siguiente sección se profundizará en la arquitectura propuesta y se expondrán las soluciones adoptadas para cada uno de los aspectos más críticos de esta aportación.

3.3 Descripción detallada de la arquitectura de conmutación

En la Figura 3.5 se muestra la arquitectura de una tarjeta de línea del conmutador *GMDS* para el caso particular de cuatro fuentes, donde se diferencia el puerto de entrada, *Ingress*, y el puerto de salida, *Egress*. Los paquetes se reciben en el módulo *Ingress* y se transmiten a través del *backplane* serie *multidrop* de forma transparente. Los módulos receptores, *Egress*, reciben los paquetes transferidos a través del *backplane* por todos los módulos transmisores. Dichos paquetes se filtran en función de la dirección del destinatario en el submódulo *Filtro*. Posteriormente, el submódulo *Gestor de Colas* clasifica dichos paquetes sobre la base de la tarjeta de línea origen y el tipo de servicio. Con esta información, el submódulo *Multiplexor* los almacena en la cola correspondiente de la memoria *RAM*. Toda la información relativa al número de paquetes almacenados y su clasificación se mantiene en el submódulo *Estado*, información con la cual el planificador solicita la lectura de los paquetes.

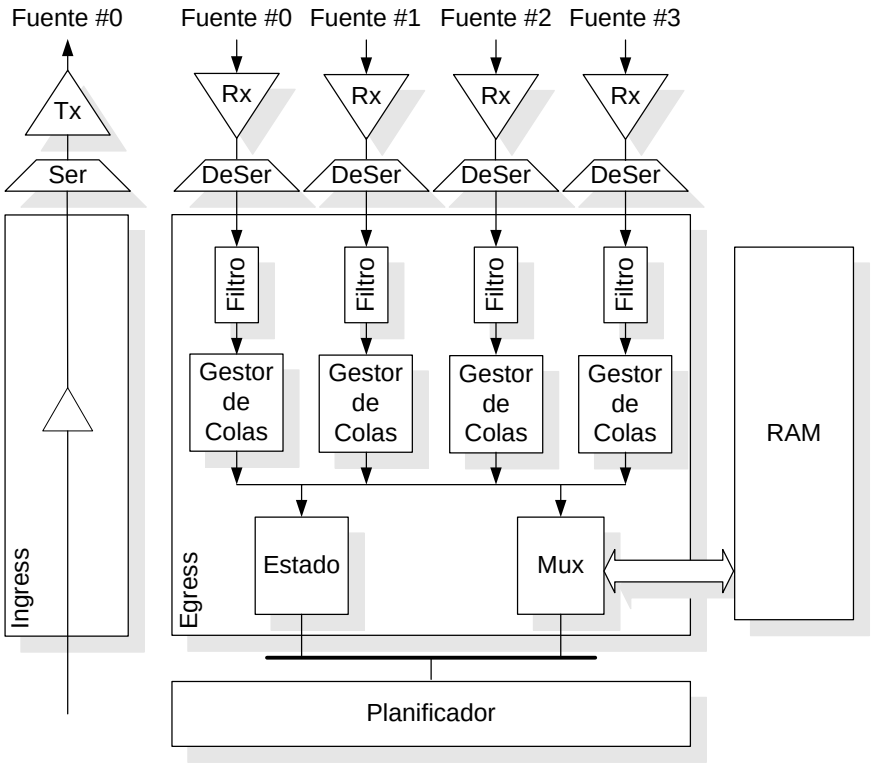


Figura 3.5: Tarjeta de línea del conmutador *GMDS* 4×4

La recepción y transmisión de paquetes en la tarjeta de línea sigue la interfaz *CSIX* propuesta por las empresas más importantes dentro del sector del diseño y fabricación de conmutadores y procesadores de red [CSIX]. Este consorcio promovió con esta interfaz la reducción de los costes de fabricación y la mejora de la interoperabilidad de los sistemas de conmutación de paquetes. En la sección 3.3.5.1 se presenta una visión más detallada de las características de esta interfaz.

3.3.1 Arquitectura del módulo *Ingress*

El módulo *Ingress* se encarga básicamente de recibir las diferentes tramas *CSIX*[1] procedentes de la fuente de tráfico y reconducirlas hacia el *backplane*

[1]Las características más relevantes de las tramas *CSIX* son la longitud variable del paquete, la clase de tráfico situada en la cabecera a fin de soportar calidad de servicio y la paridad vertical incluida en la cola del paquete con el objetivo de verificar su integridad. En la sección 3.3.5.1 se presenta una descripción detallada de la interfaz *CSIX*.

serie *multidrop*. Durante este proceso se mezcla el flujo de datos entrante con la información del control de flujo que le llega, por un lado, desde su propio módulo *Egress* y, por otro lado, desde la fuente de tráfico externa a través de la interfaz de entrada. Con dicho flujo de datos se compone la *multitrama*[2], formato propuesto en la presente Tesis Doctoral para la comunicación interna entre las tarjetas de línea. La Figura 3.6 muestra la estructura interna del módulo *Ingress* de una tarjeta de línea.

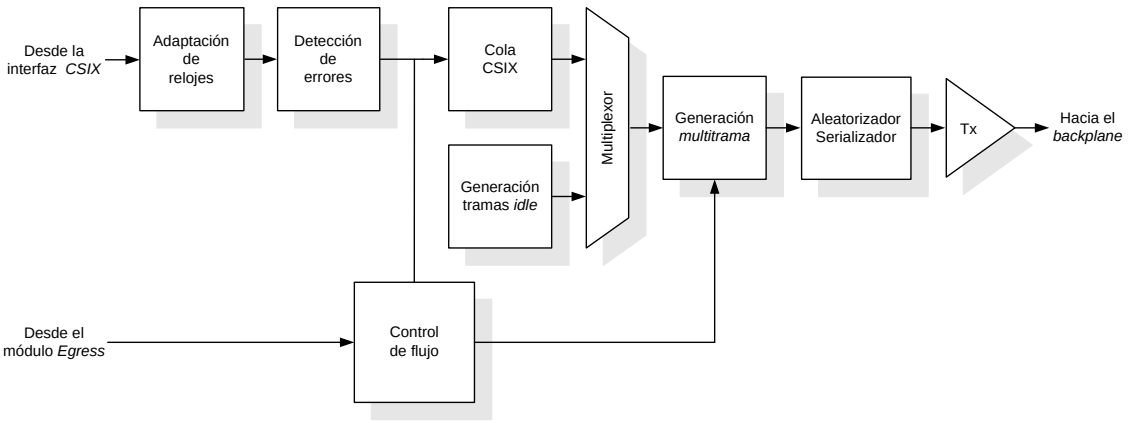


Figura 3.6: Arquitectura interna del módulo *Ingress*

En primer lugar se encuentra el adaptador de dominios de reloj entre la interfaz *CSIX* y el conmutador *GMDS*, ya que éste último posee su propia referencia temporal interna. En caso necesario también convierte el bus de comunicaciones externo en palabras de 32 bits, formato interno del conmutador *GMDS*.

Cuando el módulo *Ingress* recibe una trama *CSIX*, se verifica la paridad, la longitud del paquete y el tipo de trama y, de esta manera, se evita la posibilidad de transmitir una trama errónea. Paralelamente a la verificación y almacenamiento de la trama actual, se transmite la trama almacenada y verificada previamente. Al existir solapamiento entre la transmisión y la recepción, la capacidad de almacenamiento del submódulo *Cola CSIX* se debe establecer en dos tramas. En el caso de que la fuente de tráfico detenga la

[2]Las características de la *multitrama* se exponen en profundidad en la sección 3.3.5.2.

transmisión de tramas *CSIX*, la memoria se vacía. En consecuencia, se generan automáticamente tramas *CSIX idle* para mantener el enlace activo y no perder el sincronismo entre el módulo *Ingress* y el módulo *Egress*.

Los mensajes de control de flujo procedentes del módulo *Egress* y de la interfaz *CSIX* se almacenan en el submódulo *Control de Flujo* y se transmiten hacia las tarjetas remotas con el fin de soportar control de flujo extremo-a-extremo.

El submódulo *Multiplexor* combina las tramas procedentes de la interfaz externa con las tramas vacías generadas internamente. Dicho flujo de tramas se combina con la información de control de flujo con el objetivo de componer la *multitrama*.

Por último, la *multitrama* se aleatoriza y se genera el flujo de datos serie a partir de las palabras de 32 bits. Dicho flujo se transmite a través del *backplane* serie *multidrop* con la ayuda del transmisor serie encargado de adaptar las impedancias y generar la señal diferencial que se transmite a través de las líneas del *backplane*.

3.3.2 Arquitectura del módulo *Egress*

En la arquitectura propuesta del conmutador *GMDS*, el módulo *Egress* es la parte clave que define las prestaciones del conmutador. Previamente, en la Figura 3.5 se presentó la estructura general de la tarjeta de línea compuesta por los módulos *Ingress* y *Egress*. En este apartado sólo se profundiza en los detalles sobre la funcionalidad e implementación del módulo. La Figura 3.7 muestra la arquitectura de este módulo *Egress*, particularizado para una única fuente.

3.3.2.1 Submódulos *Rx* y *DeSer*

Cada enlace dispone de un receptor independiente (*Rx*) en la entrada del módulo *Egress*, encargado de regenerar la señal procedente del *backplane* serie *multidrop* y adaptar la impedancia de la línea. Dicha señal se transmite a la etapa deserializadora (*DeSer*), transformando el flujo de datos serie en

palabras de 32 bits, retemporizando y recuperando los datos recibidos, manteniendo una frecuencia de funcionamiento abordable por la lógica de baja frecuencia que compone el módulo *Egress*.

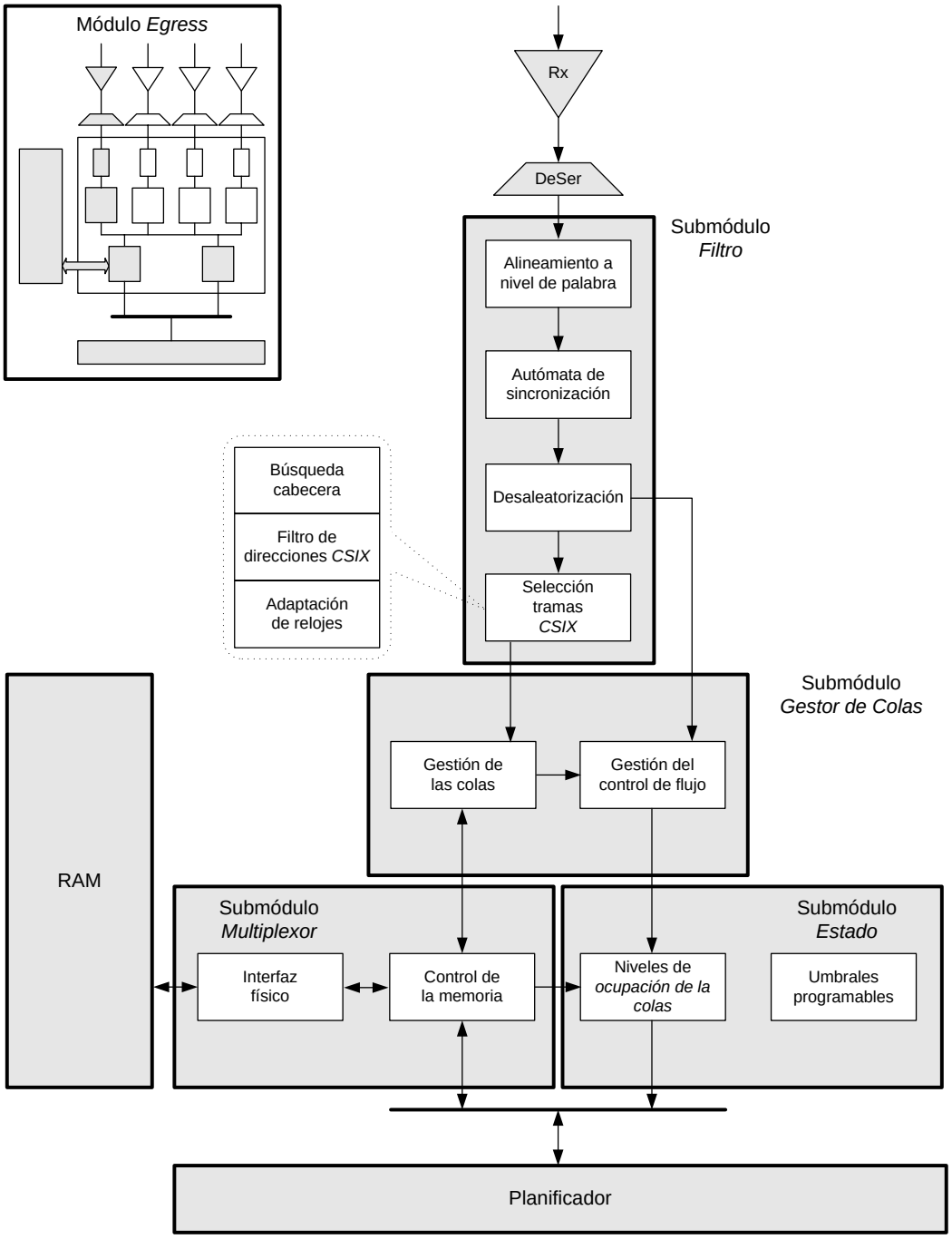


Figura 3.7: Arquitectura interna del módulo *Egress*

3.3.2.2 Submódulo Filtro

Una vez que el flujo de datos ha sido paralelizado, el submódulo *Filtro* se encarga de las tareas de sincronización de la *multitrama* y del filtrado de los paquetes.

La primera tarea del submódulo *Filtro* consiste en la sincronización del puerto de salida. Dicha tarea se lleva a cabo alineando la información entrante a nivel de palabra y activando el autómata presentado en la Figura 3.8 hasta que se encuentren tres *multitramas* correctas. Debido a las peculiaridades del *backplane* serie *multidrop*, cada submódulo *Filtro* se sincroniza con el transmisor de una fuente determinada, es decir, los receptores dentro del mismo módulo *Egress* se sincronizan independientemente. Por lo tanto, el conmutador *GMDS* puede soportar la parada, mal funcionamiento y posterior sincronización de un transmisor sin afectar al resto de los enlaces.

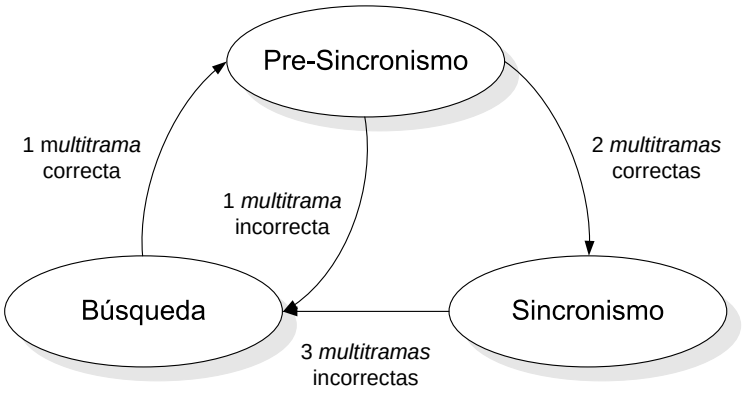


Figura 3.8: Autómata de sincronización

Una vez sincronizados el receptor y el transmisor, se desaleatoriza la *multitrama* y se localizan y analizan todas las tramas *CSIX* entrantes. Cada puerto de destino dispone de un registro de configuración local, que se compara con la información contenida en el paquete. Si el puerto de destino no coincide con la configuración local del puerto, o la trama presenta un error de paridad, se descarta. En caso contrario, se suministra la trama al submódulo *Gestor de Colas*, adaptando el dominio de reloj entrante del *backplane* se-

rie *multidrop* al dominio del reloj local del módulo *Egress*. Este proceso de análisis en el puerto de destino soporta tanto direccionamiento *unicast* como *multicast*.

La información de control de flujo insertada en la *multitrama* se filtra y se separa de las tramas *CSIX* con el fin de ser transferida por un bus dedicado al submódulo *Gestor de Colas*.

3.3.2.3 Submódulo *Gestor de Colas*

El submódulo *Gestor de Colas* se encarga del control de la escritura y la lectura de las tramas *CSIX* en memoria. Dicho control se realiza sobre la base de la fuente y la clase de tráfico de cada trama *CSIX*. Cuando se produce una escritura por parte de submódulo *Filtro*, se calcula la posición en la cola con la cual el submódulo *Multiplexor* puede calcular la dirección física base de la memoria. A partir de dicha dirección se comienza a almacenar la trama *CSIX* suministrada por el submódulo *Gestor de Colas* al submódulo *Multiplexor*. En el caso de la lectura, el planificador del conmutador solicita una trama *CSIX* de una fuente y una clase de tráfico determinada. El submódulo *Gestor de Colas* genera la posición correspondiente para que el submódulo *Multiplexor* comience la lectura de dicha trama. Con el fin de facilitar las tareas de lectura y escritura y, por lo tanto, la gestión de las colas, la distribución interna de la memoria se organiza en colas *FIFO* circulares y estáticas por cada binomio clase-fuente. En ambos casos, lectura y escritura, se informa al submódulo *Estado* con el objetivo de que mantenga el número exacto de tramas almacenadas en cada cola. Además, se le envía toda la información de control de flujo contenida en la *multitrama* que se recibe del submódulo *Filtro*.

3.3.2.4 Submódulo *Estado*

Cada vez que el submódulo *Gestor de Colas* almacena o extrae una trama de la memoria, se comunica con el submódulo *Estado* encargado de recopilar y mantener la información sobre el número de tramas *CSIX* almacenadas. Dicha información contiene el número de paquetes almacenados de cada fuente in-

dividual, de cada clase individual, de cada binomio clase-fuente de tráfico y de los valores totales del conmutador *GMDS*.

El submódulo *Estado* permite, por parte del usuario, la programación de los umbrales que controlan los estados de las palabras de control. Los diferentes controles de flujo que se pueden obtener programando adecuadamente el submódulo *Estado* son:

1. Control de flujo extremo-a-extremo por clase. Cuando el número de tramas almacenadas en una clase de tráfico de una fuente determinada cruza un umbral del tipo *ClaseFuente*, se envía un mensaje con el fin de modificar, ya sea incrementar o disminuir, la tasa de transmisión de las tramas de una fuente y clase de tráfico determinada.

2. Control de flujo extremo-a-extremo. Cuando el número total de tramas almacenadas en una fuente supera un umbral del tipo *FuenteTotal*, se modifica la tasa de transmisión de una fuente enviando un mensaje al transmisor.

3. Control de flujo por clase. Cuando el número de tramas almacenadas de una clase supera un umbral del tipo *ClaseTotal*, se envía un mensaje *broadcast* para modificar la tasa de transmisión de todas las fuentes de una clase determinada.

4. Control de flujo global. Cuando el número total de tramas almacenadas supera un umbral del tipo *TraficoTotal*, se envía un mensaje *broadcast* con el objetivo de modificar la tasa de transmisión de todas las fuentes con este destino, sin importar la clase de tráfico.

Los niveles de ocupación de las diferentes colas se comparan con una serie de umbrales programados por el usuario. En función de los resultados se generan los estados de control de flujo y éstos se envían al módulo *Ingress*. El generador de la *multitrama* inserta los estados de control de flujo en la posición correspondiente, siendo transferida por el *backplane* serie *multidrop*. En consecuencia, las restantes tarjetas de línea incrementan, reducen o mantienen

la tasa de envío de la información gradualmente en función de los estados de control de flujo recibidos.

Otra tarea de este submódulo es comunicar al planificador el estado de las colas. Concretamente, cuando así lo solicite, se indica si una cola está vacía o si dispone de una, dos o más tramas *CSIX* almacenadas. Por lo tanto, se puede consultar el estado de una clase o de una fuente asociada a una clase específica con el objetivo de planificar el paquete a extraer del conmutador *GMDS*.

3.3.2.5 Submódulo *Multiplexor* / Memoria

El submódulo *Multiplexor* recibe la instrucción de escritura de los diferentes submódulos *Gestor de Colas* a su cargo, así como las instrucciones de lectura del planificador. En ambos casos, el submódulo *Gestor de Colas* debe proporcionar la posición en la cola o dirección base a partir de la cual se calculen las direcciones físicas de la memoria. Además, ambas operaciones se suceden en paralelo, obligando a introducir aceleración en este submódulo.

3.3.3 Planificación de tráfico

El planificador, único por tarjeta de línea, secuencia la salida de los paquetes almacenados en las colas de tráfico y es el responsable de la asignación de los recursos a los flujos de datos. Por lo tanto, el propósito del planificador es priorizar el orden de salida de los paquetes de las colas de almacenamiento y su política se configura externamente, aunque requiere conocer el nivel de ocupación de las diferentes colas, información mantenida por el submódulo *Estado*.

Sin embargo, esta tarea no es trivial. Los requisitos para lograr un proceso de planificación de tráfico satisfactorio se pueden resumir en dos puntos clave: la protección y el aislamiento de los flujos de datos, y el reparto justo de los recursos. En este último punto aparecen tres elementos relacionados entre si: la probabilidad de pérdidas de paquetes, la asignación del ancho de banda y el retardo obtenido. Por ejemplo, acotar el retardo de un flujo de datos supone

asignar un ancho de banda en función de la tasa de transferencia máxima de dicho flujo. Sin embargo, si se admiten ciertas pérdidas de paquetes se puede considerar un valor medio de la tasa de transferencia, aunque se produzcan pérdidas cuando se alcance puntualmente la tasa de transferencia de pico.

El diseño del algoritmo de planificación es una tarea crítica, ya que combina muchos factores que lo hacen albergar una gran complejidad, además de determinar las prestaciones del conmutador. Por estos motivos, en esta tesis Doctoral se propone un planificador, especialmente concebido para la arquitectura de un conmutador de colas a la salida, que profundiza en los aspectos de la planificación de tráfico, entre los que se incluye el soporte de diferentes calidades de servicio. Debido a la relevancia de este aspecto, el siguiente capítulo se dedica enteramente al estudio y propuesta de un algoritmo de planificación con, entre otras, esta característica.

3.3.4 Arquitectura del *backplane*

En los enlaces serie *multidrop* las señales se transfieren de uno a múltiples puntos usando un único transmisor y múltiples receptores en paralelo, siendo la transmisión en un único sentido. Basándose en esta aproximación, un *backplane* serie *multidrop* consiste en un conjunto de tarjetas de línea interconectadas a través de un conjunto de enlaces serie *multidrop* pasivos. Esta arquitectura permite que cada receptor tenga acceso a todas las líneas de todos los transmisores, ya que tiene disponible una línea dedicada de cada uno, traduciéndose en la recepción de todos los paquetes transmitidos por todos los transmisores. La Figura 3.9 muestra la arquitectura de un *backplane* serie *multidrop* 4×4.

3.3.4.1 Descripción de un enlace serie *multidrop*

La configuración de los enlaces serie *multidrop* tradicionales se basa en una única línea de transmisión terminada con una impedancia característica Z_0 como se muestra en la Figura 3.10. Esta línea contiene un transmisor y múltiples terminaciones donde se conectan las tarjetas de línea al *backplane*. Para

minimizar el efecto de la reflexión de la señal transmitida en las líneas que no tienen conectadas una tarjeta de línea, se debe conectar una impedancia de terminación con el objetivo de lograr la adaptación de impedancias. El valor de la impedancia de terminación depende de la impedancia característica de la línea de transmisión, la distancia entre los conectores y las líneas del receptor, y la capacidad añadida por las tarjetas de línea.

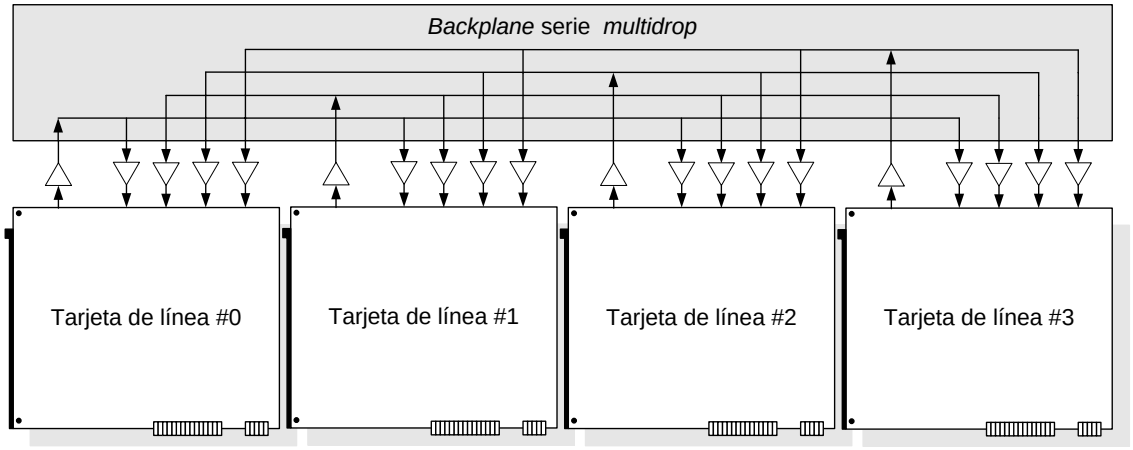


Figura 3.9: *Backplane* serie *multidrop* 4×4 con las tarjetas de línea

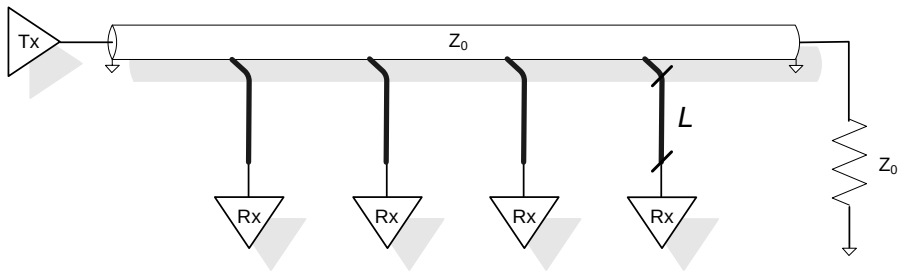


Figura 3.10: Configuración de un enlace serie punto-a-multipunto tradicional

Esta configuración de enlace serie *multidrop* representa la peor situación desde el punto de vista de la integridad de señal. Cada línea no terminada produce una serie de desadaptaciones de la impedancia característica Z_0 conduciendo a la degradación de la señal. Tanto el retardo de propagación como la impedancia característica de la línea se ven afectados. Estas reflexiones se pueden propagar en ambos sentidos de la línea de transmisión por un periodo

de tiempo substancial, degradando la integridad de la señal a alta velocidades e imponiendo limitaciones en la velocidad de los enlaces. Para minimizar la degradación de la señal debida a las líneas no terminadas, muchos *backplanes* de alta velocidad limitan la longitud de dichas líneas [NS1][NS2], siendo L la longitud mencionada mostrada en la Figura 3.10.

Dentro de esta arquitectura, se propone un *backplane* serie *multidrop* de alta velocidad basado en divisores de potencia asimétricos de banda ancha, como se muestra en la Figura 3.11, que aceptan una señal de entrada y la distribuyen entre múltiples salidas con una fase y una amplitud específica y, al mismo tiempo, mantienen la adaptación de todos puertos. Estas características hacen que la tasa máxima de transmisión de datos de un enlace serie *multidrop* esté limitada principalmente para el material de fabricación del circuito impreso, teóricamente 10 Gbps con un sustrato *FR4* [DBL+01]. Además, no se necesita limitar la longitud de las líneas de transmisión si todas las conexiones con las tarjetas de línea son terminadas adecuadamente.

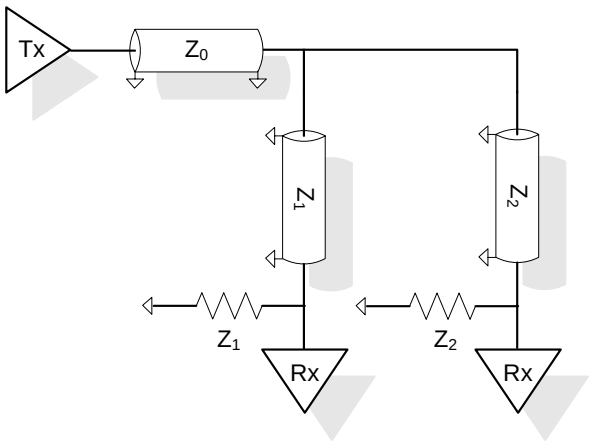


Figura 3.11: Configuración básica del enlace serie punto-a-multipunto propuesto

En [TEA+05][TET+05][ETT+05] se introduce un nuevo diseño de divisores de potencia asimétricos de banda ancha sin pérdidas. La configuración básica de este divisor de potencia consiste en una línea de transmisión bifurcada internamente en la cual las impedancias características de cada bifurcación, Z_1

y Z_2, deben cumplir que $Z_0 = Z_1//Z_2$, siendo Z_0 la impedancia característica de la línea de transmisión de la fuente. Como la impedancia formada por las dos líneas de transmisión bifurcadas en paralelo se adapta perfectamente a la impedancia característica Z_0, la transferencia de potencia es máxima. Además, como las resistencias de terminación son ajustadas de acuerdo con la impedancia característica de la línea de cada receptor, se consigue mantener un voltaje de salida igual al aplicado por el transmisor.

3.3.5 Otras características

En secciones anteriores se ha hecho mención a la comunicación entre el procesador de red y el puerto de entrada a través del interfaz *CSIX* y entre los puertos de entrada y salida a través del *backplane*. Ambos aspectos se describen detalladamente en los siguientes apartados. Posteriormente, gracias a la estructura e información contenida en la *multitrama*, se puede realizar el control de flujo extremo-a-extremo.

3.3.5.1 Interfaz con microprocesadores de red: *CSIX*

El consorcio *CSIX* fue fundado en Febrero de 1999 por algunas de las compañías más respetadas de la industria de los semiconductores, como *Applied Microcircuits Corp.*, *Broadcom Corp.*, *C-Port Corp.*, *Growth Networks*, *Maker Communications*, *Power X*, *Silicon Access*, *SiTera Corp.*, *Vertex Networks*, *Vitesse Semiconductor Corp.* y *XaQti*, para crear y promover una interfaz común para la transferencia de información entre procesadores de red y elementos de conmutación en sistemas de redes de alta velocidad, denominándose interfaz *CSIX* (*Common Switch Interface*) [CSIX].

La definición de interfaces entre componentes constituye un aspecto de especial importancia, facilitando la conexión entre circuitos integrados. Como filosofía general, en la interfaz *CSIX* se especifican únicamente aquellas funciones necesarias para la interoperación entre un procesador de red y un elemento de conmutación que necesitan ser implementados en *hardware*. De esta forma, se define una interfaz optimizada para las necesidades de comu-

nicación, incluyendo el direccionamiento *unicast*, así como diferentes mecanismos de direccionamiento *multicast* y varias clases de tráfico que permiten el aislamiento de datos dirigidos a un mismo puerto de destino, facilitando con ello la interacción con elementos de conmutación que soportan diferentes calidades de servicio.

La utilización de dispositivos que cumplan con la especificación del estándar *CSIX* permite reducir significativamente los costes de diseño y el *time-to-market*. Por un lado, los fabricantes de sistemas pueden combinar circuitos integrados de diferentes fabricantes, así como ampliar y/o modificar el diseño de productos básicos existentes añadiendo o sustituyendo componentes, pudiendo confiar plenamente en la interacción de los dispositivos. Por otro lado, la interfaz *CSIX* facilita la ampliación del mercado y reduce el coste del mantenimiento de sus productos a los fabricantes de elementos de conmutación.

La trama *CSIX*

La unidad definida en la interfaz *CSIX* para la transferencia de información y mensajes de control de flujo entre procesadores de red y elementos de conmutación *CSIX* se denomina trama *CSIX*. El formato de una trama *CSIX*, representado en la Figura 3.12, se compone de tres campos de información básicos: cabecera, datos y cola.

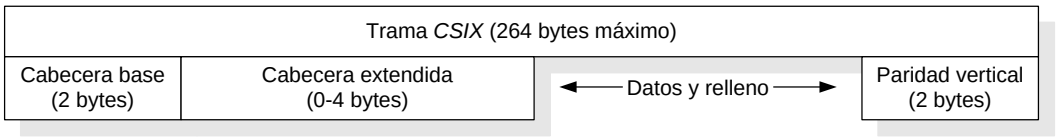


Figura 3.12: Formato de una trama *CSIX*

La cabecera de la trama *CSIX* contiene los campos de información a través de los cuales se proporciona al conmutador *CSIX* la información necesaria para realizar las tareas de planificación y conmutación. Esta cabecera se divide en una cabecera base de 2 bytes, que contiene información relativa a la longitud de los datos de la trama, el tipo de trama e información de control de flujo, y una cabecera extendida opcional de 4 bytes como máximo, específica

para cada tipo (*unicast*, *multicast*, *broadcast*, control de flujo, ...), que contiene información adicional necesaria para la transferencia de la trama, como puede ser la dirección de destino o la clase de tráfico.

La carga útil de la trama *CSIX* es de longitud variable, excepto en las tramas de control de flujo. Dicha longitud está limitada a 256 bytes como máximo.

La cola de la trama *CSIX* contiene un campo de paridad vertical de 2 bytes que se puede utilizar para la detección de errores en la interfaz *CSIX*.

Habitualmente, el bus de comunicación de la tarjeta de línea se compone de 1, 2 ó 4 bytes. En determinados escenarios puede ser necesario añadir bytes de relleno para asegurar que la trama *CSIX* sea de la longitud adecuada y, por lo tanto, múltiplo en bytes del ancho del bus de la interfaz. En caso contrario, se añaden bytes de relleno entre el final de los datos y el campo de la paridad vertical. Estos bytes de relleno se ignoran en el procesamiento de trama, pero se deben tener en cuenta en el cómputo de la paridad vertical.

3.3.5.2 Comunicación *Ingress-Egress*. Formato de la *multitrama*

La *multitrama* es la estructura de datos propuesta en la presente Tesis Doctoral con el objetivo de transmitir la información entre las tarjetas de línea a través del *backplane* serie *multidrop*. El formato propuesto se muestra de forma esquemática en la Figura 3.13.

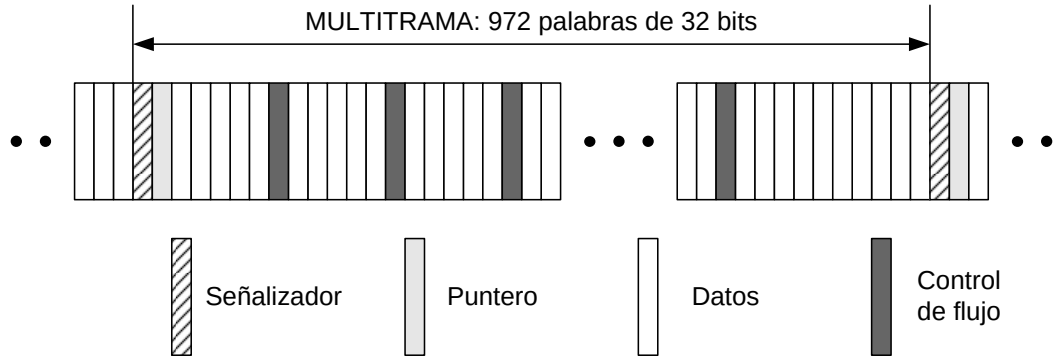


Figura 3.13: Formato de una *multitrama*

La información que contiene la *multitrama* comienza con un elemento de señalización encargado de indicar el inicio de cada una de ellas, facilitando de esta manera la sincronización entre los módulos *Ingress* y *Egress*, tanto a nivel de palabra como de *multitrama*. Por estos motivos, esta palabra es la única que no se puede aleatorizar de toda la estructura de datos. La Tabla 3.1 muestra el valor de este señalizador en hexadecimal (*E9 E9 E9 E9*).

Posición	Valor	Valor
0/1	E9	E9
2/3	E9	E9

Tabla 3.1: Formato del señalizador

A continuación del señalizador se introduce el puntero de *multitrama*, consistente en una palabra de 8 bits que indica la distancia existente entre la palabra siguiente al puntero y la posición del comienzo de la primera cabecera de una trama *CSIX* contenida en la *multitrama*, en palabras de 32 bits. Como se puede observar en la Tabla 3.2, el puntero es redundante y sólo se considera válido si al menos tres palabras contienen el mismo valor.

Posición	Valor	Valor
0/1	Puntero	Puntero (Redundante)
2/3	Puntero (Redundante)	Puntero (Redundante)

Tabla 3.2: Formato del puntero

Desde la posición 2 hasta la 968 se repite el patrón de 5 palabras de datos por cada palabra de control de flujo. Sólo al final de la *multitrama* se elimina una palabra de control de flujo, quedando 10 palabras de datos consecutivas para compensar las dos palabras de *overhead* iniciales.

Así, una *multitrama* se compone de un señalizador, un puntero, 810 palabras de 32 bits dedicadas a la información útil, y 160 palabras dedicadas a la información de control de flujo, haciendo un total de 972 palabras de 32

bits. Por lo tanto, el transmisor dedica el 83,3% de la capacidad del enlace a la transferencia de datos, dejando el 16,7% restante a la información de *overhead*. Dicha relación se mantiene siempre constante. Sin embargo, al transportar paquetes de longitud variable, no se conoce *a priori* el número exacto de tramas *CSIX* que contiene una multitrama.

3.3.5.3 Control de flujo entre *Ingress-Egress*

El conmutador *GMDS* soporta un mecanismo de control de flujo interno extremo-a-extremo que permite controlar el flujo de datos por fuente y por destino. Mediante este control, el módulo *Egress* informa al módulo *Ingress* del estado de las colas. El módulo *Ingress* recoge esta información, la procesa y la inserta en el *backplane* hacia un destino determinado.

El mecanismo de identificación de los puertos está basado en la posición de la palabra de control de flujo dentro de la *multitrama*. De esta manera, cada 5 palabras de datos se inserta una palabra de control de flujo comenzado cada *multitrama* con los puertos 0, 1, 2, y así sucesivamente hasta alcanzar la totalidad de los puertos del conmutador, repitiendo el proceso hasta que se alcance el final de la *multitrama*. Dadas las características de la arquitectura del conmutador *GMDS*, el destino conoce perfectamente cuál es el puerto de origen del que le llegan las tramas *CSIX*, por lo que se pueden emplear los 32 bits de una palabra exclusivamente para la información de control. Las clases de tráfico se numeran en orden ascendente empezando por los bits menos significativos de la palabra de 32 bits, de forma que a los bits 0 y 1 les corresponde la clase 0 (*spd_0*), a los bits 2 y 3 la clase 1 (*spd_1*) y así sucesivamente. En la Tabla 3.3 se representa el formato de la palabra de control de flujo generada.

