**UNIVERSIDAD DE CASTILLA-LA MANCHA**
**ESCUELA POLITÉCNICA SUPERIOR**

**INGENIERÍA**
**EN INFORMÁTICA**

# PROYECTO FIN DE CARRERA

Estudio, adaptación y evaluación de algoritmos de arbitraje para entornos con control de flujo a nivel de enlace de red.

Jorge Juan García Rodríguez

**Julio, 2007**

**UNIVERSIDAD DE CASTILLA-LA MANCHA**
**ESCUELA POLITÉCNICA SUPERIOR**
Departamento de Sistemas Informáticos

# PROYECTO FIN DE CARRERA

Estudio, adaptación y evaluación de algoritmos de arbitraje para entornos con control de flujo a nivel de enlace de red.

Autor: Jorge Juan García Rodríguez

Directores: Francisco José Alfaro Cortés
Raúl Martínez Moráis

**Julio, 2007**

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

**“Titulo del proyecto”**

Presentado por D/Dª

**Jorge Juan García Rodríguez**

y siendo su/s tutor/es

**Francisco José Alfaro Cortés y Raúl Martínez Moráis**

se otorga la calificación de _____

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

Albacete, a ..... de ..... de 200...

PRESIDENTE: _____

SECRETARIO: _____

VOCAL: _____

SECRETARIO PRESIDENTE VOCAL

# RESUMEN

Los avances que se han producido en las tecnologías de los computadores y en las redes de computadores permiten obtener muy buenas prestaciones de las mismas y han propiciado la proliferación de nuevas aplicaciones como la videoconferencia, difusión de televisión digital, telemedicina o video bajo demanda. Estas nuevas aplicaciones son grandes consumidoras de recursos y necesitan unos requisitos cada vez más estrictos. Dichas necesidades han provocado que los mecanismos de control de flujo hayan tenido que revisarse para que se acomoden a los nuevos requisitos, que además son cambiantes.

Una muestra del creciente interés que genera este tema es la incorporación de mecanismos para proporcionar QoS en los últimos estándares de las tecnologías de red. Una componente clave para la diferenciación del tráfico en las redes con soporte de QoS es el algoritmo de planificación que se emplea en los enlaces de salida. Dicho algoritmo está supeditado a la tecnología de red sobre la cual actúa. Si la tecnología tiene soporte para QoS mediante canales virtuales, el algoritmo de planificación debe encargarse de distribuir el tráfico en unos enlaces u otros según su destino, y dentro de cada enlace de salida decide a cuál de los canales virtuales corresponde dicho tráfico.

Por todo lo anterior, en este trabajo se pretende diseñar unos algoritmos que se integren a la perfección con los mecanismos de control de flujo a nivel de enlace de red. Aún así, no tenemos redes capaces de aprovechar todos los avances de los que disponemos, ya que, por ejemplo, en los paquetes IP hay campos específicos para dar QoS, pero que el hardware disponible en las redes no lo utilizan.

Quiero dedicar este proyecto a mis padres **Juan** y **Mª Angeles** que me han apoyado en todo momento, a mi novia **Yolanda**, a mis tios **Enrique**, **Ascensión**, **Mariano**, **Consuelo**, **José Luis**, **Carmelo** y **Trini**, a todos mis prim@s, a mis abuelas **Mª Angeles** y **Carmen**, a mis hermanos **Ángel** y **Ana**, y a mis amigos, incluidos aquellos con los que no he tenido tanto tiempo para salir. Especialmente quiero agradecer a mi abuelo **Carmelo**, al que puedo considerar mi primer profesor de informática, utilizando el Spectrum 64k o el Amstrad CPC-6128, espero que, desde arriba, esté muy orgulloso de mí.

Quiero agradecer también especialmente a mis tutores **Francisco José** y **Raúl**, como también a **José Luis**, por su inestimable ayuda y paciencia que han tenido conmigo.

Por último, no olvidarme de esos compañeros de angustias, con los que he compartido la experiencia que da esta titulación, acordarme de todos los profesores que he tenido, y también de las noches en vela y los cafés reparadores.

# ÍNDICE GENERAL

# 1 INTRODUCCIÓN.

El incremento espectacular que se ha producido en prestaciones de los computadores y las redes de interconexión durante los últimos años y la disminución de sus costes ha permitido el gran desarrollo de las aplicaciones multimedia, siendo estos los principales clientes que aprovechan las mayores capacidades de las redes de computadores. Las aplicaciones de videoconferencia, difusión de televisión digital, telemedicina o video bajo demanda, no sólo requieren un gran ancho de banda y una baja latencia, sino que tienen otros requisitos adicionales diferentes de las aplicaciones tradicionales, como pueden ser las transferencias de archivos o conexiones remotas, entre otras.

Las redes de alta velocidad son cada vez más frecuentes y el modelo de servicio "best effort" no puede dar una respuesta adecuada para los requisitos actuales de las diversas aplicaciones. Esto ha producido qué en las últimas décadas se haya investigado mucho para obtener un mecanismo que ofrezca calidad de servicio según que tipo de aplicación esté en el extremo de la comunicación. Además, el tráfico que tienen que soportar las redes, y la congestión que puede producir el tráfico, hace que haya que idear soluciones para estos casos, pero para obtener una buena solución hay que estudiar cuidadosamente la naturaleza de estos conflictos que se producen.

Estos nuevos requisitos se reflejan en lo que se denomina Calidad de Servicio (Quality of Service, QoS). El término QoS fue empleado por primera vez hace unos veinte años en el modelo de Interconexión de Sistemas Abiertos (OSI). Hacía referencia a la capacidad de un proveedor de servicios, o un operador de red, para soportar los requisitos de las aplicaciones de usuario con respecto a cuatro parámetros de servicio: ancho de banda, latencia o retardo, jitter o variación en la latencia y pérdidas de datos.

Una muestra del creciente interés que genera este tema es la incorporación de mecanismos para proporcionar QoS en los últimos estándares de las tecnologías de red. Las principales características incorporadas en las tecnologías de red que les permiten proporcionar QoS y tratamiento diferenciado al tráfico son: control de flujo, gestión de la congestión y diferenciación del tráfico.

Una componente clave para la diferenciación del tráfico en las redes con soporte de QoS es el algoritmo de planificación que se emplea en los enlaces de salida. Dicho algoritmo está supeditado a la tecnología de red sobre la cual actúa. Esta última se encarga de distribuir el tráfico en unos enlaces u otros según su destino. Si la tecnología tiene soporte para QoS mediante canales virtuales, también debe decidir, dentro de cada enlace de salida, qué tráfico corresponde a uno u otro canal virtual. El algoritmo de

planificación se encarga de seleccionar qué canales virtuales pueden enviar sus paquetes. Para ello se basa en los parámetros de QoS que debe ofrecer a cada uno de los canales virtuales, y con ello diferencia un tipo de tráfico de otro.

Un planificador ideal implementado en una red de altas prestaciones con soporte para QoS debería satisfacer dos propiedades principales: buena latencia y simplicidad. Los planificadores se pueden clasificar como "work-conserving" o conservativos y "non-work-conserving" o no conservativos. Los primeros son aquellos en los que el servidor no está nunca desocupado mientras haya algún paquete en alguna de sus colas. Esto les permite tener un menor retardo medio. Los segundos pueden mantenerse desocupados incluso aunque queden paquetes por transmitir. Se utilizan cuando se quiere controlar el *jitter*, siendo capaces de retrasar los paquetes que hayan llegado antes de tiempo. En este trabajo nos centraremos en los algoritmos de planificación conservativos, ya que estos aprovechan todo el tiempo mientras haya paquetes en el sistema. Para un mejor estudio, los planificadores conservativos se pueden dividir, atendiendo a su estructura interna, en: Sorted priority o basados en prioridades y frame based o basados en frames.

La tendencia actual en las redes de interconexión con mecanismos de QoS es incorporar mecanismos de control de flujo a nivel de enlace. Estos mecanismos evitan que se descarten paquetes debido a la congestión. El problema de la mayoría de las técnicas de planificación tradicionales es que fueron diseñadas para redes sin control de flujo a nivel de enlace, por ejemplo ATM o Internet. Estas circunstancias son las que han motivado el desarrollo de este trabajo. Es decir, adaptar algunos de los algoritmos de planificación conservativos propuestos en la literatura a las nuevas necesidades impuestas por el mecanismo de control de flujo, de forma que el algoritmo de planificación interactúe de manera adecuada con el control de flujo. También se probará el funcionamiento de los mismos, realizando una comparación entre las dos grandes ramas de funcionamiento existentes, los algoritmos basados en frames y los algoritmos basados en prioridades.

Para las pruebas de funcionamiento y la obtención de resultados se utilizará la simulación, este método pondrá de manifiesto lo adecuado o no que resulta la técnica o mecanismo propuesto. El proceso de evaluación consiste básicamente en someter al modelo de red que incorpora las propuestas realizadas a diferentes pruebas. Este modelo viene caracterizado por una serie de parámetros (como puedan ser el planificador, el tipo de tráfico, el espacio de almacenamiento en los buffers, etc…).

# 2 REVISIÓN DE DIVERSAS TECNOLOGÍAS DE RED Y ALGORITMOS DE ARBITRAJE.

En los últimos años se ha producido un incremento espectacular en las prestaciones de los computadores [SW97, CJ97]. Como consecuencia, en las tecnologías de red actuales se dispone de un gran ancho de banda[1], capaz de proporcionar varios gigabits por segundo, y una latencia[2] muy baja, que permite la transferencia en pocos milisegundos o incluso decenas de nanosegundos, todo esto con un bajo coste. Esta disponibilidad de las redes ha permitido un gran desarrollo de las aplicaciones multimedia como los principales clientes que aprovecharán las mayores capacidades de estas redes de computadores. Ejemplos de aplicaciones multimedia pueden ser: videoconferencia, difusión de televisión digital, vídeo bajo demanda o telemedicina. Sin embargo, estas aplicaciones no sólo requieren un gran ancho de banda y una baja latencia, sino que tienen otros requisitos adicionales diferentes de las aplicaciones tradicionales, como pueden ser las transferencias de archivos o conexiones remotas, entre otras.

Las aplicaciones multimedia integran varios tipos de medio, como vídeo, audio, imágenes fijas, gráficos y texto. La transmisión de estos tipos, y sobre todo del audio y del vídeo, introducen nuevos requisitos en las redes de transmisión. Concretamente, tanto el audio como el vídeo generan un tráfico continuo, en el sentido de que requiere disponibilidad de ancho de banda durante toda la duración de la transmisión para que la aplicación no se vea interrumpida en el nodo receptor. En el caso del vídeo, el tráfico suele ser a ráfagas, es decir, no inyecta la misma cantidad de información por unidad de tiempo durante el progreso de dicho flujo. Esto es debido a los formatos de compresión actuales, como por ejemplo MPEG-4 [ISO99], que se emplean para la transmisión de vídeo. A este tipo de tráfico variable se le denomina Variable Bit Rate (VBR), como contraposición a un tipo de tráfico constante que inyecta una cantidad fija de información por unidad de tiempo, y que suele recibir el nombre de Constant Bit Rate (CBR). Además, en el caso de la señal de vídeo, se debe mostrar al receptor una imagen cada cierto tiempo de forma continua. Para ello, estas imágenes, una vez enviadas desde el emisor, se deben presentar en el destino dentro de unos límites de retardo adecuados.

Todos estos nuevos requisitos se reflejan en lo que se denomina Calidad de Servicio (Quality of Service, QoS). El término QoS fue empleado por primera vez hace unos veinte años en el modelo de Interconexión de Sistemas Abiertos (OSI). Hacía referencia a la capacidad de un proveedor de servicios (un operador de red) para soportar los requisitos de las aplicaciones de usuario con respecto a cuatro parámetros

[1] Ancho de banda: cantidad de datos que se pueden transmitir en una unidad de tiempo.
[2] Latencia: suma de los retardos temporales dentro de la red. Normalmente la suma del tiempo de propagación del paquete, el tiempo de transmisión del paquete y el tiempo de espera en el conmutador.

de servicio: ancho de banda, latencia o retardo, jitter o variación en la latencia y pérdidas de datos [Bla00]:

- La provisión de **ancho de banda** para una aplicación significa que la red tenga la suficiente capacidad para soportar los requisitos de productividad de la aplicación.
- La **latencia o retardo** describe el tiempo que se tarda en enviar un paquete de usuario desde un nodo origen a un destino. Hay algunas aplicaciones, aquellas con un índice de interactividad elevado, que tienen marcados requisitos en cuanto a latencia. Por ejemplo, aplicaciones de voz o videoconferencia.
- En cuanto al **jitter**, se trata de la variación del retardo entre la llegada al receptor de paquetes consecutivos. Estos retardos son causados por diferentes tiempos de espera provocados a los paquetes por la contención al acceder a los recursos. Una variación elevada en los retardos experimentados por los paquetes complica la reproducción en el destino de secuencias de vídeo o voz, al vaciarse o desbordarse los buffers que recogen los paquetes pertenecientes a dichos flujos. Si un paquete llega lo suficientemente tarde como para que el buffer se vacíe, se traduce en una posible degradación en la percepción de las imágenes o el sonido por parte de los usuarios. Además, cuando el paquete finalmente llegue será descartado traduciéndose en un ancho de banda desperdiciado. Por el contrario, si un paquete llega demasiado pronto y no hay sitio en el buffer para acomodarlo también tendrá que ser descartado produciendo los mismos efectos negativos.
- La **pérdida de datos** es el último de los parámetros generalmente considerados cuando se habla de QoS a nivel de red. En el caso de aplicaciones no multimedia los datos descartados pueden ser retransmitidos y el único efecto es la sobrecarga que esto produce y el aumento en el retardo de la aplicación para obtener los datos. En las aplicaciones multimedia la retransmisión no suele ser posible debido a sus restricciones en cuanto a latencia y jitter, por lo que puede traducirse, como ya se ha dicho, en la degradación de la señal que recibe el usuario final.

Una componente clave para redes con soporte de QoS es el algoritmo de planificación que se emplea en los enlaces de salida. Dicho algoritmo está supeditado a la tecnología de red sobre la cual actúa. Esta última se encarga de distribuir el tráfico en unos enlaces u otros según su destino. Si la tecnología tiene soporte para QoS mediante canales virtuales, también debe decidir, dentro de cada enlace de salida, qué tráfico corresponde a uno u otro canal virtual. El algoritmo de planificación se encarga de seleccionar qué canales virtuales pueden enviar sus paquetes. Para ello se basa en los

parámetros de QoS que debe ofrecer a cada uno de los canales virtuales, y con ello diferencia un tipo de tráfico de otro.

Dada la importancia que tienen los algoritmos de planificación en este trabajo, a continuación se incluye una revisión de algunas de las más importantes tecnologías de red, y de los algoritmos de planificación que en ellas se proponen.

## 2.1 TECNOLOGÍAS DE RED.

Durante los últimos años los avances en teoría, arquitectura y tecnología de las redes de computadores han permitido mejorar las prestaciones de éstas enormemente. Se han desarrollado algoritmos de encaminamiento más eficientes y fiables, mejores topologías de red, mecanismos de control de flujo y de la congestión, esquemas de utilización de los medios compartidos, y técnicas de conmutación más eficientes. Más aún, durante la pasada década hemos presenciado como técnicas empleadas tradicionalmente en redes de interconexión de multicomputadores y multiprocesadores han saltado las barreras de dichas máquinas extendiendo su aplicación al ámbito de las redes de área local (LAN, Local Area Network).

La necesidad de proporcionar QoS en las redes se ha producido cuando el tráfico ha evolucionado de formas diferentes y con necesidades diferentes [Rei06]. Hace relativamente poco tiempo, las redes utilizaban enlaces compartidos (buses), que se convirtieron en el cuello de botella del sistema. Las investigaciones consiguieron mejorar su rendimiento pasando a unas configuraciones conmutadas. Así, cuando se dispuso de un buen ancho de banda, llegaron las aplicaciones en tiempo real y multimedia. Ya en la década de los 90 se pensó mejorar el aprovechamiento de la red, para lo cual, la solución era dividir el tráfico según sus requisitos.

Las principales características incorporadas en las tecnologías de red que les permiten proporcionar QoS y tratamiento diferenciado al tráfico son:

- **Control de flujo**: trata de reducir o eliminar la pérdida de paquetes en los buffers de los conmutadores. Normalmente, la pérdida de paquetes es resultado de la contención y el sobreflujo, y su uso puede dar como resultado la generación de congestión en parte de la red, extendiéndose al resto en forma de enlaces transmitiendo más lentamente o incluso parados temporalmente. Este fenómeno se conoce como *back-pressure*.

- **Gestión de la congestión**: trata de prevenir o reaccionar ante estados de congestión, reduciendo o controlando sus efectos sobre el rendimiento total de la red. Es el mecanismo que ayuda a los enlaces y conmutadores de la red para no estar sobrecargados. Cuando aparece la congestión se merman los créditos y se empiezan a llenar las colas del conmutador, las colas de los conmutadores vecinos e incluso en los peores casos llega a

los nodos, lo que crea árboles de congestión, que pueden interferir en otros flujos que incluso no vayan destinados a la zona congestionada.

- **Diferenciación del tráfico**: el conmutador aplica diferentes tratamientos a cada uno de los flujos particulares, que sumados forman todo el tráfico. Así, con ello se puede asegurar un cierto nivel de servicio a cada uno de esos flujos. El núcleo de las capacidades de QoS se basa en la disponibilidad de recursos independientes para cada canal virtual, junto con el uso de mecanismos de diferenciación de canales virtuales. Ciertos flujos pueden asignarse a baja prioridad, mientras otros pueden requerir estrictas garantías de latencia o retardos acotados.

A continuación vamos a mostrar las principales características de algunas de las últimas tecnologías de red propuestas.

### 2.1.1 ATM.

El estándar Asynchronous Transfer Mode (ATM) está definido por el ATM Forum (fundado en 1991) y por los estándares ITU [Dom]. Las redes ATM comenzaron con las compañías de telecomunicaciones, y con las redes MAN o WAN en la troncal de Internet. Ahora, muchas empresas tienen sus propias redes ATM con soporte para Internet o para LAN.

Hoy en día las redes ATM soportan grande rangos de velocidad de transmisión, 25, 51, 155 y 622 Mbps. A menor velocidad, menor es el coste de los switches y de los enlaces, pero cada vez más se intenta incrementar la velocidad.

ATM usa unidades de transferencia de tamaño fijo llamadas celdas (Figura 2.1) y son la unidad básica de transmisión [Gir98]. Los usuarios establecen conexiones virtuales (VC) para transportar las celdas desde una fuente a un destino. Varios VCs pueden ser agrupados en un camino virtual (VP) para simplificar la gestión de la red, al reducir el número de elementos a gestionar. Cada conexión tiene un identificador único que se obtiene de combinar los identificadores VC y VP. Las conexiones pueden ser establecidas como permanentes (PVC) a través del sistema de gestión de red. De forma alternativa, las conexiones conmutadas (SVC) pueden ser establecidas y eliminadas de forma dinámica. Los caminos pueden ser determinados, por ejemplo, usando el protocolo de enrutamiento "*private network-network interface*" o por medio de tablas de enrutamiento estático.

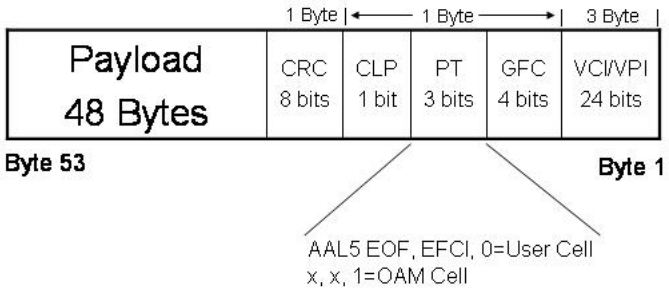



**Figura 2.1 Formato de una celda ATM.**

Algunos conmutadores soportan VCs y VPs, mientras que otros conmutadores sólo soportan VPs y no se preocupa de qué VCs estén usando los VPs. Estos últimos no llevan a cabo el proceso de llamada de los VCs y no mantienen ninguna información de VCs. Es decir, los conmutadores más cercanos al usuario discriminan entre caminos virtuales y canales virtuales, mientras que el resto de la red, sólo atiende a caminos virtuales para así tener menos información que gestionar.

Las aplicaciones generalmente acceden a la capa de transporte ATM por medio de la capa de adaptación ATM (*ATM adaptation layer*). Las más comunes son AAL1, para tráfico que requiere sincronización de tiempo, y AAL5, que lleva tráfico que no tiene requisitos de tiempo.

El campo GFC de la celda ATM se designa para controlar el ratio de un terminal usando el esquema de control de flujo *Stop-and-Go*.

El campo CRC se usa para detectar errores en la cabecera y prevenir el envío de paquetes a destinos equivocados (*cell missinsertion*). En caso de que sólo haya un único bit equivocado, éste se puede corregir. La carga del paquete no está protegida por el CRC de la cabecera.

El campo tipo de carga (PT) consta de 3 bits. El primero es 0 para datos de usuario, y 1 en caso de señalización o información de mantenimiento. El segundo (EFCI) es el indicador explícito de congestión hacia delante, e indica que la congestión impide el paso. Este bit puede usarse para el control de flujo. El tercero se utiliza con tráfico AAL5 para indicar que la celda es la última de una unidad de datos de nivel superior (PDU).

El bit CLP se utiliza para indicar que la celda tiene una prioridad de descarte menor que otra celda que no lo tiene activado. A esas celdas no se les garantiza recibir la calidad de servicio convenida.

### 2.1.2 Gigabit Ethernet.

Desde 1973, la tecnología Ethernet ha destacado en el ámbito de las redes de área local, tanto cableadas como inalámbricas, así que ahora se está estudiando cómo adaptarla a nuevas áreas como clustering, almacenamiento e incluso backplane. En 1982 apareció la Ethernet de 10-Mbps, que se ha comprobado que no es nada adecuada para satisfacer todo el ancho de banda que requieren los clusters de computadores o las aplicaciones de Internet. En 1994 aparecieron dos versiones Fast Ethernet de 100-Mbps (100BaseT y 100VG-AnyLan). En 1997 el grupo de trabajo de la IEEE 802.3 anunció la salida de Gigabit Ethernet (Tabla 2.1). La tecnología que mostramos en nuestro estudio es la última propuesta.

| Características | Gigabit Ethernet |
|---|---|
| **Enrutamiento** | Destination lookup |
| **Ancho de enlace (nº canales)** | 1x, 4x (para 3.125Gb/s) |
| **Ancho de banda por canal** | 1, 3.125, 10 Gb/s |
| **Ancho de banda** | 1 – 10 Gb/s |
| **Tamaño máximo de paquete** | 1522 bytes |
| **Tamaño mínimo de paquete** | 64 bytes |
| **Codificación de transmisión** | 8B/10B, 64/66B (para 10 Gb/s) |
| **Longitud máxima del cable** | 5000 m |

**Tabla 2.1 Características de Ethernet.**

- **Control de flujo**: El mecanismo de control de flujo a nivel de enlace se especifica para ser usado en redes sin pérdidas. El nodo receptor envía un mensaje explícito, *off-message,* directamente al emisor. Así se detiene toda la transmisión durante un cierto periodo de tiempo, evitando sobrecargar al receptor con paquetes que no puede almacenar. Si la transmisión se puede reanudar antes del tiempo especificado, se envía un mensaje *on-message*. Para que funcione correctamente, se debe tener en cuenta el retardo entre la emisión del paquete de pausa y su activación. Este retraso está en función del tiempo de transmisión y del tiempo de procesado. Hay que tener en cuenta que para mantener el mismo ancho de banda, el control de flujo xon/xoff requiere el doble de espacio en buffer que el sistema de créditos.

- **Gestión de la congestión**: Actualmente el estándar no ofrece ningún mecanismo de control de la congestión, pero se está investigando para incluir un mecanismo que avise directamente a los nodos causantes de la congestión. También se está considerando el uso de una notificación explícita de congestión hacia atrás, lo cual tiene un corto bucle de control al

no necesitar llegar hasta el destino para avisar a la fuente. También se está considerando la inclusión de un mecanismo, que reduzca la inyección del tráfico en la red en base al nivel de congestión.

- **Diferenciación del tráfico**: Desde 1998 se usa el mecanismo de etiquetado de los paquetes y la implementación de múltiples colas dentro del conmutador para discriminar entre las 8 clases de tráfico diferentes. Sin embargo, esto entra en conflicto con el control de flujo, ya que éste bloquea el enlace/puerto, sin tener en cuenta las prioridades del tráfico, produciendo una degradación del servicio en todos los niveles. Para solucionarlo, el organismo de estandarización ha sugerido que el control de flujo se extienda para soportar una granularidad parecida a la del mecanismo de prioridades, realizando un control de flujo basado en clases. Esto se puede hacer usando ciertos campos reservados de la capa de enlace. Otra consecuencia del control de flujo xon/xoff es que no puede ofrecer enrutamiento libre de bloqueos, ni mejoras de prestaciones.

### 2.1.3 InfiniBand.

InfiniBand fué estandarizada en el año 2000 como una convergencia de Futura I/O y Next Generation I/O. Es una tecnología de interconexión con enlaces serie, punto a punto, full-duplex, que pretendía sustituir al bus PCI en los servidores, a FiberChannel en redes de almacenamiento y a Ethernet en redes de área local, aunque no resultó fructífero. Sin embargo, sí que se está usando InfiniBand para la conexión de sistemas de almacenamiento y computación de altas prestaciones, donde se requiere gran ancho de banda y baja latencia (Tabla 2.2). InfiniBand utiliza el concepto de "*virtual lanes*" para dividir un enlace físico en varios canales virtuales. Cada canal virtual tiene su propio buffer, su propio control de flujo y su propia gestión de la congestión. El concepto de *virtual lanes* también permite agruparlas de forma recursiva. Esto hace posible construir redes virtuales sobre la topología física, lo que permite varios propósitos como enrutamiento más eficiente, evita bloqueos, aumenta la tolerancia a fallos y permite la diferenciación de servicios

| Características | IBA |
|---|---|
| **Enrutamiento** | Destination lookup |
| **Ancho de enlace (nº canales)** | 1x, 4x, 12x |
| **Ancho de banda por canal** | 2.5, 5, 10 Gb/s |
| **Ancho de banda** | 2.5 – 120 Gb/s |
| **Tamaño máximo de paquete** | 4096 bytes |
| **Tamaño mínimo de paquete** | 24 bytes |
| **Codificación de transmisión** | 8B/10B |
| **Número máximo de hosts** | 49.152 |
| **Número máximo de puertos por switch** | 255 |

**Tabla 2.2 Características de IBA.**

Algunas características de InfiniBand son:

- **Control de flujo**: Para evitar la pérdida de paquetes debida al desbordamiento de buffers, se usa un esquema de control de flujo punto a punto basado en créditos. El receptor mantiene un contador de los recursos libres del buffer, llamados créditos y medidos en unidades de cantidad de información, decrementando dicho contador cuando se almacena algo en el buffer y aumentándolo cuando se desocupa el mismo. De manera análoga, el emisor mantiene un contador de los créditos que posee, decrementándolo cada vez que manda un paquete. Cuando llegan créditos del receptor, aumenta el número de créditos disponibles. Nunca se envía un paquete a menos que haya lugar para almacenarlo. A intervalos regulares, el receptor, actualiza el número de créditos disponible para el emisor y este intervalo varía con la carga del sistema: con carga alta se actualiza más frecuentemente, para cargas bajas se reduce la frecuencia de actualización. El uso del control de flujo permite que la retransmisión de paquetes sólo se produzca debido a errores en la transmisión y hace posible aprovechar el ancho de banda efectivo, ya que las retransmisiones no son necesarias.

- **Gestión de la congestión**: IBA tiene tres mecanismos de control de congestión: notificación explícita de congestión hacia delante (*forward explicit congestion notification* - FECN), control estático de ratio y mecanismo de drenado de la cabeza de la cola.

    - La notificación explícita de congestión hacia delante se utiliza para avisar al destino de un paquete de que el camino por el que ha llegado está congestionado, para lo que hay un flag (FECN) en la cabecera del paquete. El destino puede avisar a la fuente bien mediante un paquete de congestión o bien mediante los

reconocimientos al poner dicho flag activo. El conmutador puede también informar al nodo fuente de la congestión con 16 niveles de congestión diferentes. El conmutador se puede identificar como el causante si un canal virtual ha excedido la capacidad de su buffer y tiene créditos disponibles. Si, por el contrario, el canal virtual ha excedido su umbral y no tiene créditos, se puede considerar como una víctima de la congestión.

- El control estático de ratio se usa en los nodos finales para reducir su ritmo de envío de paquetes al recibir una notificación de congestión o un paquete de reconocimiento con el flag de congestión activado. Si llegan más notificaciones o flags de congestión, se sigue reduciendo el ratio de envío. Para volver a aumentar el ratio hay un temporizador. Los ratios de envío están en una tabla que contiene los ratios aceptables según el nivel de congestión.
- El mecanismo de drenado de la cabeza de la cola asegura que una cola sea drenada cuando un temporizador llega a cero. Esto es útil si un destino ha dejado de consumir paquetes.

- **Diferenciación del tráfico**: Ya se ha visto el soporte para canales virtuales, que es la base para la diferenciación del servicio. Los 3 mecanismos para la diferenciación del tráfico son: el nivel de servicio de un canal virtual, el peso de los canales virtuales y su prioridad. El nivel de servicio modela el tipo de servicio que recibe un paquete a lo largo del recorrido. Como normalmente no se tiene una relación uno a uno entre los canales virtuales y el nivel de servicio, se guarda una tabla con el mapeado de canales sobre los niveles de servicio. Se soportan hasta 16 canales virtuales asociados con su nivel de servicio, el canal 15 corresponde a la gestión del tráfico. Aparte del canal de control que tiene la mayor prioridad, los demás canales pueden tener prioridad alta o baja. El tráfico de alta prioridad tiene preferencia sobre el tráfico de baja prioridad, pero para asegurar el progreso de todos hay un límite de alta prioridad (LHP), que es el máximo número de paquetes de canales de alta prioridad que pueden ser planificados antes de coger un paquete de un canal de baja prioridad. Entre canales de la misma prioridad se utiliza un esquema de arbitraje ponderado. Cada canal virtual es planificado según su nivel de servicio indicado en la tabla y se le asigna un peso indicando el número de bytes que se le permite transmitir en cada turno.

### 2.1.4 RapidFabric architecture (SRIO).

Durante los últimos dos años, la Rapid IO Trade Asociation ha trabajado para extender la interconexión de Serial RapidIO (SRIO) para cubrir las necesidades de datos de las aplicaciones de redes de telecomunicaciones [Har05], aunque las extensiones para proporcionar control de flujo se han desarrollado a partir del año 2002. La especificación original de SRIO sólo permite una conexión serie simple chip-a-chip para interconexión de aplicaciones entre procesadores, que es ampliamente utilizada. Está tecnología está pensada para reemplazar tecnologías propietarias de "data fabrics" con una tecnología estándar abierta que añade multicast, control de flujo entre extremos y características de "*data streaming*". Un ejemplo de interconexión se puede ver en la Figura 2.2, y sus principales características están en la Tabla 2.3.

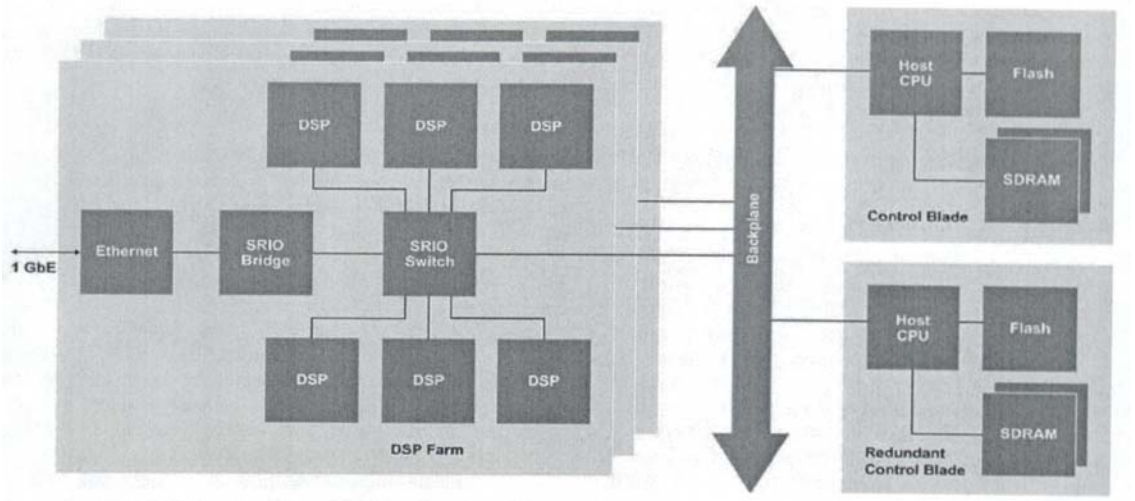



**Figura 2.2 Ejemplo de sistema SRIO.**

| Características | SRIO / RapidFabric |
|---|---|
| **Enrutamiento** | Basado en los dispositivos |
| **Ancho de enlace (nº canales)** | 1x, 4x |
| **Ancho de banda por canal** | 1.25 , 2.5 , 3.125 Gbps |
| **Ancho de banda** | 10 Gbps |

**Tabla 2.3 Características de SRIO.**

Algunas de las principales caracteristicas de SRIO son:

- **Control de flujo**: Se basa en el método xon/xoff en las peticiones de las transacciones. Los elementos del "fabric" mantienen el nivel del buffer interno entre unos límites inferior y superior variables, según el tamaño de los paquetes. Si un flujo sobrepasa esos límites el receptor envía un mensaje de control xoff para indicarle al emisor que deje de transmitir temporalmente. Dichos límites están determinados por varios factores como

la localización de los elementos del conmutador, la distancia a la fuente, la topología, y deben permitir suficiente espacio para almacenar los paquetes en vuelo y tener en cuenta las latencias en el peor de los casos. Cuando el nivel baja del límite, el receptor envía un mensaje xon a la fuente para continuar con la transmisión.

- **Control de la congestión**: El control de la congestión se realiza con un control de flujo extremo a extremo. Este control de flujo puede ser controlado por el transmisor o por el receptor. Cuando es controlado por el receptor, acepta los datos mientras tenga suficiente buffer. Al no tener espacio disponible descarta el paquete e informa al emisor con un paquete de retransmisión. El transmisor espera y retransmite el paquete. El conmutador del receptor genera paquetes de control de flujo para empezar (xon) y parar (xoff) el flujo de tráfico desde una fuente. Cuando el control de flujo es controlado por el transmisor, el sistema confía en un esquema basado en créditos. En este caso, el receptor informa a la fuente sobre cuánto espacio de buffer hay disponible y el emisor regula el flujo como corresponda.

- **Diferenciación del tráfico**: la capa física implementa un campo de 2 bits en cada paquete para asignarle la prioridad. La capa lógica define tres tipos de peticiones de transacciones (alta, media, baja) que son mapeados a una de las cuatro prioridades de la capa física. Las mejoras hechas después han incluido más clases de servicio.

### 2.1.5 Advanced Switching Interconnection (ASI).

Desde mediados de los años 90 el bus PCI ha evolucionado de paralelo a serie, y de basarse en bus a ser conmutado. PCI Express es una interconexión de sistemas de gran ancho de banda y baja latencia. Advanced Switching Interconnection (ASI) está construido sobre la base de PCI Express (PCI-X) reutilizando la capa física y la capa de enlace, pero con su propia capa de transacciones para poder ofrecer comunicaciones punto a punto en el conmutador. A finales de 2003 se publicó la versión 1.0 de las especificaciones de ASI [Adv03], cuyas principales características pueden verse en la Tabla 2.4 .

| Características | ASI |
|---|---|
| **Enrutamiento** | Source routing |
| **Ancho de enlace (nº canales)** | 1x, 2x, 8x, 16x, 32x |
| **Ancho de banda por canal** | 2.5, 5 Gb/s |
| **Ancho de banda** | 2.5 – 128 Gb/s |
| **Tamaño máximo de paquete** | 2176 bytes |
| **Tamaño mínimo de paquete** | 64 bytes |
| **Codificación de transmisión** | 8B/10B |
| **Número máximo de puertos por switch** | 256 |

**Tabla 2.4 Características de ASI Interconnection.**

- **Control de flujo**: El control de flujo utiliza un mecanismo punto a punto basado en créditos para prevenir la pérdida de paquetes. Los créditos se actualizan regularmente, y si es necesario, créditos adicionales son enviados para aquellos que gastan sus créditos más rápido. El emisor, vecino directo del receptor, mantiene el estado de la comunicación. Una limitación de las redes sin pérdidas es la posibilidad de situaciones de bloqueo cuando hay una dependencia circular entre varios recursos. ASI hereda el modelo de carga/almacenamiento de PCI-X, que dispone de un esquema de prevención de bloqueos extremo a extremo integrado en las estructuras de sus colas, de forma que puede romper esa dependencia circular al permitir a una de las operaciones sobrepasar a las demás. Esta prevención de bloqueo se consigue usando una cola *bypassable* que mantiene siempre créditos para 2 tipos de paquetes, *bypassable-ordered* y *bypassable-bypass.* Los paquetes *bypassable-bypass* no pueden impedir el progreso de los paquetes *bypassable-ordered*. Si ambos tipos de paquetes tienen créditos, las colas se comportan como FIFO, pero si los paquetes *bypassable-bypass* se queda sin créditos, los paquetes *bypassable-ordered* puede seguir emitiendo paquetes mientras tengan créditos, adelantando a los paquetes retenidos. Una vez que los paquetes *bypassable-bypass* vuelven a tener créditos se les da estricta prioridad, ya que han sido retenidos.

- **Gestión de la congestión**: La congestión como resultado de la falta de créditos es una condición transitoria. Para manejarlo, ASI utiliza un sistema de estado (*status-based flor-control* – SBFC) que permite al conmutador del receptor avisar del estado de congestión de alguno de sus puertos de salida a su vecino emisor. Así, éste asigna menor prioridad a los paquetes de los flujos que tienen como destino dicho puerto. El impacto de la congestión sólo tiene efecto en los paquetes que van al puerto congestionado, mientras los demás no serán afectados. La inyección de tráfico en el conmutador es limitada por el uso de colas asociadas con ratios de transmisión. Estas colas

pueden separar el tráfico por clase, destino, o próximo conmutador. Para controlar el ratio de inyección y las ráfagas producidas, se dispone de un *token bucket*. Aunque la notificación de la congestión extremo a extremo no está soportada de forma explícita dentro de ASI, los paquetes normales de ASI llevan un campo, llamado FECN, para informar al destino del estado de congestión, y éste puede informar a la fuente para que limite el ratio al que está enviando paquetes.

- **Diferenciación del tráfico**: Se soporta usando clases de tráfico que separan los flujos en varias clases. Ocho clases de tráfico se mapean a los canales virtuales, permitiendo priorizar los flujos en el conmutador. El número máximo de canales virtuales que se pueden gestionar son veinte, limitados según sus tipos de tráfico: 8 canales virtuales del tipo ordered, 8 canales virtuales del tipo bypass y 4 canales virtuales del tipo multicast. Los mensajes de control utilizan los canales bypassable con la máxima prioridad (TC7), que disponen de la máxima prioridad en el planificador. Se soportan 2 mecanismos diferentes de planificación en los buffers de salida, uno basado en una tabla y el otro basado en el mínimo ancho de banda. El mecanismo basado en la tabla es legado de PCI Express; mientras, el de mínimo ancho de banda (MinBW) permite servicios diferenciados para varios canales virtuales, y proporciona un mínimo compartido para cada uno, evitando la contención/inanición. Normalmente usa alguna variante del algoritmo de planificación *weighted fair queuing*. Durante la negociación del enlace, el nodo emisor y el receptor deciden el número de canales virtuales que emplearán, eligiendo el menor número que implementen los dos extremos, adaptando sus estructuras de colas y vigilando el cumplimiento de los parámetros.

### 2.1.6 Consideraciones finales.

Para continuar se considera, en las distintas tecnologías, si se dispone de control de flujo a nivel de enlace, y si éste se realiza a nivel de puerto, o a nivel de canal virtual.

ATM no dispone de un mecanismo de control de flujo a nivel de enlace, sino un mecanismo de control de flujo extremo a extremo del tipo *Stop-and-Go*, SRIO dispone de un mecanismo similar, lo cual crea el inconvenientes de que, puede haber momentos en los que los paquetes crucen los conmutadores y no dispongan de espacio para almacenarse, o sean descartados frente a otros flujos más prioritarios. Esto origina retransmisiones que producen la degradación de las prestaciones. En el caso de aquellas aplicaciones con estrictos requisitos de latencia, es conveniente que dichas retransmisiones no se produzcan, ya que una vez que lleguen los paquetes perdidos, ya será demasiado tarde para utilizar su información. Incluso en el caso del video, que tiene

una cierta resistencia ante la perdida de paquetes, es conveniente que no se produzca dicho descarte, ya que así se podrá optimizar la transmisión de este tipo de datos al no tener que emplear costosas técnicas de recuperación de errores.

Ethernet utiliza un sistema que limita el control de flujo a los puertos, lo cual limita la efectividad para dar QoS cuando el control de flujo está activado. Al mandar un mensaje *xoff* el enlace es bloqueado, y no se permite que otros flujos puedan utilizar ese puerto, aunque los demás puertos del conmutador pueden seguir funcionando normalmente.

Las tecnologías más recientes, ASI e IBA, sí han tenido en cuenta las cuestiones anteriores, de forma que permiten que distintos flujos reciban diferentes niveles de QoS. En estos casos el control de flujo a nivel de enlace se encarga de dividir los enlaces en canales virtuales, de forma que aunque haya algunos canales bloqueados en el enlace, los demás canales sí pueden continuar emitiendo paquetes. Más aún, en el caso de ASI, el control de la congestión, SBFC, puede detener flujos de un canal virtual, basándose en el destino de la siguiente etapa de la red, y permitir a otros flujos del mismo canal virtual seguir avanzando. Esto hace que las prestaciones se degraden en mucha menor medida que en las tecnologías de red más antiguas.

En la Tabla 2.5 se incluye una comparativa de los elementos que componen el control de flujo de estas tecnologías de red.

| Tecnología de red | Control de flujo a nivel de enlace | Clases de servicio | Planificador de prioridades |
|---|---|---|---|
| **ATM** | no | 5 | Sin determinar |
| **Ethernet** | XOn/XOff (nivel de puerto) | 8 | SPQ, WFQ (optional) |
| **IBA** | Credit-based | 16 | Table (WRR) |
| **SRIO / RapidFabric** | no | tres tipos de peticiones de transacciones que son mapeados a cuatro prioridades | Sin determinar |
| **ASI** | Credit-based | 20 | Table, MinBW |

**Tabla 2.5 Elementos de control de flujo.**

## 2.2 ALGORITMOS DE PLANIFICACIÓN.

La arquitectura general [Zha02] de un router con buffer a la entrada y a la salida puede verse en la Figura 2.3. El factor de *speedup*[3] varía entre 1 y N, donde N es el número de puertos de entrada/salida del conmutador. Los conmutadores de gran velocidad actuales tienen un *speedup* pequeño, algo mayor que 1 y son llamados Combined Input and Output Queued (CIOQ) switch ya que necesitan buffers tanto en la entrada como en la salida.

Todos los paquetes en los buffers se organizan en diferentes flujos o conexiones, cada una de las cuales forma una cola. Cada una de estas colas compite por un enlace de salida común. Diferentes servidores tienen diferentes disciplinas de servicio, y por tanto diferentes algoritmos de planificación.

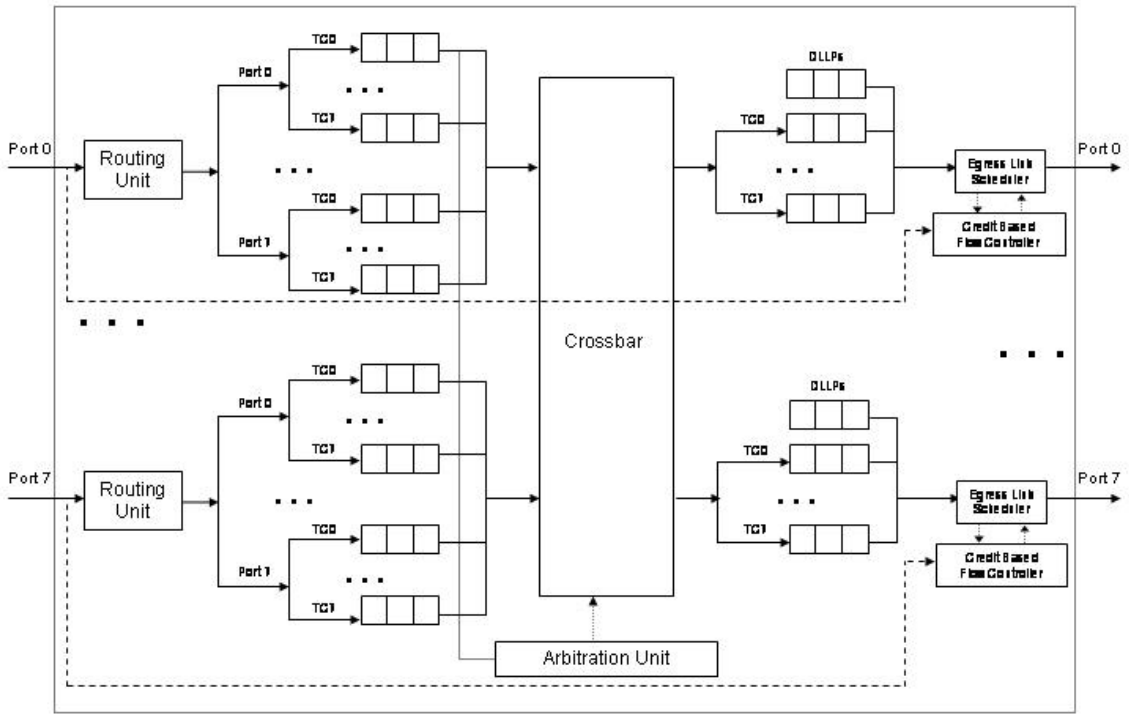



**Figura 2.3 Modelo básico de un planificador.**

Un planificador ideal implementado en una red de altas prestaciones con soporte para QoS debería satisfacer dos propiedades principales: buena latencia y simplicidad. En la Figura 2.4, se puede ver una clasificación de algunos de los algoritmos de planificación que hay propuestos hoy en día.

[3] Speed-up: relación entre la velocidad de los enlaces y la velocidad del elemento de cruce del conmutador.

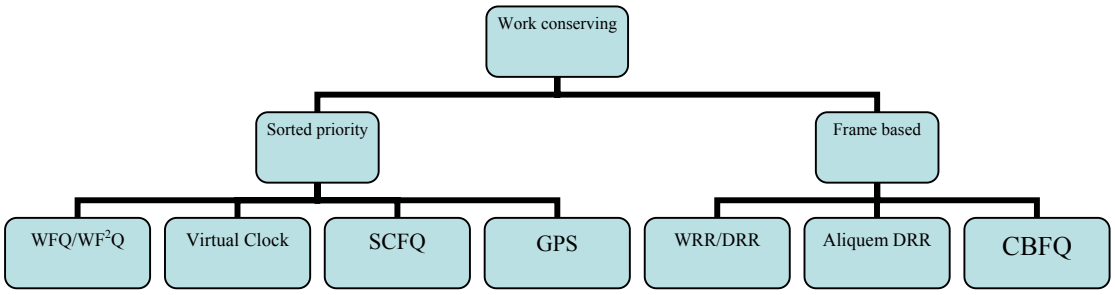



**Figura 2.4 Planificadores work conserving.**

Podríamos clasificar los planificadores como "work-conserving" o conservativos y "non-work-conserving" o no conservativos. Los primeros son aquellos en los que el servidor no está nunca desocupado mientras haya algún paquete en alguna de sus colas. Esto les permite tener un menor retardo medio. Los segundos pueden mantenerse desocupados incluso aunque queden paquetes por transmitir. Se utilizan cuando se quiere controlar el *jitter*, siendo capaces de retrasar los paquetes que hayan llegado antes de tiempo.

En base a la estructura interna del planificador, los planificadores "work-conserving" se pueden dividir en:

- Sorted priority: tienen una variable global, normalmente llamada *virtual time*, que es mantenida y actualizada cada vez que llega o se sirve un paquete. Un *timestamp* o etiqueta, se asocia con cada paquete en el sistema como función de esta variable, y esta marca se utiliza para ordenar y transmitir los paquetes. Los algoritmos sorted priority, por definición, son conservativos, ya que ordenan los paquetes según la etiqueta que se les asocia, y por tanto mientras el planificador tenga paquetes, va eligiendo el siguiente paquete que corresponda, atendiendo al valor de su etiqueta.

- Frame based: estos planificadores dividen el tiempo en ranuras de longitud fija o variable. Las reservas se hacen en términos de cantidad de tráfico que le está permitido transmitir a un flujo durante una ranura de tiempo. Si el servidor usa ranuras de tiempo constantes y el flujo no utiliza todo lo que ha reservado, el servidor puede permanecer desocupado.