Los mecanismos de control de flujo se basan habitualmente en dos estados: detener o reanudar una transmisión de datos. En el caso del conmutador *GMDS*, se propone un mecanismo de control de flujo que permite parar (*STOP*), acelerar o frenar por pasos (*SPEED*+/-), acelerar al 100% (*FULL SPEED*) y mantener la tasa de bits de una fuente de tráfico determinada. A diferencia del control de flujo desde la fuente de tráfico hacia el conmutador

GMDS, en este control de flujo se permiten diferentes niveles, tantos como el usuario quiera definir hasta un máximo de 16. Estos niveles permiten un mejor control de la tasa de transferencia, por lo que evitan el colapso de la memoria y la pérdida de paquetes sin necesidad de detener totalmente las fuentes de tráfico. Además, los controles de flujo extremo-a-extremo cobran especial relevancia en el rendimiento del conmutador *GMDS*, ya que permiten controlar individualmente una clase o fuente de tráfico específica logrando, por un lado, evitar el colapso de las colas congestionadas y, por otro lado, mantener una tasa menor de transmisión de paquetes sin necesidad de detener totalmente la fuente. Gracias a este control de flujo es posible controlar cualquier cola de almacenamiento individualmente. En la Tabla 3.4 se muestra la codificación de cada uno de los estados posibles.

Bytes	Posición del bit															
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0/1	spd_15		sp_14		spd_13		spd_12		spd_11		spd_10		spd_9		spd_8	
2/3	spd_7		spd_6		spd_5		spd_4		spd_3		spd_2		spd_1		spd_0	

Tabla 3.3: Formato de la palabra de control de flujo extremo-a-extremo

Codificación	Interpretación
00	STOP
01	SPEED -
10	SPEED +
11	FULL SPEED

Tabla 3.4: Codificación de los 2 bits de velocidad por clase

Dado que no se pueden codificar más de 4 estados con 2 bits, para controlar la tasa de bits se debe recurrir a un mecanismo de funcionamiento basado en el estado anterior y actual. La Figura 3.14 muestra este mecanismo de funcionamiento en el caso concreto de configurar cuatro niveles diferentes.

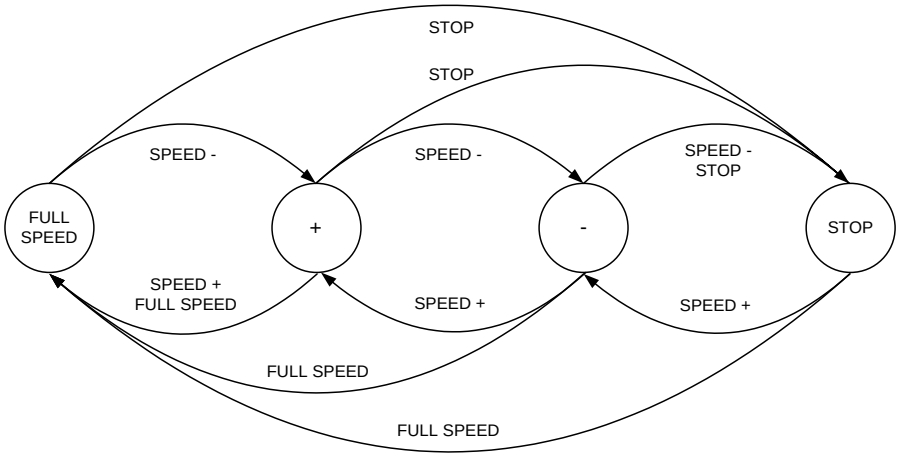


Figura 3.14: Máquina de estados con 4 niveles de control de flujo

Resumiendo, incrementar o reducir la tasa de bits de una fuente implica el envío consecutivo del estado *SPEED* + o *SPEED* -, respectivamente. Si el objetivo es mantener la tasa de bits, se deben mandar oscilaciones de *SPEED (+/-)*. Evidentemente, el sistema entiende que los incrementos y decrementos de la tasa de transferencia pueden alcanzar unos límites, *FULL SPEED* y *STOP*, los cuales no se pueden sobrepasar.

3.4 Implementación del prototipo para validación

En esta sección se presenta un prototipo de la arquitectura propuesta, el conmutador *GMDS* [ATE+08]. Sus especificaciones deben ser lo suficientemente restrictivas para que en el sistema se reproduzcan los problemas habituales de las arquitecturas con colas a la salida y de la integridad de la señal en el *backplane*. Desde el punto de vista tecnológico, las opciones elegidas para la fabricación del prototipo son *FPGAs*, debido a su capacidad de reprogramación, y circuitos integrados comerciales discretos debido a su coste. En consecuencia, se establece como objetivo la implementación de un conmutador de dimensiones 8×8 y con una tasa de transferencia de datos en sus puertos de

1 Gbps. El número de puertos establecido se considera idóneo para producir contención en los puertos de salida. Internamente, y debido a la información de *overhead*, la tasa de transferencia de los enlaces serie *multidrop* se establece en 1,2 Gbps. A esta frecuencia, las pistas del *backplane* se comportan como líneas de transmisión con todos los efectos asociados a ellas. Ambos valores de partida, 8 puertos y 1 Gbps, permitirán evaluar las prestaciones del conmutador en condiciones reales sin incurrir en un alto coste.

Según estas especificaciones, los módulos *Egress* de cada tarjeta de línea deben procesar los paquetes procedentes de ocho fuentes de tráfico. Para mejorar la escalabilidad y la modularidad del sistema, se definen bloques individuales capaces de procesar las fuentes en grupos de cuatro con su propia memoria. Consecuentemente, cada tarjeta de línea del prototipo se compone de un módulo *Ingress* y un módulo *Egress* formado por dos bloques que dan servicio a cuatro fuentes cada uno. La memoria utilizada por el módulo *Egress* es un circuito integrado externo a la *FPGA* con el fin de incrementar la capacidad de almacenamiento, obtener mayor fiabilidad y reducir el coste y la complejidad.

Un último punto sobre las especificaciones del conmutador es el soporte de la calidad de servicio. Aunque en esta sección sólo se considera el conmutador a efectos de su validación, se debe fijar el número de clases de tráfico para estructurar adecuadamente la memoria de almacenamiento. Por lo tanto, se establecen 16 clases de tráfico para diferenciar los flujos de información. El número de umbrales de control de flujo se establece en cuatro por lo que coincide con los estados de control de flujo.

3.4.1 Dominios de reloj

El conmutador *GMDS* es un sistema complejo compuesto por un gran número de módulos, descritos en los apartados anteriores. A lo largo de la ruta que recorre una trama *CSIX* entre el puerto de origen y su destino, atraviesa diferentes elementos y se realizan diferentes transformaciones, como por ejemplo la multiplexación con información de control de flujo o la paralelización. Por lo

tanto, la señal de referencia de cada módulo o submódulo puede cambiar sustancialmente. Esto supone que dentro del sistema existan diferentes dominios de reloj, como se observa en la Figura 3.15, que se describen a continuación.

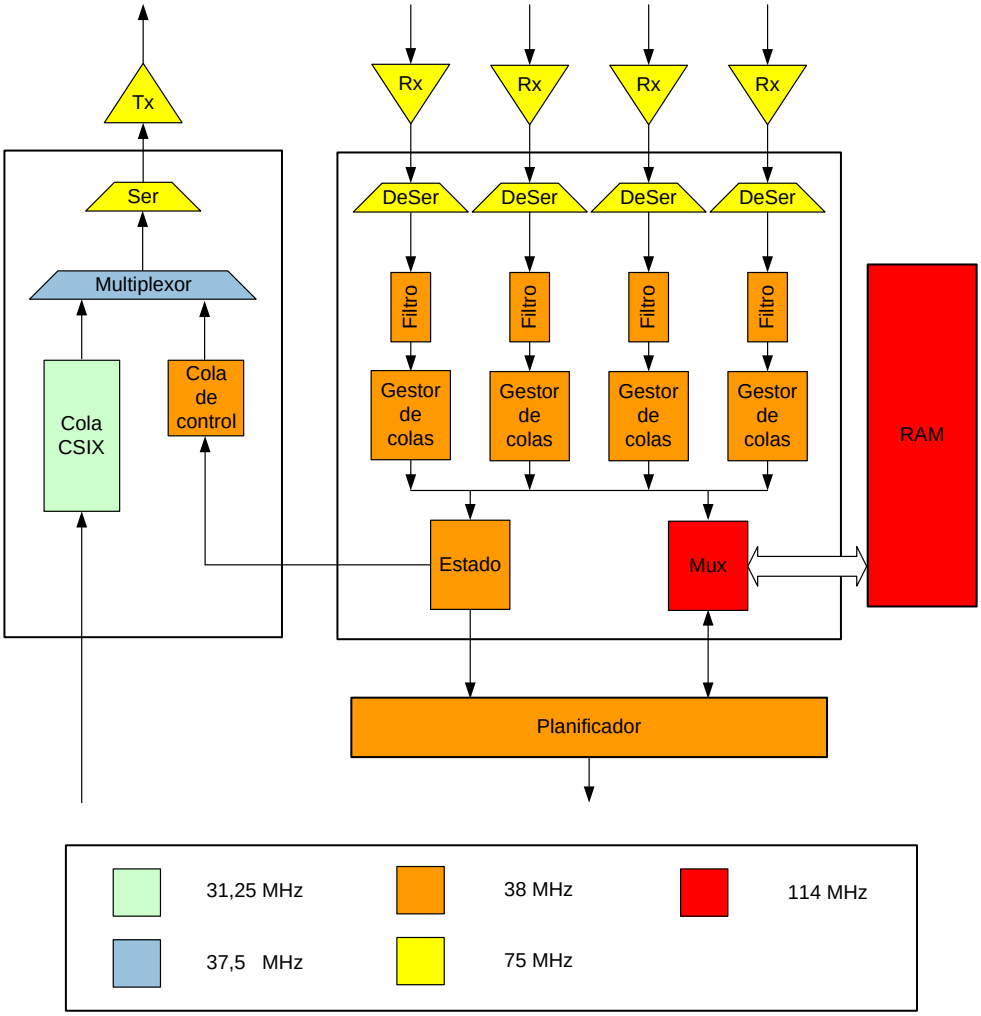


Figura 3.15: Dominios de reloj en una tarjeta de línea

- El módulo *Ingress* recibe los datos procedentes de la interfaz *CSIX* con una tasa de 1 Gbps. Los datos se almacenan en la cola *CSIX* de este módulo en palabras de 32 bits, resultando una frecuencia de funcionamiento de 31,25 MHz. El flujo de datos se combina con la información procedente de su propio módulo *Egress*. La tasa de transferencia de los datos de control de flujo es de 200 Mbps por lo que la tasa total del módulo *Ingress* es de 1,2 Gbps, es decir, 37,5 MHz en palabras de 32 bits.

- El dispositivo DS92LV16 de *National Semiconductor* [NS] es un conversor serie/paralelo y paralelo/serie de 16 bits con un rango de frecuencia de funcionamiento hasta 80 MHz. Al tratarse el conmutador *GMDS* de un sistema de 32 bits, la frecuencia de estos dispositivos debe ser el doble que la frecuencia del flujo de datos de salida del módulo *Ingress*, es decir, 75 MHz.

- El submódulo *Filtro* del módulo *Egress* se encarga de recibir los datos procedentes de cada fuente (*Ingress*) a través del *backplane* serie *multidrop* y adaptarlos al módulo *Egress*. La frecuencia nominal de este módulo es de 38 MHz. La frecuencia es ligeramente superior al mínimo necesario con el objetivo de evitar problemas de sincronización. Cada tarjeta de línea tiene su propio reloj local. La intención de incrementar ligeramente la frecuencia es evitar pérdidas de datos debido a una posible deriva temporal del oscilador generador del reloj del transmisor. Como segunda ventaja cabe destacar que no es necesario un dispositivo de enganche (*PLL - Phase Locked Loop* o *DLL - Delay Locked Loop*), sino un pequeño registro de adaptación más una señal de validación [EGT+03][ETG+04] que proporciona un *throughput* del 100%.

- El módulo *Egress* de un conmutador con colas a la salida debe ser capaz de realizar $(N+1)$ accesos a la memoria, siendo *N* procesos de escritura y *1* lectura. Como se ha definido que cada bloque del módulo *Egress* debe ser capaz de procesar cuatro fuentes de tráfico, la frecuencia de funcionamiento debería ser $38\ MHz \times (4\ escrituras + 1\ lectura) = 190\ MHz$. En la sección 3.4.2.1 se analiza en profundidad esta restricción de diseño y se propone una solución *hardware* a medida.

3.4.2 Implementación de la tarjeta de línea

La funcionalidad de la tarjeta de línea se implementa en una *FPGA* Virtex-II 4000 utilizando la herramienta *ISE* (*Integrated Software Environment*)

versión 8.2.02i de *Xilinx*. La Figura 3.16 muestra dicho dispositivo, además de varios conectores de alta densidad.

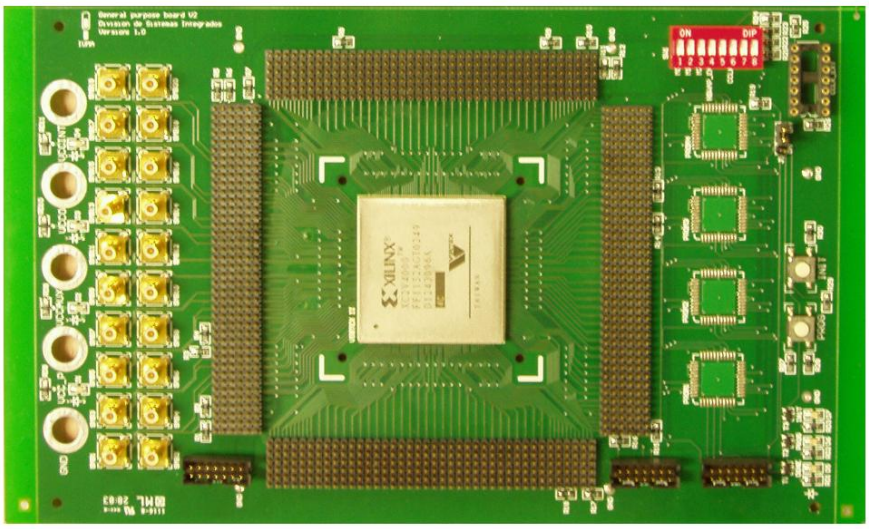


Figura 3.16: Fotografía de la tarjeta de línea

Con el fin de poder reutilizar la tarjeta de línea en posteriores proyectos, se decidió diseñar una placa de propósito general que tuviera parte de los recursos necesarios para la implementación de los módulos *Ingress* y *Egress*. Por lo tanto, dicha placa carece de determinados elementos necesarios para la implementación del conmutador *GMDS*, como los dispositivos conversores paralelo/serie y serie/paralelo DS92LV16 de *National Semiconductor* [NS], las memorias externas de almacenamiento, y las conexiones, por un lado, entre la *FPGA* y las memorias, y, por otro lado, entre la *FPGA* y el *backplane* serie *multidrop*. El resultado de esta política de reutilización supone la adición de una pequeña placa de interconexión entre el *backplane* serie *multidrop* y la tarjeta de línea.

En la Tabla 3.5 se presenta la lógica utilizada por un módulo *Egress* capaz de procesar cuatro fuentes. Cabe destacar el gran tamaño del submódulo *Filtro* debido a la inclusión dentro de la *FPGA* de las listas que contienen las direcciones de paquetes admitidos y rechazados.

Lógica utilizada	*Filtro*	*Gestor de Colas*	*Multiplexor*	*Estado*
Número de *Slices*	6779	3393	410	5808
Número de *Slices Flip Flops*	5580	2786	557	4369
Número de *LUTs*	10019	5906	400	8830
Número de *BRAMs*	8	0	0	0
N.^{o} de puertas equivalentes	638585	62361	9752	101736

Tabla 3.5: Resultados de la síntesis de un módulo *Egress* 1×4

La Tabla 3.6 muestra los resultados de la síntesis de los dos módulos principales, *Ingress* y *Egress*, así como la utilización porcentual de los recursos de la *FPGA*. Como era de esperar, la lógica del módulo *Ingress* es muy reducida en comparación a la del módulo *Egress*. En esta síntesis se han alcanzado las diferentes frecuencias descritas en el apartado anterior. Por último, se añade la estimación del número de puertas equivalentes de cada módulo, apreciándose su relativamente pequeño tamaño — aproximadamente un millón de puertas equivalentes—. Es importante remarcar que cada bloque de memoria equivale aproximadamente, y siempre según la propia herramienta de síntesis, a 65000 puertas equivalentes. Por lo tanto, si el sistema completo ocupa aproximadamente un millón de puertas, el 70% corresponde a las memorias utilizadas en el módulo *Ingress* y en el submódulo *Filtro*, y el 30% restante a la lógica del conmutador *GMDS*.

Lógica utilizada	*Ingress*	*Egress*	Total	FPGA(%)
Número de *Slices*	1018	15864	16882	50%
Número de *Slices Flip Flops*	998	12203	13201	20%
Número de *LUTs*	1334	24085	25419	38%
Número de *BRAMs*	3	8	11	8%
Número de puertas equivalentes	213884	790744	1004628	

Tabla 3.6: Resultados de la síntesis de la tarjeta de línea 1×4

3.4.2.1 Memoria

Según se estableció al comienzo de la sección 3.4, cada módulo *Egress* debe ser capaz de procesar cuatro fuentes de tráfico, por lo tanto, su memoria debe ser capaz de, en el peor caso, realizar cuatro operaciones de escritura y una operación de lectura en un ciclo de reloj. Es decir, la frecuencia de funcionamiento de la memoria debe alcanzar $38\ MHz \times (4\ lecturas + 1\ escritura) = 190\ MHz$. Este valor se debe analizar desde el punto de vista del dispositivo de almacenamiento y desde la lógica que controla dicho dispositivo, ya que ambos están sujetos a la misma restricción de velocidad.

La solución del almacenamiento de los paquetes consiste, bien en encontrar una memoria que alcance esta frecuencia, o bien la utilización de dos o más memorias en paralelo para lograr reducir la velocidad de funcionamiento. Comercialmente, la búsqueda de memorias que alcancen la frecuencia de funcionamiento de 190 MHz tuvo un éxito limitado. Aunque varias de las memorias disponibles sobrepasaban la velocidad exigida por el conmutador *GMDS*, dichas memorias estaban enfocadas a la utilización en ordenadores personales por lo que la lectura y escritura se realizaba en bloques. Por lo tanto, no tiene sentido diseñar un sistema de conmutación con tráfico de longitud variable si posteriormente se pierde la ventaja rellenando los paquetes en las operaciones de acceso a memoria.

La búsqueda de un dispositivo de almacenamiento comercial para el conmutador *GMDS* condujo al modelo MT55L512L32PT-6 de la empresa *Micron* [MT], caracterizado por su alta capacidad de almacenamiento, 8 MB, la arquitectura *ZBT* (*Zero Bus Turnaround*), que facilita la interfaz con la *FPGA*, y una frecuencia máxima de funcionamiento de 166 MHz. Sin embargo, ésta no es la única opción disponible en el mercado. La empresa *Integrated Device Technology* [IDT] dispone de las memorias *ZBT* 75K62234 y 75K72234, de 9 MB y 18 MB respectivamente, capaces de alcanzar 250 MHz. Dichas memorias son soluciones propuestas por la misma empresa para los sistemas de procesado inteligente de paquetes, ya sea en el mecanismo de clasificación

de paquetes o en el propio módulo *Ingress*, al no tratarse de un conmutador basado en colas a la salida.

La característica clave de estos dispositivos es la relación entre la frecuencia de funcionamiento exigida por el conmutador *GMDS* para realizar todas las operaciones de escritura y lectura, y su frecuencia de operación. La memoria de la empresa *Micron* seleccionada no es suficientemente rápida para alcanzar los requerimientos del conmutador *GMDS*, a no ser que se opte por utilizar dos memorias en paralelo. De esta manera, y dando esta opción por válida, sólo se necesita realizar dos escrituras y una lectura por parte del planificador, es decir, la frecuencia mínima necesaria de funcionamiento es $38\ MHz \times (2\ lecturas + 1\ escritura) = 114\ MHz$. Al mismo tiempo, también se reduce la frecuencia de funcionamiento del submódulo *Multiplexor* del módulo *Egress*. Un último efecto secundario beneficioso de la utilización de una segunda memoria es el incremento de la capacidad de almacenamiento. Además, esta memoria, al ser de gran tamaño, facilita el mantenimiento de las clases de tráfico, puesto que se usan bloques separados por clase y fuente de tráfico. El siguiente análisis muestra la estructuración de la memoria:

- Tamaño de la *RAM* de $256\ K\ direcciones\ \times\ 32\ bits = 8\ MB$.
- La memoria *RAM* se distribuye en bloques de $64\ posiciones\ \times\ 32\ bits = 2048\ bits = 256\ bytes$. Es decir, aunque los paquetes son de tamaño variable, se establece en 256 octetos el tamaño máximo por trama *CSIX*, de los cuales 248 bytes corresponden a la carga útil y 8 bytes al *overhead*, reservándose 64 posiciones de memoria para el almacenamiento de una trama *CSIX* de longitud máxima. De esta manera, se facilita el mecanismo de direccionamiento implementado en el submódulo *Multiplexor*, pero se reduce el aprovechamiento total de la memoria.
- De los cálculos anteriores también se puede extraer que la capacidad total de la memoria es de 4096 paquetes.
- Como cada memoria se comparte por dos fuentes, se dispone de 2048 paquetes por fuente.

- Como además se soportan 16 clases de tráfico diferentes, se pueden almacenar hasta 128 tramas por clase y fuente de tráfico.

A la vista de la arquitectura del conmutador *GMDS* y las características propias de un conmutador con colas reales de salida, la memoria elegida debe soportar la posibilidad de recibir paquetes de cuatro fuentes simultáneamente y, al mismo tiempo, realizar la lectura de un paquete que habrá sido solicitado por el planificador. Por lo tanto, los dispositivos comerciales de la empresa *Integrated Device Technology* [IDT] también son una solución válida para los requisitos de frecuencia del demostrador propuesto. Desde el punto de vista del proceso de síntesis en la *FPGA*, el cambio de frecuencia de 114 MHz a 190 MHz es asumible. Los resultados de la síntesis muestran que la FPGA Virtex-II 4000 es capaz de alcanzar frecuencias en torno a los 200 MHz en módulos o submódulos de área reducida.

Entre ambas opciones se ha elegido la disposición de dos memorias en paralelo, a pesar de incurrir en un incremento de coste *hardware*, ya que con la frecuencia de funcionamiento resultante se puede abordar el incremento de la velocidad del enlace serie *multidrop* para aumentar las prestaciones del conmutador *GMDS* en futuras versiones. Al tratarse la implementación propuesta de un demostrador, varias de sus características están limitadas. Por lo tanto, la escalabilidad del sistema supone un reto que se debe tener en cuenta y, dentro de lo posible, debe ser estudiado y analizado a partir de la implementación actual del sistema.

3.4.3 Implementación del *backplane* serie *multidrop*

El *backplane* serie *multidrop* construido [TEA+05][TET+05][ETT+05] soporta la interconexión de ocho tarjetas de línea a través de ocho enlaces series *multidrop* 1-a-8, donde todos los receptores reciben el mismo conjunto de enlaces serie *multidrop* dedicados. La adecuada selección de la impedancia característica de la línea de transmisión en este *backplane* es una cuestión clave en el diseño de cada enlace serie *multidrop* 1-a-8.

En este sentido, para la implementación de cada enlace serie *multidrop* 1-a-8 se ha adoptado 25 Ω como la impedancia característica Z_0 de la línea de transmisión, por representar un buen compromiso entre los requisitos del ancho de las pistas y las impedancias más altas controlables que se pueden alcanzar en el sustrato FR4 estándar. Además, para minimizar el máximo valor requerido de impedancia en la construcción de un *backplane* serie *multidrop*, cada enlace serie *multidrop* diferencial 1-a-8 se implementa con dos divisores de potencia asimétricos 1-a-4 diferenciales en paralelo. Esta estructura se muestra en la Figura 3.17, donde cabe destacar la asimetría en el reparto de potencia y la perfecta adaptación. En esta implementación, el transmisor debe ser capaz de atacar una carga formada por una impedancia de 25 Ω, mientras que el valor de las resistencias de terminación debe adaptarse a 100 Ω. Para atacar de forma efectiva una resistencia de 25 Ω se utilizan amplificadores *MAX9400*, de la empresa *Maxim* [MX], en paralelo, caracterizados por entradas abiertas y salidas en emisor abierto, mientras que un divisor adicional 1-a-2 con conexiones cortas entre el par de *buffers* proporciona la mejor adaptación de impedancias y minimiza la capacidad en las salidas del transmisor.

En la Figura 3.18 se muestra la foto del prototipo implementado del *backplane* serie *multidrop* 8×8 con una tarjeta transmisora y cuatro tarjetas receptoras. Las dimensiones finales del esta placa son 34 cm $\times$ 52 cm, y se han utilizado dos planos de señal y dos planos de masa. La principal característica del diseño de esta tarjeta es que el espesor del sustrato entre la cara superior y el primer plano de masa es de medio milímetro, mientras que el espesor del sustrato entre la cara inferior y el segundo plano de masa es de un milímetro y medio. Gracias a estas diferencias se facilita el proceso de fabricación de las líneas de transmisión *microstrip*, ya que las impedancias más bajas, 25 Ω y 33 Ω, se implementan en las capas más cercanas al plano de masa, mientras que las impedancias más altas, 50 Ω y 100 Ω, se implementan en las capas más alejadas del plano de masa.

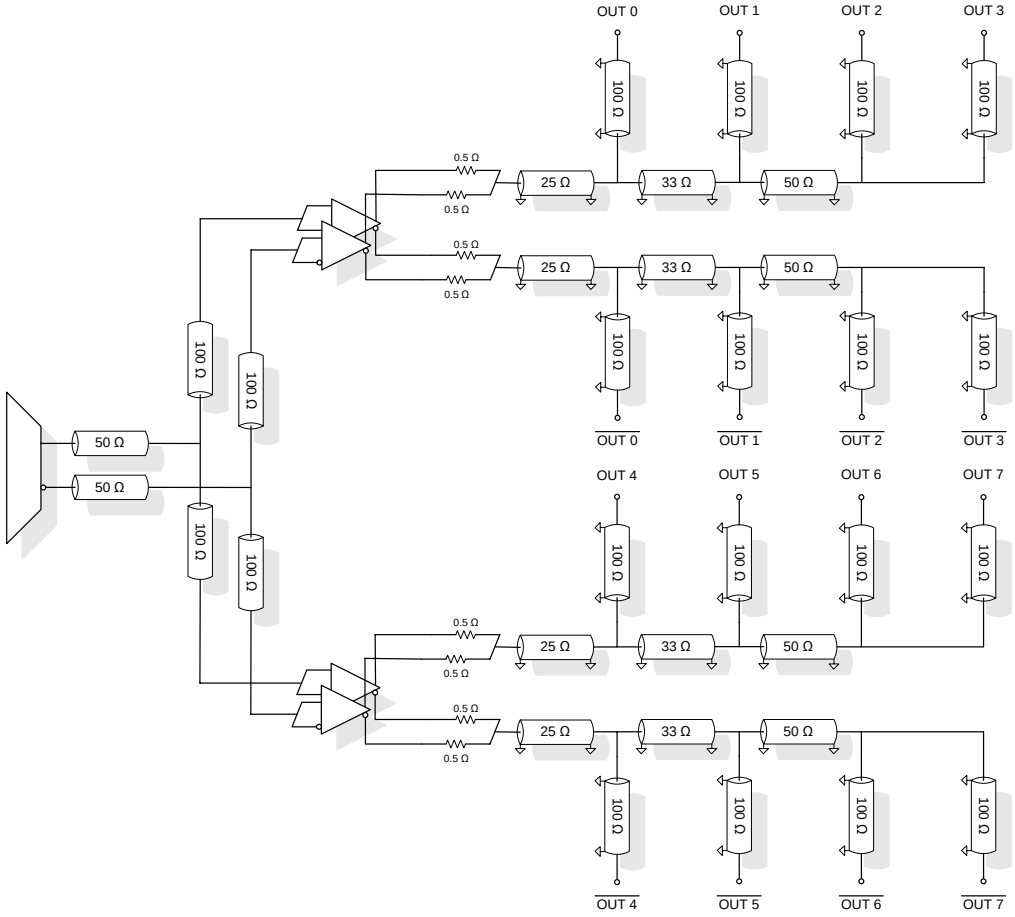


Figura 3.17: Implementación de un enlace serie *multidrop* 1-a-8

Este prototipo se ha alimentado con tensiones de 2 y -0,5 voltios, introduciéndose una señal diferencial de 630 mV_{pp} y 780 mV de *offset* con una tasa de transferencia de 1,5 Gbps en la entrada del *buffer*. De las cuatro salidas disponibles del divisor de potencia, la Figura 3.19 muestra las tensiones diferenciales de la más cercana mientras que la Figura 3.20 de la más alejada. En ambos casos, los niveles de tensión en la salida son muy cercanos a 300 mV_{pp}, suficiente para que el dispositivo DS92LV16 detecte las transiciones.

Las Figuras 3.21 y 3.22 muestran el diagrama de ojo de la salida más cercana y más alejada, respectivamente. El *jitter* medido en cada una de sus correspondientes salidas es de 16,7 ps y 34,1 ps. Se destaca, en ambos casos, la integridad de la señal obtenida, estando las restantes salidas adaptadas con una impedancia de 100 Ω.

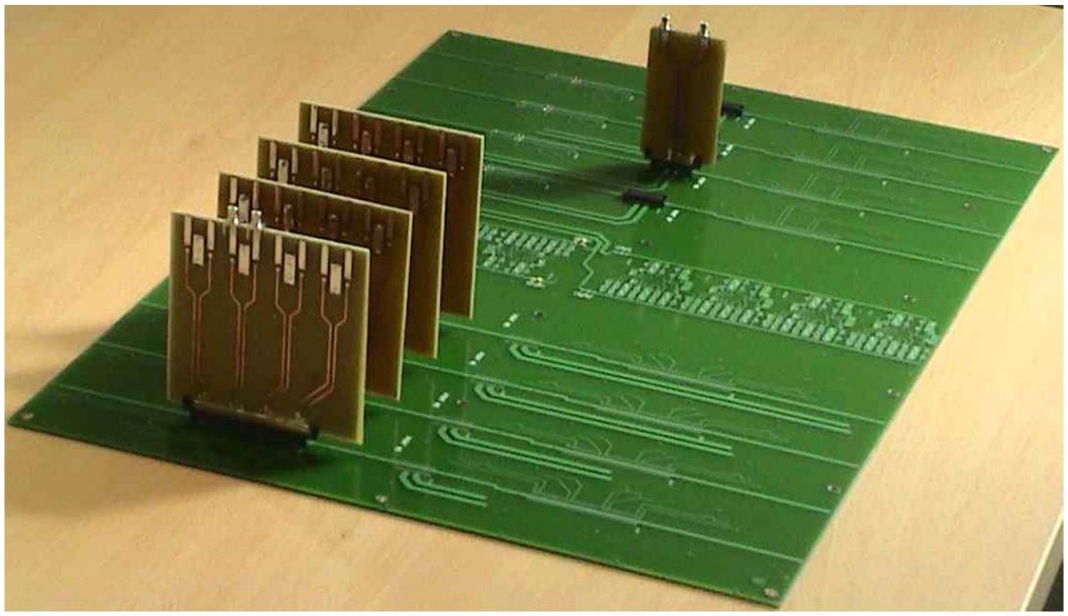


Figura 3.18: Prototipo del *backplane* serie *multidrop* 8×8

Figura 3.19: Tensión en el receptor más próximo

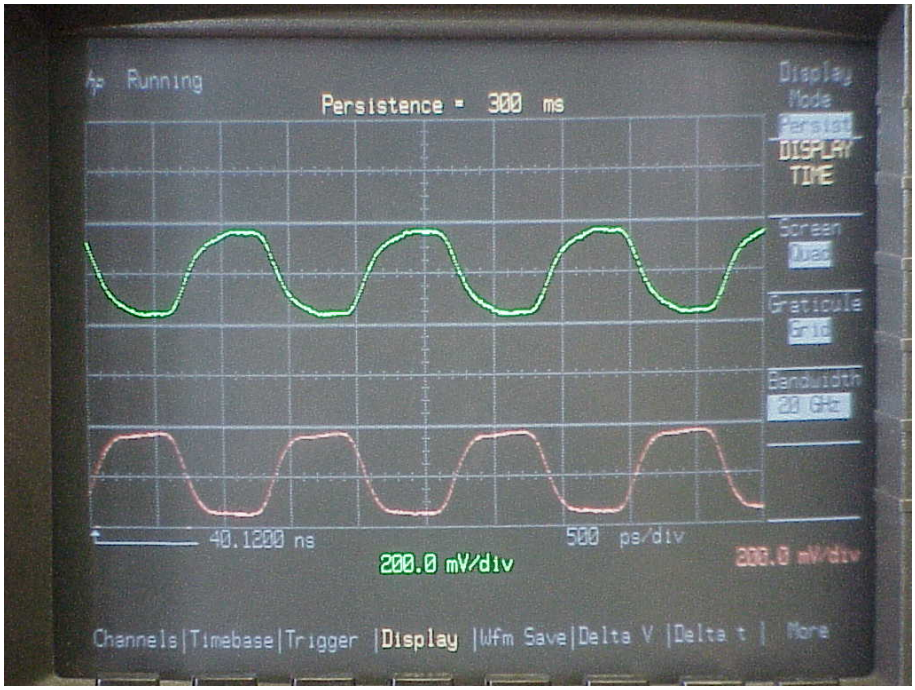


Figura 3.20: Tensión en el receptor más alejado

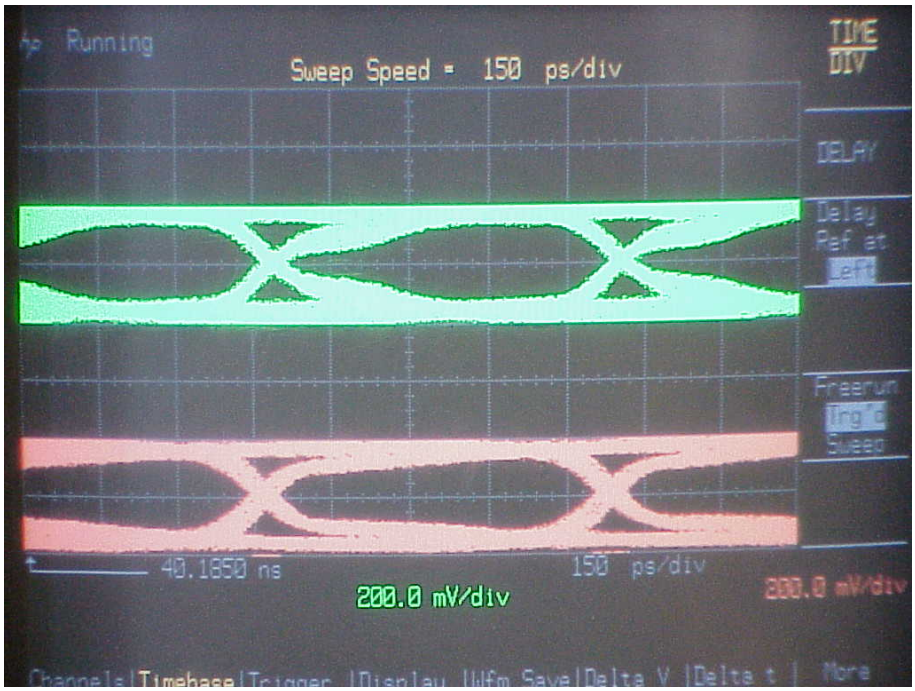


Figura 3.21: Diagrama de ojo en el receptor más próximo

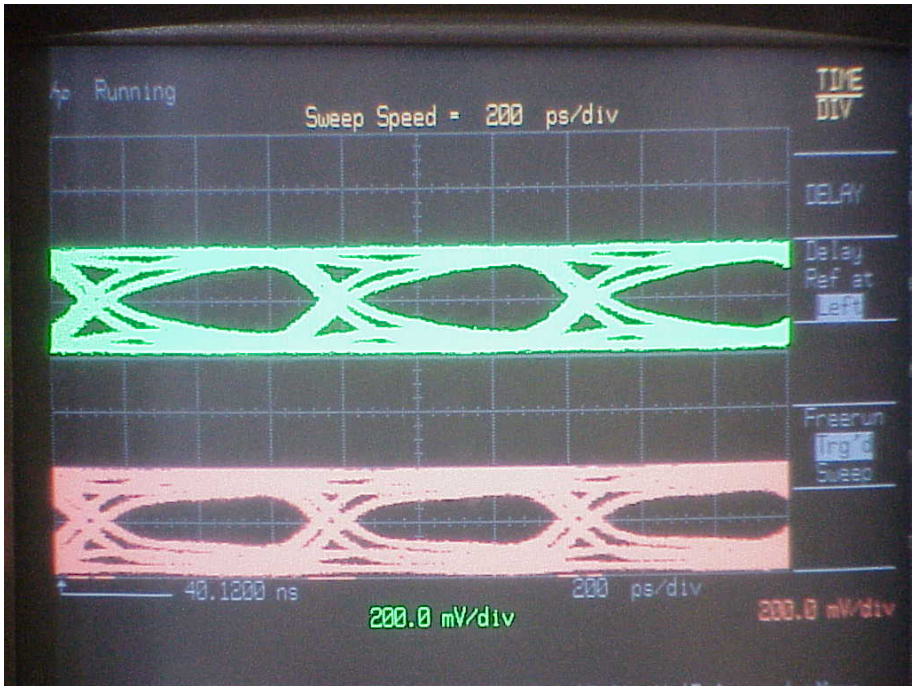


Figura 3.22: Diagrama de ojo en el receptor más alejado

3.4.4 Escalabilidad del conmutador *GMDS*

La escalabilidad del conmutador *GMDS* se debe afrontar desde la perspectiva del aumento de la tasa binaria o desde el incremento del número de puertos.

El incremento de la tasa binaria afecta a la tasa interna de transmisión del *backplane* serie *multidrop* y a la frecuencia de operación de las memorias. En las condiciones actuales y con el diseño del *backplane* propuesto previamente, se han realizado medidas con tasas de hasta 3 Gbps, obteniéndose una transmisión satisfactoria. No se han podido transmitir tasas mayores debido a las limitaciones de la instrumentación disponible. Con una tasa de 2 Gbps, la frecuencia de operación de la memoria se incrementa al doble, 380 MHz en el caso de disponer de una única memoria, y 228 MHz en el caso de la estructura elegida con dos memorias en paralelo. Por lo tanto, el dispositivo *ZBT* 75K62234 cumpliría los requisitos de capacidad y velocidad necesarios. Una mayor velocidad de transferencia requiere una búsqueda más exhaustiva de dispositivos con un coste económico más alto o un diseño específico

que integre la memoria con la tarjeta de línea. En el caso del *backplane*, la solución para tasas superiores de transferencia pasa por un nuevo diseño del *backplane* siguiendo la misma filosofía con un sustrato de alta velocidad.

Asimismo, si se evalúa la escalabilidad del conmutador *GMDS* desde el punto de vista del incremento del número de puertos, la solución consiste en organizar las tarjetas de línea en grupos de cuatro fuentes, donde cada grupo incluye un amplificador *MAX9400* de la empresa *Maxim*. En el caso de que aparezca un problema de variación del retardo debido a la distribución o las longitudes de las líneas de transmisión, los módulos *Egress* se sincronizan independientemente sin perder información. Por otro lado, el problema de la velocidad del acceso a la memoria de almacenamiento aparece nuevamente, siendo la solución propuesta mantener la arquitectura actual, es decir, separar la escritura de las fuentes en diferentes memorias. Sin embargo, los mecanismos de control de flujo global y por clase necesitan información sobre todas las fuentes y, por lo tanto, no son válidos, ya que cada módulo *Egress* dispone de la información de un grupo de cuatro fuentes. A fin de resolver este problema, los diferentes submódulos *Estado* transmiten en cadena la información de sus contadores parciales de tramas *CSIX* almacenadas. El último submódulo de la cadena dispone de los valores totales extremo-a-extremo. En consecuencia, el módulo *Ingress* tan solo se comunica con este último submódulo para recibir los controles de flujo globales. La Figura 3.23 muestra una tarjeta de línea con 8 fuentes de tráfico. Su módulo *Egress* se compone de dos bloques que reciben los paquetes de cuatro fuentes cada uno. Además, se muestra el conexionado en cadena entre los submódulos *Estado* y el módulo *Ingress*.

Si se plantea una configuración típica de 32×32 puertos de 1 Gbps, los recursos *hardware* por tarjeta de línea ascienden a un módulo *Ingress* y 8 módulos *Egress* conectados en cadena. El resultado serían 16 memorias *RAM* de 8 MB cada una y una tarjeta de línea de, aproximadamente, seis millones y medio de puertas equivalentes. En el caso del diseño del *backplane* serie

multidrop, se incluirían 16 dispositivos *MAX9400*. El coste total de esta configuración es completamente asumible por la tecnología actual.

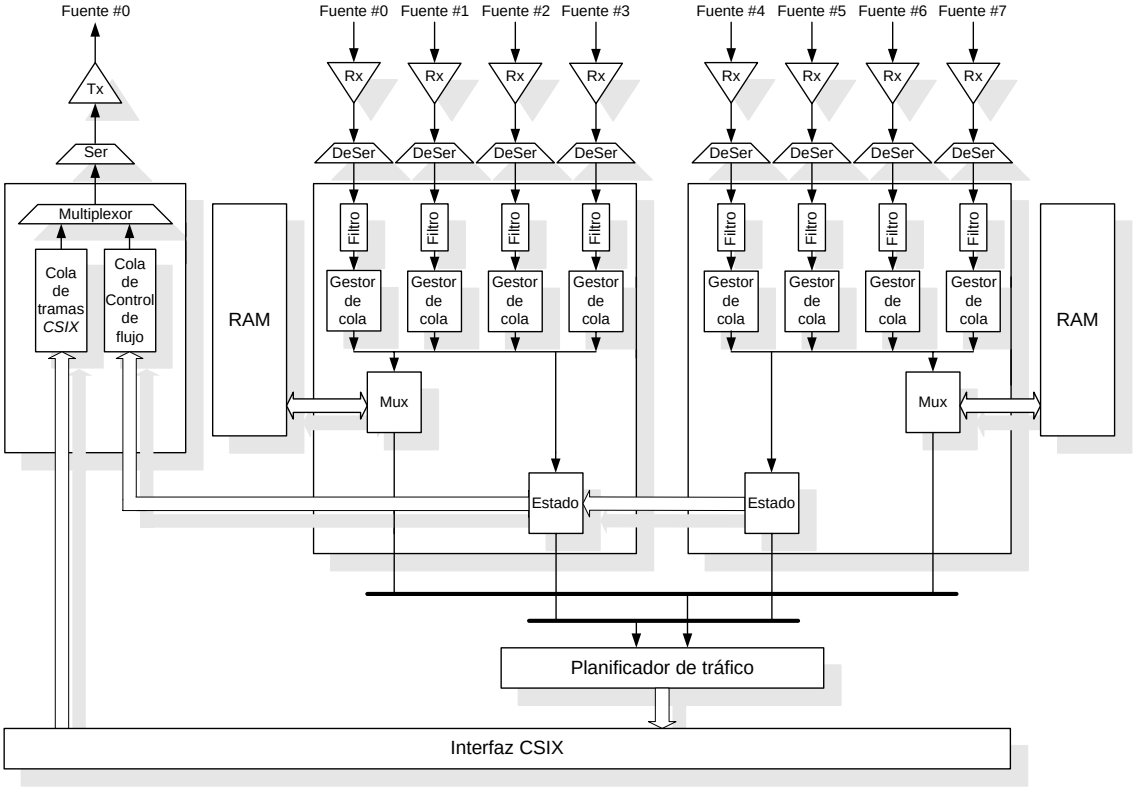


Figura 3.23: Tarjeta de línea 1-a-8

3.5 Resumen

En este capítulo se ha revisado el modelo y la implementación de un conmutador de alta velocidad basado en colas reales de salida. Este conmutador se ha denominado *GMDS* (*Gigabit MultiDrop Switch*) y se basa en la sustitución de la matriz de conmutación por un *backplane* serie *multidrop*, siendo ésta la característica distintiva en torno a la cual gira el desarrollo del conmutador. Otras características, como la estructura con múltiples colas reales de salida para soportar diferentes calidades de servicio, la capacidad de conmutar directamente tráfico de longitud variable, o el aumento de la fiabilidad del sistema al eliminar el elemento de conmutación incrementan el valor de la propuesta

hecha a lo largo de este capítulo. Otro aspecto destacable es la capacidad inherente del *backplane* para la transferencia de paquetes *multicast*/*broadcast* al tratarse de una transmisión *multidrop*.

El conmutador *GMDS* se compone de las tarjetas de línea y el *backplane* serie *multidrop*. Los módulos principales de las tarjetas de línea se denominan *Ingress* y *Egress*. El módulo *Ingress* verifica las tramas *CSIX* entrantes, las multiplexa con la información de control de flujo y transmite toda la información encapsulada en una *multitrama*. El receptor, en el módulo *Egress*, filtra los paquetes entrantes y los almacena a la espera de las instrucciones del planificador de tráfico. Dicha arquitectura ha sido concebida de forma modular para mejorar el acceso a memoria y su escalabilidad, y ha sido dotada de mecanismos de control de flujo extremo-a-extremo excepcionalmente precisos.

Al mismo tiempo, esta arquitectura es idónea para el desarrollo de un planificador con soporte de diferentes calidades de servicio. Los paquetes se clasifican en las múltiples colas a la salida en función de su fuente y clase de tráfico. Dicha diferenciación permite introducir el concepto de prioridad y ancho de banda asignado a una clase de tráfico siendo posible, por lo tanto, procesar independientemente cada una de las clases de tráfico. Este aspecto de la planificación del tráfico se estudia en profundidad en el siguiente capítulo.

Capítulo 4

Planificación de tráfico

4.1 Introducción

En una red como Internet, donde los flujos de información de diversas aplicaciones se multiplexan y compiten por los mismos recursos, surgen políticas de servicio con el fin de priorizar el acceso de los paquetes a los elementos compartidos en la red. En el caso concreto de los conmutadores, el algoritmo de planificación se encarga de gestionar el acceso de los paquetes a sus correspondientes puertos de salida, lo que representa que determina el retardo y el ancho de banda de los flujos de datos. Además, las diversas aplicaciones requieren diferentes calidades de servicio, es decir, al seleccionar el paquete a transmitir se deben tener en cuenta aspectos como la probabilidad de pérdidas de paquetes, la latencia, o la distribución del ancho de banda.

Desde un punto de vista cualitativo, el resultado de priorizar los paquetes se traduce en un principio básico de funcionamiento. Si se reduce el retardo o se incrementa la tasa de transferencia de un flujo de datos, otro flujo incrementa su retardo o reduce su tasa de transferencia. Esta ley de conservación fue enunciada por Kleinrock en 1975 [Kle75] con el objeto de relacionar el retardo y la tasa de transferencia de los flujos de datos en un algoritmo de planificación conservador. Esta ley se utiliza para estudiar las prestaciones del algoritmo de planificación propuesto en el presente capítulo sobre la base del conmutador *GMDS*.

4.2 Modelos de tráfico

Antes de comenzar con el estudio de las prestaciones de la arquitectura de conmutación y del algoritmo de planificación propuesto, se deben introducir las distribuciones de probabilidad utilizadas para este fin. Las distribuciones elegidas son un proceso de llegadas *Bernoulli* y un proceso de *Bernoulli* interrumpido. El modelo *Bernoulli* es un modelo de tráfico común para evaluar el rendimiento de un elemento de conmutación [AOS+93][McK95], mientras que el proceso de *Bernoulli* interrumpido se utiliza para aproximar la naturaleza del tráfico de longitud variable que genera ráfagas del tráfico [Li92][NHL99] [YC01]. En ambos casos, representan distribuciones usadas como referencia en la bibliografía para la caracterización de elementos de conmutación. En las siguientes secciones se describe brevemente las características de cada una.

4.2.1 Tráfico uniforme

La generación de tráfico uniforme está basada en un proceso de llegadas *Bernoulli*. Así, para la caracterización de las prestaciones de los algoritmos de conmutación propuestos, ante tráfico uniforme, en los ciclos de reloj de decisión se genera un número aleatorio para determinar si se produce un periodo de inactividad o se genera un paquete de datos. En el caso de seleccionarse un paquete de datos, puede tener uno o varios destinos, es decir, se genera tanto tráfico *unicast* como *multicast*. En función de las decisiones anteriores se eligen los destinos uniformemente. Además, se selecciona aleatoriamente una clase de tráfico entre todas las disponibles. En el siguiente paso, se genera la longitud de la carga útil del paquete con un valor comprendido entre 1 y 248 bytes debido a la restricción impuesta por la memoria de almacenamiento. Por lo tanto, el paquete de tamaño mínimo son 12 bytes (8 bytes de *overhead*, 1 byte de datos y 3 bytes de relleno) y el tamaño máximo son 256 bytes. Finalmente, el paquete se envía al módulo *Ingress* del conmutador *GMDS*. Cuando finaliza la transmisión del paquete o el periodo de inactividad, se produce un nuevo ciclo de reloj de decisión y se repite el proceso.

4.2.2 Tráfico a ráfagas

La naturaleza del tráfico a ráfagas se modela siguiendo un proceso de *Bernoulli* interrumpido (*IBP - Interrupted Bernoulli Process*). Dicho proceso requiere los siguientes parámetros:

$$\alpha = Probabilidad[Llegada(t)_{|Estado\ ON}]$$

$$p = Probabilidad[Estado\ ON\ (t+1)_{|Estado\ ON(t)}]$$

$$q = Probabilidad[Estado\ OFF\ (t+1)_{|Estado\ OFF(t)}]$$

La longitud media de un estado ON es $\frac{1}{(1-p)}$, y la longitud media de un estado OFF es $\frac{1}{(1-q)}$. En el estado OFF no se generan paquetes y, al menos, su longitud es de valor 1. La tasa media de llegadas de paquetes, o tráfico ofrecido, ρ, es:

$$\rho = \frac{\alpha(1-q)}{2-p-q}$$

El coeficiente de variación (*CoV - Coefficient of Variation*) es:

$$CoV = 1 + \alpha\left(\frac{(1-p)(p+q)}{(2-p-q)^2} - 1\right)$$

En la presente Tesis Doctoral, α siempre toma el valor 1, es decir, en el estado *ON* siempre se generan paquetes y en el estado *OFF* nunca, con el fin de generar ráfagas de tráfico continuas. La Figura 4.1 muestra esta condición modelada como una cadena de *Markov* de 2 estados. Los parámetros p y q se calculan para obtener la carga de tráfico ofrecida sabiendo que el coeficiente de variación se fija al valor 2. El resultado es una duración media de los estados ON y OFF de 5 paquetes si la carga de tráfico se establece en el 50%. A medida que la carga de tráfico aumenta, la duración del estado ON es mayor y, de forma inversa, la duración del estado OFF se reduce hasta el punto de que con una carga de tráfico del 90% son 105 y 11,67 paquetes, respectivamente.

Todos los paquetes que pertenecen a la misma ráfaga comparten el mismo puerto de destino, seleccionado uniformemente al comienzo de la ráfaga.

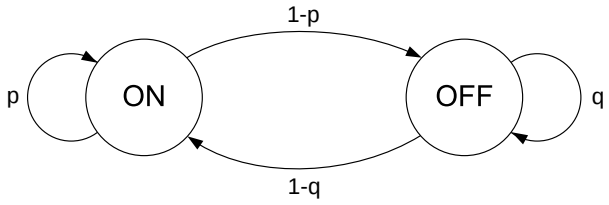


Figura 4.1: Cadena de *Markov* de 2 estados

El resto del proceso de generación de paquetes y los parámetros correspondientes son similares al caso expuesto anteriormente para el tráfico uniforme.

4.3 Caracterización del conmutador *GMDS*

En esta sección se presentan los resultados de la simulación del conmutador *GMDS*. En primer lugar hay que remarcar que las presentes simulaciones tienen dos objetivos principales: verificar la funcionalidad de la arquitectura y obtener unos resultados bajo los patrones teóricos más habituales de caracterización, *Bernoulli* y *Bernoulli* interrumpido. El algoritmo de planificación utilizado en esta sección sólo tiene un propósito de evaluación, por lo que se ha elegido un algoritmo *Deficit Round Robin* que recorre en orden todas las clases de tráfico de una fuente antes de pasar a la siguiente fuente de tráfico.

El conmutador *GMDS* se ha modelado tanto a alto nivel, en lenguaje C con la ayuda de la librería *CSIM* [CSIM], como a bajo nivel, en el lenguaje *Verilog HDL* a nivel *RTL*. En ambos casos de simulación se ha elegido como criterio de comparación el retardo medio de encolado, mostrado en paquetes, compuesto del retardo de almacenamiento en el conmutador y el tiempo de transmisión del paquete. En primer lugar, se muestran los resultados obtenidos a partir de la caracterización del modelo de alto nivel.

4.3.1 Caracterización del modelo de alto nivel del conmutador *GMDS*

A continuación se presentan los resultados obtenidos de la caracterización del modelo de alto nivel del conmutador *GMDS* con el algoritmo de planificación *DRR* sin soporte de calidad de servicio.

4.3.1.1 Tráfico uniforme

En el caso de la generación de tráfico uniforme se necesita fijar únicamente el rango de carga del tráfico, 90-99%, y la longitud de la trama, en este caso, variable entre 1 y 248 bytes. Todas las clases de tráfico son tratadas de la misma manera, es decir, no existen prioridades y se les asigna el mismo ancho de banda a cada una. El criterio de finalización de la simulación consiste en alcanzar una precisión del 2% con un intervalo de confianza del 95%, es decir, se deben recibir, aproximadamente, al menos 10000 paquetes por destino, clase y fuente de tráfico para detener una simulación con garantías de obtener unos resultados significativos.

La Figura 4.2 muestra el retardo medio de todas las clases de tráfico del conmutador *GMDS*, medido en paquetes. El resultado esperado depende principalmente del algoritmo de planificación elegido. Por lo tanto, en el caso de un algoritmo *Deficit Round Robin* modificado se espera que todas las clases reciban el mismo tratamiento. Dicha figura muestra el comportamiento igualitario de todas las clases de tráfico en términos de retardo medio de encolado.

La Figura 4.3 muestra el reparto del ancho de banda entre las diferentes clases de tráfico. Como el algoritmo de planificación no asigna anchos de banda diferentes a las clases de tráfico, la figura muestra una única línea constante que corresponde a la superposición del ancho de banda asignado a todas las clases y destaca por su independencia respecto a la carga de tráfico.

Los resultados de esta simulación se deben comparar con el rendimiento de otras arquitecturas. Por un lado, se ha seleccionado un conmutador con colas a la salida (*OQ*) de tamaño infinito, política de planificación de tráfico *FIFO*,

y paquetes *ATM*, ya que esta arquitectura representa el límite inferior e ideal en términos de retardo hacia el que se tiende en cualquier propuesta dentro del campo de la conmutación y la planificación de paquetes. Por otro lado, se ha utilizado un conmutador con colas virtuales de salida (*VOQ*), política de planificación *iSLIP* (4 iteraciones) y, al igual que en el caso anterior, paquetes *ATM*. Las prestaciones de ambas arquitecturas se pueden consultar en [MA98]. Por último, y dadas sus buenas prestaciones, se ha añadido una arquitectura de conmutación *CICQ* [Yos04] con colas de tamaño infinito en los puertos de entrada y de tamaño unitario en los puntos de interconexión, política de planificación *Round Robin*, tanto en los puertos de entrada como en los nodos de interconexión, y paquetes de longitud fija.

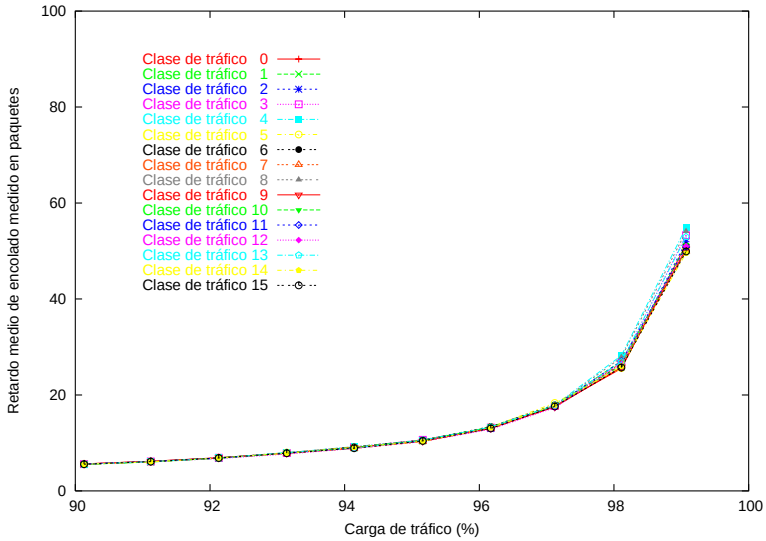


Figura 4.2: Retardo medio de encolado del modelo de alto nivel con tráfico uniforme en el conmutador *GMDS*

La Figura 4.4 muestra el retardo medio del conmutador *GMDS* y de las tres arquitecturas de referencia propuestas. Como era de esperar, las prestaciones de esta arquitectura no pueden alcanzar el rendimiento de un conmutador ideal con colas a la salida al introducir una estructura de almacenamiento compleja en los puertos de salida. Sin embargo, al tratarse de una

arquitectura basada en una modificación de la arquitectura de colas a la salida, sus prestaciones son mucho mejores que una arquitectura con colas a la entrada y ligeramente mejores que una arquitectura *CICQ*. Actualmente, esta última arquitectura, evaluada en [Yos04], presenta un excelente compromiso entre complejidad y rendimiento. La propuesta arquitectural realizada en esta Tesis Doctoral mejora ligeramente sus prestaciones en términos de retardo, sin olvidar que su arquitectura es mucho más sencilla, facilita las tareas de planificación con calidad de servicio y soporta la transferencia de tráfico de longitud variable sin requerir mecanismos de segmentación y reensamblado.

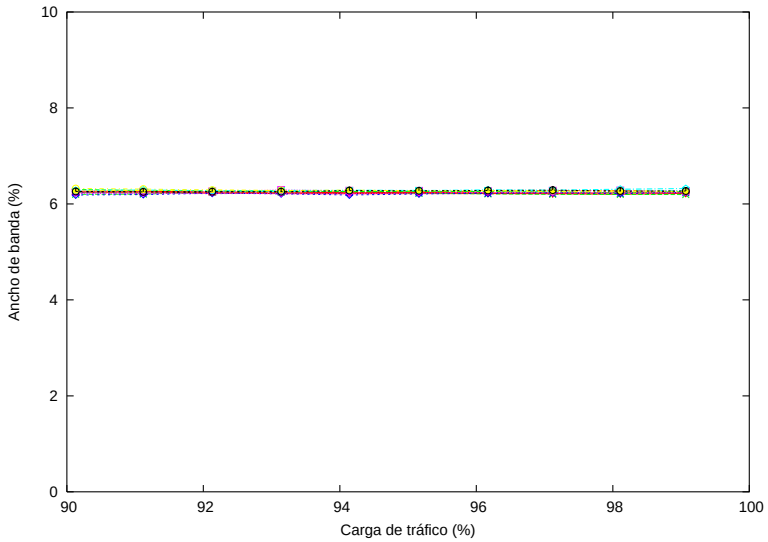


Figura 4.3: Distribución del ancho de banda del modelo de alto nivel con tráfico uniforme en el conmutador *GMDS*

4.3.1.2 Tráfico a ráfagas

Al igual que en el escenario del tráfico uniforme, es necesario establecer el rango de la carga de tráfico, que en este caso varía entre el 70% y el 90%, y la longitud de la trama, que se va a seguir manteniendo variable. No se introduce ningún tipo de diferenciación entre las clases de tráfico, por lo que todas

son tratadas de la misma manera. Siguiendo el mismo criterio de finalización de la simulación, se establece una precisión del 2% con un intervalo de confianza del 95%, es decir, se deben recibir aproximadamente 200000 paquetes por destino, clase y fuente.

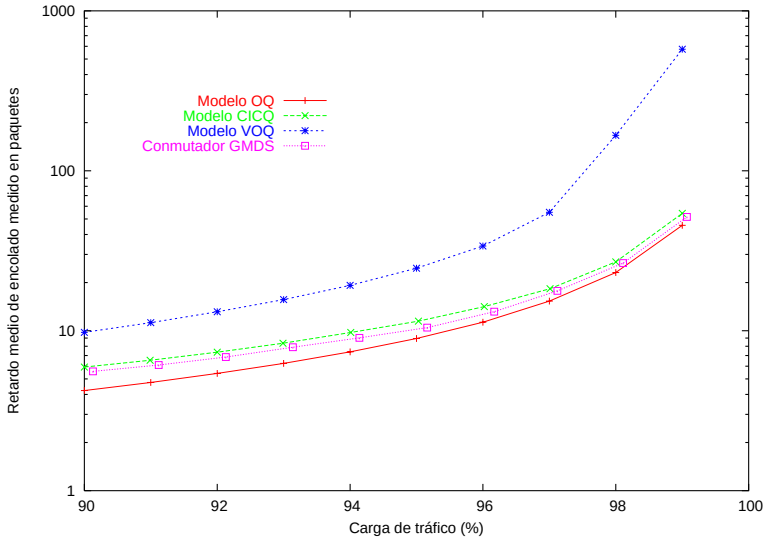


Figura 4.4: Retardo medio de encolado del modelo de alto nivel con tráfico uniforme en los conmutadores *OQ*, *CICQ*, *VOQ* y *GMDS*

Las Figuras 4.5 y 4.6 muestran que el comportamiento frente al tráfico a ráfagas es similar al tráfico uniforme, salvando las magnitudes de los retardos y el rango de las cargas de tráfico.

Sin embargo, la Figura 4.5 presenta una limitación. Se ha considerado una arquitectura con múltiples colas en los puertos de salida de tamaño infinito. Si dichas colas se limitan a 128 paquetes cada una, se obtiene un resultado más cercano a una implementación real. La Figura 4.7 muestra la simulación de este caso y permite apreciar que el retardo se mantiene constante a partir del 80% de la carga de tráfico. En realidad, a partir de este punto las colas localizadas en los puertos de salida están llenas y el retardo mostrado corresponde al tiempo necesario para atender el último paquete de la correspondiente cola.

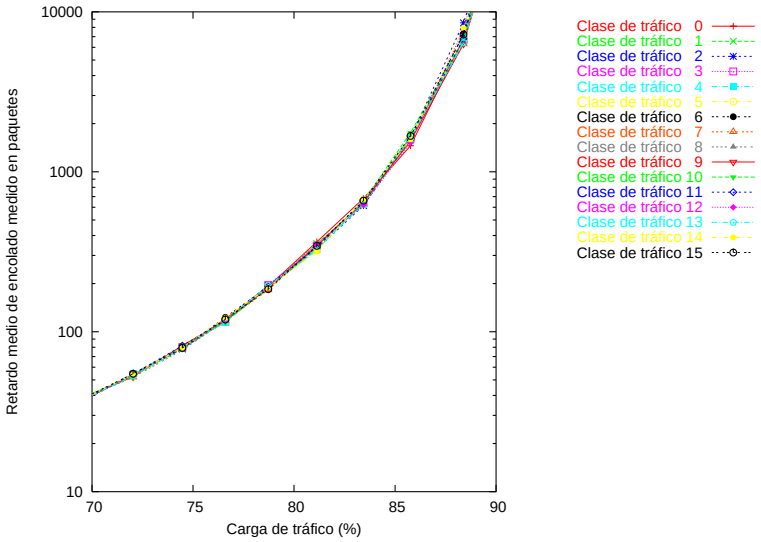


Figura 4.5: Retardo medio de encolado del modelo de alto nivel con tráfico a ráfagas en el conmutador *GMDS*

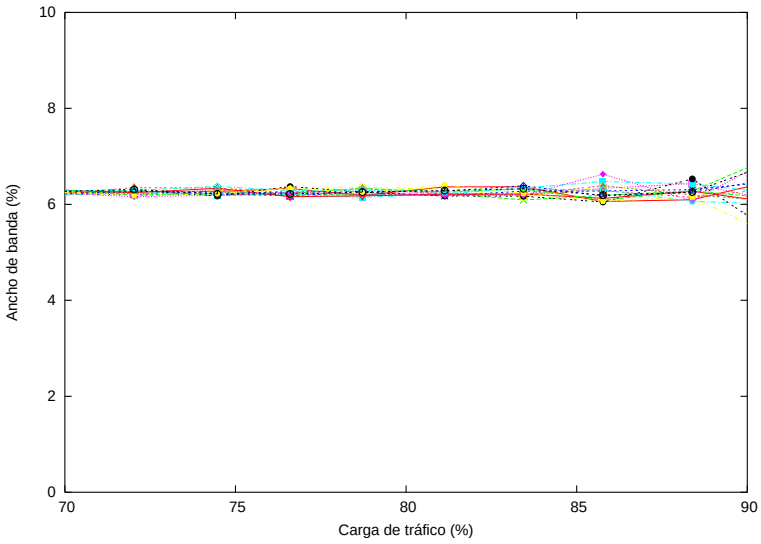


Figura 4.6: Distribución del ancho de banda del modelo de alto nivel con tráfico a ráfagas en el conmutador *GMDS*

Este valor de carga de tráfico, 80%, correspondería al *throughput* soportado por el conmutador *GMDS* con la capacidad de memoria de 128 paquetes según el modelo de alto nivel. Si se incrementa la capacidad de almacenamiento de las memorias, este límite se incrementa por lo que este valor es representativo únicamente para el demostrador propuesto en esta Tesis Doctoral.

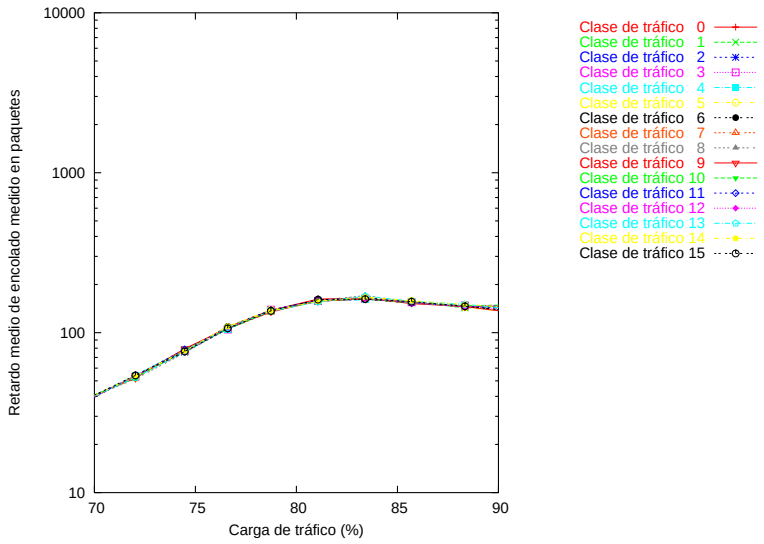


Figura 4.7: Retardo medio de encolado del modelo de alto nivel con tráfico a ráfagas en el conmutador *GMDS* y colas finitas

4.3.2 Caracterización del modelo de bajo nivel del conmutador *GMDS*

En la sección anterior se presentaron los resultados del modelo de alto nivel del conmutador. A continuación, se presentan los resultados obtenidos a partir de las simulaciones del modelo de bajo nivel implementado en *Verilog HDL* a nivel *RTL*. Por lo tanto, en dichas simulaciones se incluyen todos los retardos de almacenamiento y lectura de los paquetes, proceso que se produce en el módulo *Ingress*, en el submódulo *Filtro* y en la memoria externa del conmutador *GMDS*. Consecuentemente, se espera un comportamiento similar al

mostrado en las simulaciones del modelo de alto nivel, pero con un incremento generalizado del retardo en todas las clases de tráfico.

4.3.2.1 Tráfico uniforme

Las condiciones de simulación a nivel *RTL* son similares a las presentadas en el modelo de alto nivel, con la salvedad de que el generador de tráfico no puede ajustar de forma tan precisa la carga de tráfico. Al no producirse ningún tipo de diferenciación entre las clases de tráfico, la Figura 4.8 presenta por simplicidad el retardo medio de todas las clases de tráfico en lugar de mostrar el retardo individual de cada una. Cabe destacar que ambos modelos muestran la misma tendencia. Sin embargo, el modelo de bajo nivel muestra un ligero incremento del retardo debido a los procesos de almacenamiento y lectura de los paquetes en las memorias. El ancho de banda se reparte equitativamente, como se aprecia en la Figura 4.9.

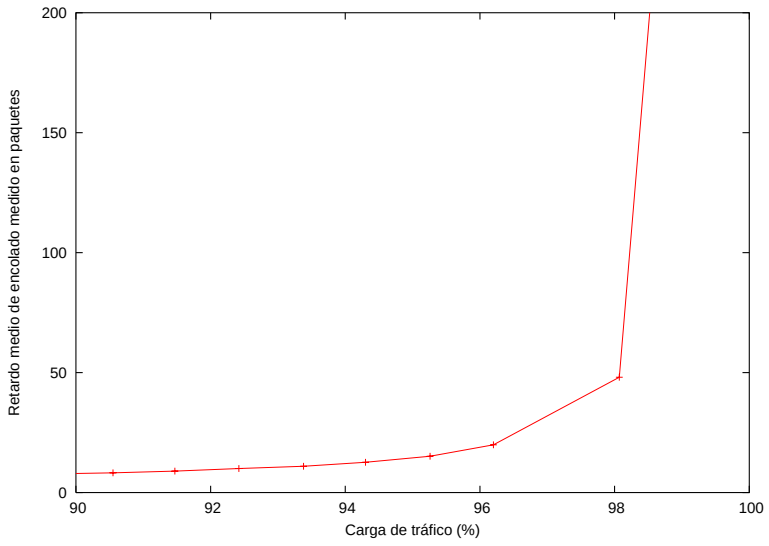


Figura 4.8: Retardo medio de encolado del modelo de bajo nivel con tráfico uniforme en el conmutador *GMDS*

El retardo del conmutador *GMDS* presentado en la Figura 4.8 se puede analizar en profundidad. Concretamente, y dada la arquitectura de un con-

mutador con colas a la salida, se desglosa en dos componentes: un retardo fijo y un retardo variable. El retardo fijo se compone del tiempo que un paquete tarda en atravesar el módulo *Ingress*, el *backplane* serie *multidrop*, los submódulos *Filtro*, *Gestor de Colas* y *Multiplexor* del módulo *Egress*, y el almacenamiento en memoria. Este valor es ligeramente superior a 3, ya que el paquete se almacena tres veces además del retardo correspondiente a la transmisión y al procesado en los restantes módulos. Una vez almacenado el paquete, el planificador se responsabiliza de tomar la decisión de extracción de un paquete del conmutador. Este retardo es variable y depende, fundamentalmente, de la política de planificación elegida.

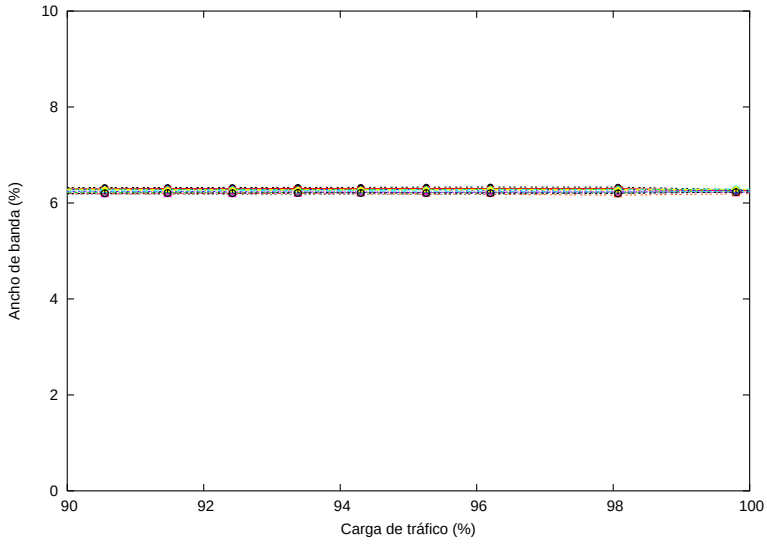


Figura 4.9: Distribución del ancho de banda del modelo de bajo nivel con tráfico uniforme en el conmutador *GMDS*

4.3.2.2 Tráfico a ráfagas

En la Figura 4.10 se observa el retardo medio de las clases con tráfico a ráfagas. Este retardo está directamente influenciado por la capacidad limitada de las memorias de almacenamiento diferenciándose dos tramos con un punto de inflexión en torno al 50% de la carga de tráfico. El primer tramo

muestra el incremento del retardo debido al aumento de la carga de tráfico. A partir de este punto, las colas de almacenamiento se llenan y se muestra el retardo constante correspondiente al tiempo necesario para extraer el último paquete de cada cola con un algoritmo *DRR* modificado.

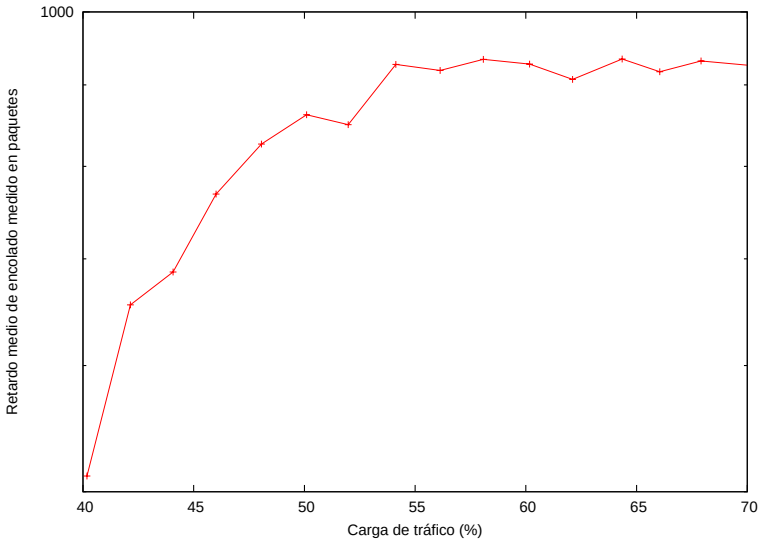


Figura 4.10: Retardo medio de encolado del modelo de bajo nivel con tráfico a ráfagas en el conmutador *GMDS*

Si se analizan las pérdidas de paquetes del conmutador *GMDS*, en la Figura 4.11 se aprecia el colapso de las colas en torno al 50% de la carga de tráfico. Para poder realizar esta simulación se ha desactivado el control de flujo, ya que, en caso contrario, limitaría la carga de tráfico al reducir la tasa de transferencia de la fuente.

Independientemente de la distribución y de la carga de tráfico, el mecanismo basado en créditos para paquetes de longitud variable implementado en el algoritmo *DRR* mantiene el reparto equitativo del ancho de banda, como se muestra en la Figura 4.12.

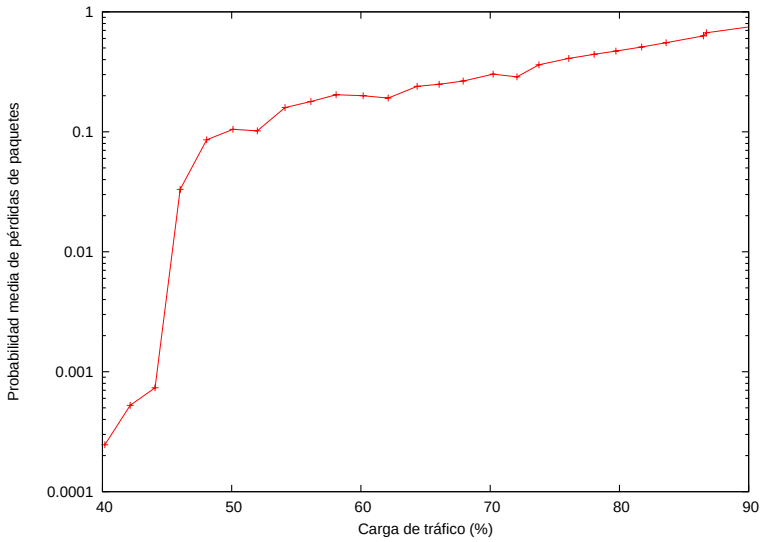


Figura 4.11: Pérdidas de paquetes del modelo de bajo nivel con tráfico a ráfagas en el conmutador *GMDS*

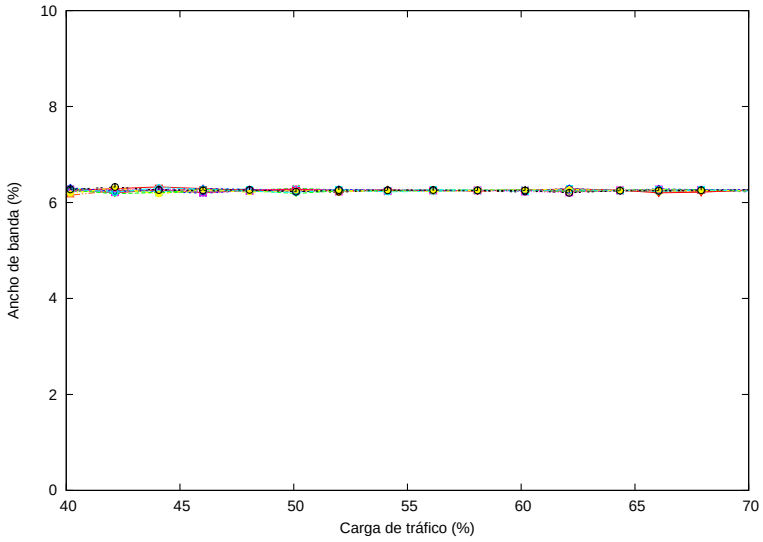


Figura 4.12: Distribución del ancho de banda del modelo de bajo nivel con tráfico a ráfagas en el conmutador *GMDS*

4.4 Descripción del algoritmo de planificación

En la revisión del estado del arte se encuentran numerosas referencias de algoritmos de planificación. Cada uno de ellos reúne diversas características que se pueden clasificar en generales y/o deseables en cualquier algoritmo de planificación, y específicas de la planificación del tráfico con calidad de servicio. En la presente Tesis Doctoral se ha desarrollado un algoritmo de planificación que, además de aunar todas las propiedades anteriores, incluye características propias necesarias para conseguir los hitos propuestos inicialmente. En esta sección se presenta la descripción de dicho algoritmo.

4.4.1 Características generales de la planificación

Un algoritmo de planificación debe reunir ciertas características para prestar un servicio satisfactorio. Zhang enunció en [Zha95] las características más representativas que deben guiar la concepción de este tipo de diseños.