Este trabajo, se centra en los planificadores conservativos, ya que son los que encontraremos en las tecnologías de red mencionadas en el apartado anterior. No

obstante, los algoritmos no conservativos pueden ser convertidos en algoritmos conservativos al añadirles una cola más, llamada *stand-by queue*. A la llegada de un paquete al sistema, se va encolando en la cola que le corresponda, y también en la cola stand-by. Cuando los paquetes son servidos, se quitan de la cola a la que pertenecen y de la cola stand-by. En caso de que alguna cola no tenga paquetes que enviar en su turno, se mira cuál es el primer paquete que hay en esta cola stand-by y se envía. Así se aprovechan los momentos ociosos enviando paquetes que estén esperando ser servidos.

A continuación se pasa a detallar el funcionamiento y las características más relevantes de algunos de los planificadores más importantes. Para una mejor comprensión de los algoritmos, se separarán en algoritmos basados en frames y algoritmos basados en etiquetas.

### 2.2.1 Algoritmos basados en frames.

En este apartado se muestra el funcionamiento de algunos de los algoritmos basados en frames más importantes. Se han seleccionado para este estudio los algoritmos Round Robin, Weighted Round Robin, Deficit Round Robin, Aliquem Round Robin, Smooth Aliquem Round Robin y Credit-Based Fair Queuing, aunque existen otros muchos.

#### 2.2.1.1 Algoritmo Round Robin (RR).

En un planificador Round Robin [Zha02] el eje del tiempo es dividido en ranuras de tamaño máximo fijo, y los paquetes de las diferentes colas se sirven uno tras otro. Cada cola puede tener diferentes requisitos de ancho de banda, y el número de paquetes que pueden ser servidos en cada ranura está acotado. Los contadores, que se asocian a cada una de las colas, se inicializan al principio de la ranura al máximo que pueden transmitir durante la ranura y se decrementan al servirse un paquete de dicha cola. Esta cantidad se llama quantum, y ninguna cola cuyo contador sea cero puede ser servida.

#### 2.2.1.2 Weighted Round Robin (WRR) y Deficit Round Robin (DRR).

En redes con tamaño de paquete fijo, el contador suele indicar el número de paquetes que se pueden servir. En este caso el planificador se llama WRR. Si por el contrario el tamaño del paquete es variable, el contador se especifica en bytes, y nos referimos al planificador como DRR. La simplicidad de este algoritmo permite implementaciones que consiguen gran velocidad. Sin embargo, los requisitos de un

buen reparto del ancho de banda obligan a un tamaño de ranura grande y en consecuencia, grandes retardos de extremo a extremo.

El algoritmo DRR, propuesto en [Shr95], da servicio a las colas de un conmutador. A cada flujo i se le asocia una cantidad $quantum_i$, que indica el ancho de banda que comparte el flujo i sobre el total del enlace de salida, o más genéricamente, lo que comparten todos los flujos de datos que deben compartir el mismo canal virtual del conmutador, respecto de la suma del ancho de banda de todos los canales virtuales, también se asocia a cada flujo un contador, $DeficitCounter_i$, para acumular el $quantum_i$ no utilizado, y que es inicializado a cero.

$$\phi_i = \frac{Quantum_i}{\sum_{n=1}^{N} Quantum_n} \qquad \textbf{( 2.1)}$$

A la llegada de los paquetes de los diferentes flujos, éstos se guardan en las diferentes colas. Siendo el número de bytes que se envían para la cola i en la vuelta k ($bytes_{i,k}$). A cada cola i se le permite enviar paquetes con la restricción de que ($bytes_{i,k}$ < quantum). Si no quedan más paquetes en la cola i, una variable de estado llamada DeficitCounter es puesta a 0. En caso contrario DeficitCounter se actualiza con la cantidad que queda para servir $quantum_i - bytes_{i,k}$ . En las rondas siguientes, la cantidad de ancho de banda que puede utilizar será la suma de $DeficitCounter_i$ de la ronda previa más $quantum_i$.

Este funcionamiento podría expresarse más claramente con el siguiente pseudocódigo:

1. Cuando llega el turno del flujo i:
   $DeficitCounter_i \leftarrow Quantum_i + DeficitCounter_i$
2. Enviar mientras:
   $bytes_{i,k} < Quantum_i$
3. Actualizar contador deficitario:
   a. Si quedan paquetes:
      $DeficitCounter \leftarrow Quantum_i - bytes_{i,k}$
   b. Si no quedan paquetes:
      $DeficitCounter \leftarrow 0$
4. Volver al paso 1.

Para evitar examinar colas vacías, se mantiene una lista auxiliar ActiveList que mantiene los índices de las listas que tienen por lo menos un paquete para transmitir. Cuando un paquete llega a una cola vacía, su índice se añade al final de esta lista. Cuando el índice de una cola está a la cabeza de la lista, el algoritmo la sirve hasta el

límite de $DeficitCounter_i$ + $Quantum_i$. Si al final del quantum aún quedan paquetes, se inserta su índice al final de la lista. Si no quedan paquetes por transmitir, $DeficitCounter_i$ es puesto a 0 y se elimina su índice de la lista de colas activas.

La longitud del *frame* de estos algoritmos se define como la suma de todos los quantum. Cuanto más largo es el *frame*, mayor es la latencia y peor es la equidad. Este debe ser lo menor posible, pero para obtener una complejidad O(1), cada flujo debe obtener un quantum no menor a la máxima Maximum Transfer Unit (MTU). Esto es así para que este algoritmo tenga un orden de complejidad constante O(1), el tamaño del *frame* debe ser lo suficientemente grande para que, con el quantum que se pueden obtener en cada vuelta, cada flujo activo pueda enviar un paquete por lo menos, lo que puede causar que dicho tamaño sea demasiado grande, dependiendo de la situación. El tamaño mínimo para operar con O(1) es: $F^S = \max_{j=1..N}\left(\frac{\overline{L_j}}{f_j}\right)$. De no cumplirse, la complejidad seria O(N), siendo N el número de canales virtuales.

#### 2.2.1.3 Aliquem Round Robin.

El método de Active LIst QUEue Method (ALIQUEM) [Len04] es una implementación que permite a DRR obtener un mejor retardo y equidad preservando el O(1) de la complejidad. Esto se consigue evitando que el tamaño del *frame* tenga que ser extremadamente grande. Para $\overline{L_j}$ el tamaño máximo de paquete, este algoritmo se basa en suponer la longitud del *frame* $F = \alpha F^S$, 0<α<1, y $F^S = \max_{j=1..N}\left(\frac{\overline{L_j}}{Quantum_j}\right)$, que es el tamaño mínimo de *frame* para que el algoritmo pueda funcionar con una complejidad O(1). Nótese que, tener una longitud de *frame* F, en vez de $F^S$, significa violar la condición de caber por lo menos un paquete en cada quantum. Si se permite al flujo i esa longitud de *frame*, y se utiliza una lista FIFO para los flujos activos, el flujo i se desencolará y encolará en la lista varias veces, hasta que su contador de déficit sea mayor que el tamaño de su primer paquete. Pero, sólo con tener la información del tamaño de su primer paquete de la lista y el quantum asignado al flujo i, se puede saber cuantas vueltas dará el algoritmo antes de que pueda servir su primer paquete. Así, la actualización de su contador se puede diferir hasta ese momento. Específicamente para el flujo i, siendo $L_i$ el tamaño del primer paquete de la cola i antes de que sea servido, y $DeficitCounter_i$ el valor del contador deficitario en ese momento. Podemos calcular la R-ésima ronda en la cual será servido el paquete como:

$$R = \left\lceil \frac{L_i - DeficitCounter_i}{Quantum_i} \right\rceil \qquad \textbf{(2.2)}$$

Y el contador deficitario queda como:

$$R * Quantum_i + DeficitCounter_i \quad \textbf{( 2.3 )}$$

Esto se puede aplicar incluso cuando llega un paquete a un flujo que no está activo, teniendo el contador deficitario un valor nulo.

Aliquem DRR evita encolamientos y desencolamientos innecesarios y también actualizar los contadores al utilizar las expresiones anteriores. En vez de considerar una sola cola FIFO activa, se utilizará la estructura de la Figura 2.5. Una cola circular de q>1 elementos, contiene la cabeza y la cola referentes a q colas diferentes de listas activas. Cada lista activa contiene referencias a flujos acumulados. Durante la ronda k, sólo los flujos cuyas referencias están guardadas en la (k mod q) lista activa, la actualmente activa, son considerados para la transmisión y envían los paquetes de acuerdo al algoritmo DRR.

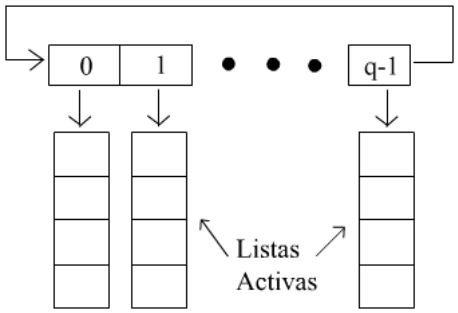



**Figura 2.5 Estructura de la lista activa.**

Cuando el flujo completa el servicio, es decir, ha transmitido tantos paquetes como le permite el contador deficitario, calculado con 2.2, si aún le quedan paquetes para transmitir, el canal se encola en otra lista activa, específicamente la que va a ser activa en la próxima ronda en la que pueda transmitir el paquete de su cabecera. La lista g en la que debe ser encolado el flujo i será.

$$g= (k+R) \bmod q \quad \textbf{( 2.4 )}$$

Para que esta implementación funcione, es obligatorio que, para cualquier posible valor de $L_i$, $DeficitCounter_i$ y $quantum_i$, para $0 \le DeficitCounter_i \le \overline{L_i}$ se cumpla.

$$R<q \quad \textbf{( 2.5 )}$$

Estos requisitos trasladan las siguientes restricciones a la longitud del quantum para que no se pueda dar una vuelta completa a la lista de listas, y por tanto atender a los canales cuando no corresponde.

$$\forall \text{ i}, \ \phi_i \geq \frac{\overline{L_i}}{q-1} \qquad \textbf{( 2.6 )}$$

Despejando queda el tamaño mínimo de la lista de listas.

$$q \geq \frac{\overline{L_i}}{Quantum_i} + 1 \qquad \textbf{( 2.7)}$$

#### 2.2.1.4 Smooth Aliquem Round Robin

De acuerdo a DRR, durante cada vuelta se sirven los flujos hasta que éstos no tienen más quantum para transmitir. Aliquem DRR obtiene un subconjunto de los flujos que pueden transmitir durante la vuelta, lo que puede resultar en ráfagas de tamaño $\overline{L_i} + \phi_i$ que pueden obtener servicio. Claramente, reduciendo el tamaño del frame se reduce el tamaño de las ráfagas, pero implica un aumento del trasiego de flujos al tener un quantum menor. También se puede reducir el tamaño de las ráfagas al limitar el número de paquetes que puede transmitir a uno por cada transmisión [Len04]. Este flujo puede ser encolado al final de la actual lista activa o en una nueva cola, según tenga aún quantum o no, respectivamente. Esto tiene varias ventajas, como unos requisitos de buffers menores y una mejor equidad y retardo medios.

#### 2.2.1.5 Credit-Based Fair Queuing (CBFQ).

Para este algoritmo, además del ancho de banda compartido $S_i$, un contador $K_i$ se asocia al flujo i, donde i= 1,…, J, J < N, siendo J el número de colas activas y N el número total de colas [Ben01]. Este contador mantiene la cuenta del peso acumulado conseguido por el correspondiente flujo. Basándose en el ancho de banda compartido, los valores de los contadores y los tamaños de la cabecera de los paquetes, el algoritmo decide qué paquete será el próximo en servirse. En otras palabras, en vez de esperar a que tenga suficiente peso acumulado para transmitir el paquete, lo que resulta en un desperdicio del ancho de banda, el flujo que necesita el menor tiempo para obtener el peso acumulado necesarios para transmitir el paquete, se servirá primero. Cada vez que un paquete es servido, su contador es puesto a cero y los demás contadores de los flujos son actualizados como corresponda. El algoritmo actúa de la siguiente forma:

1) Fase inicial:
2) $\forall l, l \in \{1,...,J\}, Poner \quad K_l \leftarrow 0$

3) Operaciones:
4) Sea J = $\{j_1, \ldots, j_k\}$ tal que las colas $j_1, \ldots, j_k$ están activas
5) Sea $L_{jn}$ el tamaño del primer paquete de la cola $j_n \in J$

6) Para todo $j_i \in J$

$$T_{j_i} \leftarrow \frac{L_{j_i} - K_{j_i}}{S_{j_i}}$$

7) Ordenar las colas activas respecto a

$$T_{j_1} \leq T_{j_2} \leq \cdots \leq T_{j_n}$$

8) Transmitir el paquete de la cola $j_1$, y actualizar los contadores así:

$$K_{j_n} \leftarrow K_{j_n} + T_{j_1} \cdot S_{j_n}, \forall j_n \in J / \{j_1\}$$

$$K_{j_1} \leftarrow 0$$

9) Ir al paso 4)

Este algoritmo, inicialmente pone a cero los contadores de las colas. Durante el ciclo de operación del algoritmo, calcula $T_i$ en base a su ancho de banda compartido, tamaño del paquete y valor del contador, este número nos indica cual de todas las colas será la primera en obtener el peso suficiente para enviar el primero de sus paquetes, ordena las colas respecto a $T_i$, y transmite el paquete de la primer cola de dicha ordenación. A los contadores de las demás colas se les suma el peso que habrían acumulado hasta el momento en que la cola seleccionada emite su primer paquete, es decir, el valor del contador del canal que se ha emitido, y pone a cero este último. Luego volvemos a repetir el algoritmo desde la comprobación de las colas que están activas.

### 2.2.2 Algoritmos basados en prioridades.

A continuación se muestra el funcionamiento de algunos de los algoritmos basados en prioridades más importantes. Se han seleccionado para este estudio los algoritmos Generalized Processor Sharing, Weighted Fair Queueing, Worst-case Fair Weighted Fair Queuing, Self-Clocked Fair Queueing y Virtual Clock aunque existen otros muchos.

#### 2.2.2.1 Generalized Processor Sharing (GPS).

Es una disciplina de planificación ideal, definida por [Zha02] respecto a un modelo de fluidos, en la cual los paquetes son considerados infinitesimalmente divisibles. El ancho de banda reservado para el flujo i se representa por un número real $Ø_i$. Las sesiones a las que aún les quedan paquetes por transmitir, se sirven con un ratio mínimo igual al que han reservado en cada instante, pero si algún flujo no lo consume, el exceso es repartido entre los demás flujos proporcionalmente al ratio que tienen reservado individualmente. Como resultado obtenemos un buen aislamiento entre canales, equidad ideal y un bajo retardo extremo a extremo. Pero esta técnica no se puede implementar, debido a la imposibilidad de dividir los paquetes infinitesimalmente, lo que atrajo a muchos investigadores que desarrollaron aproximaciones a este algoritmo.

Suponiendo, durante un intervalo de tiempo dado, un ratio de servicio R, N flujos, con un peso de $Ø_i$ para el i-ésimo flujo, y siendo V el conjunto de flujos activos en dicho periodo, el flujo i recibe un ancho de banda ($B_i$) proporcional a su peso.

$$B_i = \frac{\phi_i}{\sum_{j=1}^{V} \phi_j} * R \qquad \textbf{( 2.8 )}$$

En el sistema GPS cada flujo tiene exactamente un paquete en servicio y el ratio de servicio que está obteniendo para el flujo i es $\frac{\phi_i}{\sum_{j \in B(\tau)} \phi_j} r$ donde $B(\tau)$ es el conjunto de flujos acumulados en tiempo $\tau$.

#### 2.2.2.2 Weighted Fair Queueing (WFQ).

También conocido como Packet GPS (PGPS), es decir, versión GPS para paquetes [Zha02]. El sistema GPS se simula en paralelo con el sistema WFQ para saber el conjunto de flujos a los que les quedan paquetes para transmitir en cada momento. El tiempo virtual v(t) es una función lineal del tiempo real t, y su pendiente cambia dependiendo del número de conexiones activas y sus anchos de banda asignados respectivos. Siendo $\sum_{i \in B(\tau,t)} \phi_i$ la sumatoria de los anchos de banda compartidos por los flujos que aún tienen paquetes para transmitir, WFQ se define en términos de un reloj virtual que se incrementa con un ratio igual a

$$\frac{1}{\sum_{i \in B(\tau,t)} \phi_i} \qquad \textbf{( 2.9 )}$$

Cuando llega un nuevo paquete, primero se debe calcular su tiempo virtual, y luego el *timestamp* del k-ésimo paquete del flujo i ($TS_i^k$) se calcula de acuerdo a (2.9), donde $L_i^k$ es el tamaño del k-ésimo paquete.

$$TS_i^k \leftarrow \max(TS_i^{k-1}, v(t)) + \frac{L_i^k}{\phi_i} \qquad \textbf{( 2.10 )}$$

Un número máximo de V eventos pueden ocurrir en el simulador GPS durante la transmisión de un paquete, siendo V el número máximo de flujos a los que les quedan paquetes por transmitir. Así, la sobrecarga para tomar una decisión es O(V), lo cual hace muy difícil y costoso la implementación de este algoritmo.

#### 2.2.2.3 Worst-case Fair Weighted Fair Queuing ($WF^2Q$).

Al diseñar un sistema de paquetes que emule el sistema de fluidos GPS tan fielmente como sea posible, surge el problema de que en el sistema de fluidos en cada momento hay n paquetes cruzando el sistema, cosa que no ocurre en el sistema de paquetes, pues sólo hay un paquete cada vez. Como en el sistema GPS cada flujo acumulado tiene exactamente un paquete en servicio y el ratio de servicio que está obteniendo para el flujo i es $\frac{\phi_i}{\sum_{j \in B(\tau)} \phi_j} R$ donde B(τ) es el conjunto de sesiones acumuladas en tiempo τ. Mientras el tiempo de servicio de un paquete con L bits en el sistema de paquetes es $\frac{L}{R}$, puede ser mucho mayor en el sistema GPS dependiendo de la fracción de ancho de banda garantizado al flujo y el número de sesiones activas. Así pues, aunque el paquete en el sistema de paquetes puede empezar a ser servido después que en el sistema GPS, puede acabar de servirse mucho antes que en el sistema GPS, y por tanto, los siguientes paquetes pueden empezar a obtener servicio antes que en el sistema GPS, esto puede dar como resultado grandes diferencias entre ambos sistemas. Si tenemos en cuenta que en un periodo de tiempo grande, en ambos sistemas se recibe el mismo servicio, el sistema de paquetes compensa los tiempos en los que ha recibido mucho servicio con tiempos en los que no recibe apenas servicio.

Para minimizar este problema $WF^2Q$ [Ben96b] cuando elige el siguiente paquete que va a ser servido en tiempo τ, en vez de elegir sobre todos los paquetes que están acumulados en ese momento, sólo considera el conjunto de paquetes que han empezado y posiblemente finalicen a recibir servicio en el sistema GPS en tiempo τ, y selecciona el paquete que terminará primero en el sistema GPS.

#### 2.2.2.4 Self-Clocked Fair Queueing (SCFQ).

Otro concepto importante es el esquema del reloj virtual en [Gol94], ya que se ha mostrado como una buena herramienta para formular la equidad y representar el progreso del sistema, aunque no provee un servicio equitativo y la complejidad computacional es demasiado elevada para llevarla a cabo en un sistema de alta velocidad, debido a su definición respecto al esquema basado en el sistema del fluido.

El esquema SCFQ adoptar una noción diferente del tiempo virtual, que depende del progreso del trabajo y no del sistema de fluidos. La generación del tiempo virtual, dentro del propio esquema, permite una sobrecarga insignificante, pues el tiempo virtual es extraído del paquete que se encuentra primero en la cola. Esto elimina el coste computacional del sistema de fluidos y nos permite una implementación simple y viable en sistemas de banda ancha.

SCFQ es otra aproximación de GPS para reducir el coste del cálculo de v(t). En este algoritmo el cálculo de v(t) se define respecto al *timestamp* del paquete en servicio $TS_{current}$ . El *timestamp* del k-ésimo paquete se calcula como:

$$TS_i^k \leftarrow \max\{TS_i^{k-1}, TS_{current}\} + \frac{L_i^k}{\phi_i} \qquad \textbf{( 2.11 )}$$

El precio pagado con esta simplificación es que el retardo extremo a extremo crece de manera lineal al número de conexiones que están compartiendo la línea de salida. Tampoco se puede controlar el retraso de un flujo en el peor de los casos simplemente controlando sus reservas, como sí se puede en WFQ.

#### 2.2.2.5 Virtual Clock (VC).

En [Zha91] se propone Virtual Clock (VC) como algoritmo de planificación apropiado para redes de alta velocidad. También [Cam00] muestra las propiedades y el funcionamiento de VC. Este algoritmo lleva a cabo las siguientes funciones:

- Controla la tasa media de transmisión de los flujos de datos.
- Hace que la utilización media de recursos por parte de cada usuario se corresponda con la productividad especificada.
- Proporciona protección entre los flujos individuales.
- Soporta servicios de prioridad multinivel.

En este algoritmo, cada flujo se modela mediante dos variables: tasa media (AR: average rate) e intervalo medio (AI: average interval). Esto significa que, al dividir la cantidad de datos recibidos durante cada período de tiempo AI, debería resultar AR. El

valor mínimo de AI es 1/AR. En ese caso, estaríamos ante un flujo CBR. El valor máximo es evidentemente toda la duración del flujo, con lo que tendríamos un patrón de tráfico totalmente incontrolado. Un buen valor de AI debe permitir un control de flujo por parte de la red, a la vez que debe poder reflejar posibles variaciones en la llegada de paquetes, dentro de cada AI. De esta forma la tasa media medida sobre cada periodo AI, permanecerá relativamente constante.

El algoritmo VC está inspirado por las redes TDM (Time Division Multiplexing). En estas redes, cada usuario tiene garantizada una tasa de transmisión, al reservar determinados slots, dentro de una trama, para transmitir sus paquetes. Esta reserva es fija, con lo cual es posible que se desaprovechen slots al no haber suficientes paquetes, pertenecientes al flujo correspondiente, listos para ser transmitidos. Esta limitación implica que sólo soporte, adecuadamente, tráfico CBR. Por el contrario, lo flujos están perfectamente aislados entre sí, y su productividad está garantizada.

VC intenta conseguir las mismas ventajas ofrecidas por TDM, a la vez que intenta preservar las ventajas de la multiplexación estadística. Por tanto, la red debería asignar slots a los flujos bajo demanda y regular el uso de los recursos únicamente cuando aparecen conflictos en la demanda, así se garantiza la productividad que reservó cada flujo. Un sistema TDM regula el uso de recursos mediante un reloj de tiempo real: todos los usuarios de los canales envían datos por turnos cuando el reloj avanza. Un sistema multiplexado estadísticamente puede emplear un reloj virtual de una forma similar.

Al hacer que un flujo de datos VBR sea similar a un canal TDM, imaginamos que los paquetes del flujo llegan espaciados por intervalo constante en tiempo virtual, de manera que la llegada de cada paquete indica que ha pasado un slot de tiempo. Se puede asignar a cada flujo de datos un reloj virtual, que avanza con cada llegada de un paquete de ese flujo. Si se hace que el paso de reloj se corresponda con el tiempo entre dos paquetes, el valor del reloj virtual denotará el tiempo de llegada esperado para el paquete entrante. Para imitar el orden de transmisión de un sistema TDM, se hace que cada nodo marque los paquetes de cada flujo con su valor de reloj virtual. Las transmisiones de paquetes se ordenan de acuerdo con los valores marcados, como si la marca de reloj virtual fuese el número de slot en un sistema TDM.

La implementación real del algoritmo emplea dos variables en lugar de una sola para representar el tiempo virtual: VirtualClock, y auxVC, para controlar que el flujo se atiene a los valores especificados de AI y AR. Cada conmutador lleva a cabo las dos siguientes funciones básicas:

**Avance de datos**: hay una cola de paquetes por cada puerto de salida. Los paquetes se sirven de la siguiente forma:

1. Al recibir el primer paquete del flujo i:
VirtualClock $_i \leftarrow$ auxVC $_i \leftarrow$ tiempo real

2. Al recibir cada paquete del flujo i:
   (a) $auxVC_i \leftarrow max($ tiempo real, $auxVC_i )$
   (b) $VirtualClock_i \leftarrow (VirtualClock_i + Vtick_i )$
   $auxVC_i \leftarrow (auxVC_i + Vtick_i )$
   (Para paquetes de tamaño constante, $Vtick_i = 1/AR_i$ paquetes/seg )
   (c) Marcar el paquete con el valor de $auxVC_i$

3. Insertar el paquete en su cola de salida. Los paquetes se insertan y se sirven según el orden creciente de sus valores marcados.