- Eficiencia. Para asegurar los requisitos de un sistema de conmutación se necesita disponer de un control de admisión de conexiones (*Call Admission Control - CAC*) para limitar la carga de tráfico con servicio garantizado. Cuanto más eficiente sea el algoritmo, más conexiones se pueden admitir, manteniendo la calidad de servicio de las conexiones previas, y mayor utilización se puede hacer de los recursos de la red.

- Protección. Un algoritmo de planificación debe proteger las conexiones de tráfico con buen comportamiento de tres fuentes principales de variabilidad: conexiones con mal comportamiento, fluctuaciones de carga de la red, y tráfico sin calidad de servicio o sin ningún tipo de restricción.

- Flexibilidad. Debido a la gran diversidad de aplicaciones existentes, así como a la incertidumbre sobre los futuros servicios, cabe esperar que un algoritmo de planificación sea flexible para asignar diferentes retardos, anchos de banda y probabilidades de pérdidas a diferentes conexiones con calidad de servicio.

- Simplicidad. Un algoritmo de planificación debería ser conceptualmente y arquitecturalmente simple para permitir su análisis e implementación. Este punto cobra especial relevancia si se enfoca el algoritmo a sistemas de conmutación de alta velocidad o con un gran número de puertos, ya que la simplicidad del algoritmo mejora su escalabilidad.

4.4.2 Características específicas del soporte de *QoS*

Junto a las características generales expuestas previamente, y con el objetivo de alcanzar los requisitos de calidad de servicio exigidos por los usuarios, se propone un planificador de paquetes para el conmutador *GMDS* siguiendo la filosofía de Servicios Diferenciados [BBC+98]. Sin embargo, la clasificación de los flujos de datos en clases de tráfico no es suficiente por sí misma para alcanzar el objetivo de soporte de calidad de servicio propuesto en esta Tesis Doctoral. Por lo tanto, se añaden parámetros adicionales para incrementar la eficiencia y la flexibilidad de la planificación, necesarios para poder proponer un algoritmo de planificación con soporte de calidad de servicio capaz de explotar las características del conmutador *GMDS*.

- Clases de tráfico. Cada uno de los paquetes que componen un flujo de datos se asigna a una clase de tráfico. Todos los paquetes pertenecientes a la misma clase reciben el mismo servicio, según el modelo de Servicios Diferenciados.

- Peso. El ancho de banda se reparte entre las clases de tráfico de acuerdo al peso. Este parámetro es la cuantificación de la proporción que tiene asignada una clase de tráfico respecto al ancho de banda total. Cuando se utilizan paquetes de longitud fija, hay una relación directa entre el ancho de banda utilizado y el número de paquetes transmitidos. En el conmutador *GMDS* no existe dicha relación, por lo que la complejidad del algoritmo de planificación aumenta. Propuestas como los algoritmos *WRR* [KSC91] o *HRR* [KKK90] para paquetes de longitud fija o *DRR*

[SV96] para paquetes de longitud variable, pueden garantizar el ancho de banda.

- Prioridades. Es posible asignar una prioridad a cualquier clase de tráfico con el objetivo de reducir o incrementar el retardo medio de los paquetes de dicha clase. Sin embargo, independientemente de la configuración de las prioridades, el planificador de paquetes debe mantener la distribución del ancho de banda de banda en función de los pesos, como en el algoritmo *PQWRR* [MMW01].

- Paquetes de longitud variable. El ancho de banda se comparte teniendo en cuenta el peso de las clases de tráfico. Sin embargo, una aproximación más clásica, como por ejemplo los algoritmos *WRR*, o *DRR*, no puede utilizarse debido a la combinación de prioridades, pesos y paquetes de longitud variable en el mismo planificador.

4.4.3 Características propias del algoritmo propuesto

Sobre la base de las características de los algoritmos de planificación, tanto generales del servicio prestado como las específicas del soporte de calidad de servicio, se plantean tres consideraciones adicionales a seguir con el fin de alcanzar los objetivos propuestos en la presente Tesis Doctoral.

- Servicio conservador. Siempre que existan paquetes almacenados en la memoria del conmutador *GMDS*, el planificador selecciona y transmite uno de ellos a fin de maximizar el *throughput*. Esta política de servicio es independiente del número de clases de tráfico, pesos o prioridades configuradas.

- Paquetes de longitud variable. Propuestas basadas en sellos de tiempo, como el algoritmo *WFQ* [PG93][DKS89], requieren un alto coste *hardware* debido a las operaciones matemáticas relacionadas con el procesado de los paquetes de longitud variable. Como en el presente trabajo

de investigación se ha valorado el enfoque arquitectural y la simplicidad a fin de implementar físicamente la propuesta realizada frente a una mera visión algorítmica, se ha considerado más adecuado utilizar un mecanismo basado en créditos con el objetivo de controlar la relación entre los paquetes de longitud variable y el ancho de banda.

- Marco de tiempo. Se define un marco de tiempo como un intervalo temporal donde el planificador mantiene los pesos y las prioridades de manera justa, de acuerdo con la configuración definida por el procesador de red. Por lo tanto, cada clase de tráfico reserva una parte del marco de tiempo en función de su peso. El orden de salida de los paquetes depende principalmente de la prioridad asignada a cada clase de tráfico. En el caso de que todas las clases de tráfico consuman la parte proporcional de ancho de banda que les corresponde en el marco de tiempo actual, o estén vacías, el marco de tiempo se inicializa, volviéndose a asignar el ancho de banda a todas las clases de tráfico.

4.4.4 Descripción funcional

Abordar el diseño de un planificador de paquetes con calidad de servicio requiere un enorme esfuerzo. El diseño propuesto en la presente Tesis Doctoral se ha llevado a cabo descomponiendo el funcionamiento del planificador en dos mecanismos, uno de control del retardo y otro de control del ancho de banda, basados en prioridades y créditos, respectivamente.

La información proporcionada por el conmutador *GMDS* al planificador se compone de dos bits sobre las clases y las fuentes de tráfico que indican si cada una de ellas está vacía o si hay encolados uno, dos o más paquetes. Sobre la base de esta información se selecciona una clase de tráfico perteneciente a la máxima prioridad disponible mediante un mecanismo *Round Robin* entre las clases de tráfico que pertenezcan a dicha prioridad. En un segundo paso, se selecciona una fuente de tráfico teniendo en cuenta la cantidad de información enviada por los paquetes previos. Este esquema sigue una filosofía jerárquica

de prioridad máxima a mínima, seleccionando primero la clase y después la fuente de tráfico, como se muestra esquemáticamente en la Figura 4.13. Desde el punto de vista arquitectural se selecciona una entre las múltiples colas agrupadas por clase y fuente de tráfico, como se observa en la Figura 4.14.

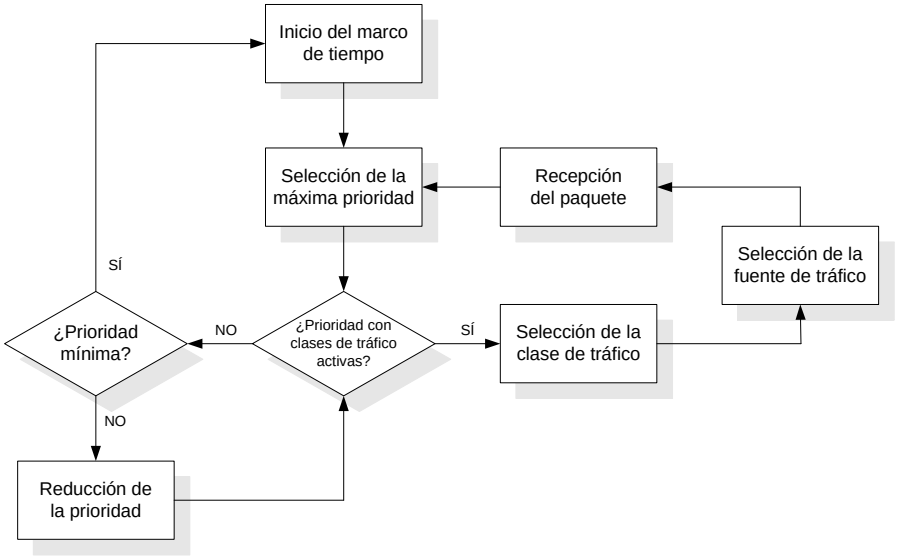


Figura 4.13: Flujo de control del algoritmo de planificación

El mecanismo de control del ancho de banda se basa en la asociación de un crédito a cada clase de tráfico. Cada paquete transmitido consume parte de este crédito en función de su longitud. Cuando se ha agotado todo el crédito de una clase de tráfico, dicha clase no puede ser elegir en el proceso de planificación. En el caso de que ninguna clase de tráfico se pueda seleccionar, se inicializa el marco de tiempo, proceso que calcula los créditos de las clases de tráfico dependiendo del peso asignado externamente a cada clase de tráfico.

4.4.4.1 Selección de la clase de tráfico

En la primera fase del algoritmo propuesto en esta Tesis Doctoral se selecciona la máxima prioridad y se comprueba que existan clases de tráfico activas. Entre todas ellas se aplica una política *Round Robin*. Se considera que una clase está activa si está asociada a la prioridad seleccionada, no está vacía y dispone de crédito. El concepto de crédito se traduce directamente como an-

cho de banda, es decir, una clase de tráfico con crédito implica que esta clase no ha utilizado todo el ancho de banda reservado para ella en el marco de tiempo actual. Si la máxima prioridad no tiene asociada ninguna clase activa, se reduce la prioridad. Una vez elegida una clase de tráfico, esta información se utiliza para comenzar la selección de la fuente de tráfico. En el caso de comprobar todas las prioridades infructuosamente, se activa el mecanismo de inicio del marco de tiempo.

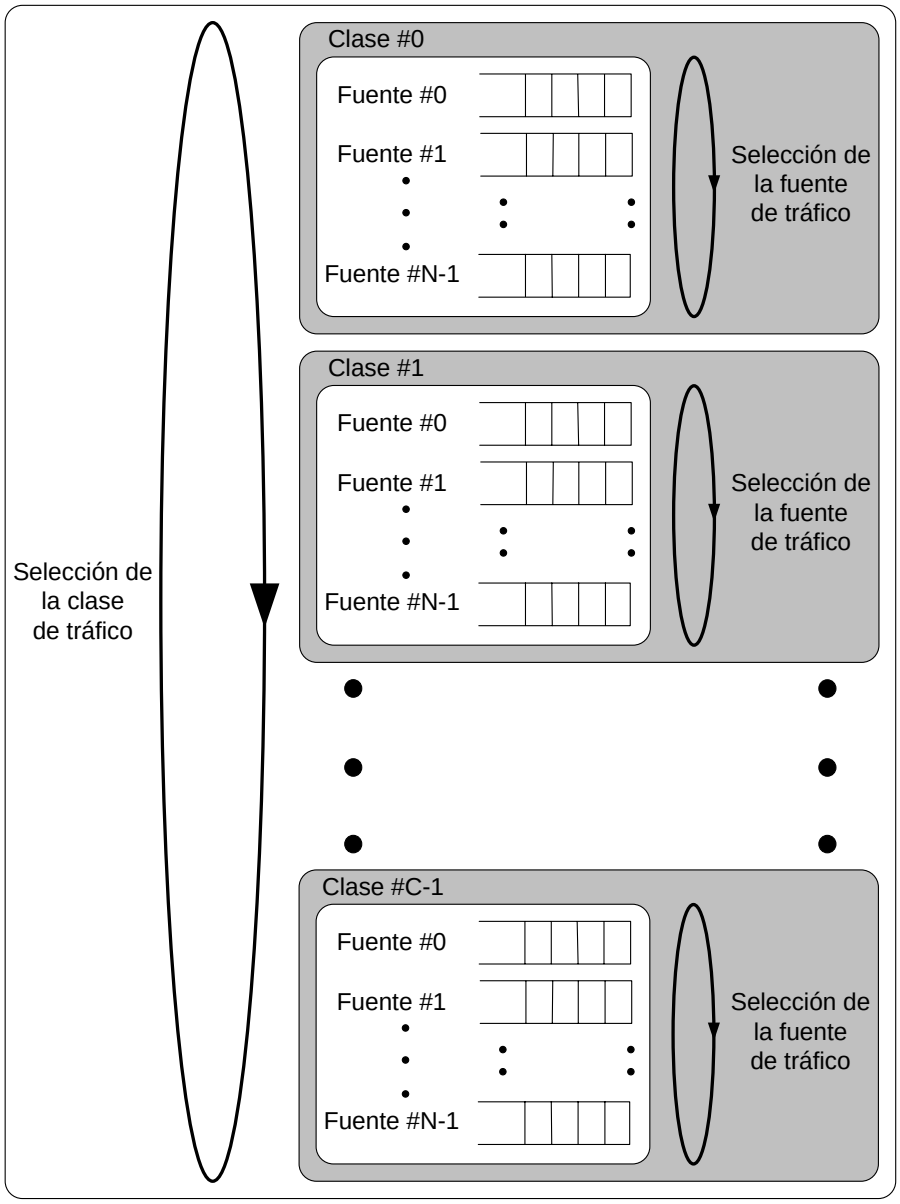


Figura 4.14: Estructura de almacenamiento del algoritmo de planificación

4.4.4.2 Selección de la fuente de tráfico

Una vez seleccionada una determinada clase de tráfico, se elige la fuente de tráfico. Esta elección conlleva la comprobación de las fuentes de tráfico no vacías asociadas a dicha clase y la comparación del tráfico acumulado enviado por cada una de ellas. A partir de esta comparación, la fuente de tráfico que haya enviado la menor cantidad de tráfico podrá enviar el próximo paquete.

4.4.4.3 Recepción del paquete. Control de ancho de banda

A priori, no se conoce la longitud de los paquetes encolados en la memoria. Por lo tanto, una vez extraído un paquete, el planificador conoce su longitud. Posteriormente, actualiza los registros de los créditos de las clases de tráfico y del valor acumulado del tráfico enviado por cada fuente y clase de tráfico.

4.4.4.4 Marco de tiempo

Al comienzo de cada marco de tiempo se calcula el valor de los créditos en función del peso asignado a cada clase de tráfico. Este valor representa la parte proporcional de ancho de banda que le corresponde a cada clase de tráfico en un marco de tiempo. Independientemente de la configuración de los pesos, siempre se garantiza que se pueda transmitir un paquete de longitud máxima del menor peso establecido. La ecuación 4.1 representa el valor del crédito de la clase de tráfico *n* al inicio del marco de tiempo actual:

$$Valor\ del\ crédito\ _{Clase\ de\ tráfico\ n} = L_{MAX}\ \frac{Peso\ _{Clase\ de\ tráfico\ n}}{Peso\ _{MIN}} + C_n, \qquad (4.1)$$

donde $Peso\ _{Clase\ de\ tráfico\ n}$ es el peso de la clase de tráfico actual, $Peso\ _{MIN}$ es el peso mínimo resultante de la comparación de todas las clases de tráfico, L_{MAX} es la longitud máxima de los paquetes, y C_n un factor de corrección procedente del marco de tiempo anterior.

El cálculo del valor de los créditos se realiza cuando no se puede elegir ninguna cola de tráfico para la transferencia de un paquete debido a que las

colas están vacías, o han consumido todo su crédito. En el caso de considerar que las colas del conmutador siempre almacenan algún paquete, la condición de inicio de los marcos de tiempo es únicamente el valor del crédito de las clases de tráfico. En consecuencia, la longitud máxima del marco de tiempo es:

$$Longitud\ _{MT} = \sum_{n=0}^{n=C-1} (L_{MAX} \frac{Peso\ _{Clase\ de\ tráfico\ n}}{Peso\ _{MIN}} + C_n), \qquad (4.2)$$

siendo C el número total de clases de tráfico soportadas por el planificador.

Según la ecuación 4.2, la longitud del marco de tiempo se determina sobre la base de la longitud máxima de los paquetes y de la relación entre los pesos de las clases de tráfico. Por lo tanto, la longitud del marco de tiempo se adapta a la configuración para garantizar el reparto del ancho de banda, motivo por el cual se le denomina *marco de tiempo de longitud variable*.

Con el objetivo de clarificar el comportamiento del planificador y del marco de tiempo, se presenta un ejemplo con cuatro clases de tráfico y dos configuraciones diferentes de ancho de banda. A cada clase de tráfico se le asocia un peso, en este primer caso, del 10%, 20%, 30% y 40% del ancho de banda total, y no se considera el factor de corrección C_n. La longitud máxima de los paquetes se establece en 256 bytes. Por lo tanto, si el peso mínimo es de un 10%, los valores de los créditos asociados a cada clase de tráfico son 256 (10%), 512 (20%), 768 (30%) y 1024 (40%). Cada crédito equivale a un byte enviado por la clase de tráfico correspondiente. La sumatoria de las longitudes de todos los paquetes enviados en un marco de tiempo alcanza 2560 bytes de carga útil. En el segundo caso, con igual ancho de banda (25%) para cada una de las cuatro clases de tráfico, la sumatoria total son 1024 bytes. Con este ejemplo se ilustra la variación de la longitud del marco de tiempo y su dependencia de los pesos bajo la premisa de asegurar la transferencia de, al menos, una trama de longitud máxima en las clases de tráfico de menor peso.

Otro de los aspectos que determinan el rendimiento y el comportamiento del planificador es el paso del factor de corrección del marco de tiempo previo

al actual, C_n. El algoritmo sólo conoce si las clases de tráfico disponen de crédito, y el nivel de ocupación de las colas. Al elegir un paquete de longitud variable, su carga útil no suele coincidir con el crédito disponible. Por lo tanto, al final del marco de tiempo puede aparecer un valor negativo en el registro de créditos, que corresponde a un remanente de crédito C_n. Este valor negativo se interpreta como la cantidad de información en bytes que ha transferido una clase de tráfico en el marco de tiempo previo, sobrepasando el valor del crédito calculado con la ecuación 4.1 para mantener el reparto del ancho de banda. Al comienzo del marco de tiempo actual, dicha clase de tráfico se habilita y, al calcular el valor del crédito, el exceso de ancho de banda en el marco previo se compensa sumando el valor negativo almacenado. De esta manera, aunque el marco de tiempo previo no sea completamente justo, se penaliza a dicha clase en el marco de tiempo actual y se compensa el exceso de tráfico enviado en el marco de tiempo previo. Si dicha clase está vacía, no se considera el valor del registro, de forma que C_n es ignorado.

Además de los créditos, cada clase de tráfico dispone de un registro por cada fuente de tráfico que controla la cantidad de datos transferidos por cada cola, es decir, por cada binomio clase-fuente de tráfico. Al comienzo de cada marco de tiempo, los registros se restablecen comparando los valores de los registros de la misma clase de tráfico y se mantiene únicamente la diferencia entre ellos. Por lo tanto, se conoce la fuente de tráfico que ha transmitido menor cantidad de información dentro de una misma clase de tráfico. Si existe una fuente de tráfico vacía, todos los registros asociados a esa fuente toman directamente un valor nulo y no se comparan.

4.5 Caracterización del modelo de alto nivel del algoritmo de planificación

Antes de proceder a la implementación del algoritmo de planificación, se simula un modelo de alto nivel con el fin de obtener sus prestaciones estimadas y disponer de una descripción funcional detallada. En el caso particular de la

presente Tesis Doctoral, este modelo se ha realizado en el lenguaje C con la ayuda de la librería *CSIM* [CSIM]. Este *software* aporta un conjunto de mecanismos que permiten modelar el paralelismo de un sistema *hardware* y las herramientas matemáticas para el análisis de los parámetros de un sistema de conmutación. El objetivo principal de esta sección es conocer el rendimiento bajo diferentes configuraciones a fin de demostrar el soporte de diferentes calidades de servicio por parte del planificador bajo tráfico uniforme [ATE+06] y tráfico a ráfagas [ATE+07].

4.5.1 Tráfico uniforme

En las simulaciones que se presentan en esta sección se modifica la asignación del ancho de banda y de las prioridades manteniendo fijo el número de clases de tráfico. Como el conmutador *GMDS* soporta la transferencia de paquetes de longitud variable, todas las simulaciones se han realizado variando la longitud de la carga útil siguiendo una distribución uniforme entre 1 byte y 248 bytes. Para garantizar la fiabilidad de los datos presentados se asegura que, al menos, se reciban 10000 paquetes por cada fuente/clase de tráfico, valor suficiente para calcular el intervalo de confianza al 95% con una precisión del 2%.

4.5.1.1 Escenario 1. Test inicial

La primera simulación se utiliza como punto de partida del proceso de caracterización del algoritmo propuesto en un escenario sin calidad de servicio. Todas las clases de tráfico son tratadas de igual manera, es decir, no se diferencian ni prioridades ni anchos de banda. Los resultados muestran un retardo similar entre todas las clases de tráfico y el mismo ancho de banda consumido por cada una de ellas, como se aprecia en las Figuras 4.15 y 4.16, respectivamente.

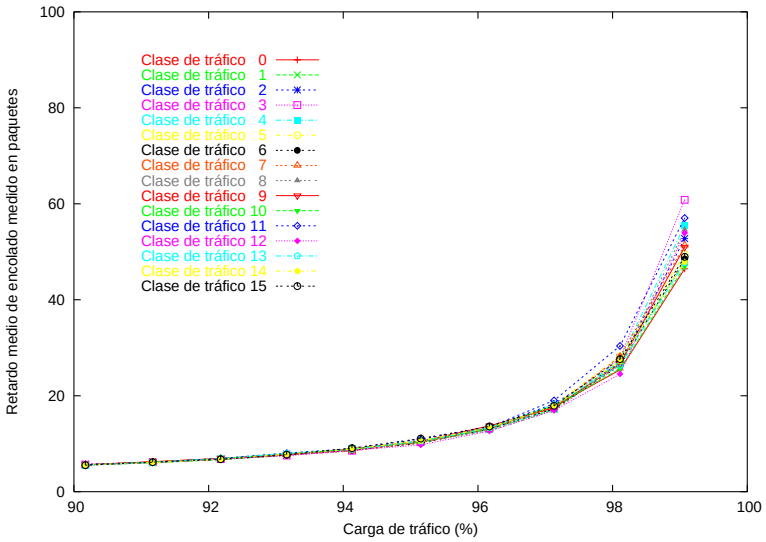


Figura 4.15: Retardo medio de encolado del modelo de alto nivel con tráfico uniforme en el escenario 1

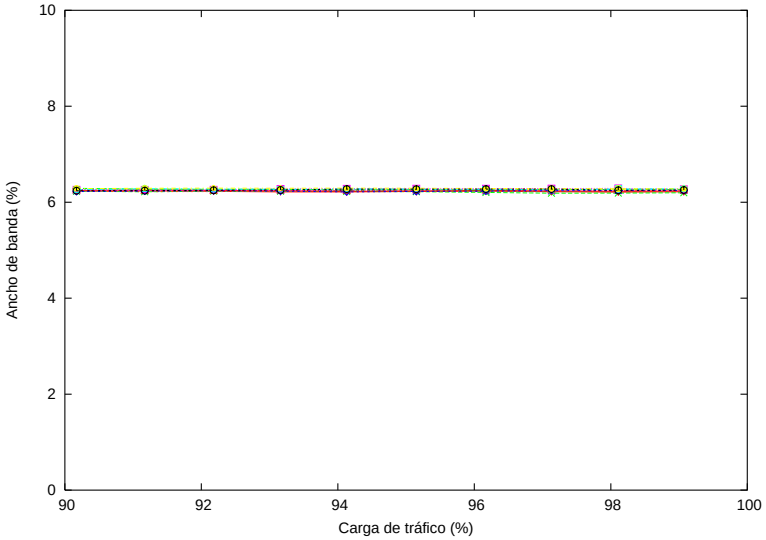


Figura 4.16: Distribución del ancho de banda del modelo de alto nivel con tráfico uniforme en el escenario 1

4.5.1.2 Escenario 2. Control del ancho de banda

El objetivo de esta simulación es estudiar el efecto del ancho de banda asignado a una clase de tráfico, en el retardo de dicha clase, por lo que se configuran las clases de tráfico con diferentes anchos de banda sin ninguna prioridad. Si el ancho de banda asignado a una clase de tráfico es relativamente elevado respecto a las restantes clases, el algoritmo atiende constantemente dicha clase de tráfico y, en consecuencia, su retardo se reduce. Si este ancho de banda es reducido, se produce el efecto contrario. Los valores asignados a los pesos de las diferentes clases de tráfico son 1%, 4%, 8% y 12% en grupos de cuatro clases de tráfico con el fin de crear un caso extremo de funcionamiento del algoritmo de planificación que acentúe las diferencias de prestaciones entre las clases de tráfico soportadas. En la Figura 4.17, las clases de tráfico con un 12% del ancho de banda reciben un mejor servicio que aquéllas con un 1% del ancho de banda asignado.

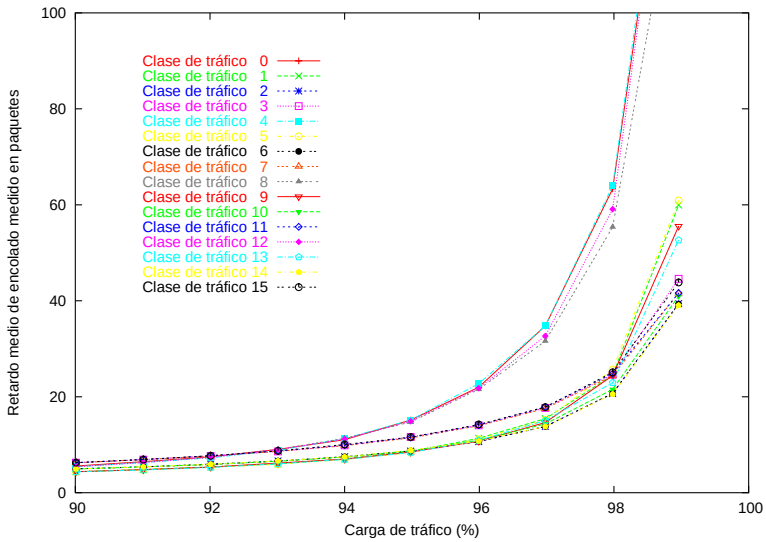


Figura 4.17: Retardo medio de encolado del modelo de alto nivel con tráfico uniforme en el escenario 2

Sin embargo, aunque la diferencia de retardo parezca elevada, se debe remarcar que las cuatro clases de tráfico que reciben el mejor servicio representan el 48% del ancho de banda disponible. Cuando la carga de tráfico alcanza cotas muy elevadas, las curvas del retardo se cruzan. Este efecto es completamente indeseado en el algoritmo y se analizará posteriormente. Desde el punto de vista del ancho de banda, el mecanismo basado en créditos garantiza completamente los pesos, como se aprecia en la Figura 4.18.

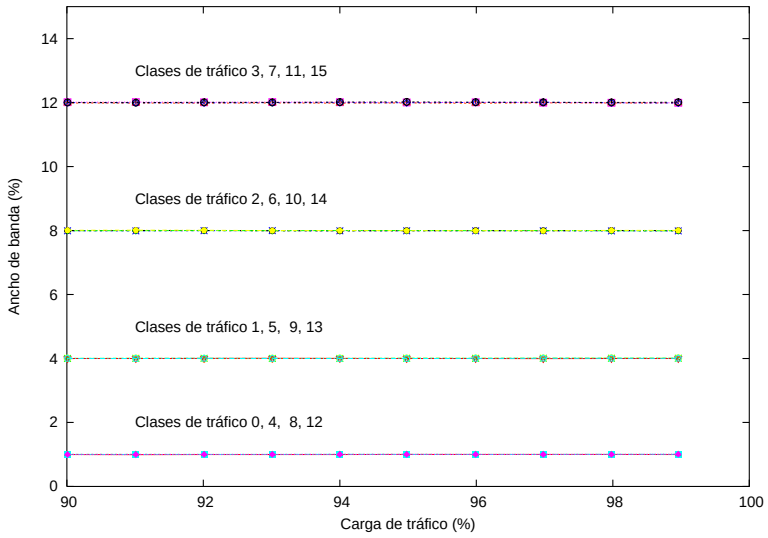


Figura 4.18: Distribución del ancho de banda del modelo de alto nivel con tráfico uniforme en el escenario 2

4.5.1.3 Escenario 3. Control del retardo

En este escenario se estudia únicamente la variación del retardo debida al efecto de las prioridades. En la Figura 4.19 se muestra la disparidad entre los retardos de las cuatro prioridades configuradas. Tomando como referencia el escenario 1, aquellas clases de tráfico con la prioridad más baja casi han duplicado su retardo. Las prioridades intermedias mantienen aproximadamente el mismo retardo y las clases de tráfico con la máxima prioridad reducen levemente su retardo hasta alcanzar prácticamente el retardo de un conmutador

ideal con colas a la salida. Evidentemente, desde el momento que se favorece una clase de tráfico, otra clase incrementa su retardo según la ley de conservación de Kleinrock. La Figura 4.20 muestra que la modificación de las prioridades no afecta a la distribución del ancho de banda.

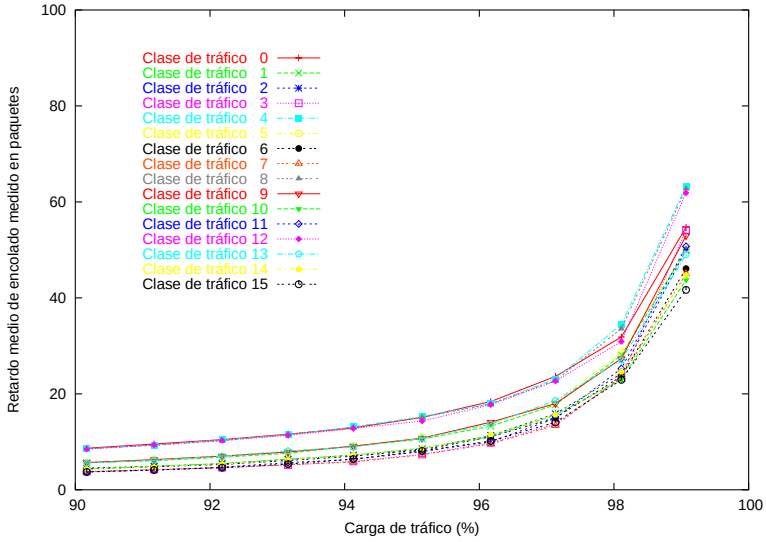


Figura 4.19: Retardo medio de encolado del modelo de alto nivel con tráfico uniforme en el escenario 3

4.5.1.4 Escenario 4. Control del ancho de banda y del retardo

Los resultados de la simulación muestran los efectos de asignar diferentes prioridades a las clases de tráfico con distinto ancho de banda. Se asigna la máxima prioridad a las clases de tráfico con peso máximo, 12%, incrementando las diferencias entre los retardos de las diferentes clases de tráfico, como se muestra en la Figura 4.21. Cabe destacar que las clases de tráfico más desfavorecidas sólo representan el 4% del ancho de banda total del conmutador. En cambio, se obtiene un 48% del ancho de tráfico total con un retardo similar al de un conmutador ideal con colas a la salida. El tráfico restante recibe un servicio intermedio. En la Figura 4.22 se comprueba la correcta distribución del ancho de banda.

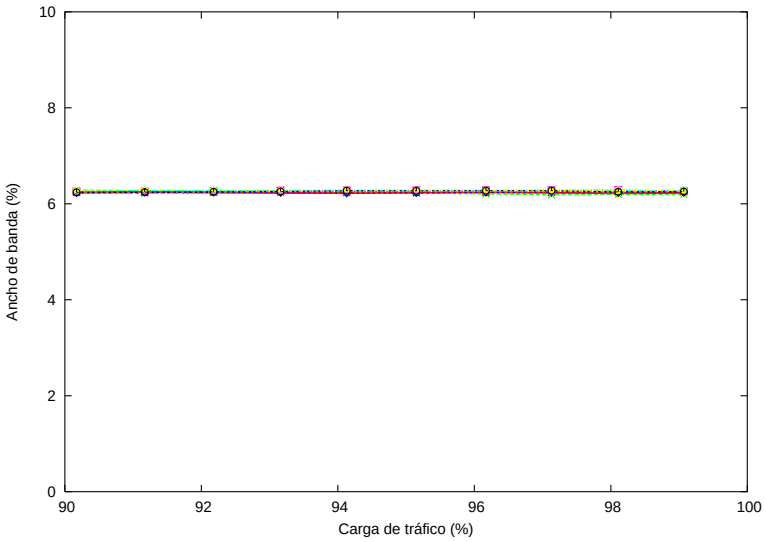


Figura 4.20: Distribución del ancho de banda del modelo de alto nivel con tráfico uniforme en el escenario 3

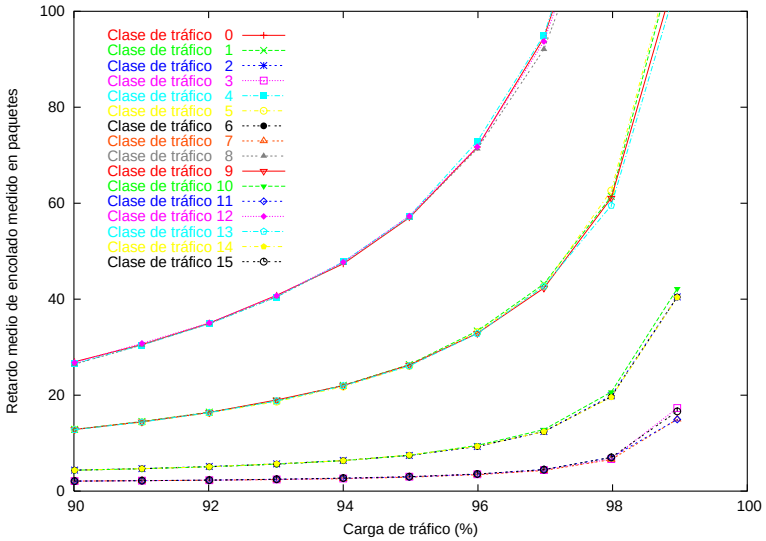


Figura 4.21: Retardo medio de encolado del modelo de alto nivel con tráfico uniforme en el escenario 4

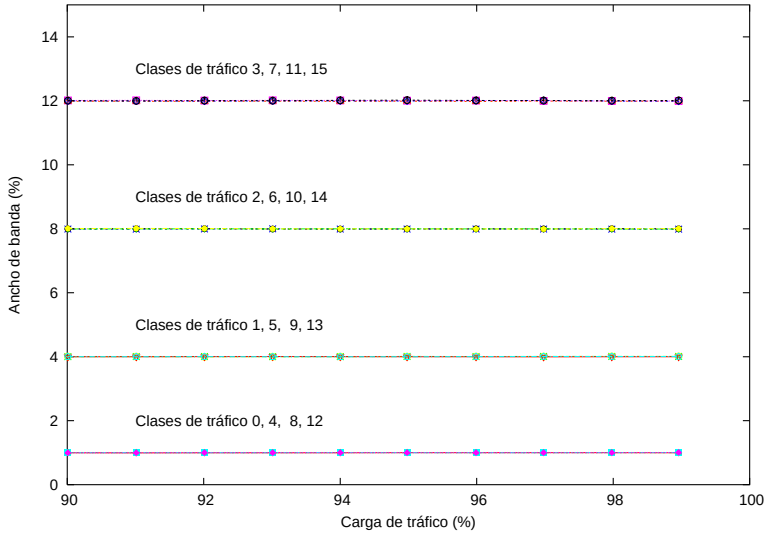


Figura 4.22: Distribución del ancho de banda del modelo de alto nivel con tráfico uniforme en el escenario 4

4.5.1.5 Escenario 5. Control del ancho de banda y del retardo

El objetivo de este escenario es estudiar la reducción del retardo de una clase de tráfico con peso mínimo y prioridad máxima. La Figura 4.23 muestra que las clases de tráfico con menor ancho de banda (1%) reciben mejor servicio que aquéllas con mayor ancho de banda (12%). Gracias al mecanismo de prioridades introducido, el algoritmo de planificación propuesto es capaz de compensar los efectos de la asignación del ancho de banda que favorece a las clases de tráfico con mayor ancho de banda. Las clases de tráfico con prioridad y peso intermedio reciben mejor servicio que las clases con pesos y prioridades extremas, ya que en estas clases no se contraponen el efecto del peso y la prioridad. Sin embargo, a partir de cargas de tráfico extremadamente altas, superiores a un 96%, el algoritmo muestra un comportamiento anómalo, siendo incapaz de mantener la configuración inicial debido a la disparidad existente entre los pesos asignados a las clases de tráfico. En todo momento se garantizan los anchos de banda, como se muestra en la Figura 4.24.

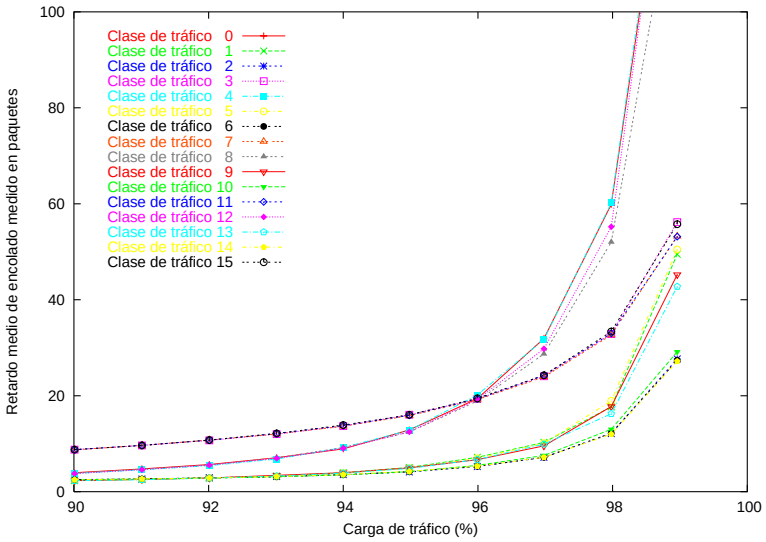


Figura 4.23: Retardo medio de encolado del modelo de alto nivel con tráfico uniforme en el escenario 5

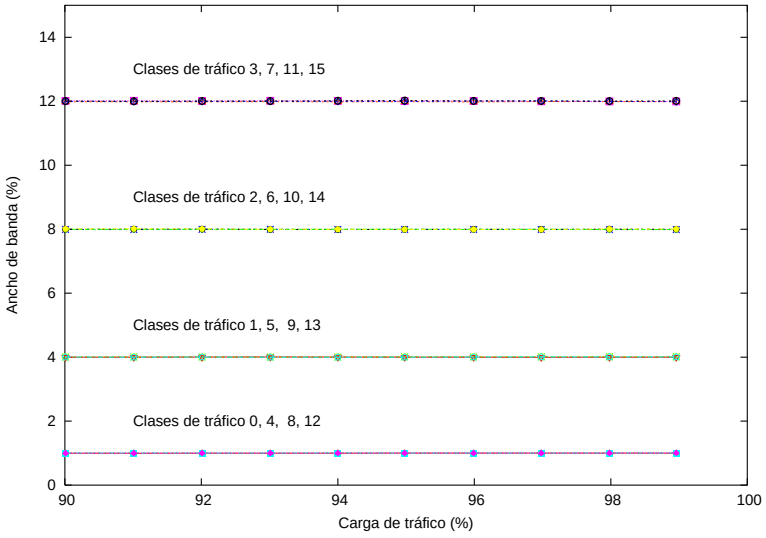


Figura 4.24: Distribución del ancho de banda del modelo de alto nivel con tráfico uniforme en el escenario 5

4.5.2 Tráfico a ráfagas

En las redes de datos actuales es común que se generen ráfagas de tráfico. Por lo tanto, es de vital importancia conocer la respuesta del algoritmo de planificación propuesto ante tal escenario. Las simulaciones que se presentan en esta sección muestran los resultados obtenidos después de que cada destino reciba al menos 200000 paquetes que aseguren el cálculo del intervalo de confianza al 95% con una precisión del 2%. Hay que remarcar que la carga de tráfico se ha reducido al intervalo comprendido entre el 70% y el 90%, no se ha limitado el tamaño de la memoria, y los escenarios de simulación se corresponden con los descritos anteriormente.