**Monitorización de los flujos:** El conmutador calcula una variable de control $AIR_i = AR_i \times AI_i$ cuando se establece cada flujo i. Al recibir cada conjunto de $AIR_i$ paquetes del flujo i, el conmutador efectúa las siguientes comprobaciones:

- Si ($VirtualClock_i$ – tiempo real) > T, donde T es un umbral de control, se envía un mensaje de aviso a la fuente del flujo. La diferencia entre $VirtualClock_i$ y el tiempo real es una medida de cuánto se ajusta la tasa a la que está transmitiendo un flujo a lo especificado. Cuanto mayor es esta diferencia, mayor es la diferencia entre las tasas especificadas y la tasa efectiva de transmisión. Dependiendo de cómo reaccione la fuente, puede que sean necesarias más acciones de control.

- Si ($VirtualClock_i$ < tiempo real), $VirtualClock_i \leftarrow$ tiempo real. En este caso, el flujo i ha transmitido por debajo de la tasa especificada. Al sincronizar $VirtualClock_i$ con el tiempo real, se evita que un flujo vaya acumulando indefinidamente sus slots de transmisión, para en un momento dado, saturar la red enviando toda la información de golpe. Esta acumulación de slots sólo puede darse dentro de un intervalo AI, no entre varios, de manera que se permite una cierta variabilidad en la transmisión de datos.

En este momento, el conmutador también sincroniza los valores de VirtualClock y auxVC, siempre y cuando esto no haga que los paquetes del mismo flujo sean servidos fuera de orden (es decir, cuando el enlace de salida del flujo i

está desocupado, o el paquete que se está transmitiendo tiene un valor de marca mayor que $auxVC_i$ ):

$$auxVC_i \leftarrow VirtualClock_i \quad \textbf{( 2.12 )}$$

Al encolar los paquetes según el orden de sus marcas de tiempo, hace que los paquetes procedentes de flujos distintos se sirvan muy intercalados entre sí, como en un sistema de servicio Round Robin.

La red puede establecer un orden de prioridades entre los flujos, sustituyendo (tiempo real) por (tiempo real – P) en el algoritmo anterior, siendo P un cierto valor que representa la prioridad. Este valor P, debe ser lo suficientemente grande para que sea capaz de separar los paquetes, primero los paquetes pertenecientes a flujos de mayor prioridad, y luego los paquetes de menor prioridad. Para los paquetes de tráfico best-effort se escoge un valor para P de $-\infty$, de manera que su marca vale $\infty$, y por tanto siempre estarán almacenados al final de la cola de servicio. Estos paquetes sólo utilizará los recursos que sobren, pues antes se atenderán los flujos que necesitan garantías de prestaciones.

Se ha visto que VC ofrece buenas propiedades de productividad y aislamiento entre flujos, así como un reparto justo del ancho de banda. Sin embargo, su mayor inconveniente es su complejidad de implementación. Otro problema es que el uso de un reloj de referencia implica que éste no puede ser puesto a cero hasta que el sistema esté inactivo. Por tanto, a menos que se empleen registros muy grandes para almacenar los tiempos virtuales, existe un problema potencial de desbordamiento numérico de los buffers si el sistema permanece ocupado durante un período largo de tiempo. En ese caso, el reloj virtual vuelve a contar desde cero, y esto lleva a una inestabilidad del algoritmo. En la Figura 2.6 se muestra un ejemplo del funcionamiento de VC frente a First Come First Serve (FCFS). El algoritmo FCFS funciona atendiendo al orden de llegada de los paquetes al conmutador, el orden en que llegan los paquetes es el orden en que son servidos.

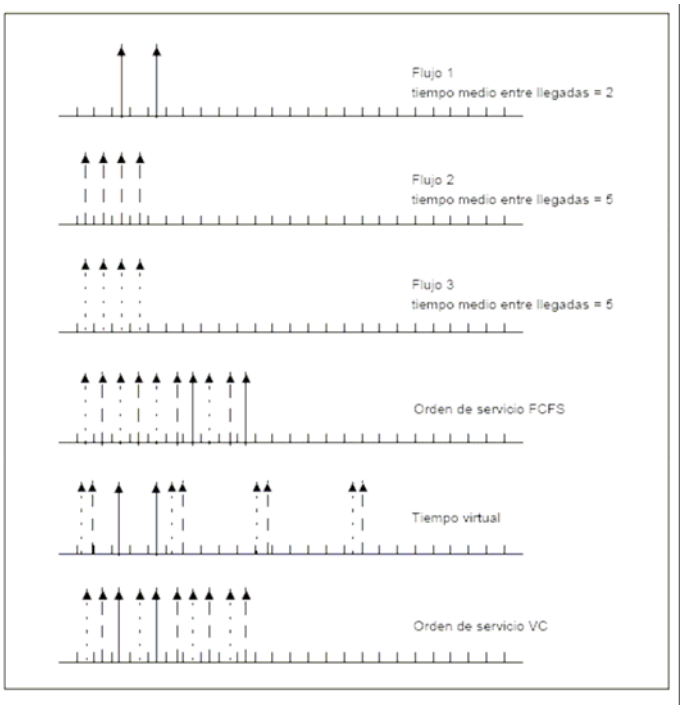



**Figura 2.6 Ejemplo de funcionamiento de VC frente a FCFS.**

### 2.2.3 Consideraciones finales.

Como se ha explicado al inicio de esta sección, un planificador ideal debe proporcionar una buena latencia y ser simple.

Los algoritmos "sorted priority", como WFQ, $WF^2Q$, SCFQ y VC ofrecen un buen retardo, pero sufren dos problemas principales. El primer problema es que requieren el cálculo de una etiqueta y la inserción en una lista ordenada de paquetes para transmitir. Por tanto, tienen la complejidad del algoritmo de inserción en la lista, como mínimo, 0(log(N)), siendo N el máximo número de paquetes de la cola, y 0(log(J)) si los buffers no son compartidos, con J el número de flujos activos. El segundo problema reside en el cálculo de la etiqueta, que es una función creciente del tiempo y depende de una referencia de tiempo virtual común, que refleja el valor de las etiquetas anteriores, y por tanto no se puede reiniciar hasta que el sistema esté vacío y todas las sesiones desocupadas. Esto, aunque es finito, puede dar lugar a periodos demasiado largos. Los contadores van aumentando durante estos periodos, y se pueden llegar a desbordar si no tienen la suficiente capacidad. Por tanto, para la implementación práctica de los algoritmos de prioridades, se debe utilizar hardware de gran velocidad que realice las ordenaciones, y se necesita de unidades de punto flotante para el cálculo de etiquetas. Todo esto hace que la implementación de estos algoritmos sea muy costosa, sino imposible.

La complejidad de los algoritmos basados en *frames* es menor. Al asignar un quantum igual o mayor que la MTU, y manteniendo una lista de flujos activos, se puede enviar, al menos, un paquete de cada flujo en cada vuelta, y no hay que recorrer varias veces la lista para ganar créditos suficientes para hacer el envío. Sin embargo, esta buena complejidad puede quedar empañada por una equidad del algoritmo que se desvía de la obtenida en los algoritmos "sorted priority". Dentro de este tipo de algoritmos encontramos DRR/WRR, Aliquem DRR y CBFQ. Para el algoritmo CBFQ, el valor de los contadores está acotado, lo cual impide el desbordamiento de los mismos. Además, la complejidad adicional de la lista ordenada es menor, pues sólo se debe insertar uno o unos pocos elementos en la lista que ya está ordenada, el identificador del canal que ha sido servido y quizá algún identificador de canal que se ha activado durante el último periodo. Para Aliquem DRR añade la complejidad espacial y computacional de la estructura de listas utilizada, aunque puede utilizarse hardware adicional para limitar el impacto producido por dicha estructura de listas.

# 3 OBJETIVOS Y METODOLOGÍA.

Como ya se ha comentado anteriormente, existen multitud de algoritmos y tecnologías de red, que han sido probados en multitud de ocasiones, y que se han comparado y obtenido multitud de índices de prestaciones. Sin embargo, en esos trabajos previos no se ha tenido en cuenta la influencia del control de flujo sobre los mismos.

Con este trabajo, se pretende obtener unos algoritmos que trabajen junto con dicho control de flujo, de forma que sea posible aprovechar todo el potencial que proporcionan las redes actuales, y así, poder saber cuáles son más apropiados dentro del nuevo entorno sobre el que trabajamos.

## 3.1 OBJETIVOS DEL PROYECTO.

El objetivo principal de este proyecto ha sido obtener unos algoritmos de planificación con control de flujo a nivel de enlace de red, dichos planificadores se basan en los algoritmos tradicionales. También se muestra el comportamiento de los planificadores mediante simulación.

Para lograr este objetivo general, inicialmente se plantearon una serie de metas u objetivos parciales que se pueden resumir de la siguiente forma:

- **Revisión del estado del arte de las tecnologías de red y los algoritmos de planificación.**

  Puesto que el trabajo desarrollado se ha centrado en ciertos algoritmos de planificación, es fundamental conocer con cierto detalle las características generales de éstos, y de las tecnologías de red sobre las que se pueden implementar. Se trata, por tanto, de estudiar la estructura y el funcionamiento de las tecnologías de red actuales, que proporcionan QoS, y comprender qué modificaciones son necesarias para obtener el mejor rendimiento del conjunto formado por la tecnología de red y el algoritmo de planificación, de forma que, los nuevos algoritmos de planificación se adapten al nuevo control de flujo, de una forma eficiente y simple.

- **Estudio de las modificaciones necesarias en los algoritmos.**

  Una vez conocidas las características de los algoritmos de planificación, debemos ver qué modificaciones son necesarias, de forma que, estos algoritmos funcionen de manera adecuada en el nuevo entorno. Esto

cambiará la forma de trabajar de los algoritmos, que ahora actúan junto al control de flujo.

- **Aprendizaje y manejo de un simulador de redes de interconexión y algoritmos de planificación.**

  El estudio no se realizará sobre sistemas reales, es decir una red de estaciones de trabajo real, sino que estará basado en el uso de un simulador. Será preciso pues conocer su estructura y funcionamiento, teniendo en cuenta además que será preciso realizar cambios en su código.

- **Creación de los nuevos algoritmos de planificación.**

  Tras estudiar qué modificaciones son necesarias en los algoritmos de planificación que hemos mostrado y seleccionar la tecnología de red, se procederá a la obtención de nuevos módulos del simulador que contendrán las nuevas versiones de los algoritmos.

- **Obtención y análisis de resultados.**

  Una vez que ya dispongamos de todos los conocimientos necesarios sobre la teoría y el simulador, se debe proceder a obtener resultados. Para ello, hay que crear los ficheros de configuración y pruebas necesarios y realizar las simulaciones. Una vez obtenidos los resultados, hay que obtener las gráficas correspondientes y realizar un análisis exhaustivo de las mismas, para así, poder explicar y comparar los resultados obtenidos.

## 3.2 METODOLOGÍA.

Para alcanzar los objetivos parciales, y con ello el objetivo principal del proyecto, se plantearon al inicio del mismo una serie de tareas adecuadamente programadas según la metodología habitual en este tipo de trabajos. Se indican en este punto cuáles han sido y en qué han consistido estas tareas:

- **Consulta bibliográfica relacionada con todos aquellos aspectos que han tenido que ver con el estudio realizado.**

  Para cubrir el primero de los objetivos antes planteados, se hace imprescindible consultar bibliografía especializada. Esas consultas deben dirigirse en dos direcciones bien distintas. Por un lado, aquella que permita localizar la información necesaria para adquirir los conocimientos requeridos

sobre las características de las tecnologías de red. Por otra parte, la que permita encontrar información, características e implementaciones, normalmente en pseudo-código, a cerca de los algoritmos de planificación. En ambos caso habrá que realizar las consultas sobre algunos libros y sobre todo artículos de revistas especializadas o actas de congresos.

- **Estudio de las modificaciones que se deben aplicar.**

  Una vez revisada la bibliografía, podemos ver qué soluciones se pueden aplicar en cada algoritmo, pues estas modificaciones dependen de las características particulares de cada uno de ellos.

- **Elección de la tecnología de red para la evaluación.**

  De la bibliografía consultada, obtendremos una tecnología que sea lo suficientemente genérica. Así se podrán generalizar los resultados obtenidos para otras tecnologías que incorporen un control de flujo similar a la utilizada. Además, esta tecnología deberá ser lo suficientemente concreta para poder modelarla sobre un simulador.

- **Revisión del código y aprendizaje del uso del simulador de redes de interconexión y algoritmos de planificación.**

  Se deberá realizar un estudio de cada uno de los módulos en los que se organiza el simulador para conocer el nivel de detalle con el que se han reflejado aspectos como los algoritmos de planificación original, la arquitectura del conmutador o las características de los enlaces. En definitiva, y como hay que realizar cambios importantes en el código, se debe comprender la estructura y funcionamiento del simulador.

- **Estudio de las modificaciones necesarias y modificación del código.**

  Partiendo de la tecnología de red seleccionada, podemos estudiar qué modificaciones son necesarias en los algoritmos de planificación que hemos mostrado. Posteriormente, este estudio resultará en la obtención de nuevos módulos del simulador, que contendrán las nuevas versiones de los algoritmos.

- **Aprendizaje de las herramientas y entornos para la ejecución y captura de resultados de simulaciones masivas.**

  Dado el elevado número de simulaciones que se han de realizar, es imprescindible hacerse con el manejo de los distintos entornos en los que se van a ejecutar las simulaciones, así como de las herramientas para extraer los resultados.

- **Desarrollo del proceso de evaluación y análisis de resultados.**

  Haciendo uso del simulador, se obtendrá un amplio conjunto de resultados. A partir de esos resultados, y del conocimiento adquirido sobre los algoritmos simulados, así como de las comparaciones entre los mismos, se deberán extraer todas las conclusiones que sea posible.

# 4 ADAPTACIÓN DE ALGORITMOS DE ARBITRAJE A ENTORNOS CON CONTROL DE FLUJO A NIVEL DE ENLACE.

Una vez visto el estado del arte, podemos decir que hay una clara tendencia en las nuevas tecnologías de red, que es incluir mecanismos que proporcionen QoS a las aplicaciones. Específicamente, la tendencia es separar el tráfico en un conjunto limitado de canales virtuales, realizando el arbitraje y el control de flujo a nivel de dichos canales virtuales. Esto conlleva que haya canales virtuales que, pese a tener paquetes para transmitir, no puedan hacerlo, al estar bloqueados debido al control de flujo, al mismo tiempo que otros canales virtuales pueden transmitir con toda normalidad. Sin embargo, hay que reseñar que los algoritmos tradicionales no están pensados para las nuevas tecnologías de control de flujo a nivel de enlace, ya que en principio no estaban pensados para trabajar con canales virtuales, sino con puertos, y no tienen en cuenta la posibilidad de deshabilitar las colas basándose en la información del control de flujo, lo cual da lugar a una serie de cuestiones relativas al funcionamiento, que se deben tratar.

En primer lugar, vamos a variar la definición de cola activa, y se habla ahora de canales virtuales activos:

- Un CV se considera que está activo sólo cuando tiene paquetes almacenados y suficientes créditos para transmitir el primer paquete de la cola del CV.
- Un CV estará inactivo si no cumple una de las dos condiciones anteriores.
- El conjunto de CVs activos puede cambiar cuando un paquete llega a un CV, cuando un paquete deja el CV, una vez seleccionado para ser transmitido, o cuando se recibe un paquete de control de flujo con créditos.

En los algoritmos “frame based”, que disponen de un contador de peso para cada CV, se puede ver un comportamiento irregular si al bloquearse el CV, y a lo largo de todas las vueltas que vaya dando el algoritmo a los contadores, se le va asignando el peso correspondiente a las vueltas en las cuales ha permanecido inactivo. Al no poder utilizar los pesos asignados va haciendo acopio de recursos que posteriormente podrá utilizar una vez que se active el CV, con el correspondiente perjuicio que se les produce a los demás CVs llegado dicho momento.

Para los algoritmos “sorted priority”, en los cuales se etiquetan los paquetes según acceden al sistema, surge el inconveniente de qué hacer cuando el CV queda bloqueado por algún motivo, y todavía quedan paquetes en sus colas. Estos paquetes ya están etiquetados para salir del sistema en un momento dado pero al quedar parados, sus

etiquetas van envejeciendo, de forma que cuando el CV vuelva a activarse, el planificador elegirá sus paquetes antes que los paquetes de otros CVs, y el CV utilizará los recursos que no utilizó antes. Esto tiene un efecto negativo sobre las demás colas, que no podrían hacer uso de sus recursos en igualdad de condiciones.

Ante las situaciones descritas anteriormente, las actuaciones aplicadas para evitar dichas anomalías atienden a dos hechos ampliamente aceptados por los investigadores: conseguir que cuando un CV se bloquea por causa del control de flujo su ancho de banda correspondiente se reparta entre el resto de canales virtuales, y también, que un CV desbloqueado no debe tomar ventaja del tiempo que estuvo bloqueado. Esta es la aproximación que se toma en Advanced Switching Interconnection (ASI).

Para corregir el problema asociado a los algoritmos basados en "frames", nuestra propuesta es aplicar la tajante solución de quitarle todos los créditos al contador del CV que se desactive. Bien cierto es que se le podrían mantener los créditos obtenidos hasta el momento y no añadirle más en las sucesivas rondas, pero creemos que así se obtiene una mejor equidad para todos, y que cuando un canal no puede emitir paquetes pierde su derecho a hacerlo a posteriori.

La adaptación de los algoritmos basados en prioridades para evitar el funcionamiento anómalo que muestran, pasa por adoptar la siguiente idea: una vez que el flujo vuelva a activarse se vuelven a etiquetar todos los paquetes que hay en la cola del mismo. De esta forma todos los paquetes tienen unas etiquetas equiparables entre sí, y no pueden obtener ventaja los paquetes que han sido retenidos. Esta operación tiene un coste computacional elevado, pero puede ser factible al pensar que sólo hay que volver a etiquetar unos pocos paquetes.

Para ilustrar más fácilmente el funcionamiento de los algoritmos, se ha desarrollado un ejemplo en el cual se representa gráficamente los instantes más representativos del funcionamiento del conmutador a lo largo del tiempo, aplicando valores concretos a los parámetros que componen la simulación, para que se pueda ver cómo va evolucionando el ejemplo gráfico. Primero vamos a indicar la disposición de cada unos de los componentes, y el significado de los valores; luego, se explicará cómo evoluciona y se detallarán los momentos más importantes para nuestro estudio.

## 4.1 MODELO DEL PLANIFICADOR.

En la Figura 4.1 se ve el modelo utilizado para ilustrar los ejemplos, del cual se detalla a continuación el significado y la simbología. En el encabezado de la figura, se muestra qué algoritmo se está aplicando, junto con los pesos de los canales virtuales y el tamaño de los paquetes. Dicho tamaño se mantiene constante para simplificar las operaciones, no obstante, también se puede utilizar un tamaño variable de los mismos. Aquí se muestran sólo dos canales virtuales, sin embargo, en un sistema real habrá varios canales virtuales más. Para mostrar la evolución del sistema son suficientes dos canales virtuales, pero el funcionamiento para n canales es el mismo.

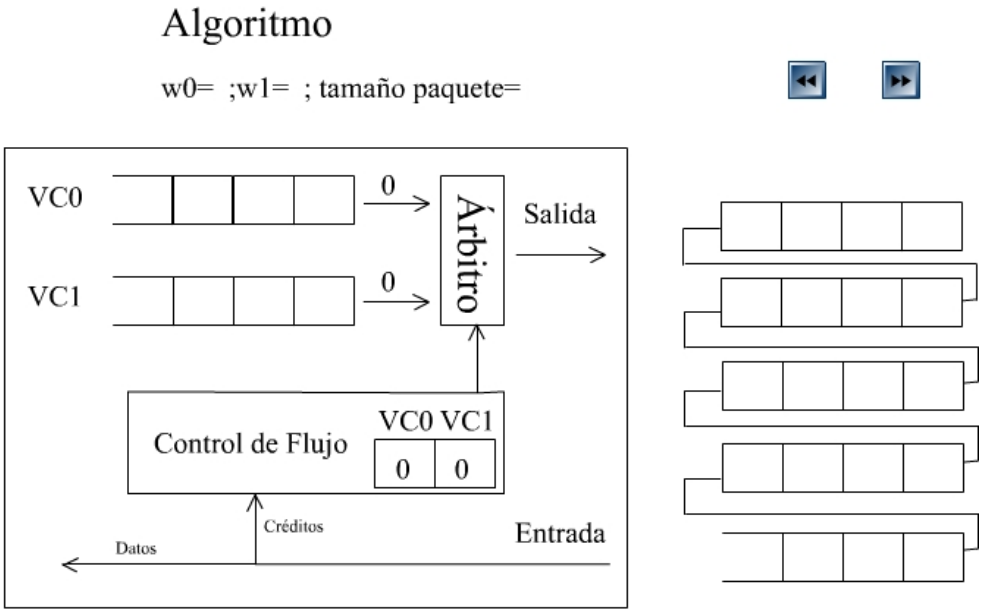



**Figura 4.1 Modelo del planificador.**

La parte que nos interesa se encuentra recuadrada justo debajo del título. En ésta se muestran:

- Los buffer de los CVs, sus identificadores (VC0, VC1, …)y los tres primeros paquetes almacenados, aunque puede haber más paquetes dependiendo de la configuración dada.
- El contador deficitario de pesos correspondiente a cada uno de los CVs, que se asigna cuando se completa una vuelta al conjunto de canales activos.
- El árbitro encargado de elegir qué CV será el próximo en transmitir.
- El control de flujo, con el número de créditos que le restan al CV por transmitir antes de desactivarse.
- El enlace de salida de paquetes, por donde se envían los paquetes hacia el siguiente conmutador o hacia el destino.

- El enlace de entrada, en el cual nos interesa la llegada de créditos del conmutador al que le enviamos los paquetes. Estos créditos se nos van proporcionando conforme dicho conmutador va disponiendo de espacio para nuestros paquetes.

En la parte derecha de la imagen se muestra el buffer de salida a falta de la lógica de control. En él se almacenan los paquetes a transmitir. Estos esperan el momento en que las señales eléctricas pueden ser emitidas. En el ejemplo no se eliminan los paquetes del mismo al ser enviados, así puede verse más fácilmente cuál es el orden en que se han servido cada una de las colas.

Para la realización de los ejemplos tomamos los siguientes valores. El tamaño máximo de paquete es cuatro, y los pesos de los canales virtuales son idénticos, con valor seis. El contador deficitario y los créditos de control de flujo son inicialmente cero, así vemos cómo se desarrolla todo desde el principio, y suponemos que hay paquetes para transmitir en ambos canales virtuales. Los paquetes de cada canal se diferencian por el color utilizado, así es más fácil distinguirlos en el buffer de salida. Para indicar la activación del CV correspondiente, el identificador del mismo cambiará a color magenta, y en caso de bloqueo del mismo su contador se pondrá en color rojo. Para indicar la llegada de créditos desde el siguiente conmutador, las palabras “Entrada” y “Créditos” pasarán a color magenta y posteriormente, se actualizarán los contadores de créditos del control de flujo, según corresponda.

En las próximas secciones se aplicarán estas consideraciones a los algoritmos presentados en el Capitulo 2. Con ello se obtendrán nuevos algoritmos que funcionan de manera adecuada en nuestro entorno de trabajo. Hemos llamado a estas nuevas versiones “credit aware”, haciendo referencia a un control de flujo basado en créditos. Aún así, son igualmente válidas para cualquier otro mecanismo de control de flujo, simplemente con ligeras modificaciones.

Los aspectos fundamentales que se han tomado en cuenta para adaptar estos algoritmos en los nuevos “credit aware” son:

- Las simplificaciones que pueden hacerse al considerar un conjunto reducido y específico de canales virtuales, en lugar de un gran conjunto de flujos.
- La propia interacción entre los canales inactivos y el mecanismo de control de flujo, respecto a la acumulación de ancho de banda para un uso futuro por parte de los primeros.

## 4.2 ALGORITMOS BASADOS EN “FRAMES”.

A continuación, se muestra un ejemplo del funcionamiento del algoritmo DRR original, en el cual no se han tenido en cuenta los efectos del mecanismo de control de flujo. Así, posteriormente, se podrán comprobar mejor los efectos de las modificaciones introducidas. Algunos pasos de la secuencia de la simulación no se detallan debido a la poca relevancia de éstos, otros más relevantes se explicarán con más detalle.

Se supone que al empezar el ejemplo hay 2 CVs, y aunque ambos tienen paquetes en sus buffers de entrada están desactivados al no disponer de créditos. Una vez que reciben los créditos del control de flujo es posible activar VC0. Una vez activo el árbitro le asigna quantum y como dispone también de créditos, puede enviar su primer paquete y actualiza dichos contadores en proporción al tamaño de paquete. Cuando vamos a enviar el segundo paquete el CV no dispone de suficiente quantum, por lo cual pasamos al siguiente canal (VC1). El quantum restante queda acumulado en el contador deficitario y los créditos en su contador correspondiente.