4.5.2.1 Escenario 1. Test inicial

Este escenario muestra todas las clases de tráfico con la misma prioridad y el mismo peso, por lo que se espera que todas reciban el mismo servicio, tal y como se deduce de los datos presentados en las Figuras 4.25 y 4.26. No obstante, debido al tráfico a ráfagas, a partir de una carga de tráfico del 85% se observa una desviación estándar que alcanza el 9% del valor medio. Además si se comparan los resultados de las simulaciones del tráfico a ráfagas con las correspondientes al tráfico uniforme, el retardo medio oscila entre los valores 5 y 10000 con un incremento exponencial, en lugar de entre los valores 5 y 100, respectivamente.

4.5.2.2 Escenario 2. Control del ancho de banda

Al igual que en el escenario 2 con tráfico uniforme, la configuración de pesos diferentes produce que las clases de tráfico reciban distinto servicio, lo que se traduce en retardos dispares, como se observa en la Figura 4.27. No obstante, el algoritmo de planificación propuesto, a pesar del tráfico a ráfagas, es capaz de evitar los efectos adversos mencionados previamente para este mismo escenario, pero con tráfico uniforme. Los resultados aseveran que el retardo es proporcional al peso configurado. Los anchos de banda de las diferentes clases

de tráfico se siguen manteniendo independientemente de los pesos, como se observa en la Figura 4.28.

4.5.2.3 Escenario 3. Control del retardo

Debido a la naturaleza del tráfico a ráfagas, cabe esperar que solamente estén ocupadas tantas colas como fuentes tenga el sistema de conmutación. Por lo tanto, la capacidad de elección entre las clases de tráfico es limitada y, aunque hay una leve diferencia de retardo, no se aprecia en la Figura 4.29 debido a la escala logarítmica del eje de ordenadas. El mecanismo de control del ancho de banda se ve ligeramente influenciado por el tráfico a ráfagas cuando la carga de tráfico alcanza valores elevados, como se aprecia en la Figura 4.30. No obstante, el valor medio del ancho de banda se corresponde con el valor configurado individualmente para cada clase de tráfico.

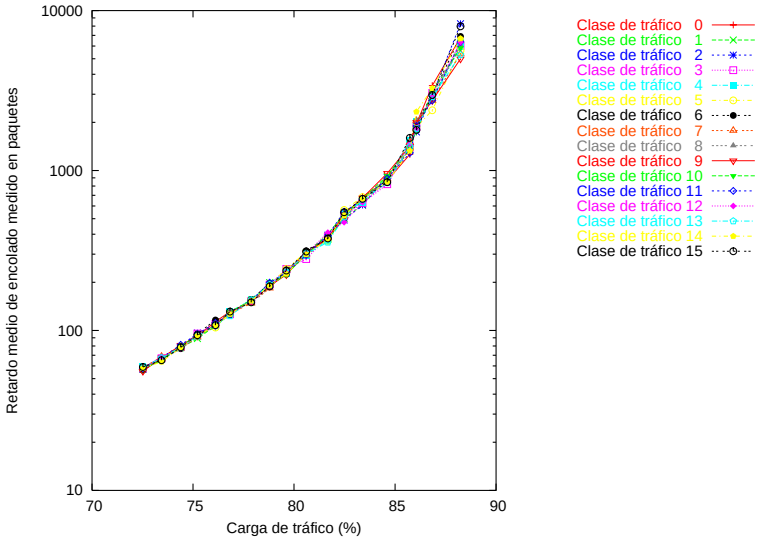


Figura 4.25: Retardo medio de encolado del modelo de alto nivel con tráfico a ráfagas en el escenario 1

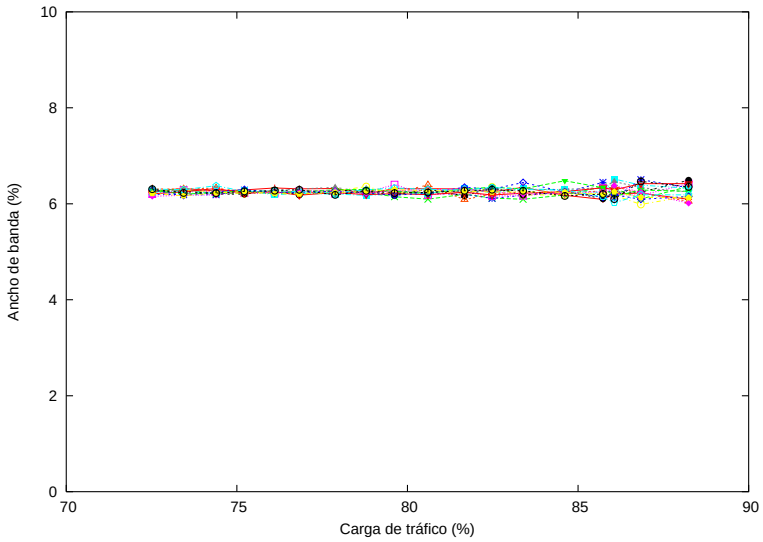


Figura 4.26: Distribución del ancho de banda del modelo de alto nivel con tráfico a ráfagas en el escenario 1

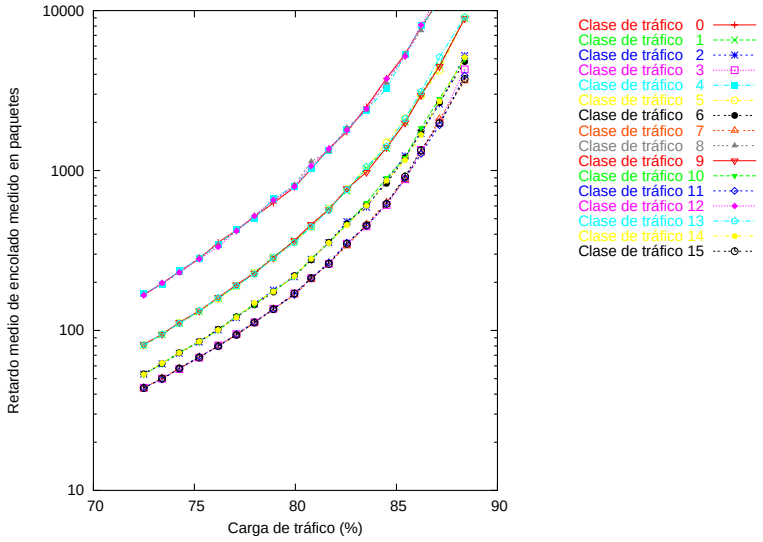


Figura 4.27: Retardo medio de encolado del modelo de alto nivel con tráfico a ráfagas en el escenario 2

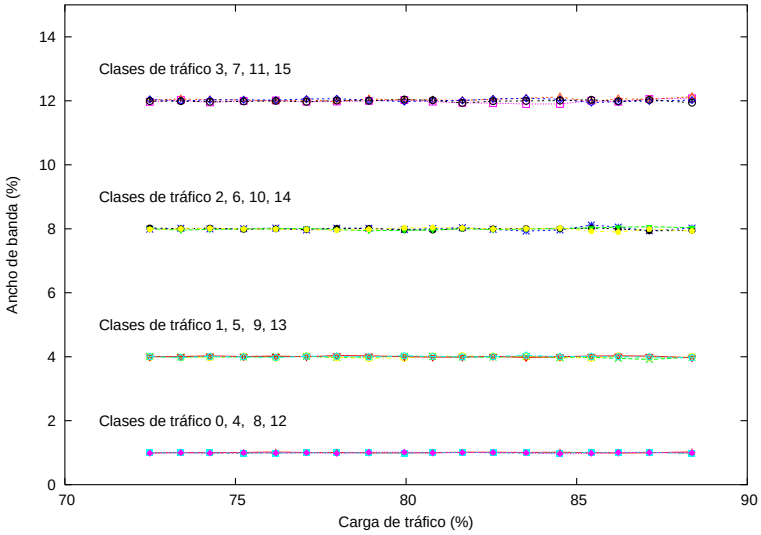


Figura 4.28: Distribución del ancho de banda del modelo de alto nivel con tráfico a ráfagas en el escenario 2

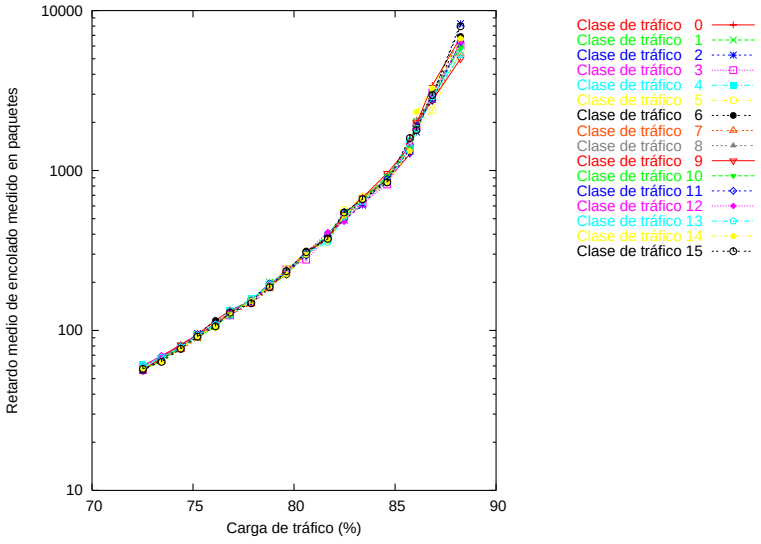


Figura 4.29: Retardo medio de encolado del modelo de alto nivel con tráfico a ráfagas en el escenario 3

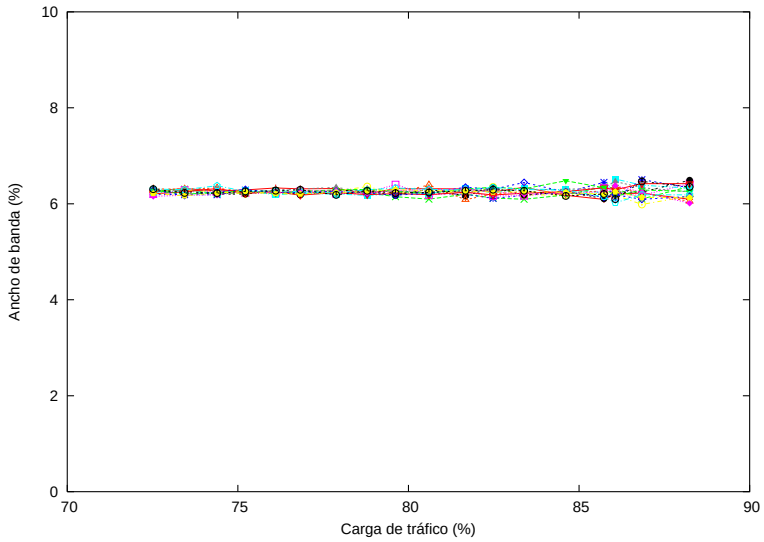


Figura 4.30: Distribución del ancho de banda del modelo de alto nivel con tráfico a ráfagas en el escenario 3

4.5.2.4 Escenario 4. Control del ancho de banda y del retardo

En este escenario se configuran las clases de tráfico con pesos de 1%, 4%, 8% y 12% en grupos de cuatro clases asignando la máxima prioridad a las clases de tráfico con mayor peso, y así sucesivamente. Las diferencias de retardo entre las clases de tráfico se incrementan notablemente respecto a este mismo escenario con tráfico uniforme debido a las características del tráfico a ráfagas y a las prioridades establecidas, como se presenta en la Figura 4.31. Además se conserva la relación entre el retardo y el ancho de banda, proporcionando un mejor servicio a las clases con mayor peso. El ancho de banda se reparte según la configuración establecida, independientemente de tipo de tráfico, como se observa en la Figura 4.32.

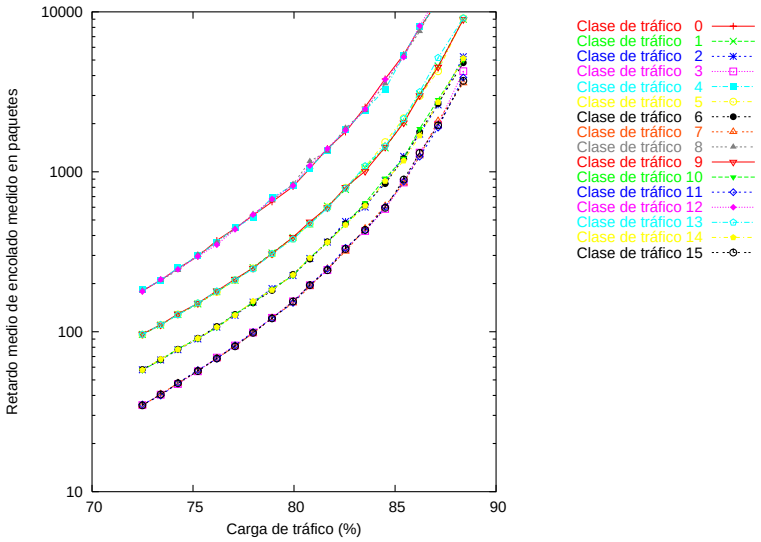


Figura 4.31: Retardo medio de encolado del modelo de alto nivel con tráfico a ráfagas en el escenario 4

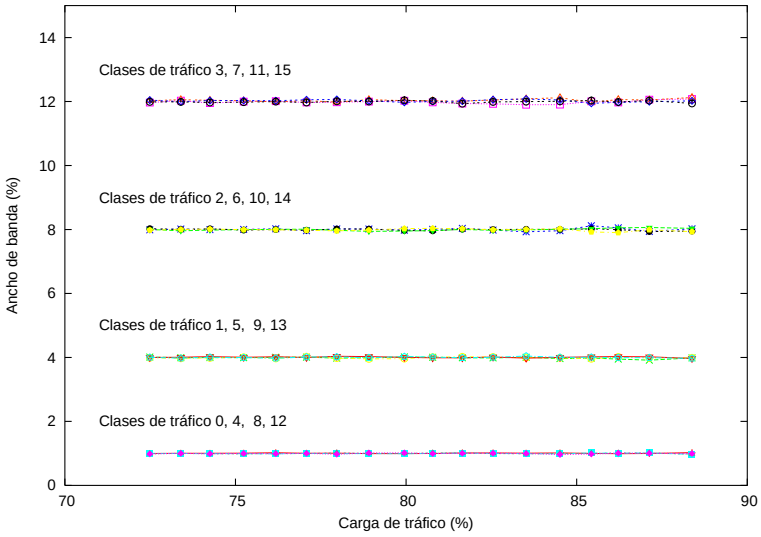


Figura 4.32: Distribución del ancho de banda del modelo de alto nivel con tráfico a ráfagas en el escenario 4

4.5.2.5 Escenario 5. Control del ancho de banda y del retardo

En la Figura 4.33 se muestra que las tendencias de las clases de tráfico con menor prioridad empeoran, si bien, debido a las grandes diferencias existentes entre los retardos de las diferentes clases de tráfico, dichas clases no presentan peor retardo que otras con mejor prioridad. Al igual que en los casos anteriores, el ancho de banda asignado a cada clase se garantiza, como se observa en la Figura 4.34.

4.6 Efecto de la longitud del marco de tiempo

La longitud del marco de tiempo depende de las proporciones de los pesos asignados a las clases de tráfico. En la sección 4.4.4 se estableció la premisa de asegurar al menos la transmisión de un paquete de longitud máxima en cada marco de tiempo sin ningún tipo de justificación. En esta sección se va a profundizar y razonar la idoneidad de esta decisión presentando varias simulaciones con diferentes longitudes del marco de tiempo. Puesto que sólo se requiere modificar los pesos para estudiar su efecto en el retardo, se ha elegido como referencia el Escenario 2 con tráfico uniforme, en el que se configura una única prioridad. Los resultados muestran el resultado de multiplicar la longitud del marco de tiempo por un factor variable. Los créditos se calculan siguiendo la ecuación propuesta:

$$Valor\ del\ crédito_{\ Clase\ de\ tráfico\ n} = Factor \times L_{MAX} \frac{Peso_{\ Clase\ de\ tráfico\ n}}{Peso_{\ MIN}} + C_n$$

Las Figuras 4.35, 4.36, 4.37 y 4.38 muestran el rendimiento del algoritmo de planificación cuando el factor multiplicador del marco de tiempo toma los valores 0,01, 0,1, 10 y 100.

Cuando se divide el marco de tiempo por un factor elevado se produce una reducción drástica del valor del crédito asignado a cada clase de tráfico al comienzo de cada marco de tiempo. Tras la transferencia de un paquete, y siempre que haya otro esperando en la misma cola, el algoritmo necesi-

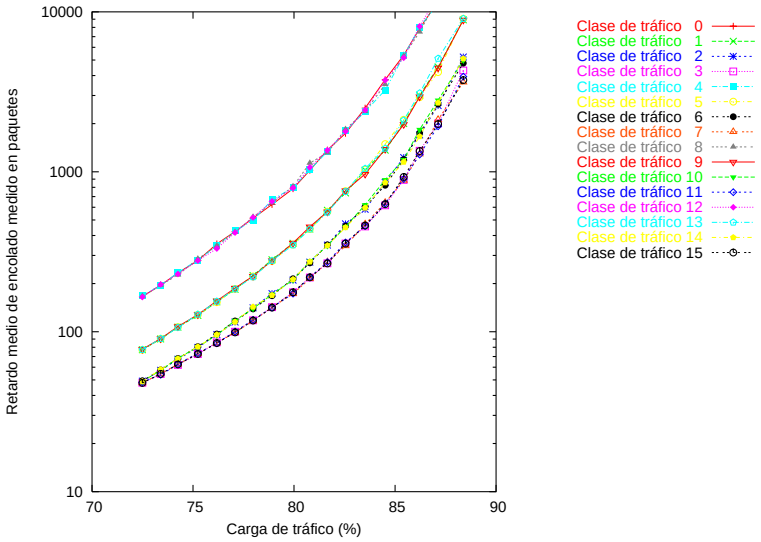


Figura 4.33: Retardo medio de las clases de tráfico del modelo de alto nivel con tráfico a ráfagas en el escenario 5

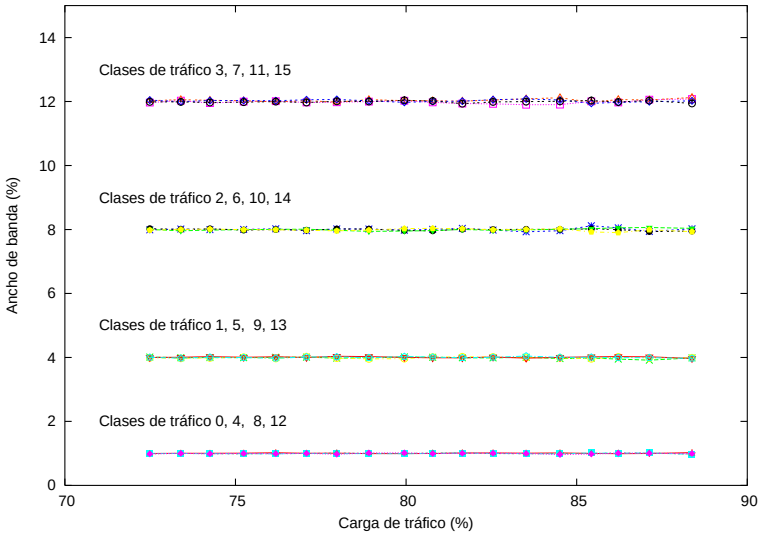


Figura 4.34: Distribución del ancho de banda del modelo de alto nivel con tráfico a ráfagas en el escenario 5

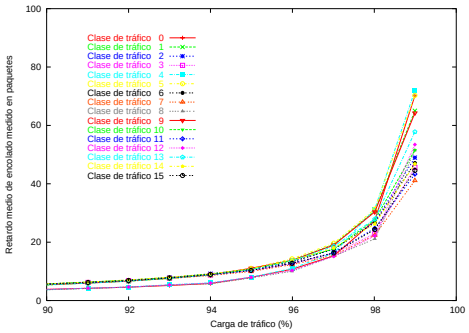


Figura 4.35: *RR*. Marco de tiempo / 100

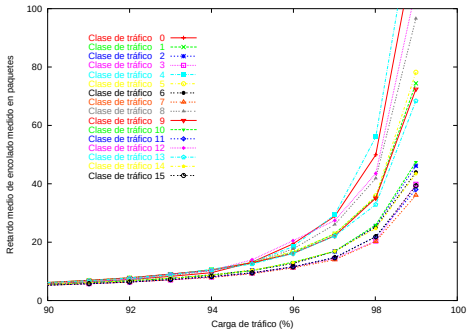


Figura 4.36: *RR*. Marco de tiempo / 10

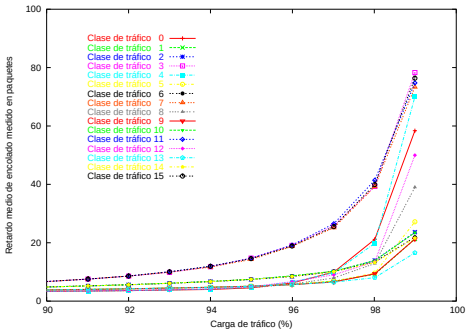


Figura 4.37: *RR*. Marco de tiempo $\times$ 10

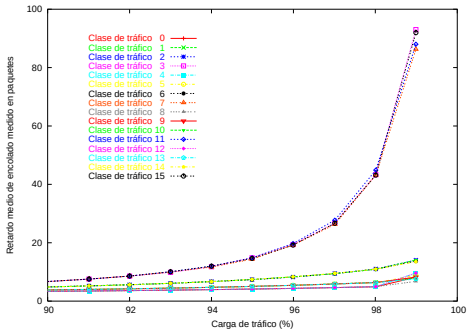


Figura 4.38: *RR*. Marco de tiempo $\times$ 100

tará reiniciar varias veces el marco de tiempo hasta acumular un crédito positivo que permita la transferencia del próximo paquete. Durante los marcos de tiempo que dicha clase de tráfico permanezca con un crédito negativo, ésta no estará disponible en el proceso de planificación. Por lo tanto, el resultado final es la imposibilidad de poder diferenciar el servicio proporcionado a las clases de tráfico, puesto que no es posible elegir entre todas al comienzo del marco de tiempo, como se observa en la Figura 4.35.

En el otro extremo, los valores de los créditos resultan muy elevados. En la mayor parte del marco de tiempo, el algoritmo de planificación es libre de elegir cualquier clase de tráfico activa. Sólo al final de cada marco de tiempo se produce un efecto de bloqueo que afecta en primer lugar a las clases de tráfico de menor peso al agotar su crédito en primer lugar, y se extiende pro-

gresivamente a las clases de tráfico de mayor peso. No obstante, el intervalo temporal donde se bloquean las clases de tráfico debido a la falta de crédito es reducido frente a la duración total del marco de tiempo, lo que representa que el algoritmo se comporta como un *RR* la mayor parte del tiempo, como se observa en la Figura 4.38.

Además, en la misma figura se advierte un comportamiento relevante del algoritmo de planificación propuesto. Las clases de tráfico con menor retardo son aquellas con menor ancho banda, mientras que las clases de tráfico que reciben peor servicio son las más congestionadas. Como todas las clases de tráfico disponen de un crédito elevado, pueden transmitir los paquetes instantes después de almacenarlos. Sin embargo, una clase de tráfico con gran ancho de banda tiene más paquetes encolados que una clase de tráfico con menor ancho de banda. Como ambas disponen del mismo número de oportunidades de transferir sus paquetes según la política *RR*, los paquetes de la clase con mayor ancho de banda deben esperar más tiempo hasta que se transmitan. En consecuencia, dos algoritmos *RR*, ya sea con créditos elevados o sin control de ancho de banda, se comportan igual y conceden mejor servicio a las clases con un menor ancho de banda. Este efecto induce a mantener un marco de tiempo unitario a fin de evitar elevados valores de los créditos.

Entre los valores extremos del factor de multiplicación, se presentan los casos intermedios que ayudan a comprender la evolución del efecto del marco de tiempo sobre el retardo. Las Figuras 4.36 y 4.37 muestran la tendencia de los retardos de las clases de tráfico en situaciones intermedias de funcionamiento en relación con la longitud del marco de tiempo.

4.7 Estudio de la justicia del algoritmo de planificación

El concepto de justicia dentro del ámbito de los algoritmos de planificación se refiere a la manera en que se asigna el ancho de banda a las conexiones. Idealmente este reparto se realiza de manera que no se produzcan ráfagas

a fin de obtener un mayor rendimiento. Dos algoritmos que tengan el mismo límite máximo de retardo, pero con diferente justicia, pueden asignarle a una misma conexión una cantidad de ancho de banda muy diferente durante un corto periodo de tiempo. Esta característica repercute en las medidas internas que toman los conmutadores para conocer el grado de carga y puede incidir negativamente en la congestión global de la red.

La medida de la justicia se puede realizar según varios índices [BZ96] [Bou08]. Entre ellos cabe destacar el índice denominado *WFI* (*Worst-case Fair Index*) presentado en [BZ97] para caracterizar servidores con política jerárquica *Packet Fair Queuing* (*PFQ*). Este índice compara los valores de un algoritmo *GPS* ideal con un algoritmo *PFQ*. Cuanto más pequeño sea el valor obtenido, más se aproxima el algoritmo bajo estudio al algoritmo ideal. Una actualización de este parámetro fue presentada en [CM04] para planificadores de paquetes dinámicos.

En [CK03] se realiza una revisión de la justicia de una arquitectura *CICQ* con soporte de calidad de servicio haciendo hincapié en el criterio de justicia máximo-mínimo (*Max-Min Fairness Criterion*) bajo elevada carga de tráfico. La diferenciación de los flujos de datos se consigue mediante el uso del algoritmo *WRR*/*WFQ*. En [ZMB07] se desarrolla una actualización en la misma línea donde se plantea un algoritmo basado en *PFQ* que asegura un *throughput* máximo para el tráfico admisible y mantiene el criterio de justicia máximo-mínimo para el tráfico no admisible.

El estudio de la justicia en los planificadores basados en políticas *Round Robin* revela cierta tendencia a generar ráfagas en periodos cortos de tiempo. Cabe destacar una aportación de Guo [Guo04] donde propone un algoritmo *Round Robin* modificado, denominado *Smoothed Round Robin* (*SRR*), que se basa en la idea de dispersar en el tiempo los paquetes de una misma clase de tráfico. Este mecanismo de dispersión se basa en la configuración de dos matrices: la matriz de los diferentes pesos de los flujos de datos y la matriz de dispersión de los pesos. Esta dispersión de los flujos de datos en función de sus pesos mejora notablemente la justicia y mantiene una baja complejidad

temporal, aunque incrementa los recursos *hardware* en la implementación del algoritmo de planificación. Una versión más compleja y con soporte de calidad de servicio se presenta en [Guo07]. En este caso se combina un algoritmo *Round Robin* recursivo con el algoritmo *SRR* para garantizar el ancho de banda, el retardo y la justicia.

Teniendo en cuenta que el algoritmo propuesto en esta Tesis Doctoral está basado en políticas *Round Robin*, la justicia y la posible generación de ráfagas requieren una atención especial. Se propone pues analizar la respuesta del algoritmo de planificación propuesto, partiendo de la configuración presentada en la Tabla 4.1, especialmente perjudicial desde el punto de vista de la generación de ráfagas de tráfico debido a la combinación de pesos muy dispares. En la Figura 4.39 se ilustra la respuesta del planificador propuesto en este escenario. Se establecen dos condiciones iniciales en la simulación. En primer lugar, se parte de una carga de tráfico elevada con el fin de disponer siempre de paquetes encolados y, en segundo lugar, de un marco de tiempo unitario que garantice el envío de, al menos, una trama de longitud máxima para la clase de tráfico con menor peso. Con el objetivo de simplificar la Figura 4.39 y facilitar el análisis del comportamiento del algoritmo, se han considerado paquetes de longitud fija.

Clase de tráfico	Ancho de banda	Prioridad	Paquetes
0	20%	Máxima	↑
1	20%	Máxima	↑
2	40%	Intermedia	↑
3	10%	Intermedia	↑
4	10%	Mínima	↑

Tabla 4.1: Configuración del tráfico para el estudio de la justicia

El principal inconveniente de esta configuración es la asignación de una misma prioridad a dos clases de tráfico con una gran diferencia de ancho de banda. El planificador combina los paquetes hasta que se consuma el ancho

de banda pero, irremediablemente, la clase que tiene asignado un 40% del ancho de banda genera una ráfaga después de que se haya consumido el ancho de banda de la otra clase de tráfico con la misma prioridad. Al definir un marco de tiempo de tamaño unitario, el tamaño de dicha ráfaga nunca podrá superar tres paquetes. Sin embargo, en el segundo caso se fija el marco de tiempo a un valor que permite a la clase de tráfico de menor peso enviar dos paquetes de longitud máxima, en lugar de uno. Al calcular los créditos de las restantes clases de tráfico, la clase de tráfico 2 puede enviar hasta 8 paquetes. En consecuencia, la ráfaga que se genera incrementa su longitud hasta 6 paquetes. Por este motivo, se aconseja dejar siempre el marco de tiempo de tamaño unitario para obligar al algoritmo a mezclar los paquetes entrantes, reduciendo al mínimo la generación de ráfagas. También hay que resaltar, siguiendo la política de dispersión, que existe un segundo mecanismo de selección de fuentes de tráfico que, aún siendo los paquetes de la misma clase de tráfico, mezcla los paquetes de diferentes fuentes evitando la ráfaga de una única fuente.

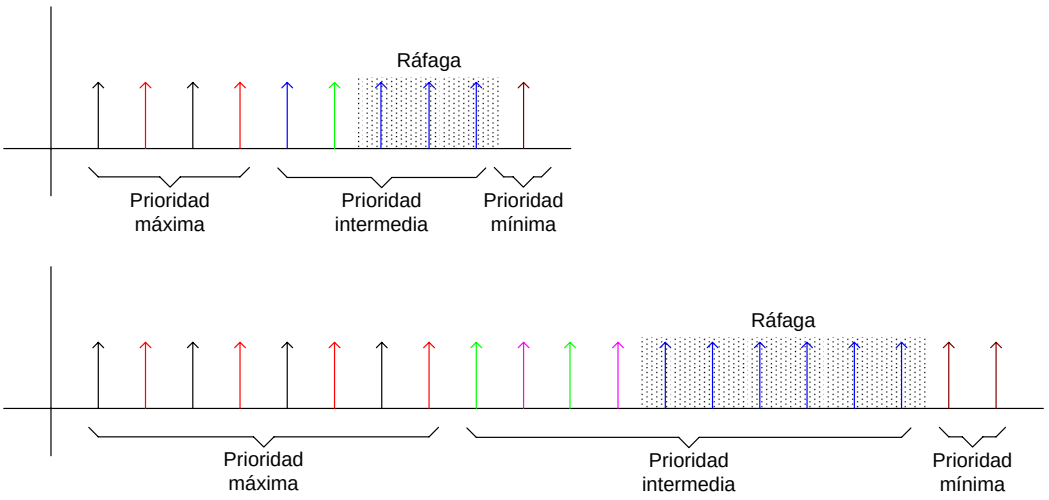


Figura 4.39: Selección de las clases de tráfico en función de la configuración de la Tabla 4.1

Por último, el marco de tiempo no se puede fijar a un valor menor que la unidad, ya que esto implica que no se puede enviar una trama de longitud

máxima. Es decir, la clase de tráfico de menor peso puede recibir un paquete de longitud máxima en el marco de tiempo actual, y dicho paquete consume todo el crédito del presente marco de tiempo y del siguiente. Este hecho empeora los retardos medios de dicha clase de tráfico y produce un *starvation* limitado temporalmente, además de empeorar la justicia del algoritmo.

Resumiendo, los efectos de la longitud del marco de tiempo en el retardo y en la justicia del algoritmo de planificación propuesto van íntimamente relacionados. Un incremento de la longitud del marco de tiempo mejora levemente los retardos, pero empeora la justicia del algoritmo. Por estos motivos, y dado que la arquitectura con colas a la salida presenta el mejor compromiso entre el retardo y el *throughput* [HK88], se prefieren minimizar las ráfagas de tráfico en lugar de reducir los retardos. Dicho aspecto se logra a través del control de la longitud del marco de tiempo y, según este análisis, se considera que la garantía de un paquete de longitud máxima en la clase de menor peso es el criterio más adecuado para lograrlo.

4.8 Influencia de la capacidad de las memorias en el rendimiento

Los escenarios de simulación presentados a lo largo de este capítulo presuponen una capacidad ilimitada de almacenamiento en los puertos de salida. Sin embargo, el prototipo del conmutador *GMDS* propuesto en el capítulo anterior dispone de una capacidad máxima para 128 paquetes por cola. Los principales efectos de la reducción de la capacidad de almacenamiento son la disminución del *throughput* del sistema [ID93][QS96] y el incremento de la probabilidad de pérdidas de paquetes [KS01].

La comparación del *throughput* entre el límite teórico [MRS03][CL07] y las simulaciones presentadas a lo largo de este capítulo no muestra una reducción significativa del rendimiento, sólo uno o dos puntos porcentuales. El principal motivo de este resultado tan positivo es la elección de la distribución de *Bernoulli*. Los paquetes generados se reparten entre todas las colas con

igual probabilidad, colapsando las colas con cargas de tráfico muy altas. En un caso real, antes de que se alcance el nivel de *throughput* máximo y se produzcan pérdidas de paquetes, se activa el mecanismo de control de flujo del conmutador *GMDS*, reduciendo la tasa de transferencia de los flujos de datos entrantes. Otro factor que contribuye a la leve pérdida de prestaciones es la estructura de memoria con múltiples colas a la salida en lugar de una cola única.

Como se presentó en las secciones 4.3.1.2 y 4.3.2.2, la capacidad de las colas del conmutador *GMDS* es insuficiente para lograr un alto *throughput* con una baja probabilidad de pérdidas de paquetes con tráfico a ráfagas. No obstante, este diseño se ha concebido como un prototipo que, en una realización física final, debe incluir mayor capacidad de almacenamiento.

4.9 Estudio de la desactivación del control de ancho de banda

En este apartado se plantea un enfoque diferente de la filosofía que se ha seguido a lo largo del capítulo. En todo momento se insiste en la protección de los flujos de datos. En ningún caso se parte del supuesto de que las fuentes de tráfico conozcan la distribución del ancho de banda — o haya otro mecanismo, que no sea el planificador, encargado de controlar el ancho de banda — y, por lo tanto, no se necesite controlar el ancho de banda en el algoritmo de planificación.

En las simulaciones que se presentan en esta sección se plantea este escenario, es decir, se asigna el mismo ancho de banda y diferente prioridad a las clases de tráfico, como en el escenario 3. Sin embargo, se ha desactivado el mecanismo de control de ancho de banda de una determinada clase de tráfico, o dicho de otra manera, dicha clase dispone de crédito ilimitado. En la simulación mostrada en la Figura 4.40, la clase de tráfico elegida se asigna a la máxima prioridad. Posteriormente, en la simulación mostrada en la Figura 4.41, la misma clase de tráfico se asigna a la mínima prioridad.

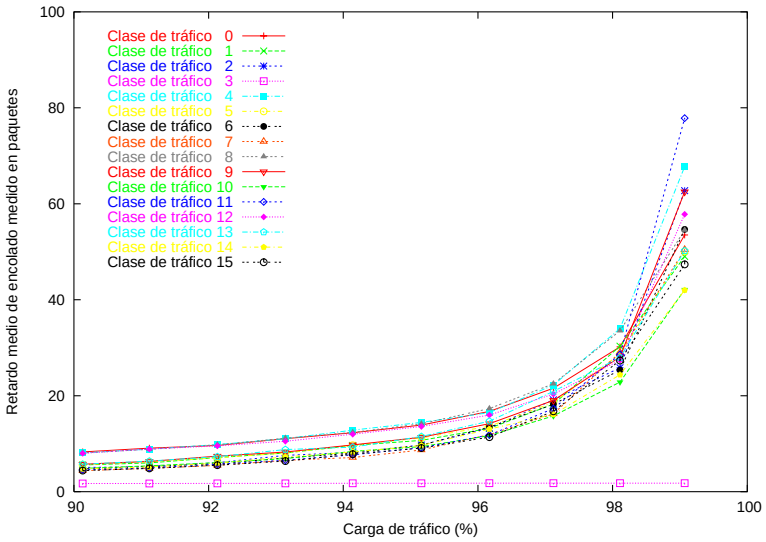


Figura 4.40: Retardo medio de encolado del modelo de alto nivel con tráfico uniforme en el escenario 3 con la clase de máxima prioridad libre

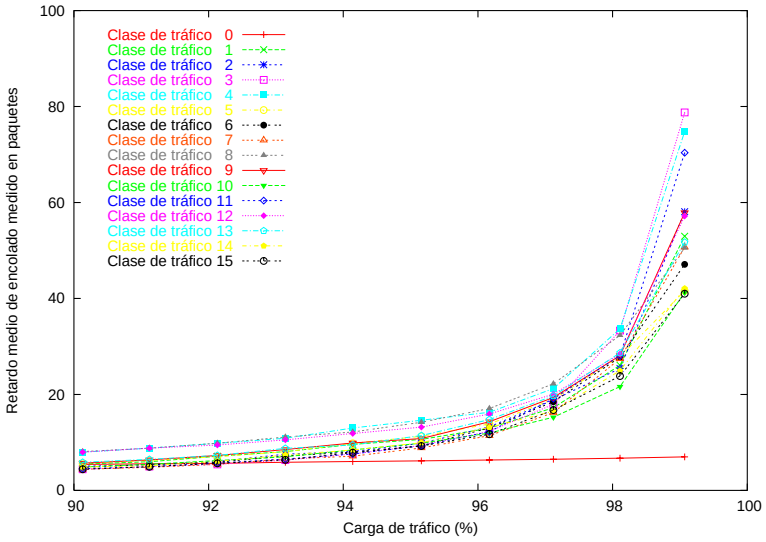


Figura 4.41: Retardo medio de encolado del modelo de alto nivel con tráfico uniforme en el escenario 3 con la clase de mínima prioridad libre

La Figura 4.40 muestra cómo se atiende a la clase de máxima prioridad de forma casi inmediata. Se aconseja esta configuración solamente en los casos de consumo mínimo de ancho de banda por parte de la clase sin control. En caso de que esta clase de tráfico demandara una tasa de transferencia elevada, utilizaría ancho de banda de las restantes clases, que sí disponen de control de ancho de banda, y produciría un incremento del retardo medio de los paquetes asignado a las demás clases de tráfico.