En el siguiente paso sucede lo mismo para VC1, y volvemos a seleccionar VC0. Este último ahora posee suficientes créditos y quantum para realizar el envío de dos paquetes. Al pasar al siguiente canal, VC1, también se procesan dos paquetes. Este funcionamiento es debido a que el algoritmo DRR original, y por tanto, nuestra nueva versión “credit aware” (DRR-CA), emiten paquetes siempre que el quantum, y si hablamos de DRR-CA también los créditos, lo permitan. En la siguiente ronda del algoritmo sale un paquete por cada CV tal y como se explica al principio. A continuación sucede un caso interesante.

Nuestro primer caso interesante puede verse en la Figura 4.2. Tiene lugar cuando el árbitro vuelve a seleccionar a VC0 para transmitir, y puesto que tiene créditos y quantum, emite el paquete número nueve del buffer de salida. Cuando se dispone a emitir el siguiente paquete, aunque tiene quantum suficiente para realizar la operación, el control de flujo se ha quedado sin créditos y por tanto, bloquea a VC0, y pasa a activar el siguiente canal VC1.

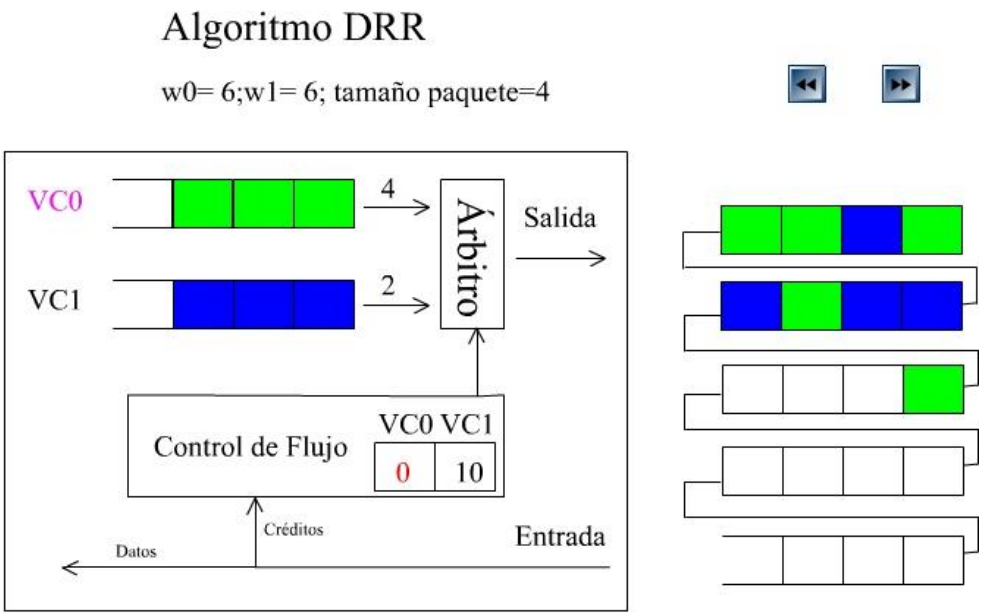



**Figura 4.2 VC0 sin créditos.**

Durante el periodo en que VC0 permanece bloqueado por la acción del control de flujo, su contador deficitario se sigue incrementando cuando le llega su turno en una cantidad igual al quantum asignado a este CV, aunque no puede hacer uso de ellos al estar bloqueado por el mecanismo de control de flujo. Los demás canales hacen uso de los recursos que le corresponderían, en este caso VC1, siempre y cuando, éste no agote sus créditos disponibles. Como esto no ocurre, VC0 no vuelve a transmitir hasta que le llegan nuevos créditos al control de flujo, provenientes del conmutador siguiente. Este control permite restringir el paso de los paquetes de un conmutador a otro de forma que no se produzcan perdidas, al regular el ritmo del envío a lo largo de la red. Cuando VC0 vuelve a tener créditos para poder enviar los paquetes que le quedan, ha acumulado un quantum de dieciséis, que le permiten utilizar los recursos que no había utilizado antes (Figura 4.3). Ahora VC0 envía todos los paquetes que le quedan de una sola vez, y aún le queda un quantum de cuatro, suficiente para enviar otro paquete.

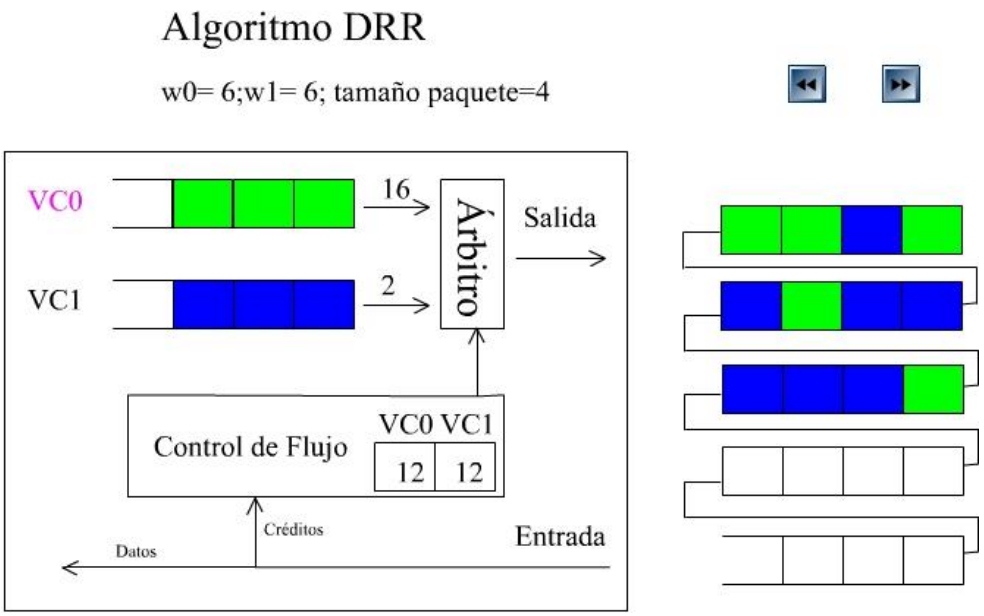



**Figura 4.3 VC0 tras recibir créditos.**

Para el resto del ejemplo, llegan créditos suficientes para emitir los paquetes restante, los cuales, llegado el momento, se acaban. En la Figura 4.4 podemos ver que VC0 ya ha transmitido todos sus paquetes, y por tanto, se ha desactivado. Luego es VC1 el que termina de emitir sus paquetes, y se va a desactivar el canal debido a la falta de paquetes. En este caso, el crédito disponible es cero, aunque sería posible que fuese distinto de dicha cantidad, y esa cantidad sí que se guarda para sucesivas rondas.

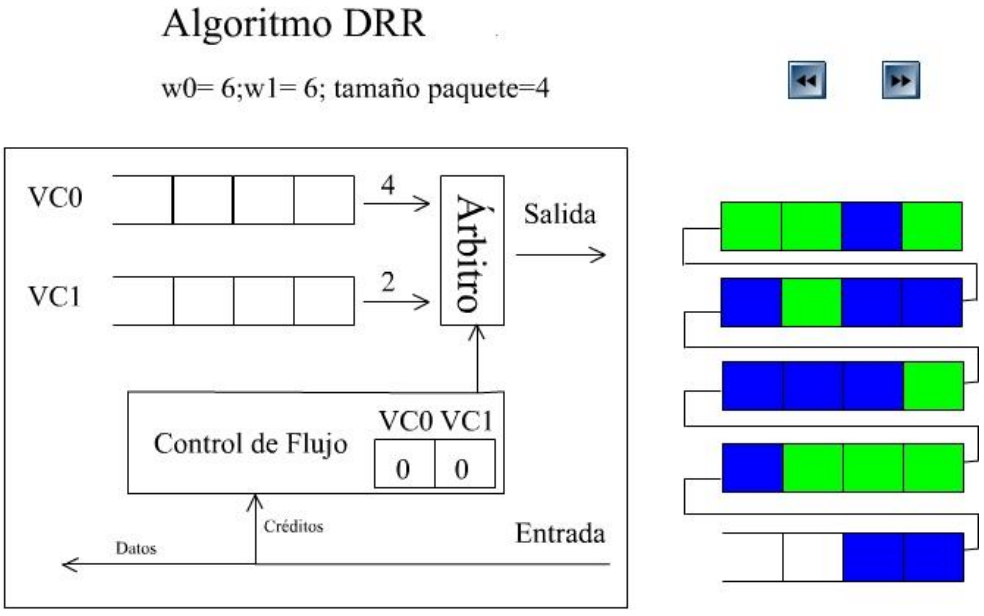



**Figura 4.4 DRR al finalizar.**

Como puede verse en la figura anterior, los paquetes 10, 11 y 12 son enviados por VC1, mientras VC0 está desactivado a causa del control de flujo. Por el contrario,

VC0 emite los paquetes 13, 14 y 15, aprovechando el acopio de recursos que se ha producido mientras permanecía desactivado, lo cual afecta de manera negativa al tráfico de VC1. Queda pues comprobado que hace falta algún mecanismo que controle esta situación creada por el control de flujo. A continuación mostramos las soluciones adoptadas en los diferentes algoritmos para mitigar dicho efecto.

### 4.2.1 Deficit Round Robin Credit Aware (DRR-CA).

El algoritmo DRR-CA que mostramos difiere del algoritmo DRR original en los siguientes puntos: Un CV se considera que está activo sólo cuando tiene, al menos, un paquete almacenado y suficientes créditos para transmitir el primer paquete de la cola del CV. Cuando se emite un paquete, se selecciona el siguiente CV cuando alguna de las siguientes condiciones se cumple:

- No hay más paquetes que transmitir en el CV actual o los créditos están agotados. Entonces, el CV se desactiva y el contador de créditos deficitarios se pone a cero.
- El quantum restante no es suficiente para enviar el siguiente paquete. En este caso, el contador de créditos deficitarios se actualiza con el peso acumulado en ese instante.

En el siguiente ejemplo también se han omitido algunos pasos de la secuencia, y se muestran aquellos pasos que tienen mayor interés para nuestro estudio. Se utilizarán las mismas suposiciones que para el algoritmo original. Al empezar, con los créditos que llegan al control de flujo es posible activar el VC0. Una vez activo el VC0, el árbitro le asigna quantum, y al disponer también de créditos puede enviar su primer paquete y actualiza dichos contadores en proporción al tamaño de paquete. Cuando vamos a enviar el segundo paquete no dispone de suficiente quantum, por lo cual pasamos al siguiente canal (VC1). El quantum restante queda acumulado en el contador deficitario y los créditos en su contador correspondiente, tal y como puede verse en la Figura 4.5.

En el siguiente paso sucede lo mismo para VC1, y volvemos a seleccionar VC0, sólo que esta vez posee suficiente quantum y créditos para realizar el envío de dos paquetes. Al pasar al siguiente canal, VC1, también se procesan dos paquetes. Este funcionamiento es debido a que el algoritmo DRR original, y por tanto, nuestra nueva versión, emite paquetes siempre que el quantum, y si hablamos de la versión “credit aware” también los créditos, lo permita. Luego sale un paquete por cada CV como se explica al principio. Seguidamente sucede un caso interesante.

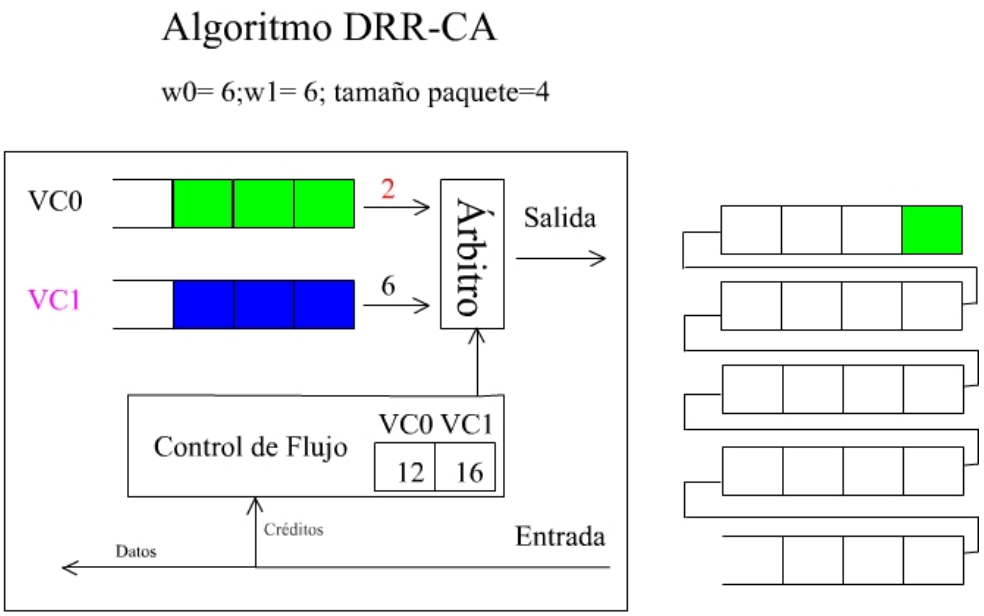



**Figura 4.5 CV0 sin quantum.**

El caso interesante puede verse en la Figura 4.6, y tiene lugar al volver al canal VC0. Puesto que tiene créditos y quantum, el VC0 transmite el paquete número nueve del buffer de salida, pero cuando se dispone a emitir el siguiente paquete, aunque tiene quantum suficiente para realizar la operación, el control de flujo se ha quedado sin créditos y por tanto bloquea a VC0, le quita el quantum restante y pasa a activar el siguiente CV. Simplificando, el cero del control de flujo obliga a poner a cero el quantum. Es a partir de este momento cuando hay diferencias entre el algoritmo original, explicado en el apartado anterior, y el algoritmo "credit aware" que estamos viendo.

Durante el periodo en que VC0 permanece bloqueado por la acción del control de flujo, los demás canales hacen uso de los recursos que le corresponderían a VC0 y la parte correspondiente de ancho de banda compartido por el mismo. Por ejemplo, si hubiese 4 CVs, el ancho de banda asignado a VC0 se repartiría entre esos 4 CVs. En este caso es VC1 el que se beneficia de dicha situación siempre y cuando éste no agote sus créditos disponibles. Como esto no ocurre, VC0 no vuelve a transmitir hasta que le llegan nuevos créditos al control de flujo, provenientes del conmutador siguiente. El funcionamiento volverá a ser el descrito al principio de este apartado cuando VC0 se active. Aquí se encuentra la diferencia de funcionamiento entre ambas versiones del algoritmo DRR.

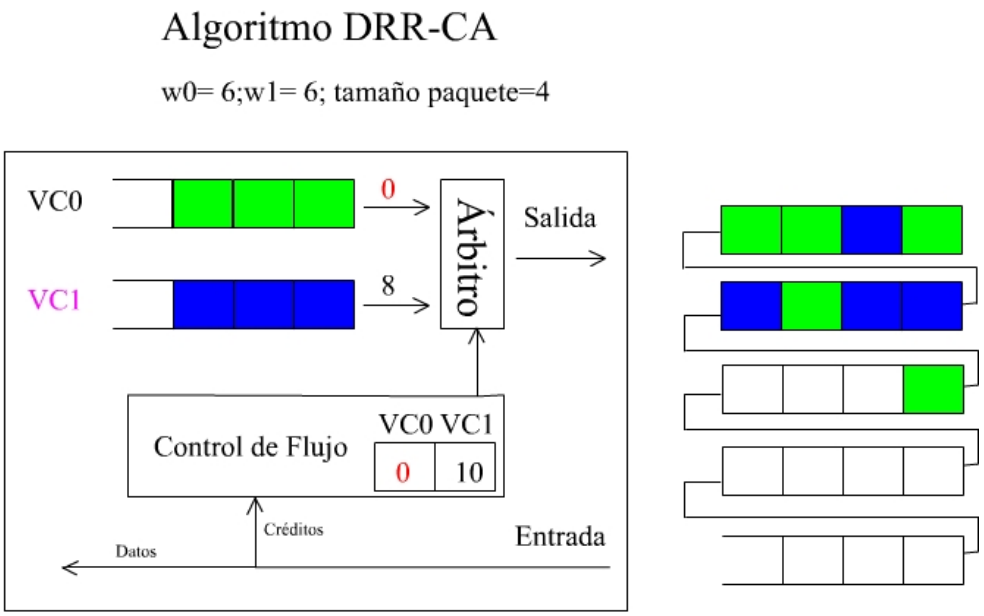



**Figura 4.6 CV0 sin créditos.**

Para el resto del ejemplo, vamos a suponer que llegan créditos suficientes para emitir los paquetes restantes. En la Figura 4.7 podemos ver que VC0 ya ha transmitido todos sus paquetes, y por tanto, se ha desactivado y su quantum se ha puesto a cero. Ahora es VC1 el que ha terminado de emitir sus paquetes, y el quantum que le resta debe ser puesto a cero, ya que se va a desactivar el canal debido a la falta de paquetes. En este caso, el crédito del control de flujo disponible es cero, aunque sería posible que fuese distinto de dicha cantidad, y esa cantidad sí que se guarda para sucesivas rondas.

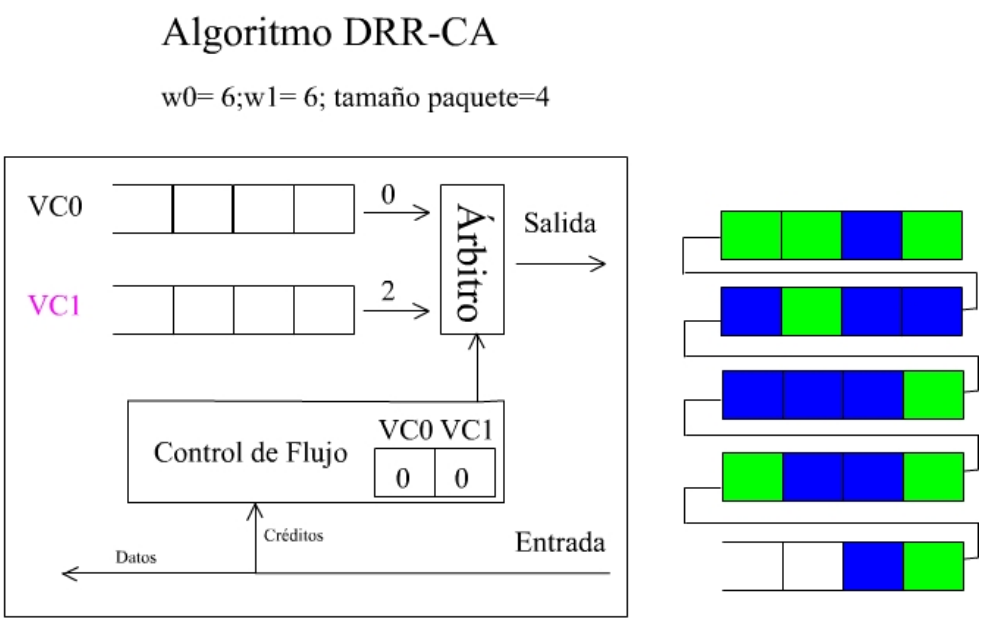



**Figura 4.7 CVs sin paquetes.**

Observando la Figura 4.7, se puede ver que los paquetes número diez, once y doce son los que ha transmitido VC1 aprovechando el bloqueo de VC0, luego ambos canales vuelven a emitir con la misma secuencia de paquetes que llevaban antes del bloqueo. En este caso, el tráfico de VC1 no se ve perjudicado por el tráfico de VC0.

### 4.2.2 Active List Queuing Method DRR Credit Aware (ALIQUEM DRR-CA).

La adaptación del algoritmo ALIQUEM DRR para tener en cuenta el control de flujo la hemos llamado ALIQUEM DRR-CA. Las modificaciones efectuadas son las siguientes:

- Si un CV está inactivo y llega un primer paquete, el paquete se inserta en su buffer, pero no se calcula cuando se emitirá hasta que se active el canal.
- Un CV estará inactivo hasta el momento en que tenga, al menos, un paquete en su buffer y disponga de créditos suficientes para la emisión del paquete, momento en que se activará el canal.
- Al llegar el primer paquete al buffer de entrada y el CV está activo, se almacena en el buffer, y se calcula dónde insertar el identificador del CV en la estructura de listas.
- Al llegar un paquete al buffer de entrada, que no es el primero de la cola, y el CV está activo, dicho paquete se guarda en el buffer. Cuando llegue al principio de la cola será cuando se calcula dónde insertar el identificador del CV en la estructura de listas.
- Cuando el CV se desactiva, se quita su índice de la estructura de listas.

Véase que en este caso no hay que preocuparse de la cantidad de quantum durante las iteraciones del algoritmo de planificación pues, gracias a la estructura de cola, se sabe en qué momento le tocará el turno, y si tiene suficiente quantum se insertará en la lista actual, sino en alguna de las posteriores. También se ha querido obtener una mejor equidad y retardo medio, para lo que se ha limitado el número de paquetes que se pueden transmitir en cada turno a uno, lo que da como resultado un tamaño de las ráfagas menor. Si al CV actual le quedan créditos después de transmitir su paquete, su identificador será insertado al final de la lista actual y, por tanto, los demás canales de esta lista podrán enviar su paquete correspondiente antes que el canal actual envíe su siguiente paquete.

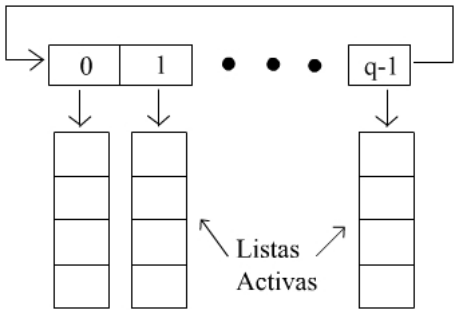



**Figura 4.8 Estructura de colas.**

Para el ejemplo gráfico sólo se muestra el funcionamiento de la estructura de colas mostrado en la Figura 4.8. La evolución del conmutador es similar al explicado en el algoritmo DRR-CA. Como ventaja de este algoritmo, se puede reducir el tamaño del quantum a uno y se mantiene el tamaño máximo de los paquetes en cuatro, aunque utilizamos paquetes de diferentes tamaños para no alargar innecesariamente dicho ejemplo. Retomando la expresión

$$\forall \mathbf{i},\ \phi_i \geq \frac{\overline{L_i}}{q-1} \qquad (2.6)$$

vemos que la longitud de la cola debe ser cinco, pues $1 \geq \frac{4}{q-1} \rightarrow q \geq 5$. También se podría utilizar para el tamaño máximo de paquetes cuatro, un quantum de dos, con lo que la anterior ecuación dará como resultado $q \geq 3$, pero se opta por la primera posibilidad, aunque es igualmente válida la segunda opción. Dado el gran número de colas que se utilizan y para mostrar una situación más real, incrementamos a ocho el número de canales virtuales que envían información, numerándolos desde VC0 hasta VC7. Para no dilatar el ejemplo, se parte de una situación intermedia, en la cual todos los VCs han enviado varios paquetes y el estado de la estructura de colas puede verse en la Figura 4.9.

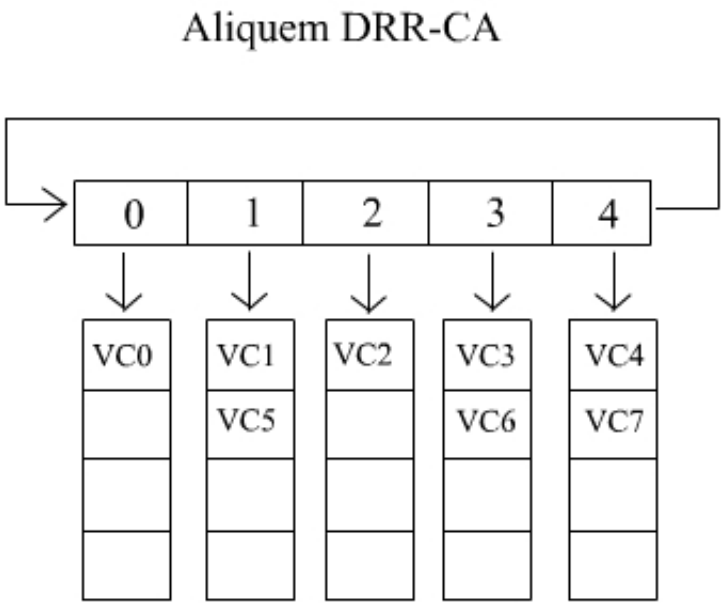



**Figura 4.9 Situación inicial.**

A continuación se analizará cómo va evolucionando la estructura de colas atendiendo los VCs según el orden mostrado en la misma a lo largo del ejemplo.

- Para VC0, supongamos que envía su paquete y se queda sin quantum suficiente para enviar el siguiente paquete. Si el siguiente paquete tiene un tamaño dos, como el quantum aumenta de uno en uno, se quita el identificador de VC0 de la lista 0 y se incluye en la lista 2 detrás de VC2. Una vez que lleguemos a dicha lista, ya dispondrá de suficiente quantum para poder emitir el siguiente paquete. Al no quedar más elementos en la lista 0, se pasa a la siguiente lista, la lista 1.