En el caso presentado en la Figura 4.41 se observa una mejora en el retardo de la clase de tráfico libre, frente a las restantes clases de tráfico de la misma prioridad. Destaca que el retardo de la clase de tráfico de baja prioridad se muestra constante, ya que sólo necesita esperar a que transcurra el marco de tiempo, que es un valor temporal constante, para enviar todos sus paquetes. En el caso de máxima prioridad es mucho menor, puesto que se atienden al principio del marco de tiempo en lugar de al final del mismo.

4.10 Implementación del algoritmo de planificación

La implementación del algoritmo de planificación propuesto en esta Tesis Doctoral se ha estructurado en tres módulos: selección de la clase de tráfico, selección de la fuente de tráfico y la máquina de estados finitos del sistema. El planificador es independiente de la tarjeta de línea y el *backplane*. Se ha implementado en el lenguaje *Verilog HDL* a nivel *RTL* y se ha sintetizado en una *FPGA Spartan-3 1000* (*xc3s1000-4FT256*) a fin de dotar de flexibilidad el sistema y permitir la implementación de nuevas propuestas. El algoritmo de planificación recibe la configuración con la información de los pesos y las prioridades desde un módulo externo.

4.10.1 Información proporcionada por el conmutador *GMDS*

El conmutador *GMDS* incluye un submódulo *Estado* en el módulo *Egress* encargado de recopilar la información sobre el nivel de ocupación de las colas correspondientes a cuatro fuentes de tráfico. Parte de esta información se envía al planificador para la selección de los paquetes. Concretamente existen dos mecanismos de consulta. En el primero, se envían dos bits por cada fuente de la clase seleccionada (8 bits) en respuesta a la clase indicada (4 bits). En el segundo se envían dos bits por clase de tráfico en grupos de 4 clases (8 bits) en respuesta al grupo elegido (2 bits). La información obtenida se interpreta según la Tabla 4.2. Ambos mecanismos están diseñados para realizar un *polling* continuo al conmutador *GMDS*, ya que, por un lado, no disponen de validación ni ningún otro mecanismo de control y, por otro lado, el planificador obtiene la respuesta un ciclo después de haber realizado la petición de información al mismo tiempo que envía la siguiente lectura. Ambos mecanismos funcionan en paralelo.

Bits	Interpretación
00	Cola vacía
01	Existe un paquete en la cola
10	Existen dos paquetes en la cola
11	Existen 3 paquetes en cola

Tabla 4.2: Niveles de ocupación de las colas

Es importante remarcar que el planificador únicamente conoce el nivel de ocupación real de una cola si tiene almacenados dos o menos paquetes, como se muestra en la Tabla 4.2. Tampoco conoce el tiempo de espera de dichos paquetes ni sus respectivas longitudes.

4.10.2 Selección de la clase de tráfico

La selección de la clase de tráfico se realiza a partir de la información inicial proporcionada por el módulo externo de configuración, el nivel de ocupación de las colas, y las últimas clases de tráfico seleccionadas. La Figura 4.42 muestra el proceso de selección de las clases de tráfico, así como las estructuras de datos y las operaciones implicadas. Los diferentes vectores y matrices están parametrizados en función de P, C y N siendo P el número de prioridades, C el número de clases de tráfico y N el número de fuentes de tráfico o puertos de entrada.

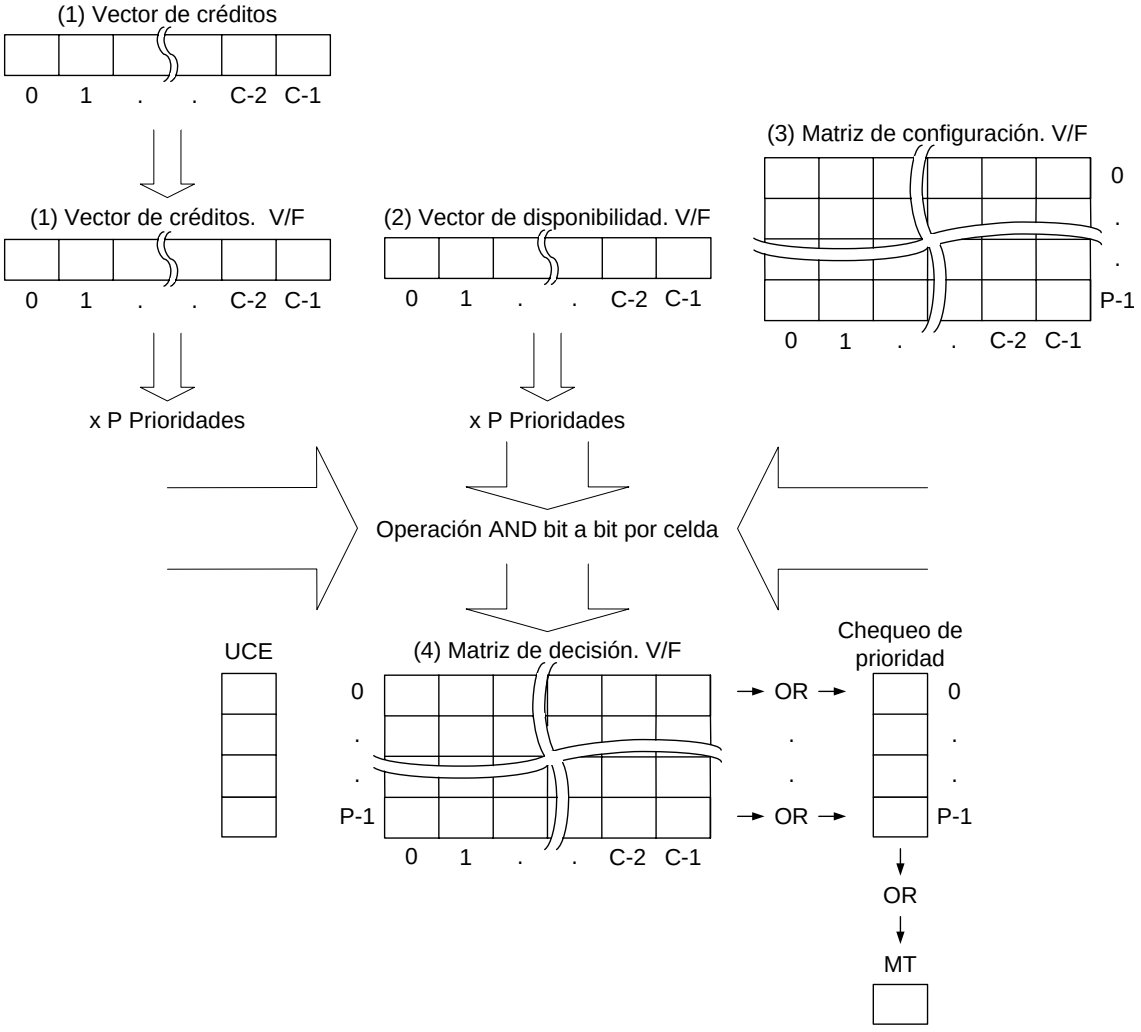


Figura 4.42: Proceso de selección de las clases de tráfico

En primer lugar, se crea el *vector de créditos* (1) de C posiciones donde cada celda almacena el crédito disponible asociado a una clase de tráfico específica. Al comienzo de cada marco de tiempo se incrementa este valor y, cuando un paquete se transfiere hacia el puerto de salida del conmutador, se reduce el crédito asignado a su clase de tráfico en función de su longitud. Sin embargo, solamente se comprueba si el valor de cada celda es mayor que cero, creando un nuevo vector con esta información y almacenando únicamente un bit por celda, verdadero o falso.

En segundo lugar, con la información suministrada por el submódulo *Estado* del conmutador *GMDS*, se genera un *vector de disponibilidad* (2) de C posiciones de un bit cada una, siendo C el número de clases de tráfico, donde se almacena si cada clase de tráfico está vacía o no.

Por último, se define una *matriz de configuración* (3) de tantas filas como prioridades y tantas columnas como clases de tráfico se configuren en el planificador. Cada celda de la matriz controla la asignación de una clase de tráfico a una determinada prioridad mediante un único bit.

La información almacenada en los *vectores de créditos* (1) procede del propio planificador. Asimismo, el *vector de disponibilidad* (2) obtiene la información del conmutador *GMDS* y la *matriz de configuración* (3) del módulo de configuración externa. Con el objetivo de discernir las clases de tráfico elegibles, se combinan todas las celdas en la *matriz de decisión* (4) mediante una operación *AND* bit a bit. Por lo tanto, si una clase de tráfico tiene crédito, dispone de paquetes, y se asigna a una prioridad determinada, la celda correspondiente en la *matriz de decisión* (4) almacena un valor verdadero y dicha clase de tráfico se considera en el proceso de planificación.

Durante el proceso de selección de las clases de tráfico, el algoritmo de planificación propuesto comprueba el vector *chequeo de prioridad* generado a partir de realizar la operación *OR* en cada fila de la *matriz de decisión* (4). Si una posición del registro almacena un valor verdadero, se interpreta como una prioridad con, al menos, una clase de tráfico elegible. La importancia de este registro radica en acelerar la elección de la máxima prioridad. En el caso

de no haber disponible ninguna prioridad, es decir, encontrarse todos los registros con un valor falso, se activan en la máquina de estados las señales del comienzo de un nuevo marco de tiempo. El registro denominado *MT*, abreviación de *Marco de Tiempo*, no es más que la operación *OR* de las prioridades con el fin de acelerar el proceso de detección y comienzo de un nuevo marco de tiempo.

Una vez que se ha elegido la prioridad con la ayuda del registro anterior, se elige la clase de tráfico según un mecanismo *Round Robin*. En este caso, se debe conocer la última clase de tráfico que recibió servicio, por lo que existe un vector auxiliar, denominado *UCE* (*Última Clase Enviada*). Finalmente, se comunica al módulo de selección de la fuente de tráfico la clase de tráfico planificada.

4.10.3 Selección de la fuente de tráfico

La información sobre el nivel de ocupación de las fuentes de tráfico, procedente del conmutador *GMDS*, se almacena en una *matriz de disponibilidad* de C filas por N columnas, siendo C el número de clases de tráfico y N el número de fuentes. Una segunda *matriz de tráfico enviado*, con dimensiones similares, y en paralelo, almacena el tráfico enviado por cada fuente de cada clase de tráfico. Por lo tanto, una vez que se elige la clase de tráfico, se comparan los valores acumulados del tráfico enviado por todas las fuentes con paquetes de la clase de tráfico seleccionada y se selecciona la fuente que haya enviado la menor cantidad de información en el marco de tiempo actual.

La *matriz de tráfico enviado* se implementa con las memorias *RAM* propias de la *FPGA* para reducir el número de registros y comparadores. Una vez elegida la clase de tráfico, se leen en paralelo todos los registros de las memorias y, con la ayuda de la *matriz de disponibilidad*, se comparan sus valores, eligiendo la fuente de tráfico.

Hay que destacar el funcionamiento del mecanismo que actualiza los valores almacenados en la *matriz de tráfico enviado*. Este proceso se realiza, por lo general, al comienzo del marco de tiempo y entraña cierta dificultad. Debido

a la posibilidad de configurar diferentes pesos, se deben comparar los valores de tráfico enviado asociados a la misma clase de tráfico y substraer a todos ellos el menor valor dentro de la misma clase de tráfico. Solamente interesa mantener las diferencias entre ellos para elegir la fuente que haya enviado la menor cantidad de tráfico. Como toda la información sobre el tráfico enviado por cada fuente se almacena en la memoria, es imposible actualizar el contenido al comienzo del marco de tiempo. Para solventar este problema existe un vector que almacena un bit por cada clase de tráfico con el fin de indicar si ha sido actualizada. Por lo tanto, la primera vez que se produzca la selección de la clase de tráfico en un marco de tiempo, se actualizan los valores de todas las fuentes de tráfico y se desactiva el bit correspondiente del vector. Si alguna fuente está vacía, el registro se anula automáticamente en el inicio del marco de tiempo.

4.10.4 Arquitectura *hardware*

La Figura 4.43 muestra la arquitectura *hardware* del planificador propuesto en esta Tesis Doctoral. Los módulos se encargan de la gestión de los créditos, del control de la disponibilidad de los paquetes en sus correspondientes clases y fuentes de tráfico, y de la configuración del planificador. Además, se comunican con el módulo externo de configuración del planificador y la tarjeta de línea para obtener la información que, posteriormente, se procesa y almacena con el fin de seleccionar una determina cola. Este último proceso se compone del mecanismo de prioridad y el algoritmo *RR* implementado en el módulo *selección de la clase de tráfico*, y de las comparaciones entre el tráfico enviado de las fuentes de la misma clase implementado en el módulo *tráfico enviado*.

La modularidad de la arquitectura permite la modificación del número de clases o de fuentes de tráfico y que sólo afecte a los módulos con conectividad externa. Además, es posible la modificación del núcleo del algoritmo de planificación, es decir, la selección de las clases de tráfico, manteniendo la arquitectura propuesta con el fin de estudiar nuevas ideas o mejoras parciales siempre que la información de partida sea el nivel de ocupación de las colas.

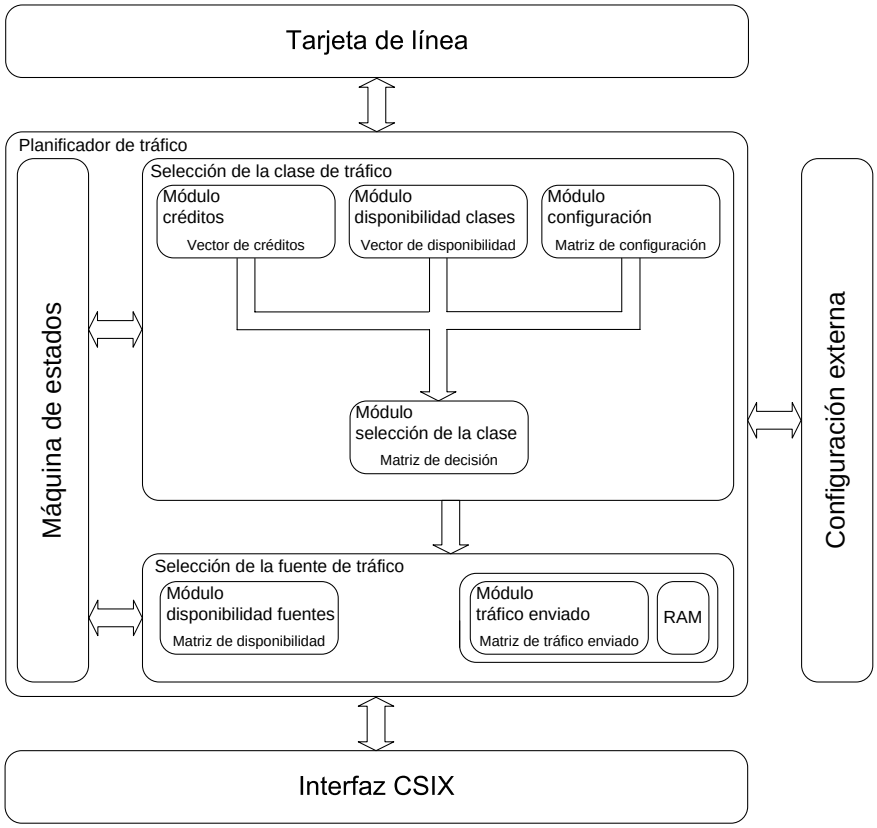


Figura 4.43: Arquitectura *hardware* del planificador

4.10.5 Restricciones temporales

Además de las consideraciones algorítmicas descritas en el apartado anterior y la descripción de la implementación, hay que remarcar varios detalles sobre las restricciones temporales del planificador de paquetes propuesto.

En primer lugar, la frecuencia de funcionamiento es la misma que el módulo *Egress* del conmutador *GMDS*, es decir, 38 MHz. En segundo lugar, la respuesta del conmutador a la petición de un paquete es de 4 ciclos de reloj. Al mismo tiempo, el planificador requiere tres ciclos para decidir el paquete a solicitar una vez conocida la longitud del paquete previo. Por último, y como consecuencia de considerar los tiempos de respuesta del conmutador y el planificador, se pueden producir varios ciclos de reloj, dependiendo su número de la longitud de los paquetes, en los cuales el planificador se mantiene a la espera de la respuesta del conmutador.

La Figura 4.44 muestra los ciclos de espera en el peor caso, considerando un paquete de tamaño mínimo. En el ciclo 0 se produce la petición activando la señal *sc_read* e indicando la clase y la fuente a través de las señales *sc_class [3:0]* y *sc_source [1:0]*, respectivamente. En el ciclo 4 dicho paquete sale del conmutador y, por lo tanto, se conoce la longitud del paquete. Al tratarse de un paquete de tamaño mínimo, en ese mismo ciclo se elige la clase y, en los dos siguientes, la fuente de tráfico. En el ciclo 7 se realiza la petición de un nuevo paquete, que se recibe en el ciclo 11 con un retardo de 4 ciclos de reloj.

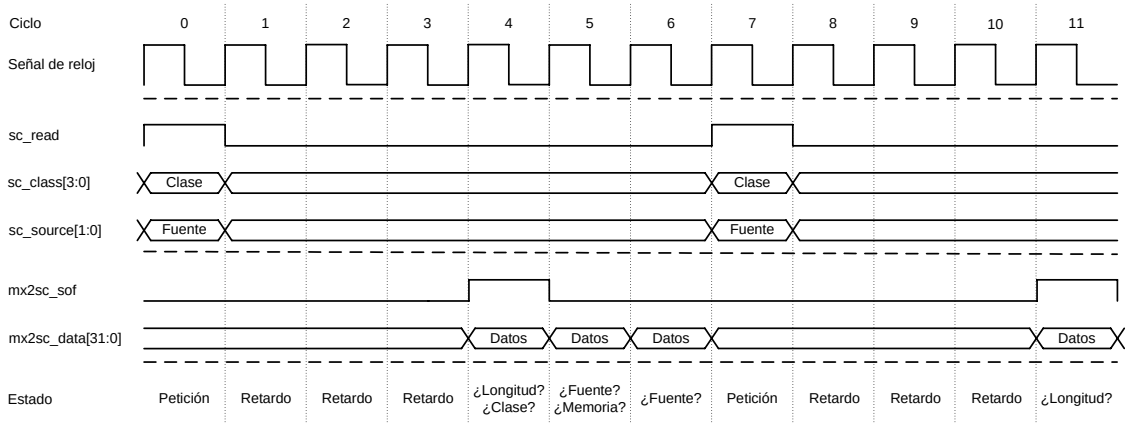


Figura 4.44: Ciclos de espera

La Tabla 4.3 expresa matemáticamente los ciclos de espera debidos al procesado de paquetes de longitud excesivamente reducida. Aunque se pierdan varios ciclos de reloj hay que recordar que, habitualmente, las fuentes tienden a generar paquetes de longitudes medias o elevadas, ya que el *overhead* siempre es constante y no depende de la longitud del paquete. Además, en el conmutador *GMDS* se ha introducido una ligera aceleración, de 37,5 MHz en el módulo *Ingress* a 38 MHz en el módulo *Egress*, para compensar, entre otros, este efecto.

Overhead	Carga útil	Relleno	Longitud del paquete	Ciclos de espera
8 bytes	1-4 bytes	0-3 bytes	3 ciclos de reloj (12 bytes)	4
8 bytes	5-8 bytes	0-3 bytes	4 ciclos de reloj (16 bytes)	3
8 bytes	9-12 bytes	0-3 bytes	5 ciclos de reloj (20 bytes)	2
8 bytes	12-15 bytes	0-3 bytes	6 ciclos de reloj (24 bytes)	1
8 bytes	$\geq$16 bytes	0-3 bytes	$\geq$7 ciclos de reloj ($\geq$28 bytes)	0

Tabla 4.3: Ciclos de espera

4.10.6 Síntesis

La Tabla 4.4 muestra los resultados de la síntesis del algoritmo de planificación realizada en la herramienta *ISE* versión 8.2.02i de *Xilinx*. La implementación se ha llevado a cabo en el lenguaje *Verilog HDL* a nivel *RTL* y la síntesis en una *FPGA Spartan-3* 1000 (xc3s1000-4FT256), caracterizada por la excelente relación entre el coste económico y las prestaciones que proporciona, y por la posibilidad de disponer de dispositivos con más capacidad en caso de nuevas propuestas de algoritmos de planificación. No se han introducido restricciones en el proceso de síntesis.

Lógica utilizada	Usado	Disponible	Utilizado
Número de *Slices*	1326	7680	17%
Número de *Slices Flip Flops*	796	15360	5%
Número de *LUTs*	2241	15360	14%
Número de *Block RAMs*	2	24	8%
Número de *MULT18X18s*	16	24	66%
Número de *GCLKs*	1	8	12%
Número de puertas equivalentes	219174		

Tabla 4.4: Resultados de la síntesis del algoritmo de planificación basado en política *RR*

Debido a las restricciones temporales de la planificación, explicadas detalladamente en el apartado anterior, la mayor parte de la lógica es combinacional, introduciendo grandes retardos en las rutas de datos y con pocas posibilidades de segmentación. La frecuencia máxima alcanzada son 39,207 MHz, lo que representa un valor suficiente para el prototipo propuesto. El número de puertas equivalentes es 219174. Aproximadamente, la mitad de la lógica se concentra en la matriz que contiene el tráfico enviado por cada fuente, compuesta por los dos bloques de memoria *RAM*.

La comparación de los recursos utilizados en la implementación por el planificador propuesto y por una tarjeta de línea capaz de atender a cuatro fuentes de tráfico es un aspecto que resulta interesante. El planificador ocupa, aproximadamente, el 22% de las puertas equivalentes utilizadas por la tarjeta de línea. Desde el punto de vista de la comparación entre los módulos, el planificador y el módulo *Ingress* son similares. Con estos resultados, se aprecia que los recursos necesarios por el planificador no son elevados frente a la tarjeta de línea, si bien, debido a la flexibilidad con la que se quiere dotar al mecanismo de planificación haciendo uso de un dispositivo programable independiente de la tarjeta de línea, esta relación de ocupación de recursos puede cambiar con una nueva propuesta de planificación.

4.11 Caracterización del modelo de bajo nivel del algoritmo de planificación

Una vez finalizada la descripción del algoritmo de planificación y el estudio de sus prestaciones y propiedades con el modelo de alto nivel, se ha implementado el planificador en el lenguaje *Verilog HDL* a nivel *RTL* con el objetivo de caracterizar nuevamente sus prestaciones incluyendo los retardos propios de la implementación y de la comunicación con el conmutador *GMDS*. Los casos de estudio coinciden con los considerados en la caracterización del modelo de alto nivel, a fin de facilitar su comprensión. Sin embargo, las condiciones de simulación varían ligeramente respecto a un modelo de alto nivel. En primer

lugar, el estudio de este capítulo se centra en el planificador de tráfico, por lo que los resultados presentados corresponden a la latencia de la extracción de los paquetes almacenados en el conmutador *GMDS*. No se considera el retardo de un paquete desde que entra al sistema hasta que se almacena en memoria, puesto que, al tratarse de una arquitectura con colas a la salida, éste es prácticamente constante, como se explicó en la sección 4.3.2.1. En segundo lugar, y a tenor de los resultados obtenidos en la sección 4.3.2.2, la capacidad actual de almacenamiento del prototipo propuesto se torna insuficiente para realizar simulaciones con tráfico a ráfagas.

4.11.1 Tráfico uniforme

A continuación, se muestra y analiza el rendimiento del modelo de bajo nivel del algoritmo de planificación con tráfico uniforme.

4.11.1.1 Escenario 1. Test inicial

Los resultados de la simulación del modelo de bajo nivel en este escenario se muestran en la Figura 4.45. El punto de inflexión de la curva estaba situado en torno al 97% de la carga de tráfico en el modelo de alto nivel, presentado previamente en la Figura 4.15. En el modelo de bajo nivel se ha desplazado hacia la izquierda reduciéndose hasta el 96%, lo que representa una ligera pérdida de prestaciones debida a la inclusión de los retardos propios del conmutador *GMDS*. Además, la comparación de la caracterización de ambos modelos muestra que el retardo medio de las clases de tráfico en el modelo de bajo nivel es mayor que en el modelo de alto nivel. Este incremento varía entre 2 y 7 paquetes en función de la carga de tráfico con un valor medio de 3,25 paquetes, siempre que no se supere el punto de inflexión. Estos valores tan próximos indican que los resultados obtenidos a partir del modelo de alto nivel son estimaciones fiables.

Si el análisis se centra únicamente en la caracterización del modelo de bajo nivel, la desviación típica de los retardos de las diferentes clases de tráfico alcanza un reducido valor de 1,35 paquetes en el punto de inflexión, lo que

representa que se proporciona el mismo servicio a todas las clases de tráfico. Independientemente de los retardos y de la carga de tráfico, el planificador es capaz de mantener el ancho de banda asignado a cada clase de tráfico, como se observa en la Figura 4.46.

4.11.1.2 Escenario 2. Control del ancho de banda

En la Figura 4.47 se espera que la clase de tráfico con mayor ancho de banda asignado reciba el mejor servicio. Al mismo tiempo, no se concibe que las curvas de las clases de tráfico con diferentes anchos de banda se crucen, ya que no se ha configurado ninguna prioridad. Ambos efectos son indeseados en el comportamiento del algoritmo de planificación y, aunque en el modelo de alto nivel no se apreciaba tan claramente, este escenario es un caso de funcionamiento indeseado de la planificación debido a las características del algoritmo *RR* que selecciona las clases. El mecanismo de control de ancho de banda permanece inalterado, como se aprecia en la Figura 4.48, frente al comportamiento del retardo de las clases de tráfico.

4.11.1.3 Escenario 3. Control de retardo

Las Figuras 4.49 y 4.50 muestran que el planificador es capaz de introducir una diferenciación en el tiempo de atención de las clases de tráfico sin modificar el ancho de banda asignado a cada una. Sin embargo, a diferencia de la simulación en alto nivel de este escenario, se aprecia una ligera varianza en el retardo, lo que da lugar a un cruce de los retardos de clases de tráfico asignadas a diferentes prioridades con altas cargas de tráfico. Al igual que en los escenarios anteriores, se ha incrementado ligeramente el retardo de todas las clases de tráfico y se ha desplazado ligeramente el punto de inflexión del retardo, quedando situado, aproximadamente, en el 96% de la carga de tráfico.

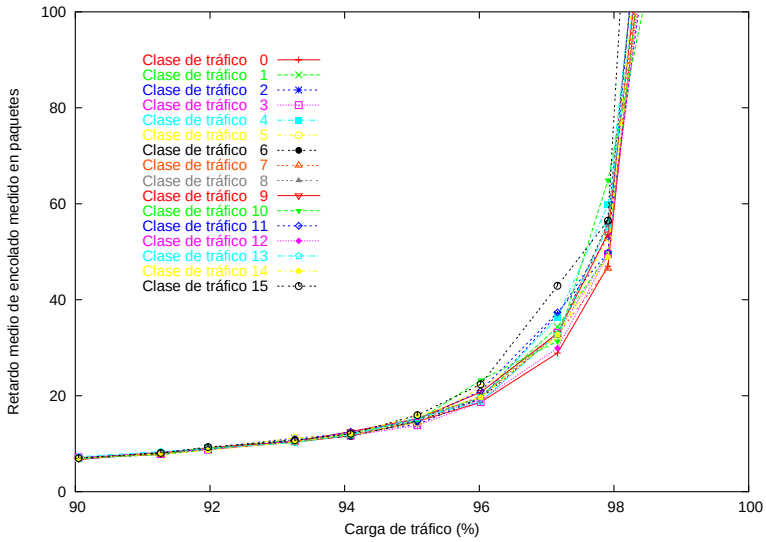


Figura 4.45: Retardo medio de encolado del modelo de bajo nivel con tráfico uniforme en el escenario 1

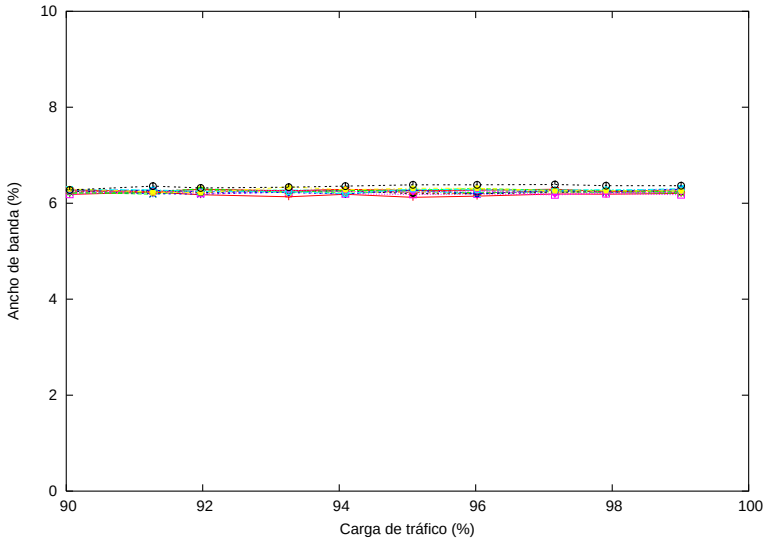


Figura 4.46: Distribución del ancho de banda del modelo de bajo nivel con tráfico uniforme en el escenario 1

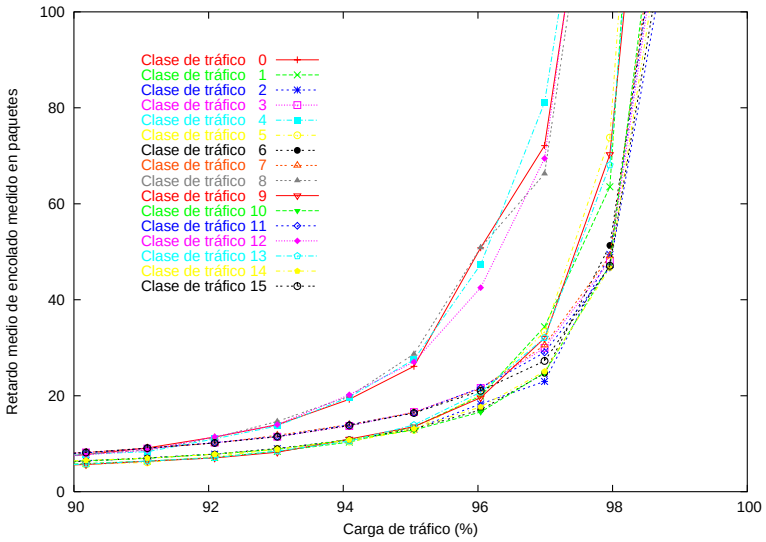


Figura 4.47: Retardo medio de encolado del modelo de bajo nivel con tráfico uniforme en el escenario 2

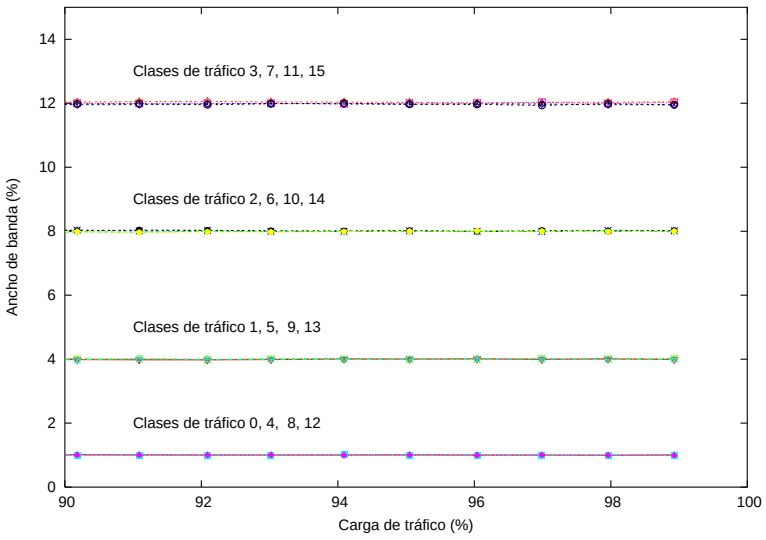


Figura 4.48: Distribución del ancho de banda del modelo de bajo nivel con tráfico uniforme en el escenario 2

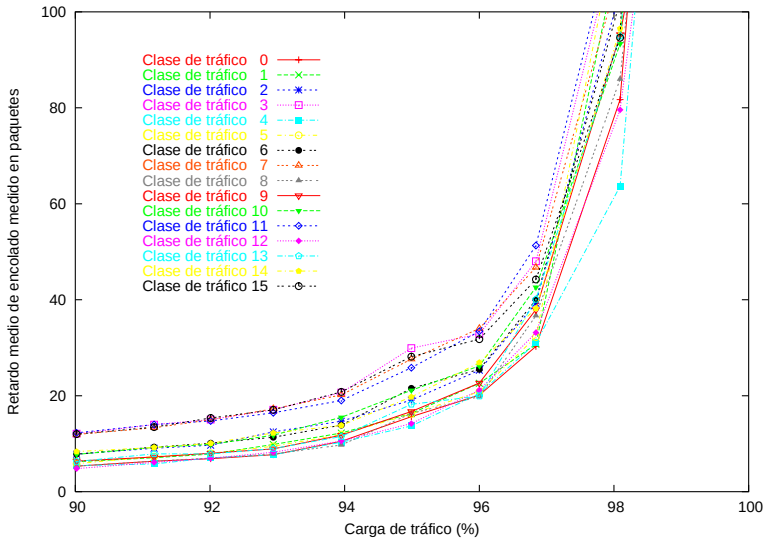


Figura 4.49: Retardo medio de encolado del modelo de bajo nivel con tráfico uniforme en el escenario 3

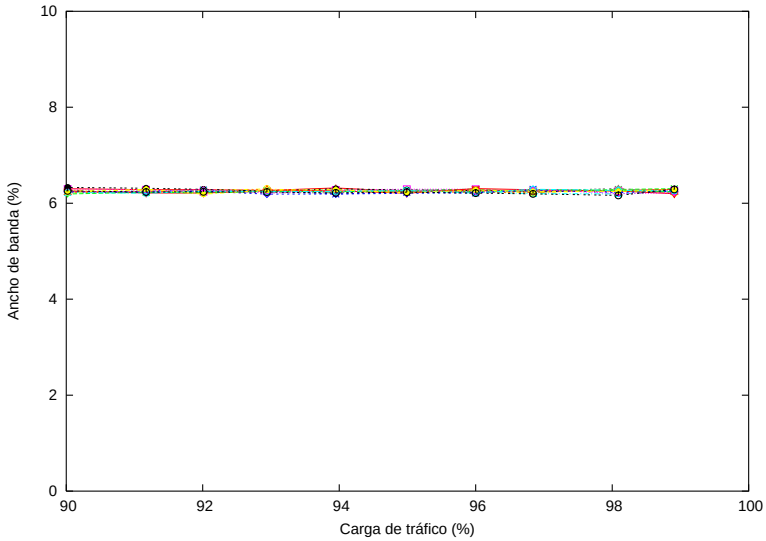


Figura 4.50: Distribución del ancho de banda del modelo de bajo nivel con tráfico uniforme en el escenario 3

4.11.1.4 Escenario 4. Control del ancho de banda y del retardo

En la Figura 4.51, se muestra el resultado de asignar la prioridad máxima las clases con mayor ancho de banda y la prioridad mínima a aquéllas con menor ancho de banda. Esta configuración conduce al incremento del retardo de todas las clases sin perjuicio del ancho de banda, como se observa en la Figura 4.52. El resultado es ligeramente peor que el obtenido a partir del modelo de alto nivel al incorporar los retardos propios de la implementación.

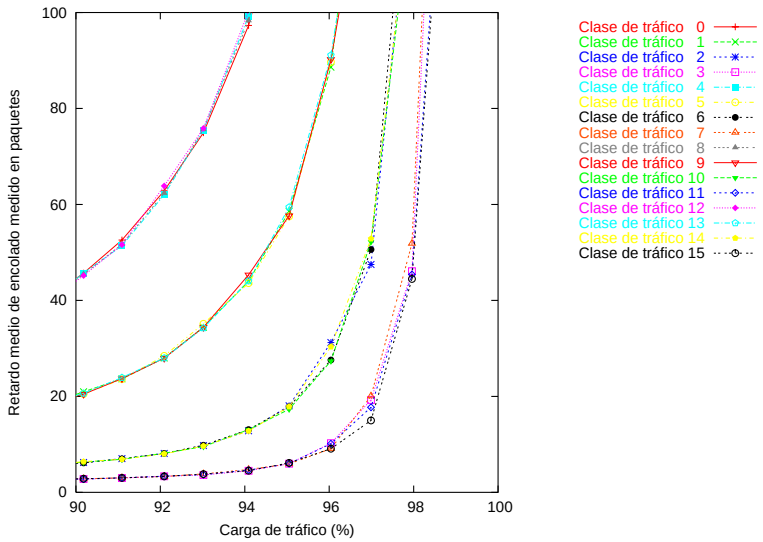


Figura 4.51: Retardo medio de encolado del modelo de bajo nivel con tráfico uniforme en el escenario 4

4.11.1.5 Escenario 5. Control del ancho de banda y del retardo

La Figura 4.53 muestra que el efecto de la asignación desequilibrada del ancho de banda se compensa con la asignación adecuada de las prioridades. Se puede apreciar una reducción del retardo de las clases de tráfico con mayor retardo y menor ancho de banda, si bien no se consigue compensar totalmente el efecto de los diferentes anchos de banda. La Figura 4.54 muestra la protección de los anchos de banda establecidos para cada clase de tráfico.

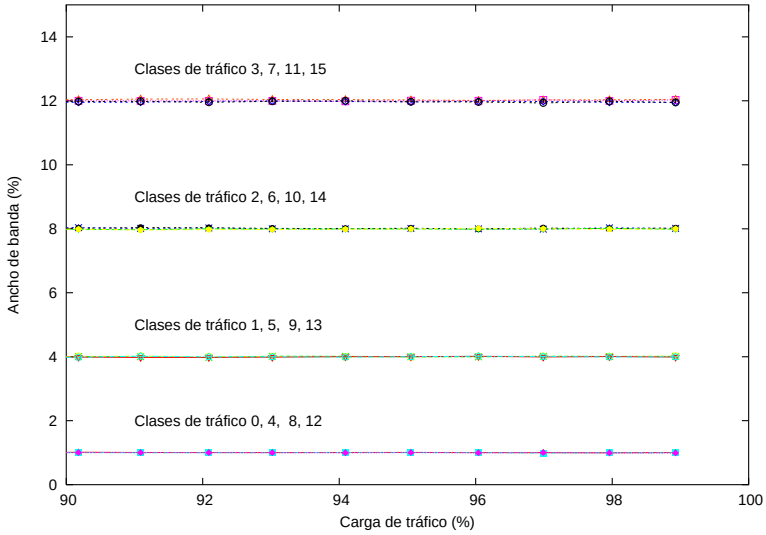


Figura 4.52: Distribución del ancho de banda del modelo de bajo nivel con tráfico uniforme en el escenario 4

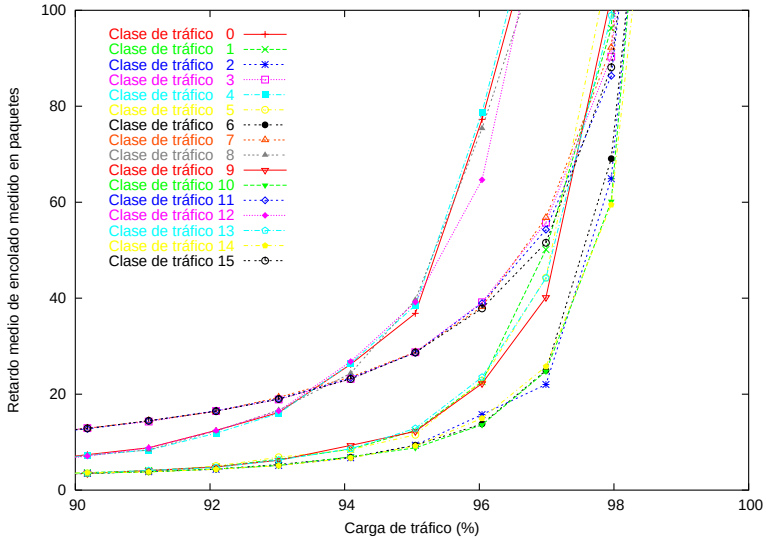


Figura 4.53: Retardo medio de encolado del modelo de bajo nivel con tráfico uniforme en el escenario 5

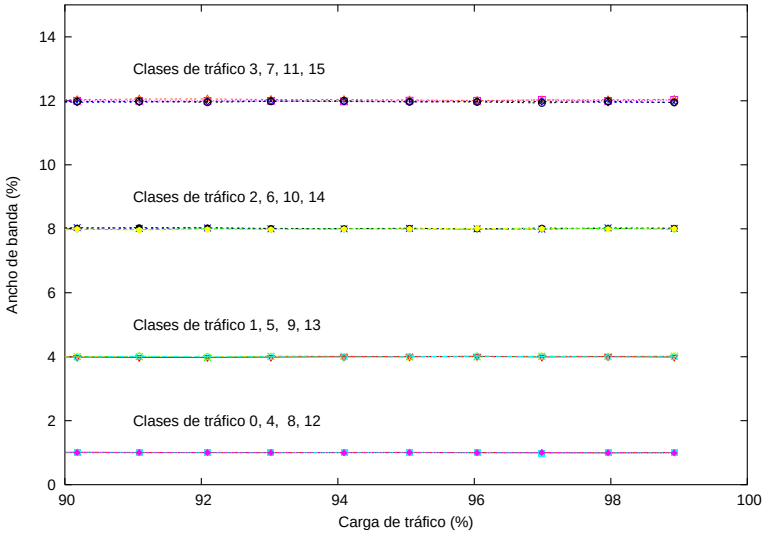


Figura 4.54: Distribución del ancho de banda del modelo de bajo nivel con tráfico uniforme en el escenario 5

4.12 Mejora del algoritmo de planificación

En la sección anterior se han estudiado las prestaciones del algoritmo de planificación propuesto sobre la base de varios escenarios de simulación. Entre todos ellos, el escenario 2 muestra un comportamiento anómalo al producirse cruces en las curvas de los retardos de las clases de tráfico con distinto peso. En consecuencia, se plantea un caso más sencillo con condiciones similares a fin de profundizar en el comportamiento del algoritmo de planificación propuesto ante esta configuración.

En el escenario 2 se configuran todas las clases de tráfico con la misma prioridad y se les asignan ancho de banda muy dispares. Concretamente, y en este ejemplo simplificado, se analizan 4 clases de tráfico con los siguientes pesos 4%, 16%, 32% y 48%, como se muestra en la Tabla 4.5. Además, los paquetes generados son de longitud fija, todas las clases de tráfico se atienden siguiendo una política *Round Robin*, y las colas siempre están activas. Con

esta configuración se logra clarificar el proceso de selección de las clases de tráfico, ya que el algoritmo de planificación siempre obtiene un paquete al elegir una cola.

Clase de tráfico	Ancho de banda	Paquetes
0	4%	↑
1	16%	↑
2	32%	↑
3	48%	↑

Tabla 4.5: Configuración del tráfico en el escenario 2

En la Figura 4.55 se muestran los paquetes enviados en un marco de tiempo con esta configuración. En la iteración 0 se envía un paquete de la clase de tráfico 0. Como consume todo su crédito, no se comprueba en las posteriores iteraciones hasta que comience un nuevo marco de tiempo y se reestablezca el crédito. El envío de los paquetes de la clase de tráfico 1, con una asignación del 16% del ancho de banda, se concentra al principio del marco de tiempo y termina agotando su crédito en 4 iteraciones. En el caso de las restantes clases de tráfico sucede que, al final del marco de tiempo, sólo la clase de tráfico con mayor ancho de banda dispone de crédito y puede generar una ráfaga de 5 paquetes. En esta Tesis Doctoral se propone la utilización del algoritmo *Smoothed Round Robin* (*SRR*) [Guo04] en lugar del algoritmo *RR* en la selección de las clases de tráfico con el objetivo de eliminar o reducir estos efectos.

El algoritmo *Smoothed Round Robin* (*SRR*) propone una distribución de los paquetes basada en los pesos de las clases de tráfico con el objetivo de minimizar la generación de las ráfagas, puesto que maximiza la distancia entre el envío de paquetes de la misma clase. Cabe destacar que este algoritmo utiliza más recursos *hardware* que una política *RR* tradicional y que dichos recursos dependen de la asignación de los pesos. Sin embargo, cuando la configuración externa asigna el mismo peso a las clases de tráfico, se comporta

igual que un algoritmo *RR* sin reducir su coste *hardware*. En este caso no se justifica la elección del algoritmo *SRR*. A continuación, se describe detalladamente el funcionamiento de este algoritmo y se profundiza en las situaciones en la cual la política *SRR* es más idónea para la planificación del tráfico.

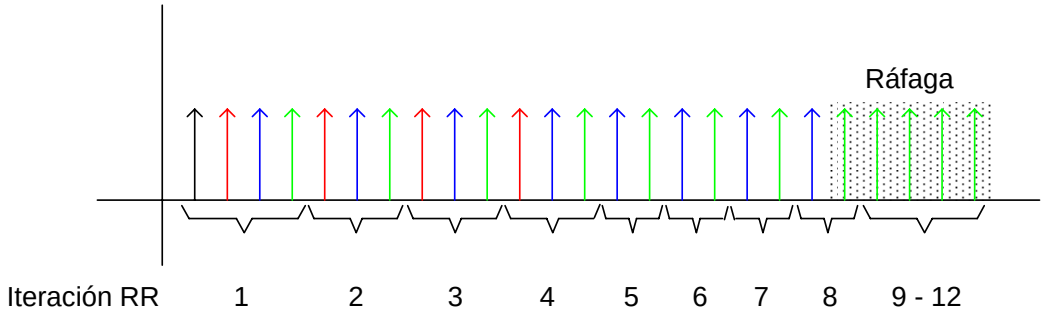


Figura 4.55: Selección de las clases de tráfico en el planificador basado en *RR*

4.12.1 Descripción

El algoritmo *Smoothed Round Robin* (*SRR*) [Guo04] se basa en una secuencia para dispersar las consultas a las clases de tráfico en función de sus pesos (*Weight Spread Sequence - WSS*) y una matriz que contiene los pesos de las clases (*Weight Matrix - WM*).

4.12.1.1 Secuencia de dispersión de los pesos

La secuencia de dispersión de los pesos *WSS* se define recursivamente de la siguiente forma:

1. El valor inicial de *WSS* es $S^1 = 1$.

2. El valor *k-ésimo* de *WSS* es,

$$S^k = a_i = S^{k-1},\ k,\ S^{k-1},$$

donde $1 \leq i \leq 2^k - 1$ para $k > 1$, siendo *k* el orden de la secuencia *WSS*.

4.12.1.2 Matriz de pesos

Se asume que el conjunto de pesos es $1, 2, 3, 4, 5, \ldots, 2^k - 1$. Eligiendo el valor apropiado de *k*, se genera la secuencia de dispersión y se ajustan las tasas de transferencia de los flujos al conjunto de pesos. El peso de cada flujo se puede codificar como

$$w_f = \sum_{n=0}^{k-1} a_{f,n} 2^n,$$

donde $a_{f,n} \in \{0, 1\}$.

Los coeficientes binarios $a_{f,n}$ forman un vector del peso que se define como

$$WV_f \ = \ \{a_{f,(k-1)},\ a_{f,(k-2)}, \ldots,\ a_{f,0}\}$$

La matriz de pesos correspondiente a los flujos $f_1,\ f_2, \ldots,\ f_N$ se define como

$$WM \ = \ \begin{bmatrix} WV_1 \\ WV_2 \\ \vdots \\ WV_N \end{bmatrix} \ = \ \begin{bmatrix} a_{1,(k-1)} & a_{1,(k-2)} & \ldots & a_{1,0} \\ a_{2,(k-1)} & a_{2,(k-2)} & \ldots & a_{2,0} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,(k-1)} & a_{N,(k-2)} & \ldots & a_{N,0} \end{bmatrix}$$

4.12.2 Funcionamiento

Se combina la secuencia *WSS* con la matriz *WM* para formar el orden de consulta de los diferentes flujos de datos en el planificador *Smoothed Round Robin*. La idea básica del algoritmo *SRR* consiste en recorrer la matriz *WM* en función del vector *WSS*. Cuando el elemento actual es i en la secuencia *WSS*, se selecciona la columna $k - i$ de la matriz *WM*. Por cada ocurrencia de $a_{f,(k-i)} = 1$ en esta columna, se consulta el flujo de datos correspondiente.

A continuación se procede a planificar el tráfico según el algoritmo *SRR* con la configuración presentada en la Tabla 4.5. Para $k = 6$ se obtiene que:

WSS = 1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,6,1,2,1,3,1,
2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1

La matriz de pesos correspondiente *WM* es:

$$WM = \begin{bmatrix} WV_1 \\ WV_2 \\ WV_3 \\ WV_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

En la Tabla 4.6 se muestra el funcionamiento del algoritmo *SRR* para las primeras posiciones del vector *WSS*. Con el fin de facilitar su comprensión se añade la posición en el vector *WSS*, el valor de su elemento, la columna a revisar de la matriz *WM* y, finalmente, las clases de tráfico que se consultan.

Posición	0	1	2	3	4
Valor del elemento i	1	2	1	3	1
Columna $k - i$	5	4	5	3	5
Clase de tráfico seleccionada	2 3	1 3	2 3		2 3

Tabla 4.6: Selección de las clases de tráfico

Si se aplica esta política en el algoritmo de planificación propuesto con el ejemplo anterior, se obtienen los resultados mostrados en la Figura 4.56. Se aprecia que se han eliminado todas las ráfagas de tráfico y las clases se seleccionan en todo el marco de tiempo intentando maximizar el espacio entre visitas.

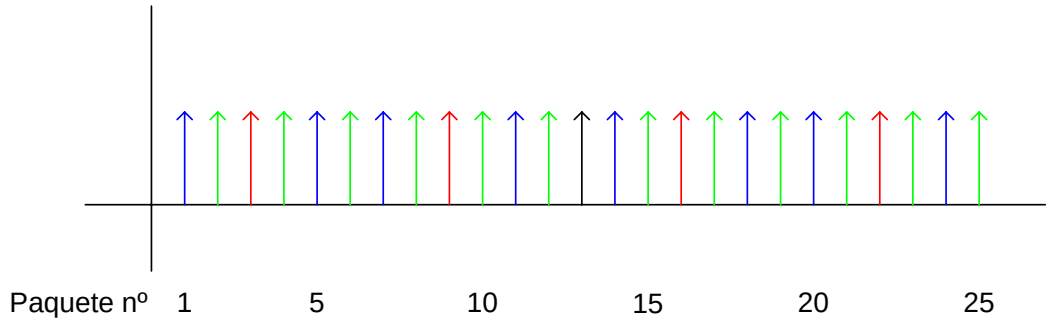


Figura 4.56: Selección de las clases de tráfico en el planificador basado en *SRR*

4.12.3 Modelo de alto nivel

Las condiciones de simulación propuestas en este apartado corresponden al escenario 2 analizado previamente en este capítulo, es decir, 16 clases de tráfico, sin prioridades y anchos de banda del 1%, 4%, 8% y 12% en grupos de 4 clases. Los resultados presentados en la Figura 4.57 muestran la eliminación de los cruces entre las clases de tráfico, así como, un menor retardo de las clases de tráfico con mayor ancho de banda asignado. Además, se sigue manteniendo el alto *throughput* característico de la implementación original.

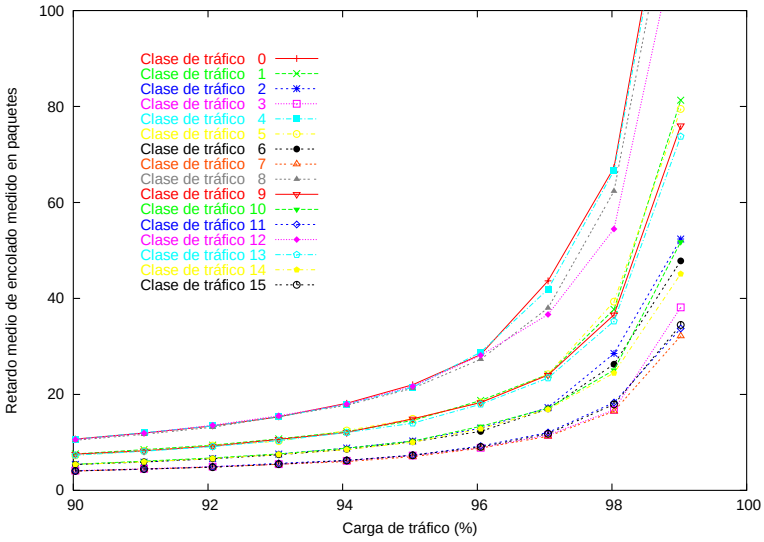


Figura 4.57: Retardo medio de encolado del modelo de alto nivel con tráfico uniforme en el escenario 2. *SRR*

4.12.4 Efecto de la longitud del marco de tiempo

El algoritmo *SRR* consulta las clases de tráfico según la secuencia generada sobre la base de los pesos de cada una. En consecuencia, si este algoritmo dispone de suficiente crédito para elegir entre todas las clases de tráfico al comienzo del marco de tiempo, proporciona las mismas prestaciones independientemente del factor multiplicador de los valores de los créditos. No se

producen más peticiones a una clase de tráfico porque ésta disponga de más crédito. Esta conclusión se muestra gráficamente en las Figuras 4.57, 4.60 y 4.61, donde el factor de multiplicación toma los valores 1, 10 y 100, respectivamente. Internamente, al comparar su funcionamiento con una política *RR*, como las peticiones de las clases de tráfico se producen sobre la base del peso, no se dedica la misma atención a todas ellas y, por lo tanto, las clases con mayor ancho de banda reciben más atención, es decir, mejor servicio. En el caso del algoritmo *RR*, todas las clases reciben la misma atención, puesto que se solicitan los paquetes en orden consecutivo. Este aspecto conlleva que un algoritmo *RR* proporcione mejor servicio a la clase de tráfico con menor peso frente a un algoritmo *SRR*, donde la clase de tráfico de mayor peso obtiene el mejor servicio al producirse más peticiones.

En el caso de un factor de multiplicación menor que la unidad, como se presenta en las Figuras 4.58 y 4.59, el algoritmo *SRR* no puede considerar todas las clases de tráfico en el proceso de planificación y, por lo tanto, no puede prestar un servicio diferenciado entre ellas.

4.12.5 Implementación

La implementación del algoritmo propuesto se ha llevado a cabo en el lenguaje *Verilog HDL* a nivel *RTL* y la síntesis en una *FPGA Spartan-3 1000* (*xc3s1000-4FT256*) con la ayuda del *software ISE* versión 8.2.02i. Los resultados se muestran en la Tabla 4.7 y corresponden a un planificador configurado con 16 clases de tráfico, sin prioridades y con anchos de banda del 1%, 4%, 8% y 12% en grupos de 4 clases. Hay que remarcar que esta configuración es el peor caso desde el punto de vista *hardware*, ya que genera la secuencia de clases de tráfico más larga posible. Además, esta implementación consume más recursos que la versión original basada en la política *RR*, presentada en la Tabla 4.4. Aproximadamente, el incremento en el número de puertas equivalentes es de un 25%. Este valor se puede asociar con el coste *hardware* individual de la mejora proporcionada por el algoritmo *SRR*.

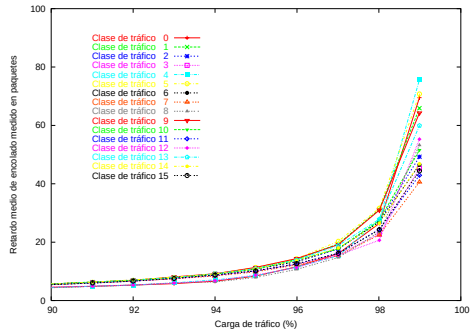


Figura 4.58: *SRR*. Marco de tiempo / 100

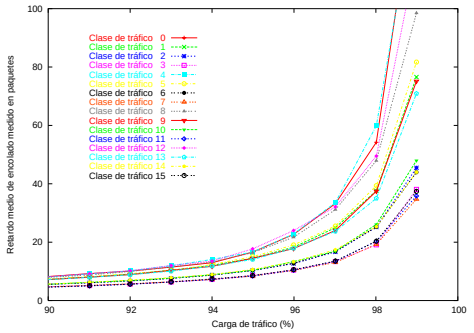


Figura 4.59: *SRR*. Marco de tiempo / 10

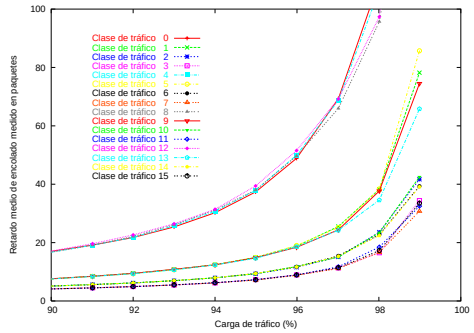


Figura 4.60: *SRR*. Marco de tiempo $\times$ 10

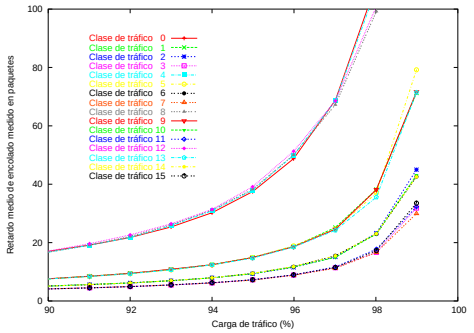


Figura 4.61: *SRR*. Marco de tiempo $\times$ 100

Lógica utilizada	Usado	Disponible	Utilizado
Número de *Slices*	5254	7860	68%
Número de *Slices Flip Flops*	1595	15360	10%
Número de *LUTs*	8050	15360	52%
Número de *Block RAMs*	2	24	8%
Número de *MULT18X18s*	16	24	66%
Número de *GCLKs*	1	8	12%
Número de puertas equivalentes	271240		

Tabla 4.7: Resultados de la síntesis del algoritmo de planificación basado en política *SRR*

4.12.6 Modelo de bajo nivel

La Figura 4.62 muestra los resultados de la caracterización de la implementación del algoritmo de planificación basado en *SRR* en *Verilog HDL* a nivel *RTL* del segundo escenario. Como se espera, los resultados concuerdan con la simulación de alto nivel y los retardos medios de las clases de tráfico no presentan efectos indeseados. No obstante, se reduce ligeramente el *throughput* máximo de salida, del 98% al 96%, si se comparan las prestaciones del modelo de alto nivel con el modelo de bajo nivel. Los retardos son proporcionales a los anchos de banda asignados. Se debe resaltar que el incremento del retardo de las clases de tráfico depende del ancho de banda. De esta manera, el valor medio de la diferencia del retardo de las clases de tráfico entre ambos modelos alcanza 30, 10, 7 y 6 paquetes en función de los pesos asignados, 1%, 4%, 8% y 12%, respectivamente. Los valores resultantes de la desviación típica son 22, 7, 4 y 2 paquetes, mostrando la misma dependencia del peso. En conclusión, entre menor sea el peso de una clase de tráfico, mayor será su retardo y la diferencia respecto al modelo de alto nivel. Sin embargo, en todo momento la tendencia de las curvas de ambos modelos es similar.

4.12.7 Discusión del algoritmo *SRR*

El algoritmo *SRR* mejora las prestaciones del conmutador *GMDS* a costa de introducir cierta lógica sobre la versión original basada en el algoritmo *RR*. Sin embargo, sólo en algunos casos de funcionamiento se justifica su coste. Su principio de funcionamiento se basa en flujos de información con diferentes pesos y, sólo en este caso, se recomienda su utilización. En consecuencia, en los escenarios de simulación 1 y 3 con igual ancho de banda, un algoritmo *RR* es suficiente. En el caso de los escenarios 4 y 5, el mecanismo de prioridad enmascara los beneficios del algoritmo *SRR*. Al elegirse una prioridad se selecciona una clase de tráfico entre las asignadas a dicha prioridad. Si todas las clases de tráfico de esa prioridad concreta tienen el mismo peso, como son los casos mencionados, tampoco se necesita considerar el algoritmo *SRR*.

En resumen, conviene aplicar la mejora del algoritmo *SRR* solamente si las clases de tráfico dentro de una misma prioridad tienen asignadas diferentes anchos de banda. En ese caso, el beneficio obtenido en la salida del conmutador resulta significativo frente al algoritmo *RR* al minimizar la generación de ráfagas.

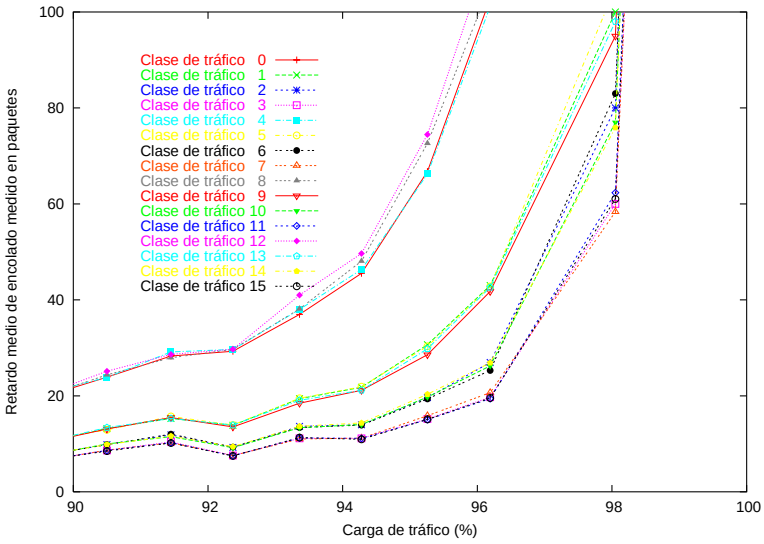


Figura 4.62: Retardo medio de encolado del modelo de bajo nivel con tráfico uniforme en el escenario 2. *SRR*

Entre las desventajas de este algoritmo conviene citar su coste *hardware* y la complejidad de su configuración. Si se modifica el número de clases de tráfico o el peso de una de ellas, se cambia el criterio de selección de las clases de tráfico, debiendo configurar nuevamente el planificador.

4.13 Comparación de prestaciones

El algoritmo de planificación propuesto en esta Tesis Doctoral se caracteriza por una técnica de almacenamiento con múltiples colas a la salida, soportar diferentes calidades de servicio utilizando pesos para garantizar el ancho de banda y prioridades para ajustar los retardos, así como la conmutación de pa-

quetes de longitud variable y el uso de una política conservadora. Todos estos aspectos combinados hacen difícil encontrar un algoritmo que permita su comparación directa en condiciones que contemplen todas estas características, si bien se pueden analizar y comparar independientemente con diferentes propuestas.

La arquitectura basada en colas a la salida proporciona el mejor compromiso entre el retardo y el *throughput* [HK88]. Además, el *backplane* serie *multidrop* permite la transmisión de paquetes de longitud variable sin mecanismos de segmentación ni reensamblado, y tráfico multidestino, lo que representa una alta eficiencia en las comunicaciones, puesto que no existe información de relleno ni la necesidad de replicar los paquetes.

Sin embargo, la transmisión de paquetes de longitud variable implica que, si se desean obtener las máximas prestaciones del conmutador, el planificador también debe soportar este tipo de paquetes. Al combinarse el soporte de calidad de servicio con el de paquetes de longitud variable, las propuestas previas, como *WRR* para paquetes de longitud fija, o *DRR* para paquetes de longitud variable, resultan insuficientes para garantizar el ancho de banda. En consecuencia, se han tomado propuestas, como por ejemplo, los algoritmos *DS* [HYG06] o *GBSBF-LBF* [OMW06] para arquitecturas *CICQ* o el algoritmo *DDS* [YLZ03] para arquitecturas con colas a la entrada, como puntos de partida. Sin embargo, estas propuestas aseguran un mínimo de ancho de banda. El algoritmo propuesto en la presente Tesis Doctoral garantiza el reparto del ancho de banda siguiendo una filosofía basada en créditos con el fin de mantener un compromiso entre las prestaciones y su complejidad.

La prioridad es el concepto utilizado en esta Tesis Doctoral a fin de determinar el orden en el que se atienden las clases de tráfico, empezando por las clases asignadas a la prioridad máxima y terminando en aquéllas asignadas a la prioridad mínima. En función de la prioridad de los paquetes, éstos son planificados antes o después, por lo que se reduce o incrementa el retardo en el conmutador. Con el fin de implementar un mecanismo de prioridades de forma eficiente, se ha elegido un algoritmo con decisiones jerárquicas tomando como

referencias a *PWDRR-CPRR* [YQW06], *HDS* [YWL+04] y *PQWRR* [MMW01] basados en arquitecturas *CICQ*, colas a la entrada y colas a la salida, respectivamente.

La justicia y la generación de ráfagas han sido dos aspectos de especial importancia al basarse el algoritmo propuesto en la filosofía *Round Robin*. La solución propuesta en [Guo04], denominada *SRR*, ha sustituido a la política *RR* en determinados escenarios con el fin de evitar la tendencia a generar ráfagas, típica de los algoritmos *Round Robin*. Además, el algoritmo *SRR* ha sido combinado satisfactoriamente con los mecanismos de control de ancho de banda y prioridades.

Un nuevo algoritmo presentado por el mismo autor en [Guo07], y basado en el algoritmo *SRR*, proporciona garantías de justicia, retardo y ancho de banda combinando un algoritmo *Round Robin* recursivo que verifica un árbol de flujos y se ayuda del algoritmo *SRR* para evitar la generación de ráfagas. No obstante, su coste *hardware* es muy elevado y además requiere reconfiguración dinámica de los flujos activos.

Por último, se pueden comparar gráficamente las prestaciones de los algoritmos y las arquitecturas comentadas a lo largo de este apartado. No obstante, sólo el hecho de que la arquitectura propuesta sea con colas a la salida mejora las prestaciones de los algoritmos de arquitecturas *CICQ* y notablemente los algoritmos de las arquitecturas con colas a la entrada.

4.14 Resumen

En el presente capítulo se ha propuesto un algoritmo de planificación caracterizado por su eficiencia, protección, flexibilidad y simplicidad. Dicho algoritmo soporta diferentes calidades de servicio diferenciando los flujos de datos en clases de tráfico. A cada una de las clases de tráfico se le asigna un peso y una prioridad con el objetivo de garantizar el ancho de banda y controlar el retardo.

La selección de los paquetes se realiza siguiendo varios pasos de forma jerárquica. En primer lugar se selecciona la máxima prioridad. Posteriormente, se elige una clase de tráfico mediante un algoritmo *RR* entre aquéllas que pertenezcan a la prioridad seleccionada. Por último, se escoge una fuente de la clase de tráfico elegida previamente que haya enviado la menor cantidad de tráfico. Como el algoritmo de planificación soporta paquetes de longitud variable, se incluye un mecanismo basado en créditos que garantiza la cantidad de información enviada en cada marco de tiempo a fin de controlar el ancho de banda y garantizar la protección entre las clases de tráfico.

Este algoritmo ha sido simulado a partir de un modelo de alto nivel y de una implementación en el lenguaje *Verilog HDL* a nivel *RTL*. Se han planteado cinco escenarios de simulación para estudiar las prestaciones modificando el ancho de banda y las prioridades asignadas a cada clase bajo las distribuciones de tráfico uniforme y tráfico a ráfagas.

Los estudios de la longitud del marco de tiempo y la justicia muestran cierta tendencia del algoritmo de planificación a generar ráfagas de tráfico en ciertos escenarios de simulación. En consecuencia, se propone una mejora del algoritmo original consistente en la sustitución del algoritmo *RR* encargado de la selección de la clase de tráfico, por el algoritmo *SRR* [Guo04], que reduce la probabilidad de generación de ráfagas.

El coste *hardware* de la versión original alcanza, aproximadamente, 220000 puertas equivalentes, el 22% si se compara con el coste *hardware* de la tarjeta de línea. La mejora funcional obtenida con el algoritmo *SRR* incrementa en un 5% el número de puertas equivalentes, hasta un total de 270000. La implementación de ambas versiones se ha llevado a cabo en una *FPGA Spartan-3 1000*, independiente de la tarjeta de línea, para dotar de flexibilidad a la arquitectura de conmutación.

Las prestaciones del algoritmo de planificación son comparables a propuestas como *PQWRR* [MMW01], *PWDRR-CPRR* [YQW06] o *DS* [HYG06]. No obstante hay que resaltar que el algoritmo propuesto combina más características que las referencias previas, se garantizan los parámetros de calidad

de servicio en lugar de un servicio mínimo y se considera la propuesta a nivel funcional y arquitectural para su implementación.

Capítulo 5

Conclusiones y líneas futuras

En este capítulo se presentan las conclusiones extraídas a partir del trabajo realizado y las líneas futuras que se derivan de esta investigación.

5.1 Conclusiones

En la presente Tesis Doctoral se ha abordado la planificación del tráfico con calidad de servicio y paquetes de longitud variable en una arquitectura de conmutación basada en colas a la salida. Este objetivo se ha alcanzado satisfactoriamente gracias a que se han conseguido los diferentes hitos que se planteaban en los objetivos de esta Tesis Doctoral.

El punto de partida de esta investigación ha consistido en el estudio de diversas arquitecturas de conmutación y técnicas de almacenamiento con el objetivo de profundizar en sus características, haciendo especial hincapié en las prestaciones de las diferentes topologías. A raíz de este estudio, se ha concluido que las arquitecturas con colas a la salida ofrecen unas prestaciones superiores al resto de las arquitecturas consideradas, ya que proporcionan el mejor compromiso entre *throughput* y retardo. Sin embargo, la velocidad de la memoria de almacenamiento de los puertos de salida limita su escalabilidad, por lo que se han propuesto arquitecturas, como el conmutador *Knockout*, que reducen la velocidad de transferencia máxima de la memoria introduciendo pérdidas de paquetes. En la presente Tesis Doctoral se han tenido en cuenta estos aspectos durante la concepción de la arquitectura propuesta.

Las arquitecturas de conmutación deben garantizar diferentes calidades de servicio en el entorno actual de múltiples aplicaciones. Por ello, en el estado del arte figura el estudio de varios modelos que tienen capacidad para soportar calidad de servicio. Entre ellos destaca el modelo de Servicios Diferenciados, ya que proporciona un tratamiento preferencial por nodo sin limitar la escalabilidad ni incrementar excesivamente el coste *hardware* de las arquitecturas de conmutación.

Otro aspecto que debe ser considerado es la conmutación de los paquetes de longitud variable. Actualmente en la mayoría de los sistemas de conmutación se requieren mecanismos de segmentación y reensamblado con el objetivo de convertir los paquetes de longitud variable en células de tamaño fijo y facilitar su transferencia a través de la matriz de conmutación. Estos mecanismos añaden memorias de almacenamiento en los puertos de entrada y salida, y aceleración interna en la matriz de conmutación a fin de compensar el *overhead* añadido en la segmentación. Por lo tanto, resulta de gran interés que la arquitectura de conmutación propuesta disponga de un mecanismo más simple y eficiente que el que representan los mecanismos de segmentación y reensamblado.

Además, en el estado del arte de esta Tesis Doctoral se ha incluido una revisión de los algoritmos de planificación y de varias arquitecturas de conmutación con soporte de calidad de servicio que proporciona una visión del estado actual de la investigación y su desarrollo comercial.

Sobre la base de estos antecedentes, se exponen, a modo de resumen, las contribuciones más significativas realizadas en esta Tesis Doctoral:

- La primera aportación de esta Tesis Doctoral consiste en la arquitectura del conmutador *GMDS* basada en la técnica de almacenamiento con colas a la salida. Las características más destacables de esta arquitectura son la técnica de almacenamiento con colas a la salida sin pérdida de paquetes, la eliminación de la matriz de conmutación gracias a la utilización de un *backplane* serie *multidrop*, el soporte de calidad de servicio basado en el modelo de Servicios Diferenciados, la conmutación de

paquetes de longitud variable sin mecanismos de segmentación ni reensamblado, la transferencia inherente de tráfico *multicast* y la gestión del control de flujo extremo-a-extremo.

- Desde un punto de vista arquitectural, el conmutador *GMDS* se compone de las tarjetas de línea, formadas a su vez por los módulos *Ingress* y *Egress*. La mayor parte de la lógica se concentra en este último módulo, que implementa una estructura con múltiples colas a la salida capaz de almacenar los paquetes en función de su clase de tráfico y origen, facilitando así el soporte de calidad de servicio al algoritmo de planificación. Además, cada puerto de salida dispone de una línea dedicada que le permite la sincronización individual y la transmisión de los paquetes de longitud variable.

- Las características más relevantes del *backplane* serie *multidrop* son la eliminación de la matriz de conmutación para interconectar las tarjetas de línea, y la comunicación directa y en paralelo entre un puerto de entrada y todos los puertos de salida, lo que representa una comunicación *multicast* con coste *unicast*. Los paquetes se transfieren entre los puertos a través el *backplane* con un formato propuesto en la presente Tesis Doctoral denominado *multitrama*. Las principales características de esta estructura de datos son la transmisión de un número indeterminado de paquetes de longitud variable en un formato de longitud fija, en el que se inserta periódicamente información de control de flujo del conmutador. Esta información es especialmente relevante, ya que permite un control por niveles de la velocidad de transferencia y un control de flujo extremo-a-extremo por clase y fuente, por clase de tráfico, y global.

- Con el objetivo de caracterizar el conmutador *GMDS*, se ha modelado el sistema a alto nivel a fin de obtener una estimación de las prestaciones. Los resultados obtenidos a partir del modelo de alto nivel se han comparado con otras referencias mostrando un mejor rendimiento que una arquitectura *CICQ* y muy cercano a una arquitectura ideal *OQ*.

- Posteriormente, se ha desarrollado el modelo de bajo nivel del conmutador *GMDS*. A partir de esta descripción, se ha propuesto un demostrador con el fin de validar la arquitectura del conmutador *GMDS*. Sus principales características son la utilización de dispositivos reconfigurables con el fin de implementar las tarjetas de línea, memoria *RAM* externa para almacenar los paquetes, y un *backplane* serie *multidrop* de interconexión de tamaño 34 cm $\times$ 52 cm. Además, la descripción de bajo nivel permite la síntesis e implementación de la arquitectura.

- El algoritmo de planificación aportado en la presente Tesis Doctoral soporta diferentes calidades de servicio sobre la base del modelo de Servicios Diferenciados. En este algoritmo de planificación se combinan aspectos como el *peso*, a fin de garantizar el ancho de banda, la *prioridad*, con el objetivo de controlar el retardo, y los *créditos*, para gestionar los paquetes de longitud variable. Sin embargo, un estudio en profundidad del comportamiento del algoritmo revela cierta tendencia a generar ráfagas de tráfico bajo determinadas condiciones. Por este motivo, se introduce una mejora basada en el algoritmo *Smoothed Round Robin*, capaz de minimizar este efecto, ya que maximiza la distancia entre las transferencias de una misma clase de tráfico. Como consecuencia, ante este escenario se propone la posibilidad de sustituir la política *Round Robin* original por el algoritmo de planificación *Smoothed Round Robin*. Ambas contribuciones muestran una gran flexibilidad, puesto que permiten la configuración del ancho de banda y las prioridades, y requieren, relativamente, pocos recursos *hardware*.

- Al igual que en el caso del conmutador *GMDS*, se han modelado el algoritmo original, basado en la política *Round Robin*, y la versión modificada, basada en la política *Smoothed Round Robin*, a alto y bajo nivel. El modelo de alto nivel del algoritmo original ofrece prestaciones similares a las obtenidas por otros algoritmos existentes, los cuales, sin embargo, no aúnan todas las características de prioridad, peso y paquetes

de longitud variable que se consideran en los algoritmos propuestos. Por otro lado, se demuestra que la modificación introducida en el algoritmo original, apropiada bajo ciertos escenarios de funcionamiento, evita la generación de ráfagas de tráfico. Además, en las simulaciones de ambos se observa que son capaces de proporcionar servicios diferenciados a las clases de tráfico y mantener el ancho de banda asignado a cada una. Por último, gracias al modelo de bajo de nivel, se han implementado en un dispositivo programable independiente de la tarjeta de línea a fin de dotar de flexibilidad el diseño de la arquitectura.

5.2 Líneas futuras

Las líneas de investigación que han quedado abiertas tras la realización de la presente Tesis Doctoral son diversas, siendo las principales las siguientes:

- El conmutador *GMDS* se ha validado gracias a la implementación de un demostrador compuesto por elementos reconfigurables. Este hecho ha motivado la elección de unos parámetros de funcionamiento restringidos por el *hardware* elegido. En consecuencia, una implementación en un dispositivo de aplicación específica con una tecnología estándar supondría eliminar estos límites y permitiría mejorar las prestaciones y reducir el *hardware* asociado.

- Debido a lo comentado en el punto anterior, en el planteamiento actual de la arquitectura del conmutación *GMDS*, el algoritmo de planificación se ha considerado como un elemento individual incluyéndose en una *FPGA* independiente en el demostrador. Esta concepción deja abierta la posibilidad de proponer diversos algoritmos siempre que sean capaces de tomar sus decisiones sobre la base de la información del nivel de ocupación de las colas. Por lo tanto, este sistema permite abrir una importante línea de investigación cuyo soporte de demostración es la arquitectura propuesta en esta Tesis Doctoral.

- El tráfico a ráfagas ha sido un aspecto muy exigente en la concepción propuesta del conmutador *GMDS*. Por sí solo, esta distribución de tráfico reduce las prestaciones de la arquitectura de conmutación, por lo que, además de la propuesta de aumentar la capacidad de la memoria de almacenamiento, se deberían considerar actuaciones especiales, como por ejemplo, transferir parte de la información de control de flujo al planificador para que sea capaz de hacer frente a las ráfagas de tráfico, o introducir memoria asignada dinámicamente a las colas de almacenamiento.

- Con respecto a la escalabilidad del *backplane* serie *multidrop*, uno de los factores limitantes en la tasa de transferencia se encuentra en el sustrato *FR4*. Por lo tanto, el estudio e implementación de un *backplane* serie *multidrop* basado en divisores de potencia asimétricos con un sustrato de alta resistividad permitiría incrementar la tasa de transferencia del *backplane* y, en consecuencia, abordar la escalabilidad de este elemento clave del conmutador *GMDS*.

Bibliografía

[ADF04] H.Z. Abidin, N.M. Din y N. Fisal, "Provisioning QOS using DiffServ with hierarchical scheduling," *IEEE Region 10 Conference TENCON 2004*, vol. B2, pp. 597-600, noviembre 2004.