- En la lista 1 le toca el turno a VC1 que envía su paquete, y aún tiene suficiente quantum para enviar el siguiente paquete, así que el identificador de VC1 se pasa a la última posición de la cola activa y pasamos el turno a VC5. En la Figura 4.10 se muestra dos veces VC1 en la lista 1, no obstante, el identificador coloreado en rojo indica la posición antes de actualizar la posición de VC1, y en color azul la posición en que quedará insertada la referencia a VC1.

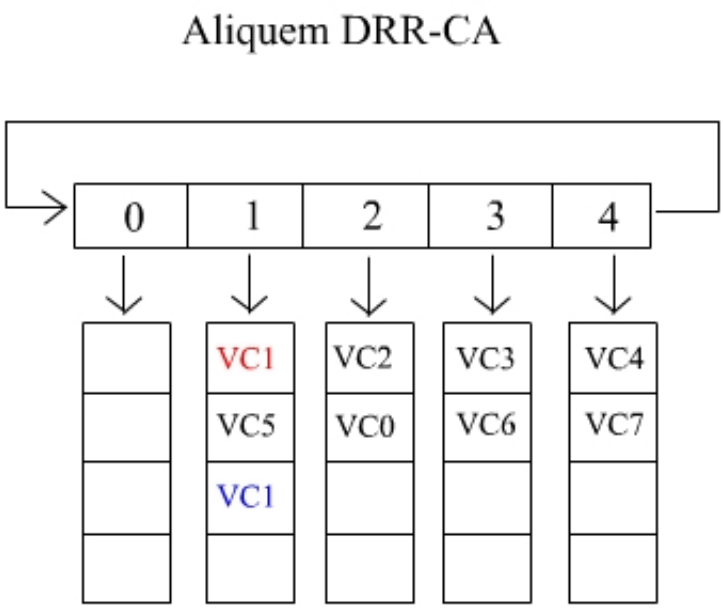



**Figura 4.10 Turno de VC1.**

- El siguiente canal al que le toca el turno es VC5, que transmite su paquete y mueve su identificador a la lista 4, que será cuando tenga suficiente quantum para continuar su transmisión debido al tamaño de su siguiente paquete.

- Al canal VC1 le vuelve a tocar el turno, enviará otro paquete y se quedará sin créditos en el control de flujo y se desactivará. En ese caso se elimina el identificador de VC1 de esta estructura de listas como resultado de su desactivación. Se volverá a incluir su identificador en la estructura de colas en el momento en que reciban créditos por parte del control de flujo. Será entonces cuando se inserte su identificador en el lugar que le corresponda.

- Continuamos con la siguiente lista, el canal VC2 recibe el turno y emite su paquete. Atendiendo al tamaño de su siguiente paquete, volverá a tener quantum tres listas después. La lista siguiente a la lista 4 es la lista 0, por eso VC2 se insertará en la lista 0. En la Figura 4.11 se ve cómo queda el ejemplo, teniendo en cuenta el código de colores explicado anteriormente.

No se continúa con el ejemplo porque las situaciones más representativas ya se han mostrado, y cualquier situación que se presente a continuación atenderá a alguna de las mostradas anteriormente.

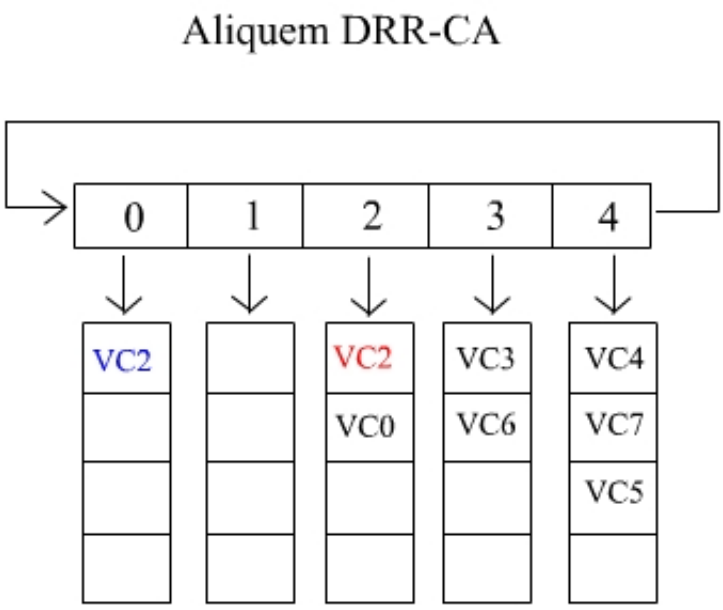



**Figura 4.11 Momento final del ejemplo.**

### 4.2.3 Credit-Based Fair Queuing Credit Aware (CBFQ-CA).

Las modificaciones realizadas en el algoritmo CBFQ para obtener la versión CBFQ-CA son las siguientes: Cuando un CV se activa, al cumplir las condiciones de tener créditos de control de flujo y paquetes para transmitir, se incluye su identificador en el conjunto de CVs que están activos. El conjunto de CVs activos se comporta de la misma manera que el algoritmo original, seleccionando el CV al que se le concederá el turno de la misma forma en la versión original y en la versión "credit aware". En caso de que el CV se quede sin créditos de control de flujo o sin paquetes, dicho CV es desactivado, se elimina su identificador del conjunto de CVs activos y se le quita su peso acumulado (quantum). En las siguientes iteraciones del algoritmo no se le asigna más peso hasta que se vuelva a activar el CV, y así no puede hacer acopio de recursos para utilizar posteriormente.

De no actuar de esta forma, si tras obtener a qué CV le toca el turno éste no puede transmitir su paquete debido al control de flujo, se habría perdido el tiempo invertido en el cálculo de qué CV es el próximo en emitir su paquete y la actualización de los contadores de pesos de los otros CVs, pues se debería volver a realizar todos los cálculos anteriormente mencionados para obtener otro CV que quizá tampoco pueda emitir sus paquetes.

## 4.3 ALGORITMOS BASADOS EN PRIORIDADES.

Para mostrar el efecto indeseado que se produce en los algoritmos basados en prioridades, se muestra en un primer ejemplo del algoritmo WFQ qué ocurre si no volvemos a etiquetar los paquetes una vez que se activa de nuevo un canal bloqueado. Partimos del momento en que ambos canales han emitido cuatro paquetes y VC0 ha utilizado todos sus créditos. Para el cálculo de las etiquetas de los paquetes se ha utilizado le ecuación 2.10, y como simplificación se considera $v(t) < TS_i^{k-1}$. La situación inicial es la mostrada en la Figura 4.12.

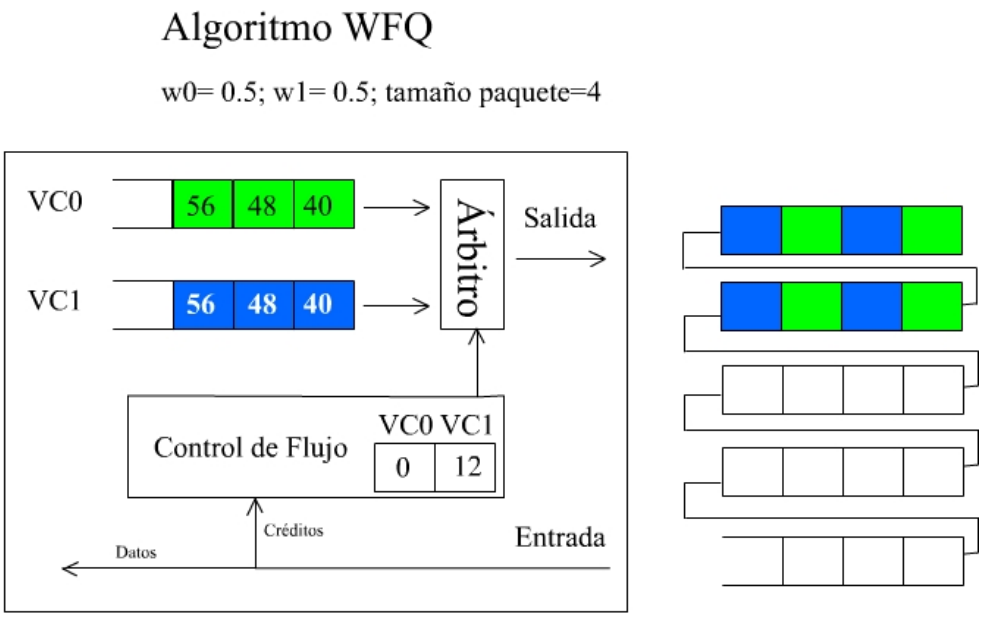



**Figura 4.12 VC0 bloqueado por el control de flujo.**

A partir de este momento, VC1 continúa con la emisión de paquetes, y la etiqueta del primer paquete del VC0 va envejeciendo. Cuando VC0 vuelve a recibir créditos para continuar su transmisión y atendiendo a sus etiquetas, los paquetes de VC0 deben se emitidos en primer lugar, y por tanto hace uso de recursos que no había podido utilizar antes. Podemos ver en la Figura 4.13 el momento en que VC0 recibe los créditos, y para entonces VC1 ha emitido dos paquetes. Al no tener en cuenta el envejecimiento de las etiquetas, VC0 toma el control y utiliza los recursos que no había utilizado antes. Entonces enviará tres paquetes de una sola vez, momento en que la etiqueta de su primer paquete es mayor que las etiquetas de los paquetes pertenecientes a VC1, véase la Figura 4.14.

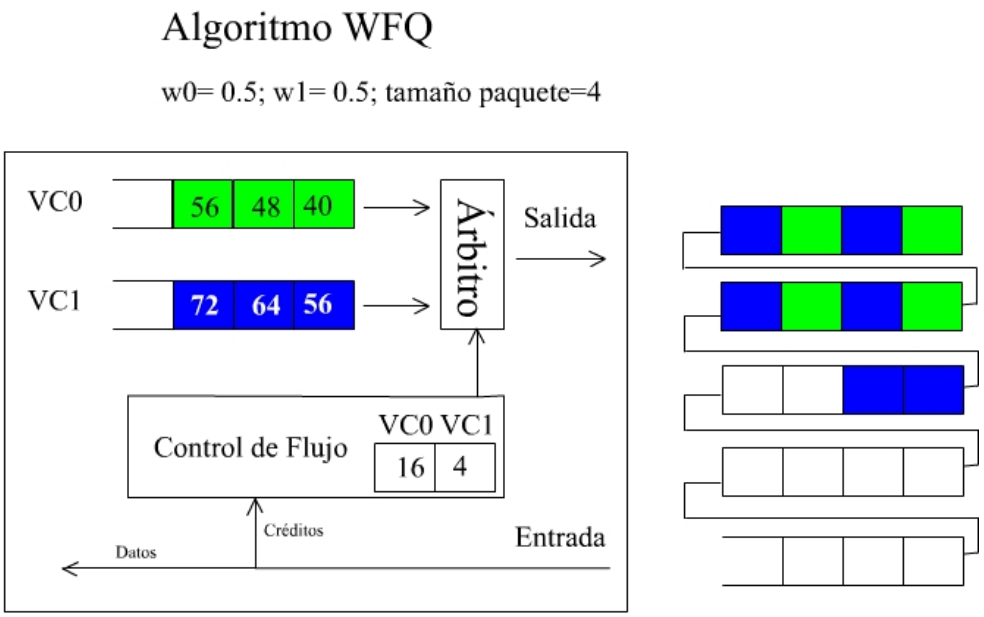



**Figura 4.13 Momento en que VC0 se desbloquea.**

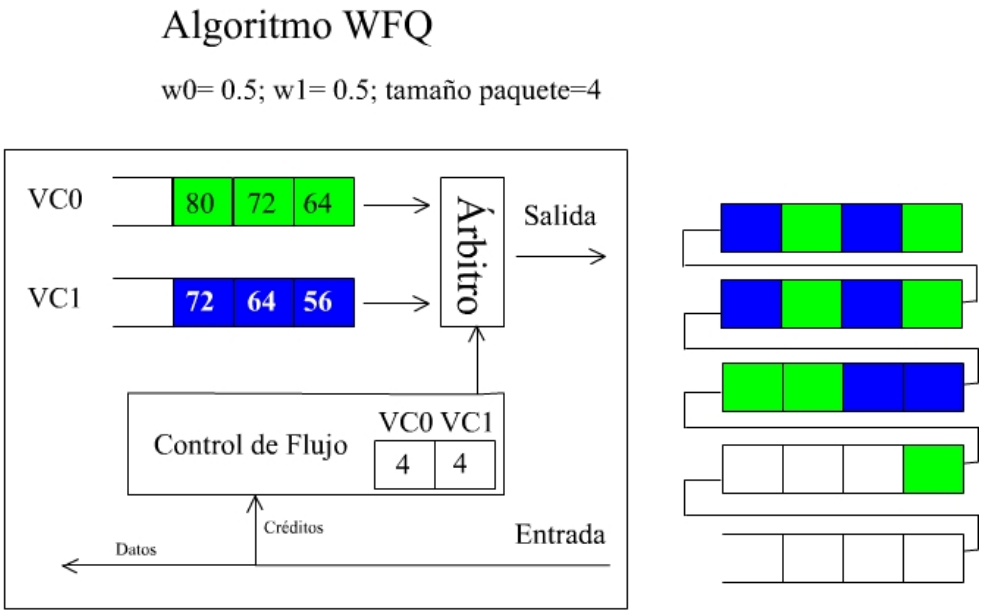



**Figura 4.14 VC0 acapara recursos.**

A partir de aquí, el ejemplo se desarrollará como al principio, emitiendo en función de la etiqueta, es decir, para este ejemplo, se intercalarán los paquetes de ambos VCs, siempre que no vuelva a suceder otra situación como la descrita anteriormente.

### 4.3.1 Weighted Fair Queuing Credit Aware (WFQ-CA).

El algoritmo WFQ-CA que implementamos, trabaja de manera similar al algoritmo WFQ original, teniendo en cuenta los siguientes aspectos:

- Cuando llega un paquete al CV, se etiqueta con el "virtual finishing time" si, y solo si, tiene suficientes créditos para transmitir el paquete que hay en la primera posición de la cola del CV.
- Los paquetes se van transmitiendo según el orden incremental de la etiqueta, pero sólo se tienen en consideración los canales con suficientes créditos para transmitir su primer paquete.
- Sí el CV estuviese inactivo debido a la falta de créditos, y recibe créditos suficientes para volver a transmitir, sus paquetes deben volver a etiquetarse, desde la cabeza hasta la cola, como si hubiesen llegado en ese instante.

De no actuar de esta forma, los paquetes tendrían etiquetas envejecidas respecto a las etiquetas de los paquetes de los demás CVs, lo cual supone la utilización de los recursos que no ha consumido durante el periodo en que ha estado desactivado.

Partiendo del ejemplo anterior se muestra el funcionamiento y las diferencias entre los dos modos de funcionamiento, lo que permite ver la necesidad de adaptar los algoritmos de planificación para que se adapten al mecanismo de control de flujo.

Partiendo de la situación inicial descrita en el subapartado 4.3, se desarrollaría de la siguiente forma. Una vez que VC0 recibe los créditos de control de flujo, se vuelven a etiquetar los paquetes de dicho VC, de forma que las etiquetas de todos los paquetes que hay en el sistema son comparables. Así ningún VC es capaz de recuperar los recursos perdidos al tener que desactivarse a causa del control de flujo. A partir de la situación mostrada en la Figura 4.15, continuarán intercalándose los paquetes de ambos VC como al principio.

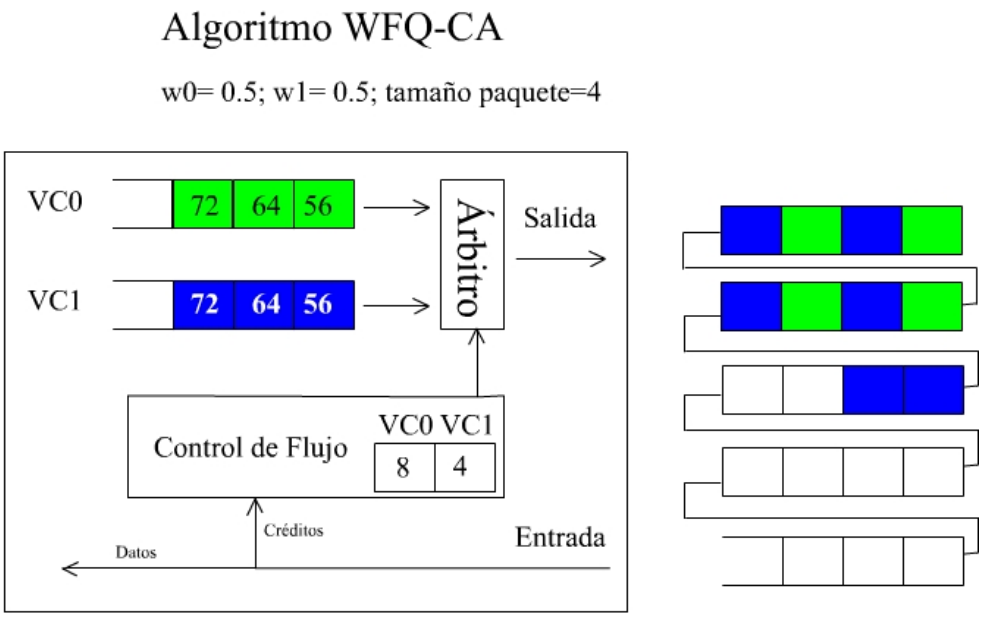



**Figura 4.15 Reetiquetado de los paquetes.**

Comparando la Figura 4.14 y la Figura 4.16, vemos que al obtener VC0 créditos de control de flujo no utiliza los recursos perdidos, y por tanto, no produce una ráfaga de paquetes. Visto de otra forma, cuando un CV queda bloqueado debido al control de flujo, los recursos que no puede utilizar son repartidos entre los demás VCs, y una vez que se vuelve a activar el canal bloqueado, éste no puede hacer uso de los recursos que haya perdido durante el periodo en que ha permanecido bloqueado.

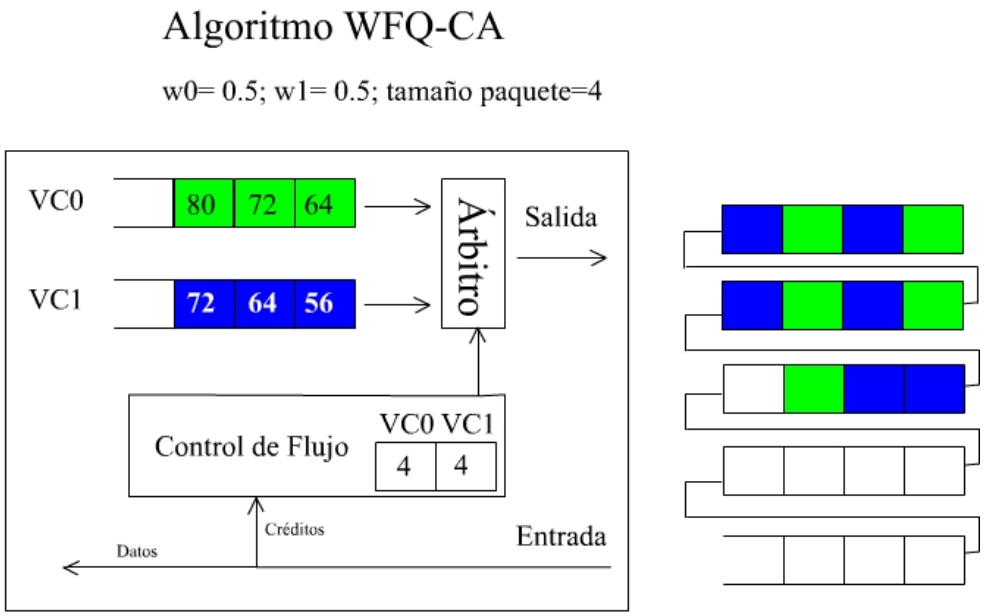



**Figura 4.16 VC0 no recupera los recursos perdidos.**

Otro aspecto a tener en cuenta es el uso del reloj real de tiempo para el cálculo del tiempo virtual. Este tiempo real incluye el tiempo usado para la transmisión de los paquetes de control y de créditos, tiempos necesarios para el buen funcionamiento del control de flujo, pero que no son computables para el algoritmo de planificación. La solución la encontramos no teniendo en cuenta dichos tiempos para el calculo del tiempo virtual. Para ello obtenemos el tiempo virtual a partir de un tiempo t', que representa al tiempo real una vez que le descontamos el tiempo empleado en las tareas del control de flujo. Esto no es trivial, ya que varios eventos pueden ocurrir en esos instantes. Un evento es aquello que cambia de estado el planificador, bien sea por llegadas o salidas de paquetes o créditos. La Figura 4.17 muestra un ejemplo del cálculo del tiempo virtual. Durante el tiempo real (t) aparecen 2 momentos en los que se transmiten paquetes de control, desde e2 a e4 y entre e6 y e7. Estos tiempos no son tenidos en cuenta en t', a partir del cual se calcula el tiempo virtual. Si durante algún intervalo de tiempo hubiese más de un evento, se consideraría que todos ellos llegaron al principio de ese intervalo.

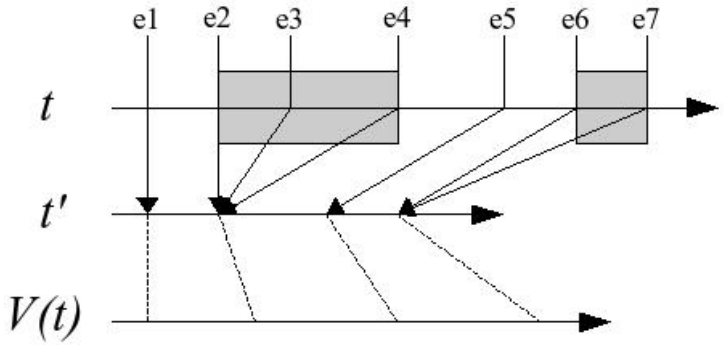



**Figura 4.17 Líneas de tiempos.**

### 4.3.2 Self-Clocked Fair Queuing Credit Aware (SCFQ-CA).

El algoritmo SCFQ-CA propuesto, actúa de forma similar al algoritmo SCFQ, modificando los siguientes aspectos:

- Cuando llega un paquete a la cola del CV, se etiqueta si y sólo si, es el primero de la cola del canal y hay suficientes créditos disponibles para transmitirlo.
- Cuando se emite un paquete, el siguiente paquete es etiquetado si hay suficientes créditos para enviarlo.
- Si un canal inactivo recibe suficientes créditos para volver a transmitir, entonces se etiqueta el paquete de la cabecera.

Para el cálculo de las etiquetas se utiliza el tiempo virtual de acuerdo a la siguiente fórmula

$$F_i^k = \max(F_i^{k-1}, F_{current}) + \frac{L_i^k}{\phi_i} \quad \textbf{( 4.1 )}$$

véase que $F_i^{k-1} \geq \mathrm{F}_{\mathrm{current}}$ si hay al menos un paquete esperando o transmitiéndose en la cola i. Esto nos permite posponer el etiquetado hasta que llegue a la cabecera del canal, haciendo innecesario volver a etiquetar todos los paquetes como en WFQ-CA, lo cual simplifica mucho el algoritmo.

Se consigue una gran ventaja al etiquetar sólo el primer paquete del CV cuando disponga de suficientes créditos, pues el bloqueo de un CV debido a la falta de créditos no implica que el paquete ya disponga de una etiqueta que vaya envejeciendo con el paso del tiempo, y por tanto, pueda obtener ventaja de dicha situación. Además evita tener que volver a etiquetar los paquetes en caso de activarse el CV que estaba desactivado y con paquetes.

### 4.3.3 Virtual Clock Credit Aware (VCK-CA).

La propuesta para el algoritmo VCK-CA es similar al algoritmo Virtual Clock original. Se define el tiempo virtual con el que etiqueta el paquete como:

$$F_i^k \leftarrow \max(F_i^{k-1}, t) + \frac{L_i^k}{\phi_i} \quad \textbf{( 4.2 )}$$

Esto nos obliga a mantener la complejidad del cálculo de V(t) para las etiquetas del paquete y no permite etiquetar el paquete “a posteriori”, de forma análoga al caso de SCFQ-CA. Como resultado, nos vemos en la obligación de etiquetar los paquetes a su llegada al sistema, y emitirlos en el orden incremental de las etiquetas siempre que el CV esté activo. Si un CV inactivo recibe créditos suficientes para transmitir, debemos etiquetar de nuevo los paquetes del canal, desde la cabeza hasta la cola. A todo esto hay que sumar el problema de utilizar el tiempo real, sin tener en consideración el tiempo usado para transmitir los paquetes de control y créditos. La solución aplicada es la misma que la utilizada en el algoritmo WFQ-CA, es decir, volver a etiquetar todos los paquetes del CV y el cálculo de un tiempo t’ para obtener V(t) libre de tiempos utilizados por la lógica de control.

## 4.4 UN NUEVO ALGORITMO: FAST COUNTER DRR (FCDRR).

La propuesta del algoritmo FCDRR ha surgido a lo largo de este estudio, es una mezcla entre los algoritmos CBFQ-CA y ALIQUEM DRR-CA. Necesitaremos, además del ancho de banda compartido $S_i$, un contador por CV que indicará en cuántas rondas, a partir de la actual, dispondrá de quantum suficiente para enviar el primer paquete de su cola, y una lista ordenada para guardar el orden en que deben emitir los canales virtuales. Así, en vez de esperar a que los canales adquieran el quantum suficientes para emitir el siguiente paquete, tenemos una lista en la que nos indica qué canal será el próximo que dispondrá de suficiente quantum para transmitir su paquete. Una vez que se transmite el paquete hay que actualizar los contadores de los demás canales y actualizar la lista con el nuevo orden de los canales virtuales. Para tener en cuenta el control de flujo nos apoyamos en la lista ordenada, en la que sólo habrá identificadores de CVs activos. Al desactivarse un canal se eliminará su identificador de la lista y perderá su quantum.

Las ideas para mejorar a sus predecesores son las siguientes, como tratamos con un número reducido de canales virtuales, ALIQUEM DRR-CA nos ha parecido poco eficiente al repartir los identificadores de canal por una compleja estructura de colas. La solución que proponemos es utilizar contadores para calcular, a partir de la ronda actual, en cuántas rondas tendremos quantum suficiente para enviar el mensaje de la cabecera y una lista para ordenar los canales según su contador de salida. Se emite el paquete del CV que vaya a enviar primero, en vez de avanzar las rondas de una en una. Aunque este funcionamiento ya está presente en CBFQ-CA, existen ciertas salvedades. Se utilizan números enteros, mantenemos el contador de quantum restante hasta la desactivación del CV y no se realiza la ordenación de todos los CVs, sino que insertamos el CV correspondiente en la lista de canales ordenada.

El arbitraje se puede resumir así:

- Si un CV está inactivo, el paquete se encola en su buffer, pero no se calculará cuando se emitirá, hasta la activación del canal.
- Si un CV está activo y es el primer paquete, se almacena en el buffer, y se calcula cuando tendrá créditos para emitir y se inserta el identificador del canal en la lista ordenada.
- Si un CV está activo y el paquete entrante no es el primer paquete, se guarda en el buffer. Cuando llegue a la cabecera de la cola se calcula cuando tendrá créditos para emitir y se inserta el identificador del canal en la lista.
- Al enviar un paquete, se le restan los créditos necesarios de su contador de créditos. Si quedan paquetes, se calcula cuándo se enviará el próximo paquete, si no quedan créditos, se desactivará el canal.

- Cuando el CV se desactiva, se le quita el quantum acumulado. Cuando se vuelve a activar se le asigna la posición que le corresponde en la lista ordenada.

El algoritmo obtenido sólo necesita operaciones con enteros en vez de con unidades de punto flotante, sin desperdicio de ancho de banda, ni complejas estructuras de datos, todo esto manteniendo una complejidad de O(log(n)), con n el número de canales activo en el momento dado. La operación más compleja es hacer una inserción en una lista ordenada, pero es una opción viable teniendo en cuenta el reducido número de canales sobre los que actuamos. Tampoco hay problemas con los contadores de rondas, ya que no mantienen un número absoluto de vuelta sino que guardan un valor relativo, indicando en cuántas rondas desde la actual, dispondremos de créditos suficientes para la emisión del primer paquete de la cola. Esto nos evita el problema de los desbordamientos de los contadores.

# 5 PRESENTACIÓN Y ANÁLISIS DE RESULTADOS.

En este capítulo se incluyen los resultados obtenidos a partir del estudio realizado. Debido a la gran cantidad de datos y gráficas que pueden generarse se ha optado por incluir aquí sólo una parte de los mismos, llevando la totalidad de ellos a los diversos anexos que se han incluido en la parte final de esta memoria. La intención no es otra que conseguir una exposición clara y sencilla que redunde en una lectura fácil y amena, sin perder por ello los contenidos que resulten de interés.

Antes de mostrar estos resultados, se incluyen a continuación una sección en las que se describen las características de la herramienta de simulación utilizada. Hay que recordar que todas las pruebas, que han sido muy numerosas, se han llevado a cabo mediante simulación y de ahí que nos parezca adecuado al menos indicar dichas características.

## 5.1 CARACTERÍSTICAS DEL SIMULADOR.

El simulador que se ha utilizado en este proyecto modela una red de conmutadores sobre la cual transmiten varios flujos sus paquetes, y puede ser configurada mediante el ajuste de una serie de parámetros. A continuación se describen estos parámetros y se indican los resultados que ofrece.

- Conmutador:
  - Flytime: tiempo desde que se decide a qué enlace se envía el paquete hasta que se envía el mismo.
  - Throughput: *speedup* del conmutador.
  - Número de puertos: número de salidas del conmutador.
  - Tiempo de enrutamiento: tiempo desde que el algoritmo de planificación envía el paquete al buffer de salida hasta que abandona el conmutador.
  - Tiempo de arbitraje: tiempo para decidir a qué enlace se envía el paquete.
  - Tiempo de procesamiento del encabezado del paquete.
  - Tamaño del buffer de entrada.
  - Tamaño del buffer de salida.
  - Threshold: nivel del buffer de entrada a partir del que se envían créditos al conmutador anterior.
  - Árbitro utilizado y ruta de su fichero de configuración.
  - Número de canales virtuales.

- Interfaz de red:
  - Tiempo de enrutamiento.
  - Tiempo de arbitraje.
  - Tiempo de procesamiento del encabezado del paquete.
  - Tamaño del buffer de entrada.
  - Threshold: nivel del buffer de entrada a partir del que se envían créditos al conmutador anterior.
  - Árbitro utilizado y ruta de su fichero de configuración.
  - Número de niveles de servicio.

- Flujo: Según el tipo de tráfico necesitaremos configurar los flujos como Burst N o CBR.
  - Burst N: Este tipo de tráfico se caracteriza por el parámetro N, donde N es la longitud de las ráfagas de paquetes que emite el canal, todos hacia el mismo destino. El tamaño de los paquetes se basa en una distribución de Pareto que generan paquetes pequeños y ocasionalmente algún paquete mayor. El periodo entre ráfagas se modela con una distribución de Poisson. Otros parámetros que caracterizan es tipo de tráfico son:
    - Tasa media de inyección por flujo en Gbps.
    - Tamaño mínimo de paquete
    - Tamaño máximo de paquete
    - Tipo de tráfico: en este caso “selfSimilar”.
    - Parámetro de forma del tráfico: en este caso “1.25”.
    - Tamaño de la ráfaga (parámetro N de Burst).

  - CBR: En este tipo de tráfico se inyectan paquetes con un ratio constante y tamaño también constante aunque sigue las distribuciones indicadas anteriormente, las diferencias son:
    - Tipo de tráfico: En este caso no se indica ni el parámetro de forma del tráfico, ni el tamaño de la ráfaga, ya que es un tráfico constante en forma y sin ráfagas.

- Parámetros de salida: Entre los muchos parámetros que nos ofrece, tanto a nivel global como a nivel de CV, nos centraremos en los siguientes:
  - Productividad.
  - Latencia.
  - Jitter.
  - Número de paquetes generados, inyectados y recibidos.
  - Tasas de generación, inyección y llegada.

## 5.2 RESULTADOS.

Para la obtención de los resultados, se han realizado las simulaciones modificando el algoritmo de planificación utilizado, el tamaño del buffer y el tipo de tráfico. Los planificadores utilizados son los mostrados en el Capitulo 4. El tamaño del buffer varía entre 2, 8 y 32 paquetes. En cuanto al tipo de trafico que se utiliza, puede ser tráfico CBR, que no tiene ráfagas, o tráfico tipo Burst, con ráfagas de tamaños 1, 15 y 60 paquetes, este último se utiliza para modelar el peor de los casos del tráfico real de una red. En total son 42 posibilidades distintas, las cuales se han simulado para 10 puntos de carga diferentes y se han repetido 3 veces, de forma que las curvas obtenidas son suavizadas, así es posible obtener mejores valores medios.

| Algoritmo de planificación | |
|---|---|
| Basados en frames | DRR-CA<br>ALIQUEM DRR-CA<br>CBFQ-CA |
| Basados en prioridades | WFQ-CA<br>SCFQ-CA<br>CVK-CA<br>FCDRR |

**Tabla 5.1 Algoritmos de planificación**

| Tipo de tráfico | Tamaño de ráfaga |
|---|---|
| Burst | 1 |
| | 15 |
| | 60 |
| CBR | No tiene |

**Tabla 5.2 Tipo de tráfico**

| Tamaño de buffer |
|---|
| 2 |
| 8 |
| 32 |

**Tabla 5.3 Tamaño de buffer**

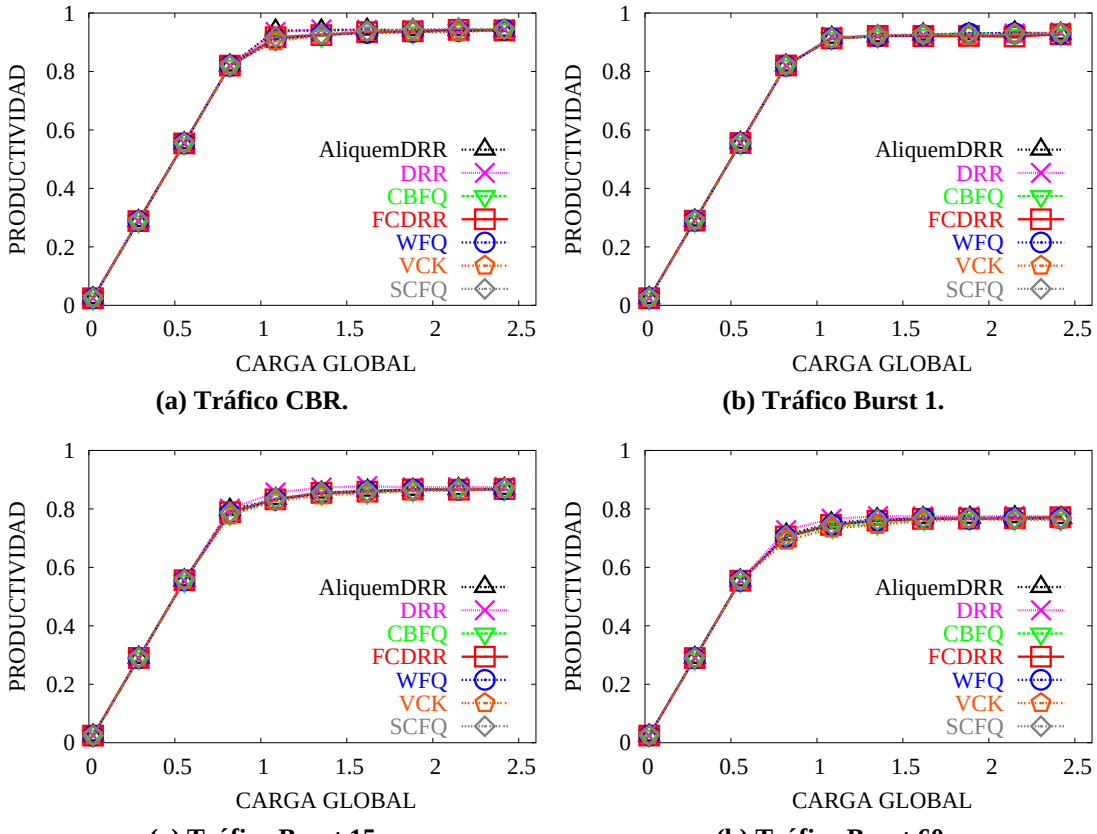
Para mostrar el grueso de los resultados, esta sección se ha organizado en varios apartados. En el primero se muestran los parámetros globales de productividad y latencia. En el segundo apartado se muestra una comparativa entre los algoritmos basados en “frames” y los algoritmos basados en prioridades. En el último apartado se muestra la importancia del parámetro “q” en el algoritmo Aliquem DRR-CA, que es el número de listas de la estructura utilizada por dicho algoritmo para saber qué CV se debe servir. Para resumir el número de figuras mostradas en esta sección, sólo se muestran aquellas que son más relevantes para nuestro estudio, aunque el conjunto total de gráficas se pueden consultar en el apéndice.

### 5.2.1 Parámetros de diseño.

Antes de considerar el funcionamiento de los algoritmos en su aspecto diferenciador del tráfico según los parámetros de los CVs, se mostrará cómo se comportan de manera global respecto a los parámetros de diseño (tamaño del buffer y tipo de tráfico), teniendo en cuenta sólo el tráfico total del conmutador. Esto nos dará una idea básica del comportamiento general del conmutador una vez que tengamos en cuenta dichos CVs.

#### 5.2.1.1 Efecto de los parámetros de diseño en la productividad.

El primer parámetro de diseño estudiado es el tipo de tráfico, comparando la Figura 5.1 a), b), c) y d), se puede observar que el tráfico con ráfagas largas obtiene una menor productividad que cuando las ráfagas son más cortas o es tráfico CBR, independientemente del algoritmo utilizado.



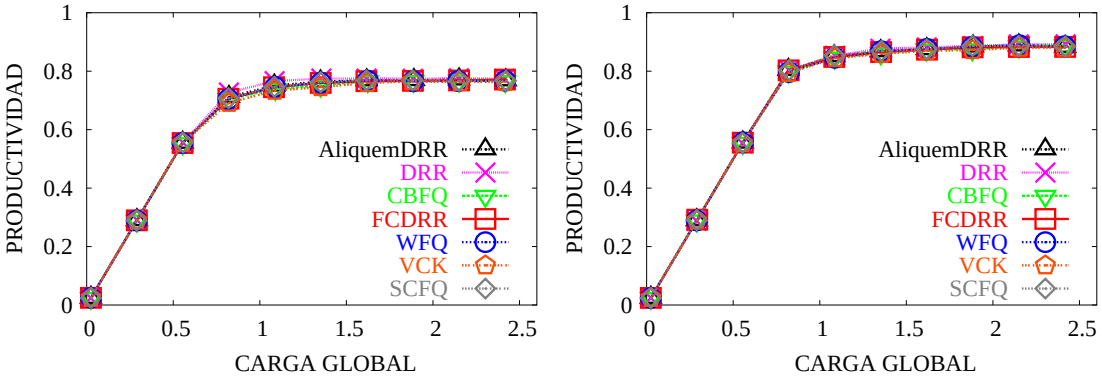

**(a) Tráfico CBR.** **(b) Tráfico Burst 1.**

**(a) Tráfico Burst 15.** **(b) Tráfico Burst 60.**

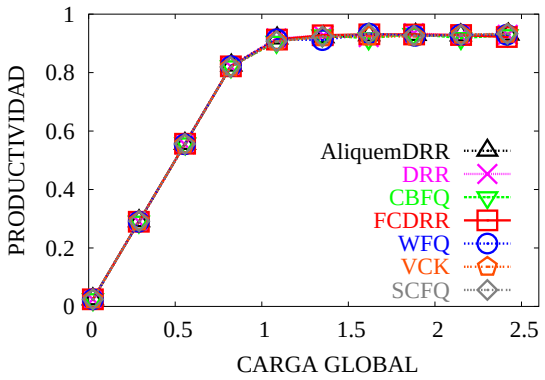
**Figura 5.1 Productividad con tamaño de buffer 2**

Este efecto tiene su explicación en que el tráfico con ráfagas largas es más probable que bloquee el CV al agotar los créditos del control de flujo, y por tanto pierda su quantum o tenga que volver a etiquetar los paquetes. Esta es una de las causas más frecuentes de que se produzcan los árboles de congestión. Estos se producen cuando en un conmutador se bloquea algún CV por alguna causa concreta. Si dicho canal bloqueado sigue recibiendo paquetes, llega un momento en que el conmutador no los puede guardar, y por tanto se almacenan en el conmutador anterior, lo que sucede recursivamente a lo largo de la red. Pero además, ese CV bloqueado puede bloquear el paso de otros CVs, y por tanto, la situación de bloqueo se expande por toda la red de conmutadores afectando cada vez a más canales y a más conmutadores. Este efecto se conoce como árboles de congestión.

Ahora estudiaremos cómo afecta el tamaño de buffer de entrada en la productividad, cuyos resultados gráficos se pueden observar en la Figura 5.2. Al reducir el tamaño del buffer de entrada, la productividad también se ve reducida, no siendo independiente del tipo de tráfico que se encamine, aunque sólo se ve ligeramente afectado por el mismo.




**(a) Tamaño buffer 2.** **(b) Tamaño buffer 8.**



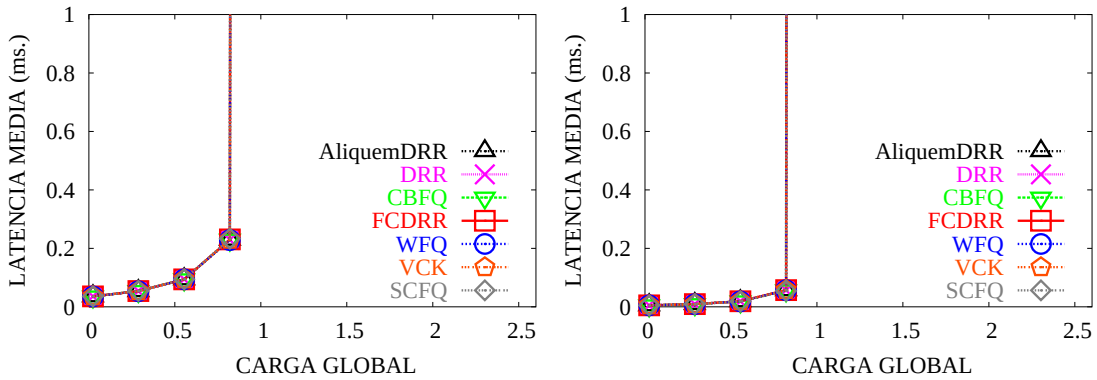

**(c) Tamaño buffer 32.**

**Figura 5.2 Productividad con tráfico Burst 60.**

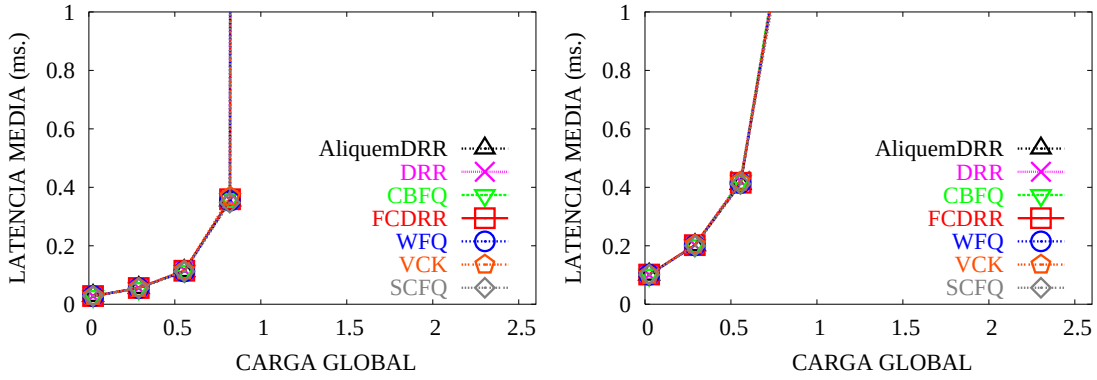
Para explicar esto sólo hay que pensar que un menor tamaño de buffer implica la recepción de un menor número de créditos, y por tanto, es más posible que un CV se bloquee, produciéndose árboles de congestión y las consecuencias que eso provoca.

#### 5.2.1.2 Efecto de los parámetros de diseño en la latencia.

Comparando las gráficas de latencia obtenidas para cada tipo de tráfico se aprecia que con tipo de tráfico "Burst n", la latencia aumenta antes cuanto mayor sea ese "n" (Ver la Figura 5.3). Este efecto se produce, lógicamente, a partir del tamaño de la ráfaga, pues cuanto mayor es la misma, mayor tiempo permanecen los paquetes en el conmutador, y por tanto la latencia aumenta.



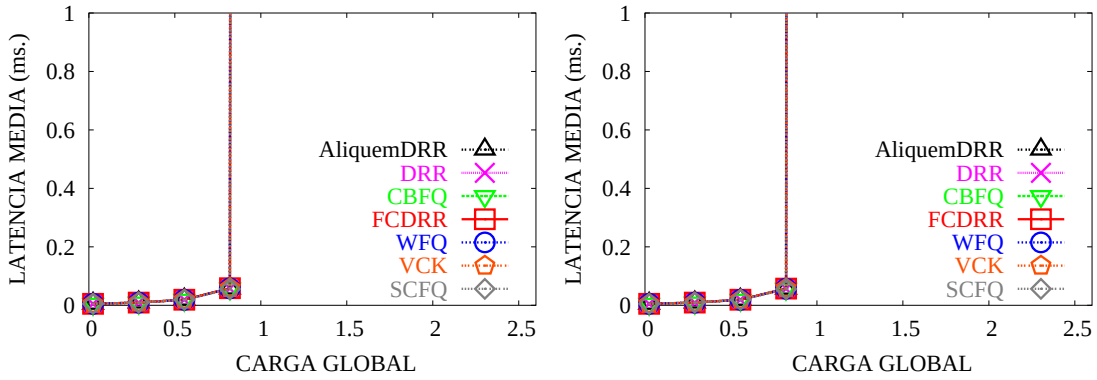

**(a) Tráfico CBR.** **(b) Tráfico Burst 1.**



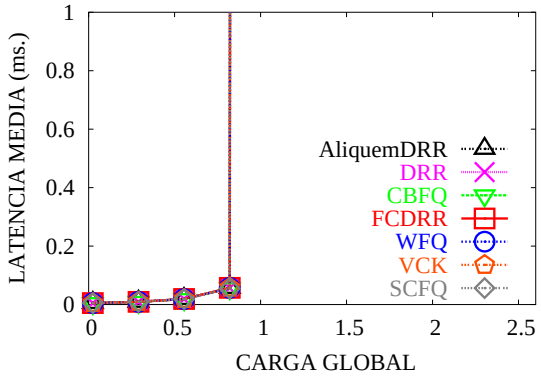

**(a) Tráfico Burst 15.** **(b) Tráfico Burst 60.**

**Figura 5.3 Latencia con tamaño de buffer 32.**

Estudiando el parámetro de tamaño de buffer y tras observar todas las gráficas obtenidas para todos los tipos de tráfico, no se encuentran diferencias significativas (ver la Figura 5.4), y sólo para Burst 60 se puede obtener una pendiente de la curva mayor al reducir el tamaño de buffer, lo cual implica mayor latencia (ver la Figura 5.5). Hay que mencionar que el tráfico CBR se comporta como el Burst 1.



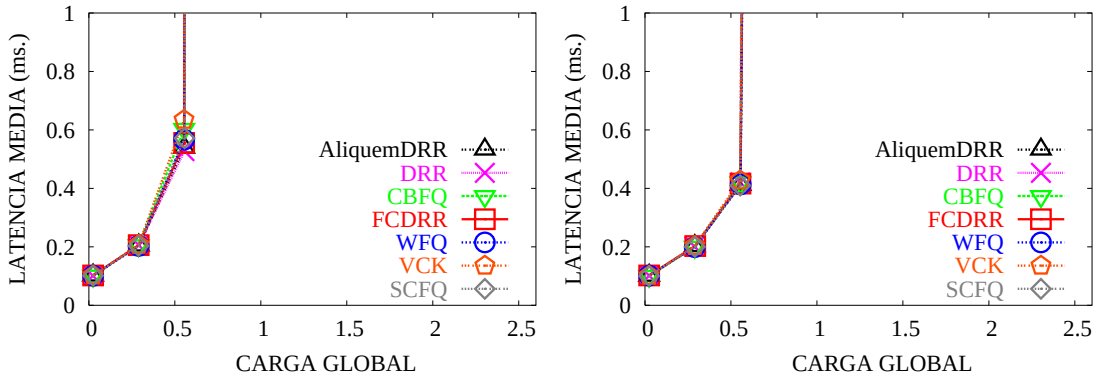

**(a) Tamaño buffer 2.** **(b) Tamaño buffer 8.**



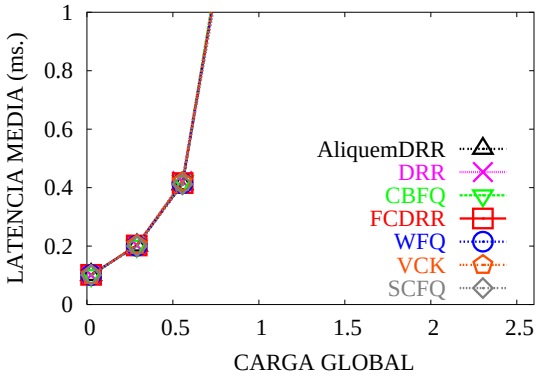

**(c) Tamaño buffer 32.**

**Figura 5.4 Latencia con tráfico Burst 1.**

En la Figura 5.4 se observa que el parámetro del tamaño de buffer tiene efectos muy leves sobre la latencia, y sólo en casos extremos de tráfico, con ráfagas largas y tamaño de buffer mayor, se obtiene una mejor latencia. Eso es debido a la menor probabilidad de bloqueo de una canal cuando el buffer es mayor, de ahí el efecto mostrado en las siguientes gráficas.