[AM95] R. Y. Awdeh y H. T. Mouftah, "Survey of ATM switch architectures," *Computer Networks and ISDN Systems*, vol. 27, num. 12, pp. 1567-1613, noviembre 1995.

[AMCC] Applied Micro Circuits Corporation. Disponible en www.amcc.com. Último acceso: 2 de noviembre del 2007.

[AMCC1] Aplied Micro Circuits Corporation. Product Brief PRS 80G, C48X and C192X Version 0.6. Disponible en www.amcc.com. Último acceso: 11 de mayo del 2008.

[AMCC2] Aplied Micro Circuits Corporation. Product Brief PRS Q-80G, C48X and C192X Version 0.61. Disponible en www.amcc.com. Último acceso: 11 de mayo del 2008.

[AN89] M.M. Ali y H.T. Nguyen, "A neural network implementation of an input acces scheme in a high-speed packet switch," *Proceedings of IEEE Global Telecommunications Conference* (GLOBECOM'89), vol. 2, pp. 1192-1196, noviembre 1989.

[AOS+93] T. E. Anderson, S.S. Owicki, J.B. Saxe y C.P. Thacker, "High-speed switch scheduling for local-area networks," *IEEE/ACM Transactions on Computer Systems* (TOCS), vol. 11, num. 4, pp. 319-352, 1993.

[ATE+06] R. Arteaga, F. Tobajas, R. Esper-Chaín y R. Sarmiento, "Variable Length Packet Scheduler Algorithm with QoS Support under Uniform Traffic," *Proceedings of XXI Conference on Design of Circuits and Integrated Systems* (DCIS'06), noviembre 2006.

[ATE+07] R. Arteaga, F. Tobajas, R. Esper-Chaín, M. A. Monzón, R. Regidor, V. De Armas y R. Sarmiento, "Variable Length Packet Scheduler Algorithm with QoS Support," *Proceedings of International Symposium on Microtechnologies for the New Millenium* (SPIE'07), vol. 6590, pp. 12-1 - 12-12, mayo 2007.

[ATE+08] R. Arteaga, F. Tobajas, R. Esper-Chaín, V. De Armas y R. Sarmiento, "GMDS: Hardware Implementation of Novel Real Output Queuing Switch," *Proceedings of Design, Automation & Test in Europe* (DATE'08), pp. 1450-1455, marzo 2008.

[Bat68] K.E. Batcher, "Sorting Networks and Their Applications", *Proceedings of the Spring Joint Computer Conference*, pp. 307-314, 1968.

[BBC+98] S. Blake, D, Black, M. Carlson, E, Davies, Z. Wang y W. Weiss, "An Architecture for differentiated services,", *RFC 2475*, *Internet Engineering Task Force* (IETF), diciembre 1998.

[BCS94] R. Braden, D. Clark y S. Shenker, "Integrated services in the Internet architecture: An overview, request for comments," *RFC 1633*, *Internet Engineering Task Force* (IETF), julio 1994.

[BD82] R. Bakka y M. Dieudonne, "Switching circuit for digital packet switching network," *United States Patent 4314367*, febrero 1982.

[BDE+04] H. Balakrishnan, S. Devadas, D. Ehlert y Arvind, "Rate guarantees and overload protection in input-queued switches," *Proceedings of IEEE Conference on Computer and Communications Societies* (INFOCOM'04), vol. 4, pp. 2185-2195, marzo 2004.

[Bou08] J.-Y. Le Boudec, "Rate adaptation, Congestion Control and Fairness: A Tutorial," Disponible en http://ica1www.epfl.ch/PS_files/LEB3132.pdf. Último acceso: 30 de julio del 2008.

[BZ96] J.C.R. Bennet y H. Zhang, "WF^2Q: worst case fair weighted fair queueing," *Proceedings of IEEE Conference on Computer and Communications Societies* (INFOCOM'96), vol. 1, pp. 120-128, marzo 1996.

[BZ97] J.C.R. Bennett y H. Zhang, "Hierarchical packet fair queueing algorithms," *IEEE/ACM Transactions on Networking*, vol. 5, num. 5, pp. 675-689, octubre 1997.

[CC95] H.J. Chao y B.S. Choe, "Design and analysis of a large-scale multicast output buffered ATM switch," *IEEE/ACM Transactions on Networking* (TON'95), vol. 3, num. 2, pp. 126-138, abril 1995.

[CF99] F.M. Chiussi y A. Francini, "Providing QoS guarantees in packet switches," *Proceedings of IEEE Global Telecommunications Conference* (GLOBECOM'99), vol. 2, pp. 1582-1590, diciembre 1999.

[CFF+97] T. Cheney, J. A. Fingerhut, M. Flucke y J. S. Turner, "Design of a gigabit ATM switch," *Proceedings of IEEE Conference on Computer Communications* (INFOCOM'97), vol. 1, pp. 2-11, abril 1997.

[CG02] H. J. Chao y X. Guo, Quality of Service Control in High-Speed Networks, John Wiley & Sons, 2002.

[CGM+99] S.-T. Chuang, A. Goel, N. McKeown y B. Prabhakar, "Matching output queuing input/output queued switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, num. 6, pp. 1030-1039, junio 1999.

[Cha00] H.J. Chao, "Saturn: a terabit packet switch using dual round-robin," *Proceedings of IEEE Global Telecommunications Confer-*

ence (GLOBECOM'00), vol. 1, pp. 487-495, noviembre-diciembre 2000.

[CK03] N. Chrysos y M. Katevenis, "Weighted fairness in buffered crossbar scheduling," *Workshop on High Performance Switching and Routing* (HPSR'03), pp. 17-22, junio 2003.

[CK04] N. Chrysos y M. Katevenis, "Multiple priorities in a two-lane buffered crossbar," *Proceedings of IEEE Global Telecommunications Conference* (GLOBECOM'04), vol. 2, pp. 1180-1186, noviembre-diciembre 2004.

[CL07] H. J. Chao y B. Liu, High Performance Switches and Routers, John Wiley & Sons, 2007.

[CLO01] H. J. Chao, C. H. Lam y E. Oki, Broadband packet switching Technology: A practical guide to ATM switches and IP routers, Wiley, 2001.

[CM04] L. Crawford y A. Marshall, "Fairness measurement for adaptive packet schedulers," *Electronics Letters*, vol. 40, num. 15, pp. 969-970, julio 2004.

[CM93] D.X. Chen y J.W. Mark, "SCOQ: a fast packet switch with shared concentration and output queueing," *IEEE/ACM Transactions on Networking* (TON'93), vol. 1, num. 1, pp. 142-151, febrero 1993.

[CMY+07] X. Chen, K. Makki, K. Yen y N. Pissinou, "A new network topology evolution generator based on traffic increase and distribution model," *Sixth International Conference on Networking* (ICN '07), pp. 56-61, abril 2007.

[Cross99] CrossStream Project (VSC870). Contrato de I+D. *Vitesse Semiconductor Corporation*. 1999. Investigador responsable: Dr. Roberto Sarmiento Rodríguez.

[CS] Cisco Systems. Disponible en www.cisco.com. Último acceso: 1 de junio del 2008.

[CS1] Cisco Systems. Data Sheet. Cisco Catalyst 6500 and 6500-E Series Switch. Disponible en www.cisco.com. Último acceso: 1 de junio del 2008.

[CS2] Cisco Systems. White paper. Series 6000 and 6500 architecture. Disponible en www.cisco.com. Último acceso: 1 de junio del 2008.

[CSIM] CSIM 19 Simulation Toolkit. Disponible en www.mesquite.com. Último acceso: 29 de noviembre del 2007.

[CSIX] *CSIX-L1: Common Switch Interface Specification - L1*. Mayo 2000.

[CYR+04] K. Christensen, K. Yoshigoe, A. Roginsky y N. Gunther, "Performance of packet-to-cell segmentation schemes in input buffered packet switches," *Proceedings of IEEE International Conference on Communications* (ICC'04), vol. 2, pp. 1097-1102, junio 2004.

[DBL+01] A.R. Djordjevi, R.M. Biljie, V.D. Likar-Smiljanic y T.K. Sarkar, "Wideband frequency-domain characterization of FR-4 and time-domain causality," *IEEE Transactions on Electromagnetic Compatibility*, vol. 43, num. 4, pp. 662-667, noviembre 2001.

[DCS88] M. Devault, J. Cochennec y M. Servel, "The Prelude ATD experiment: assessments and future prospects," *IEEE Journal on Selected Areas in Communications*, vol. 6, num. 9, pp. 1528-1537, diciembre 1988.

[DGK+04] G. Danilewicz, M. Glabowski, W. Kabacinski y J. Kleban, "Packet switch architecture with multiple output queueing," *Proceedings of IEEE Global Telecommunications Conference* (GLOBECOM'04), vol. 2, pp. 1192-1196, noviembre-diciembre 2004.

[DKS89] A. Demers, S. Keshav y S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Symposium Proceedings on Communications Architectures & Protocols* (SIGCOMM'89), pp. 1-12, septiembre 1989.

[DN] Dune Networks. Disponible en www.dunenetworks.com. Último acceso: 2 de noviembre del 2007.

[DN1] Dune Networks. Calculating Speedup in Switch Fabric Designs. Disponible en www.commsdesign.com. Último acceso: 1 de junio del 2008.

[DN2] Dune Networks. Product Brief Fabric Element FE200. Disponible en www.dunenetworks.com. Último acceso: 11 de mayo del 2008.

[DN3] Dune Networks. Product Brief Fabric Access Processor FAP10M. Disponible en www.dunenetworks.com. Último acceso: 11 de mayo del 2008.

[DN4] Dune Networks. Product Brief Fabric Access Processor FAP10/20V. Disponible en www.dunenetworks.com. Último acceso: 11 de mayo del 2008.

[DY02] V.L. Do y K.Y. Yun, "Packet latency optimization for VOQs in variable-length packet switches," *Workshop on High Performance Switching and Routing* (HPSR'02), pp. 77-82, mayo 2002.

[EDW93] J.B. Evans, E. Duron y Y. Wang, "Analysis and implementation of a priority knockout switch," *Proceedings of IEEE Conference on Computer and Communications Societies* (INFOCOM'93), vol. 3, pp. 1099-1106, marzo-abril 1993.

[EGT+03] R. Esper-Chaín, F. González, O. Tubío, F. Tobajas, V. De Armas y R. Sarmiento, "A Memory-Less Clock Domain Adaptation Unit," *Proceedings of the work in progress session held in connection with*

the 29th EUROMICRO Conference (EUROMICRO'03), septiembre 2003.

[EHY87] K. Eng, M. Hluchyj y Y.-S. Yeh, "A knockout switch for variable-length packets," *IEEE Journal on Selected Areas in Communications*, vol. 5, num. 9, pp. 1426-1435, diciembre 1987.

[EHY88] K.Y. Eng, M.G. Hluchyj y Y.S. Yeh, "Multicast and broadcast services in a knockout packet switch," *Proceedings of IEEE Conference on Computer and Communications Societies* (INFOCOM'88), pp. 29-34, marzo 1988.

[ETG+04] R. Esper-Chaín, F. Tobajas, F. González, R. Arteaga y Roberto Sarmiento, "A memoryless Clock Domain Adaptation Unit IP," *Proceedings of XIX Conference on Design of Circuits and Integrated Systems* (DCIS'04), pp. 784-789, noviembre 2004.

[ETT+05] R. Esper-Chaín, F. Tobajas, O. Tubío, R. Arteaga, V. de Armas y R. Sarmiento, "A gigabit multidrop serial backplane for high-speed digital systems based on asymmetrical power splitter," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 52, num. 1, pp. 5-9, enero 2005.

[FC02] A. Francini y F.M. Chiussi, "Providing QoS guarantees to unicast and multicast flows in multistage packet switches," *IEEE Journal on Selected Areas in Communications*, vol. 20, num. 8, pp. 1589-1601, octubre 2002.

[FN] Foundry Networks. Disponible en www.foundrynet.com. Último acceso: 1 de junio del 2008.

[FN1] Foundry Networks. Big Iron Rx Series Data Sheet. Disponible en www.foundrynet.com. Último acceso: 1 de junio del 2008.

[GBG91] A.K. Gupta, L.O. Barbosa y N.D. Georganas, "16x16 limited intermediate buffer switch module for ATM networks," *Proceedings*

of IEEE Global Telecommunications Conference (GLOBECOM'91), vol. 2, pp. 939-943, diciembre 1991.

[GG91] A.K. Gupta y N.D. Georganas, "Analysis of a packet switch with input and output buffers and speed constraints," *Proceedings of IEEE Conference on Computer Communications* (INFOCOM'91), vol. 2, pp. 694-700, abril 1991.

[Giga00] GigaStream Project (VSC872). Contrato de I+D. *Vitesse Semiconductor Corporation*. 2000. Investigador responsable: Dr. Roberto Sarmiento Rodríguez.

[GKS05] Y. Ganjali, A. Keshavarzian y D. Shah, "Cell Switching Versus Packet Switching in Input-Queued Switches," *IEEE/ACM Transactions on Networking* (TON'05), vol. 13, num. 4, pp. 782-789, agosto 2005.

[GL02] Y. Gong y B. Liu, "Performance evaluation of a parallel-poll virtual output queued switch with two priority levels," *IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions*, vol. 1, pp. 669-674, junio-julio 2002.

[Gol91] S. J. Golestani, "A framing strategy for congestion management," *IEEE Journal on Selected Areas in Communications*, vol. 9, num. 7, pp. 1064-1077, septiembre 1991.

[GP94] S. Gianatti y A. Pattavina, "Performance analysis of ATM banyan networks with shared queueing. Part I: Random offered traffic," *IEEE/ACM Transactions on Networking*, vol. 2, num. 4, pp. 398-410, agosto 1994.

[Guo04] C. Guo, "SRR: An O(1) time complexity packet scheduler for flows in multi-service packet networks," *IEEE/ACM Transactions on Networking*, vol. 12, num. 6, pp. 1144-1155, diciembre 2004.

[Guo07] C. Guo, "G-3: An O(1) Time Complexity Packet Scheduler That Provides Bounded End-to-End Delay," *Proceedings of IEEE Conference on Computer Communications* (INFOCOM'07), vol. 4, pp. 1109-1117, mayo 2007.

[HK88] M.G. Hluchyj y M.J. Karol, "Queueing in high-performance packet switching," *IEEE Journal on Selected Areas in Communications*, vol. 6, num. 9, pp. 1587-1597, diciembre 1988.

[HYG06] H. Hu, P. Yi y Y. Guo, "DiffServ Supporting Scheduling Algorithm for CICQ Switches," *Proceedings of IEEE Region 10 Conference* (TENCON 2006), 4 pp., noviembre 2006.

[ID93] I. Iliadis y W.E. Denzel, "Analysis of packet switches with input and output queuing," *IEEE Transactions on Communicactions*, vol. 41, num. 5, pp. 731-740, mayo 1993.

[IDT] Integrated Device Technology. Disponible en www.idt.com. Último acceso: 2 de noviembre del 2007.

[IM01] S. Iyer y N. McKeown, "Techniques for Fast Shared Memory Switches," Stanford University, High Performance Networking Group, Technical Report TR01-HPNG-081501, 2001.

[JK04] S. Jiwasurat y G. Kesidis, "A class of shaped deficit round-robin (SDRR) schedulers," *Journal of Telecommunication Systems on Recent Advances in Communication and Internet Technology*, vol. 25, num. 3.4, pp. 173-191, marzo 2004.

[JKM05] S. Jiwasurat, G. Kesidis y D.J. Miller, "Hierarchical shaped deficit round-robin scheduling," *Proceedings of IEEE Global Telecommunications Conference* (GLOBECOM'05), vol. 2, pp. 688-693, noviembre-diciembre 2005.

[JN] Juniper Networks. Disponible en www.juniper.net. Último acceso: 1 de junio del 2008.

[JN1] Juniper Networks. Product Brochure. Juniper Network EX-Series Ethernet Switches. Disponible en www.juniper.net. Último acceso: 1 de junio del 2008.

[KHM87] M.J. Karol, M.G. Hluchyj y S.P. Morgan, "Input versus output queueing in a space division switch," *IEEE Transactions on Communications*, vol. COM-35, num. 12, pp. 1347-1356, diciembre 1987.

[KKK90] C.R. Kalmanek, H. Kanakia y S. Keshav, "Rate controlled servers for very high-speed networks," *Proceedings of IEEE Global Telecommunications Conference* (GLOBECOM'90), vol. 1, pp. 12-20, diciembre 1990.

[KKL00] H. Kim, K. Kim y Y. Lee, "Hierarchical scheduling algorithm for QoS guarantee in MIQ switches," *Electronics Letters*, vol. 36, num. 18, pp. 1594-1595, agosto 2000.

[KL95] Y. M. Kim y K.Y. Lee, "KSMINs: knockout switch-based multistage interconnection networks for high-speed packet switching," *IEEE Transactions on Communicactions*, vol. 43, num. 8, pp. 2391-2398, agosto 1995.

[Kle75] L. Kleinrock, Queuing Systems Vol. 2 Computer Applications, *Wiley Interscience*, 1975.

[KP05] M. Katevenis y G. Passas, "Variable-size multipacket segments in buffered crossbar (CICQ) architectures," *Proceedings of IEEE International Conference on Communications* (ICC'05), vol. 2, pp. 999-1004, mayo 2005.

[KPC+99] P. Krishna, N.S. Patel, A. Charny y R.J. Simcoe, "On the speedup required for work-conserving crossbar switches," *IEEE Journal on Selected Areas in Communications*, vol. 17, num. 6, pp. 1057-1066, junio 1999.

[KR00] R. Kannan y S. Ray, "MSXmin: a modular multicast ATM packet switch with low delay and hardware complexity," *IEEE/ACM Transactions on Networking* (TON'00), vol. 8, num. 3, pp. 407-418, junio 2000.

[KS01] H. S. Kim y N. B. Shroff, "Loss probability calculations and asymptotic analysis for finite buffer multiplexers," *IEEE/ACM Transactions on Networking* (TON'01), vol. 9, num. 6, pp. 755-768, diciembre 2001.

[KSC91] M. Katevenis, S. Sidiropoulos y C. Courcoubetis, "Weighted round-robin cell multiplexing in a general-purpose ATM switch chip," *IEEE Journal on Selected Areas in Communications*, vol. 9, num. 8, pp. 1265-1279, octubre 1991.

[Lea90] C.-L. Lea, "Design and performance evaluation of unbuffered self-routing networks for wideband packet switching," *Proceedings of IEEE Conference on Computer and Communications Societies* (INFOCOM'90), vol. 1, pp. 148-156, junio 1990.

[LHD+04] H. Li, C. Huang, M. Devetsikiotis y G. Damm, "Extending the concept of effective bandwidths to DiffServ networks," *Canadian Conference on Electrical and Computer Engineering*, vol. 3, pp. 1669-1672, mayo 2004.

[Li92] S.-Q. Li, "Performance of a nonblocking space-division packet switch with correlated input traffic," *IEEE Transactions on Communications*, vol. 40, num. 1, pp. 97-108, enero 1992.

[LK05a] C.S. Lee y D. Knight, "Realization of the next-generation network," *IEEE Communications Magazine*, vol. 43, num. 10, pp. 34-41, octubre 2005.

[LK05b] T. H. Lee y Y.-C. Kuo, "Packet-based scheduling algorithm for CIOQ switches with multiple traffic classes," *Computer Communications* (Elsevier), vol. 28, num. 12, pp. 1410-1415, enero 2005.

[LKR+90] H. Lee, K. H. Kook, C. S. Rim, K. P. Jun, y S. K. Lim, "A limited shared output buffer switch for ATM," *Fourth International Conference on Data Communications, Systems and their Performance*, pp. 163-179, junio 1990.

[LL95] S.-Y.R. Li y C. M. Lau, "Concentrators in ATM switching," *Proceedings of IEEE Global Telecommunications Conference* (GLOBECOM'95), vol. 3, pp. 1746-1750, noviembre 1995.

[LL97] K.L.E. Law y A. Leon-Garcia, "A large scalable ATM multicast switch," *IEEE Journal on Selected Areas in Communications*, vol. 15, num. 5, pp. 844-854, junio 1997.

[LSC98] Y.-S. Lin, C. B. Shung y J.-C. Chen, "Design of knockout concentrators," *IEE Proceedings of Communications*, vol. 145, num. 3, pp. 145-151, junio 1998.

[LT01] J. Lu y S. Tahar, "Design and verification of an ATM Knockout switch concentrator," *Proceedings of International Conference on Microelectronics* (ICM'01), pp. 261-264, octobre 2001.

[Ma96] J. Ma, "An output buffered ATM switch with a two-class priority discarding scheme," *Proceedings of International Conference on Communication Technology* (ICCT'96), vol. 1, pp. 486-489, mayo 1996.

[MA98] N. McKeown y T. E. Anderson, "A quantitative comparison of iterative scheduling algorithms for input-queued swithes,", *Computer Networks and ISDN Systems*, vol. 30, num. 24, pp. 2309-2326, diciembre 1998.

[Mak98] I.I. Makhamreh, "Throughput analysis of input-buffered ATM switch," *IEE Proceedings of Communications*, vol. 145, num. 1, pp. 15-18, febrero 1998.

[MBG+01] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi y F. Neri, "Packet scheduling in input-queued cell-based switches," *Proceedings of IEEE Conference on Computer Communications* (INFOCOM'01), pp. 1085-1094, abril 2001.

[McK95] N. McKeown, "Scheduling algorithms for input-buffered cell switches", Ph. D. Thesis, Univerity of California at Berkeley, 1995.

[McK99] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking* (TON'99), vol. 7, num. 2, pp. 188-201, abril 1999.

[MIM+97] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick y M. Horowitz, "Tiny Tera: a packet switch core," *IEEE Micro*, vol. 17, num. 1, pp. 26-33, enero-febrero 1997.

[Mir98] N. Mir-Fakhraei, "An efficient multicast approach in an ATM switching network for multimedia applications," *Journal of Network and Computer Applications*, vol. 21, num. 1, pp 31-39, enero 1998.

[MLA+03] C. Minkenberg, R. P. Luijten, F. Abel, W. Denzel y M. Gusat, "Current issues in packet switch design," *ACM SIGCOMM Computer Communication Review*, vol. 33, num. 1, pp 119-124, enero 2003.

[MMA+99] N. McKeown, A. Mekkittikul, V. Anantharam y J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, vol. 47, num. 8, pp. 1260-1267, agosto 1999.

[MMW01] J. Mao, W.M. Moh y B. Wei, "PQWRR scheduling algorithm in supporting of DiffServ," *Proceedings of IEEE International Conference on Communications* (ICC 2001), vol. 3, pp. 679-684, junio 2001.

[MPZ97] N. McKeown, B. Prabhakar and M. Zhu, "Matching output queueing with combined input and output queueing," *Proceedings of the 35th Annual Allerton Conference on Communication, Control and Computing*, pp. 595-603, septiembre 1997.

[MRS03] R. B. Magill, C. E. Rohrs y R. L. Stevenson, "Output-queued switch emulation by fabrics with limited memory," *IEEE Journal on Selected Areas in Communications*, vol. 21, num. 4, pp. 606-615, mayo 2003.

[MS01] S.-H. Moon y D. K. Sung, "High-performance variable-length packet scheduling algorithm for IP traffic," *Proceedings of IEEE Global Telecommunications Conference* (GLOBECOM'01), vol. 4, pp. 2666-2670, noviembre 2001.

[MT] Micron Technology. Disponible en www.micron.com. Último acceso: 2 de noviembre del 2007.

[MX] Maxim. Disponible en www.maxim-ic.com. Último acceso: 10 de diciembre del 2007.

[Nab00] M. Nabeshima, "Performance evaluation of a combined Input - and crosspoint- queued switch," *IEICE Transactions on Communications*, vol. E83-B, num. 3, pp. 737-741, marzo 2000.

[New92] P. Newman, "ATM technology for corporate networks," *IEEE Communications Magazine*, vol. 30, num. 4, pp. 90-101, abril 1992.

[NHL99] G. Nong, M. Hamdi y K.B. Letaief, "Efficient scheduling of variable-length IP packets on high-speed switches," *Proceedings of IEEE Global Telecommunications Conference* (GLOBECOM'99), vol. 2, pp. 1407-1411, diciembre 1999.

[NHR96] M. Naraghi-Pour, M. Hegde y B. Reddy, "A multiple shared memory switch," *Proceedings of the Twenty-Eighth Southeastern Symposium on System Theory*, pp. 50-54, marzo-abril 1996.

[Nor] Nortel. Disponible en www.nortel.com. Último acceso: 1 de junio del 2008.

[Nor1] Nortel. Product Brief. Nortel Ethernet Routing Switch 8600 Release 5. Disponible en www.nortel.com. Último acceso: 1 de junio del 2008.

[NS] National Semiconductor. Disponible en www.national.com. Último acceso: 20 de noviembre del 2007.

[NS1] National Semiconductor Corporation. DS92LV16 Design Guide. Disponible en www.national.com/support/techdoclib.html. Último acceso: 31 de marzo del 2008.

[NS2] National Semiconductor Corporation. LVDS Owner's Manual - 4th Edition. Disponible en www.national.com/support/techdoclib.html. Último acceso: 31 de marzo del 2008.

[NS99] Y. Nishino y I. Sasase, "Multicast knockout ATM switch with batcher network for multiple priority classes," *IEEE Region 10 Conference TENCON 1999*, vol. 1, pp. 226-229, septiembre 1999.

[NTF+87] S. Nojima, E. Tsutsui, H. Fukuda y M. Hashimoto, "Integrated Services Packet Network Using Bus Matrix Switch," *IEEE Journal on Selected Areas in Communications*, vol. 5, num. 8, pp. 1284-1292, octubre 1987.

[OMK+89] Y. Oie, M. Murata, K. Kubota y H. Miyahara, "Effect of speedup in nonblocking packet switch," *IEEE International Conference on Communications* (ICC'89), vol. 1, pp. 410-414, junio 1989.

[OMW06] S. O'Neill, A. Marshall y R. Woods, "Providing Input-Output Throughput Guarantees in a Buffered Crossbar Switch," *Proceedings of 11th IEEE Symposium on Computers and Communications* (ISCC 2006), pp. 725-730, junio 2006.

[OR97] Y. Oh y S. Rai, "A non-blocking copy network for priority handling in ATM multicast switch," *IEEE International Performance, Computing, and Communications Conference* (IPCCC'97), pp. 231-237, febrero 1997.

[OSM+90] Y. Oie, T. Suda, M. Murata, D. Kolson y M. Miyahara, "Survey of switching techniques in high-speed networks and their performance," *Proceedings of IEEE Conference on Computer Communications* (INFOCOM'90), pp. 1241-1251, junio 1990.

[OYS+92] K. Oshima, H. Yamanaka, H. Saito, H. Yamada, S. Kohama, H. Kondoh y Y. Matsuda, "A new ATM switch architecture based on STS-type shared buffering and its LSI implementation," *Proceedings of IEICE*, pp. 359-363, 1992.

[Pat88] A. Pattavina, "Multichannel bandwidth allocation in a broadband packet switch," *IEEE Journal on Selected Areas in Communications*, vol. 6, num. 9, pp. 1489-1499, diciembre 1988.

[PG93] A. K. Parekh y R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Transactions on Networking*, vol. 1, num. 3, pp. 344-357, junio 1993.

[PM98] B. Prabhakar y M. McKeown, "On the speedup required for combined input and output queued switching," *Proceedings of IEEE International Symposium on Information Theory*, pp. 165, agosto 1998.

[QLY+06] H. Qiu, Y. Li, P. Yi y J. Wu, "PIFO Output Queued Switch Emulation by a One-cell-Crosspoint Buffered Crossbar Switch," *Proceedings of International Conference on Communications, Circuits and Systems*, vol. 3, pp. 1767-1771, junio 2006.

[QS96] X. Qian y J. Shen, "Performance analysis of finite output buffer ATM switch with Bernoulli arrival of two priorities," *Proceedings of International Conference on Communication Technology* (ICCT'96), vol. 2, pp. 965-969, mayo 1996.

[RCG94] R. Rooholamini, V. Cherkassky y M. Garver, "Finding the right ATM switch for the market," *Computer*, vol. 27, num. 4, pp. 16-28, abril 1994.

[RF93] E. Del Re y R. Fantacci, "Performance evaluation of input and output queueing techniques in ATM switching systems," *IEEE Transactions on Communications*, vol. 41, num. 10, pp. 1565-1575, octubre 1993.

[ROC01] R. Rojas-Cessa, E. Oki y H.J. Chao, "CIOXB-k: Combined Input-Crosspoint-Output Buffered Packet Switch," *Proceedings of IEEE Global Telecommunications Conference* (GLOBECOM'01), vol. 4, pp. 2654-2660, noviembre 2001.

[ROJ+01] R. Rojas-Cessa, E. Oki, Z. Jing y H. Chao, "CIXB-1: Combined Input-One-Cell Crosspoint Buffered Switch," *Proceedings of IEEE Workshop on High Performance Switching and Routing*, pp. 324-329, mayo 2001.

[RTH88] E.P. Rathgeb, T.H. Theimer y M.N. Huber, " Buffering concepts for ATM switching Networks," *Proceedings of IEEE Global Telecommunications* (GLOBECOM'88), vol. 3, pp. 1277-1281, noviembre-diciembre 1988.

[SCD07] O. Salami, H.A. Chan y M.E. Dlodlo, "Non-priority QoS guarantee for next generation routers," *Proceedings of IEEE AFRICON Conference* (AFRICON'07), pp. 1-7, octubre 2007.

[SIG+91] Y. Sakurai, N Ido, S. Gohara y N. Endo, "Large-scale ATM multistage switching network with shared buffer memory switches," *IEEE Communications Magazine*, vol. 29, num. 1, pp. 90-96, enero 1991.

[SG94] K. J. Schultz y P. G. Gulak, "CAM-based single-chip shared buffer ATM switch," *Proceedings of IEEE International Conference on Communications* (ICC'94), vol. 2, pp. 1190-1195, mayo 1994.

[SMT98] D. Saha, S. Mukherjee y S. Tripathi, "Carry-over round robin: a simple cell scheduling mechanism for ATM networks," *IEEE/ACM Transactions on Networking* (TON'98), vol. 6, num. 6, pp. 779-796, diciembre 1998.

[SNS+89] H. Suzuki, H. Nagano, T. Suzuki, T. Takeuchi y S. Iwasaki, "Output-buffer switch architecture for asynchronous transfer mode", *Proceedings of IEEE International Conference on Communications* (ICC'89), vol. 1, pp. 99-103, junio 1989.

[SPS99] R. Schoenen, G. Post y G. Sander, "Prioritized arbitration for input-queued switches with 100% throughput," *Proceedings of IEEE ATM Workshop*, pp. 253-258, mayo 1999.

[SS06] A. A. M. Saleh y J. M. Simmons, "Evolution toward the next-generation core optical network," *IEEE Journal of Lightwave Technology*, vol. 24, num. 9, pp. 3303-3321, septiembre 2006.

[SV96] M. Shreedhar y G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on Networking*, vol. 4, num. 3, pp. 375-385, junio 1996.

[SV97] D. Stiliadis y A. Varma, "A general methodology for designing efficient traffic scheduling and shaping algorithms," *Proceedings of IEEE Conference on Computer Communications* (INFOCOM'97), vol. 1, pp. 326-335, abril 1997.

[SZ98a] I. Stoica y H. Zhang, "Exact emulation of an output queueing switch by a combined input output queueing switch," *Sixth International Workshop on Quality of Service* (IWQoS'98), pp. 218-224, mayo 1998.

[SZ98b] D.C. Stephens y H. Zhang, "Implementing distributed packet fair queueing in a scalable switch architecture," *Proceedings of IEEE Conference on Computer Communications* (INFOCOM'98), vol. 1, pp. 282-290, marzo-abril 1998.

[TC94] Z. Tao y S. Cheng, "A new way to share buffer-grouped input queueing in ATM switching," *Proceedings of IEEE Global Telecommunications Conference* (GLOBECOM'94), vol. 1, pp. 475-479, noviembre-diciembre 1994.

[TEA+03] F. Tobajas, R. Esper-Chain, V. de Armas, J.F. Lopez y R. Sarmiento, "Cell scheduling for VOQ switches with different strict priority levels," *Electronics Letters*, vol. 39, num. 6, pp. 580-581, marzo 2003.

[TEA+05] F. Tobajas, R. Esper-Chaín, R. Arteaga, V. de Armas y R. Sarmiento, "A novel Gigabit Multidrop Serial Backplane for High Speed Digital Systems," *Proceedings of International Symposium on Microtechnologies for the New Millenium* (SPIE'05), vol. 5837, pp. 112-120, mayo 2005.

[Tera02] TeraStream+ Project (VSC873). Contrato de I+D. *Vitesse Semiconductor Corporation*. 2000-2002. Investigador responsable: Dr. Roberto Sarmiento Rodríguez.

[TET+05] F. Tobajas, R. Esper-Chaín, O. Tubío, R. Arteaga, V. de Armas y R. Sarmiento, "Experimental gigabit multidrop serial backplane for high speed digital systems," *Proceedings of IEEE International Symposium on Circuits and Systems* (ISCAS'05), vol. 4, pp. 3821-3824, mayo 2005.

[TIC2002-02998] TIC2002-02998. Diseño e implementación de un *chipset* (Transceptor y matriz de conmutación) para transmisión de paquetes en redes de alta velocidad. Plan Nacional de I+D+I. Ministerio de Educación y Ciencia. (2002-2005). Investigador principal: Dr. Roberto Sarmiento Rodríguez.

[Tob90] F. Tobagi, "Fast packet switch architectures for broadband integrated services digital networks," *Proceedings of IEEE*, vol. 78, num. 1, pp. 133-167, enero 1990.

[Tob01] F. Tobajas, "Aportaciones al diseño de conmutadores con VOQ y soporte para diferentes calidades de servicio," *Tesis Doctoral*, Universidad de las Palmas de Gran Canaria, diciembre 2001.

[Tse05] E. S. H. Tse, "Switch fabric design for high performance IP routers: A survey," *Journal of Systems Architecture*, vol. 51, num. 10-11, pp. 571-601, octubre-noviembre 2005.

[Vri90] R.J.F De Vries, "Gauss: a simple high performance switch architecture for ATM," *Proceedings of the ACM Symposium on Communications Architectures & Protocols*, pp. 126-134, 1990.

[VSC] Vitesse Semiconductor Corporation. Disponible en www.vitesse.com. Último acceso: 2 de noviembre del 2007.

[VSC1] Vitesse Semiconductor Corporation. VSC870 Data Sheet Revision 4.1: High performance serial backplane transceiver. Disponible en www.datasheetcatalog.com. Último acceso: 11 de mayo del 2008.

[VSC2] Vitesse Semiconductor Corporation. VSC880 Data Sheet Revision 4.2: High performance 16 serial crosspoint switch. Disponible en www.datasheetcatalog.com. Última acceso: 11 de mayo del 2008.

[VSC3] Vitesse Semiconductor Corporation. GigaStream Chip Set (VSC872 & VSC882): 80 Gb/s intelligent switch fabric. Disponible en www.vitesse.com. Último acceso: 11 de mayo del 2008.

[VSC4] Vitesse Semiconductor Corporation. TeraStream Chip Set (VSC871 & VSC881): 160 Gb/s intelligent switch fabric. Disponible en www.digchip.com. Último acceso: 11 de mayo del 2008.

[Wan01] Z. Wang, Internet QoS Architectures and Mechanism for Quality of Service, Morgan Kaufmann Publishers, 2001.

[WH07] F. Wang y M. Hamdi, "iPIFO: A Network Memory Architecture for QoS Routers," *Workshop on High Performance Switching and Routing* (HPSR'07), pp. 1-5, junio 2007.

[WL02] M. Y. L. Wong y C. K. Li, "Low computational complexity weighted queuing using weighted deficit round robin," *Proceedings of 20th IASTED International Conference-Applied Informatics*, febrero 2002.

[WWL05] C.-C. Wu, H.-M. Wu y W. Lin, "Efficient and fair multi-level packet scheduling for differentiated services," *Proceedings of the Seventh IEEE International Symposium on Multimedia* (ISM'05), 7 pp., diciembre 2005.

[WWM+06] C.-C. Wu, H.-M. Wu, C. Moh y W. Lin, "A Hierarchical Packet Scheduler for Differentiated Services in High-Speed Networks," *Proceedings of the fifth IEEE/ACIS International Conference on Computer and Information Science* (ICIS-COMSAR'06), pp. 63-68, julio 2006.

[XL03] J. Xie y C.-T. Lea, "Speedup and buffer division in input/output queueing ATM switches," *IEEE Transactions on Communications*, vol. 51, num. 7, pp. 1195-1203, julio 2003.

[YC01] K. Yoshigoe y K.J. Christensen, "A parallel-polled virtual output queued switch with a buffered crossbar," *Proceedings of IEEE Workshop on High Performance Switching and Routing*, pp. 271-275, mayo 2001.

[YHA87] Y.-S. Yeh, M. G. Hluchyj y A.S. Acampora, "The knockout switch: a simple, modular architecture for high-performance packet switching," *IEEE Journal on Selected Areas in Communications*, vol. 5, num. 8, pp. 1274-1283, octubre 1987.

[YLZ03] M. Yang, E. Lu y S.Q. Zheng, "Scheduling with dynamic bandwidth allocation for DiffServ classes," *Proceedings of 12th International Conference on Computer Communications and Networks* (ICCCN 2003), pp. 319-324, octubre 2003.

[YNO+02] K. Yamakoshi, K. Nakai, E. Oki y N. Yamanaka, "Dynamic deficit round-robin scheduling scheme for variable-length packets," *Electronics Letters*, vol. 38, num. 3, pp. 148-149, enero 2002.

[Yos04] K. Yoshigoe, "Design and evaluation of the combined input and crossbar queued (CICQ) swith," *Tesis Doctoral*, University of South Florida, agosto 2004.

[YQW06] P. Yi, H. Qiu y B. Wang, "Implementing priority scheduling in a combined input-crosspoint-output queued switch," *Proceedings of 20th International Conference on Advanced Information Networking and Applications* (AINA 2006), vol. 2, 5 pp., abril 2006.

[YWL+04] M. Yang, H. Wang, E. Lu y S.Q. Zheng, "Hierarchical scheduling for DiffServ classes," *Proceedings of IEEE Global Telecommunica-*

tions Conference (GLOBECOM'04), vol. 2, pp. 707-712, noviembre-diciembre 2004.

[ZH07] Y. Zhang y P.G. Harrison, "Performance of a Priority-Weighted Round Robin Mechanism for Differentiated Service Networks," *Proceedings of 16th International Conference on Computer Communications and Networks* (ICCCN 2007), pp. 1198-1203, agosto 2007.

[Zha90] L. Zhang, "Virtual clock: a new traffic control algorithm for packet switchind networks," *ACM SIGCOMM Computer Communication Review*, vol. 20, num. 4, pp 19-29, septiembre 1990.

[Zha95] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-switching Networks," *Proceeding of the IEEE*, vol. 83, num. 10, octubre 1995.

[ZMB07] X. Zhang, S.R. Mohanty y L.N. Bhuyan, "Adaptive Max-Min Fair Scheduling in Buffered Crossbar Switches Without Speedup," *Proceedings of IEEE Conference on Computer Communications* (INFOCOM'07), pp. 454-462, mayo 2007.