**(a) Tamaño buffer 2.** **(b) Tamaño buffer 8.**



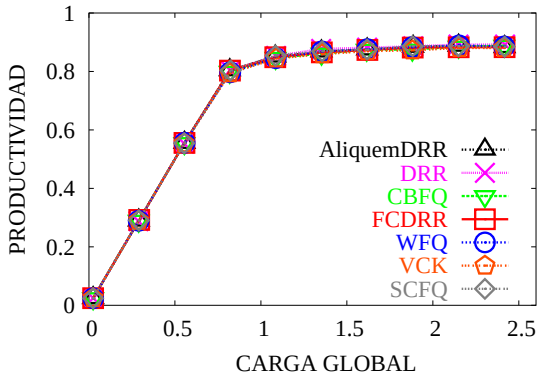

**(c) Tamaño buffer 32.**

**Figura 5.5 Latencia con tráfico Burst 60.**

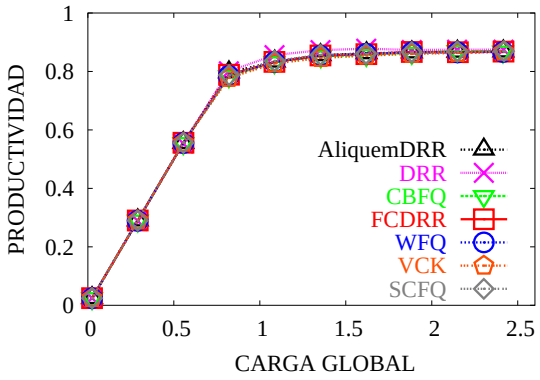
### 5.2.1.3 Otras consideraciones.

Para concluir, hay que indicar que el efecto producido por el tipo de tráfico y el tamaño de buffer sobre la productividad, es acumulativo. Es decir, si un tipo de tráfico con ráfagas largas se utiliza con un tamaño de buffer pequeño, la disminución observada en la productividad es aún más acusada.

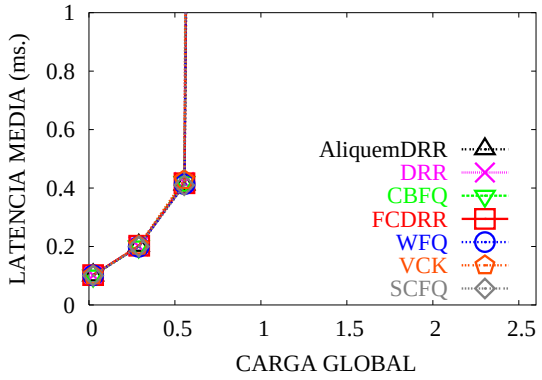
Como puede verse en la Figura 5.6 (b), un tráfico con ráfagas más cortas, pero menor tamaño de buffer obtiene una productividad parecida a la que obtiene un tráfico con ráfagas más largas y mayor tamaño de buffer (Figura 5.6 (a)). Sin embargo, al observar la latencia en la Figura 5.6 c) y d) se muestra que aunque la productividad sea parecida, la latencia obtenida en uno u otro caso es diferente. Como se explica en el apartado anterior, el tráfico con ráfagas mayores obtiene una peor latencia aunque la productividad sea similar. Esto muestra la importancia que tienen los parámetros de diseño en el funcionamiento de la red, además del tipo de tráfico que esté circulando por la red.



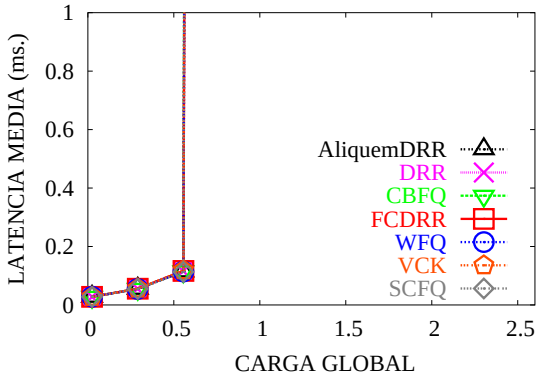

**(a) Productividad con tráfico Burst 60 y tamaño de buffer 8.**




**(b) Productividad con tráfico Burst 15 y tamaño de buffer 2.**




**(c) Latencia con tráfico Burst 60 y tamaño de buffer 8.**




**(d) Latencia con tráfico Burst 15 y tamaño de buffer 2.**

**Figura 5.6 Productividad y latencia al cambiar varios parámetros.**

### 5.2.2 Algoritmos basados en “frames” vs algoritmos basados en prioridades.

Debido a la gran cantidad de información obtenida y para poder estudiar detenidamente los casos más representativos de esta comparativa, es necesario centrarnos en un caso concreto, y luego repasar las situaciones relevantes de los casos restantes. Nótese que se han utilizado 4 CVs, identificados por VC0, VC1, VC2 y VC3, y asignándoles las proporciones mostradas en la Tabla 5.4. Los canales VC0 y VC1 también los designaremos como canales de baja prioridad, mientras que VC2 y VC3 son los canales de alta prioridad.

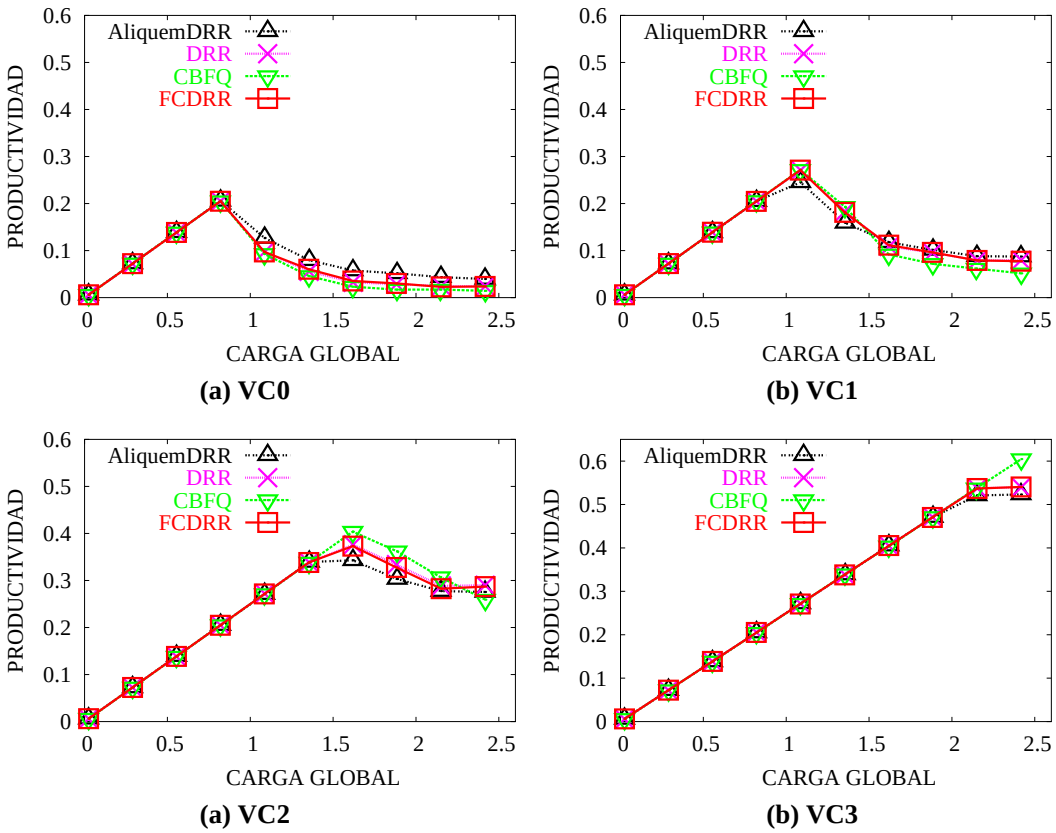
| | Peso | % ancho del enlace |
|---|---|---|
| **VC0** | 1 | 1,8867925 |
| **VC1** | 4 | 7,5471698 |
| **VC2** | 16 | 30,1886792 |
| **VC3** | 32 | 60,3773585 |
| **Total** | 53 | 100 |

**Tabla 5.4 Proporciones de los VCs.**

Para los algoritmos basados en "frames" se utiliza la primera columna, multiplicando el peso por la MTU, de forma que, se puede decir que el peso indica la cantidad de información que podemos enviar en cada iteración del algoritmo. En los algoritmos basados en prioridades utilizamos el porcentaje de ancho del enlace asignado, éste nos indica cuánto ancho de banda puede utilizar cada CV, y en caso de bloqueo de un canal dicho ancho del enlace se reparte proporcionalmente entre los demás CVs. Para empezar, los parámetros del estudio se centran en el tráfico Burst 15 con tamaño de buffer 8. Se muestra el comportamiento de todos los algoritmos de planificación para dicha situación, posteriormente se mencionarán los casos relevantes que surgen con otras combinaciones de parámetros.

#### 5.2.2.1 Algoritmos basados en "frames".

Los resultados obtenidos para la productividad de estos algoritmos se muestran en la Figura 5.7. De entre los algoritmos basados en "frames" mostrados, el algoritmo CBFQ-CA muestra la mayor diferencia de productividad entre los canales de baja prioridad y los canales de alta prioridad. La situación contraria la muestra Aliquem DRR-CA, donde los canales de menor prioridad obtienen una mejor productividad, siempre comparándolo con los demás algoritmos. Por el contrario, los canales de mayor prioridad obtienen menor productividad en Aliquem DRR-CA con respecto a los mismos canales de los otros algoritmos mostrados.




**(a) VC0** **(b) VC1**

**(a) VC2** **(b) VC3**

**Figura 5.7 Productividad de los algoritmos basados en frames.**

Para el algoritmo FCDRR, que se ha propuesto en el Capitulo 4, podemos decir que hace un reparto proporcionado, pues la productividad obtenida en cada uno de los CVs nunca muestra un comportamiento extremo como los algoritmos Aliquem DRR-CA o CBFQ-CA.

El parámetro de latencia no resulta muy significativo para el estudio, pues sólo nos permite ver que los canales de mayor prioridad del algoritmo CBFQ-CA obtienen una mejor latencia (Figura 5.8). Resulta más interesante centrarnos en la latencia máxima, donde se muestran mayores diferencias al mostrar el límite superior de la latencia.

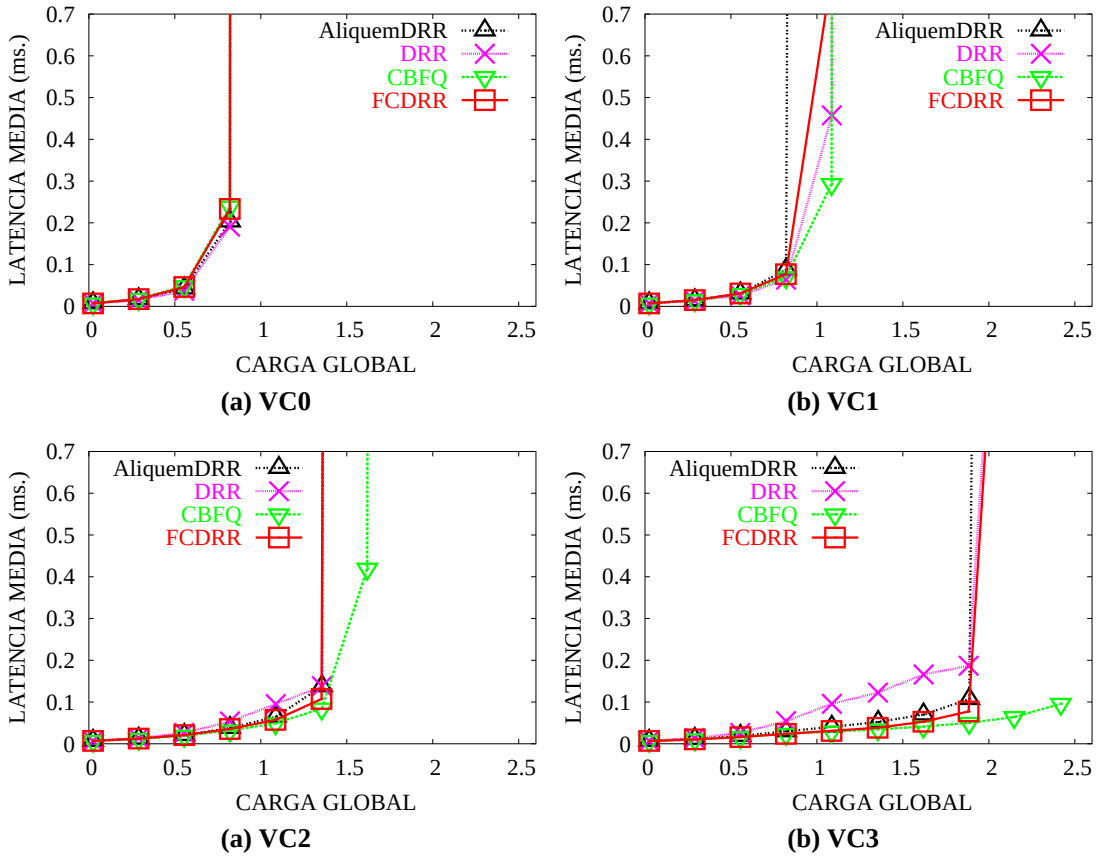
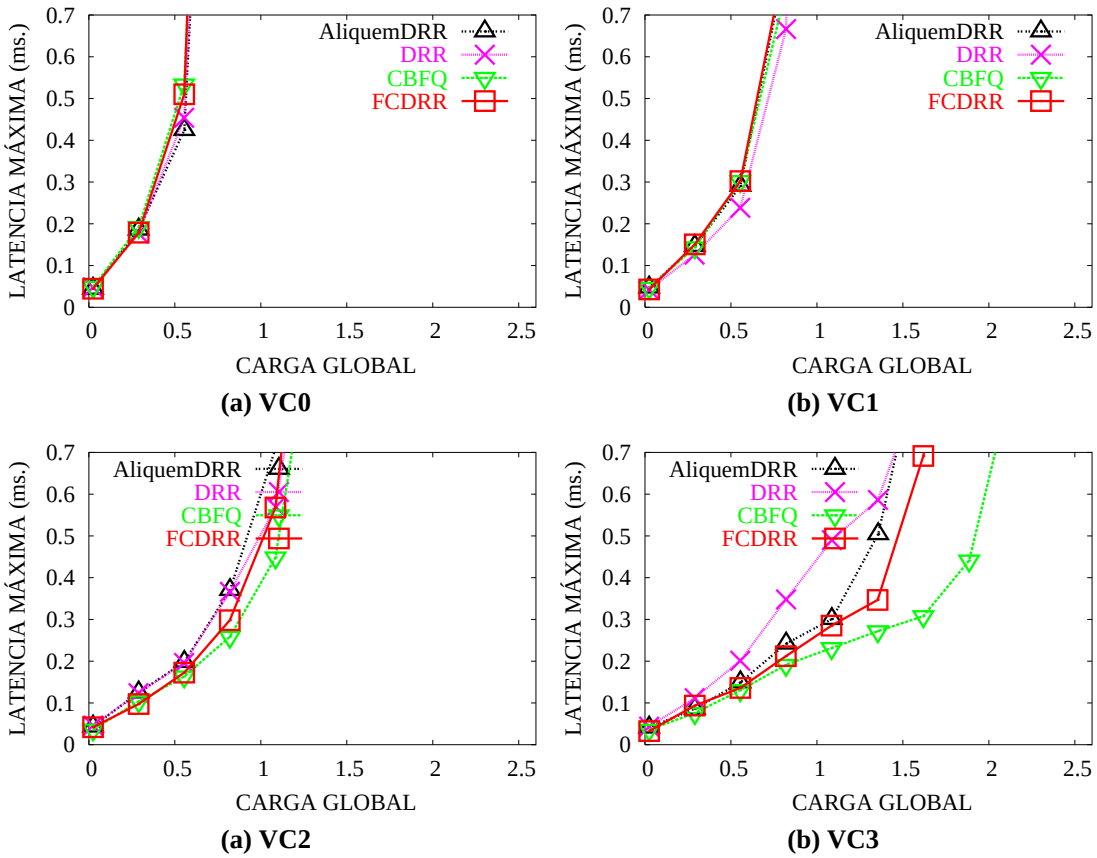



**Figura 5.8 Latencia de los algoritmos basados en frames.**

Se observa en la Figura 5.9, además de comportamiento mencionado sobre CBFQ-CA, que el algoritmo DRR-CA no se encuentra muy cómodo al asignar ancho de banda al canal de mayor prioridad, en consecuencia la latencia máxima mostrada por dicho canal es mayor que para los demás algoritmos de planificación casi desde el primer punto de carga. Las situaciones y los algoritmos no especificados en esta explicación muestran un comportamiento acorde a lo esperado, con ligeras diferencia en los parámetros de latencia media y máxima, pero que no son dignos de mención.



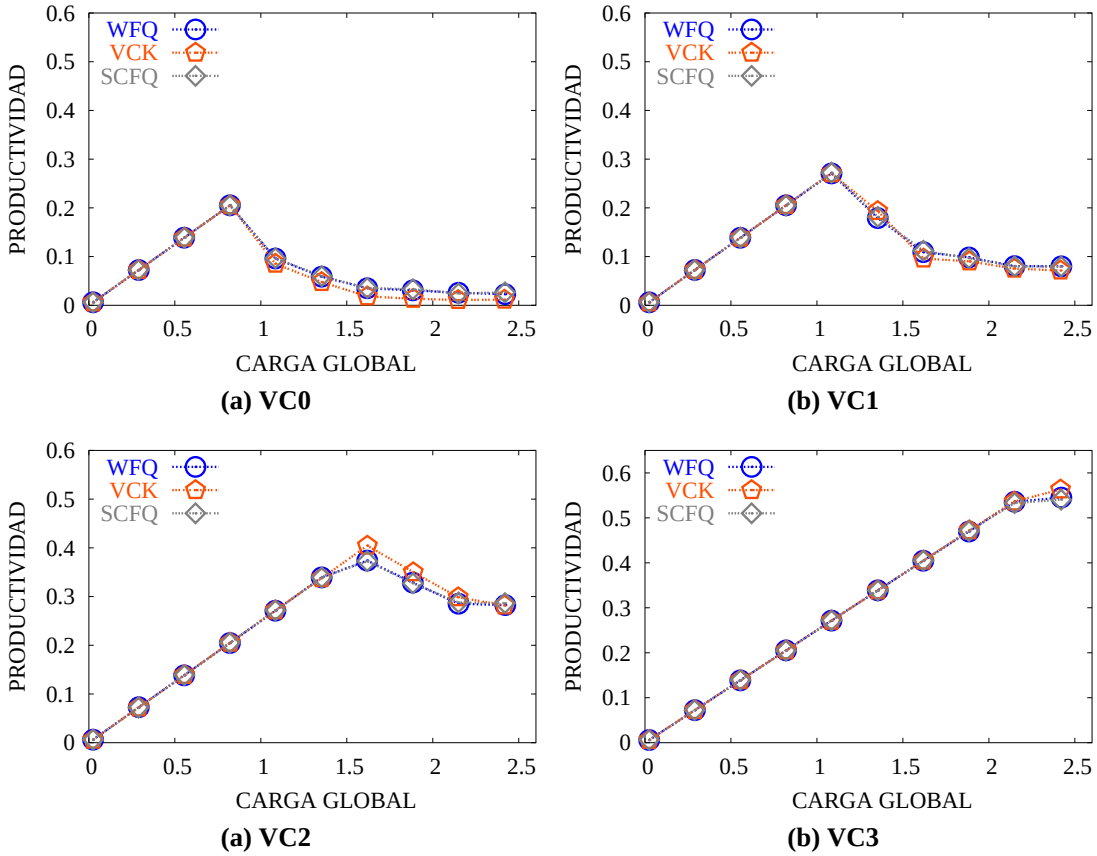

**(a) VC0** **(b) VC1**

**(a) VC2** **(b) VC3**

**Figura 5.9 Latencia máxima de los algoritmos basados en frames.**

### 5.2.2.2 Algoritmos basados en prioridades:

En el estudio de los algoritmos basados en prioridades, y atendiendo al parámetro de la productividad, se observa que las gráficas son muy parecidas. Aún así, se puede notar que el algoritmo VCK-CA tiene un reparto más extremo entre canales. Es decir, los canales de menor prioridad (Figura 5.10 (a) y (b)) obtienen una productividad inferior a la que se obtiene con los otros algoritmos. Esa diferencia es la que asigna a los canales de mayor prioridad (Figura 5.10 (c) y (d)), que para VCK obtienen mejor productividad que para WFQ-CA o SCFQ-CA.



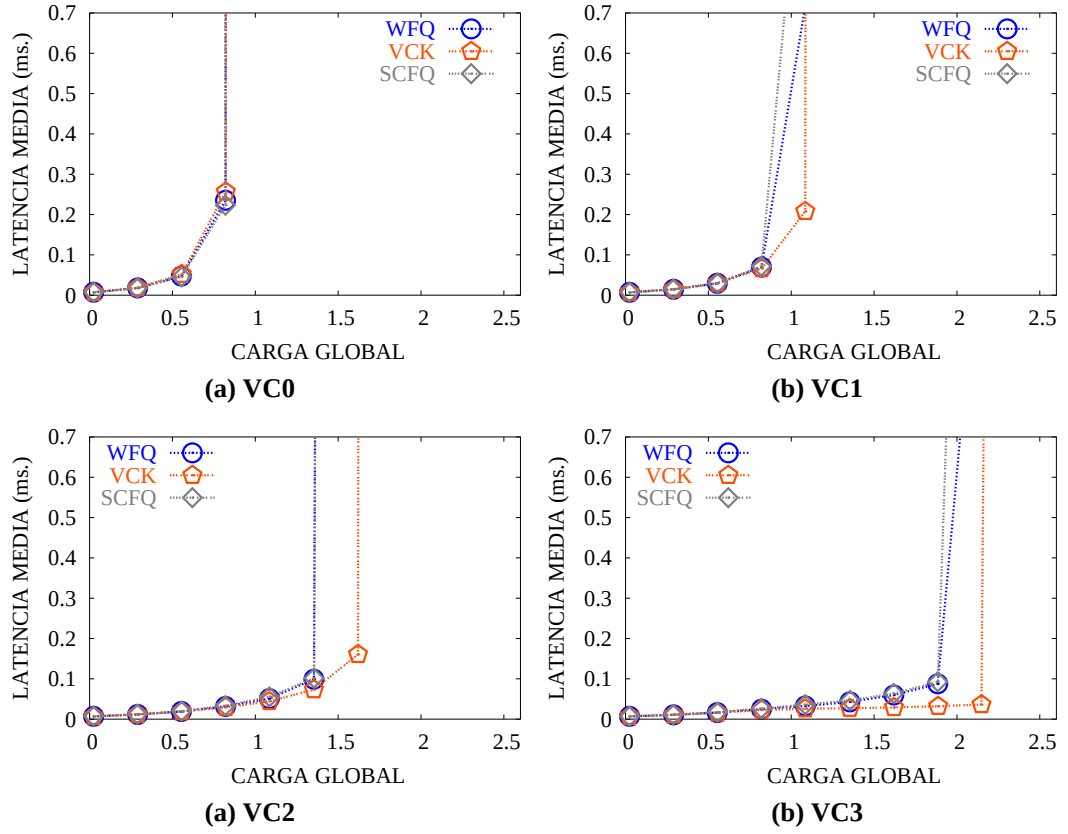

**(a) VC0** **(b) VC1**

**(a) VC2** **(b) VC3**

**Figura 5.10 Productividad de los algoritmos basados en prioridades.**

Aunque a primera vista parece que la latencia es parecida en todos los casos, hay que mencionar que utilizando más puntos de carga es posible obtener gráficas con cambios menos bruscos, pero intentar obtener la curva que mejor lo aproxima puede llevar al caso de utilizar tal cantidad de puntos que sea poco factible desde el punto de vista computacional y práctico.

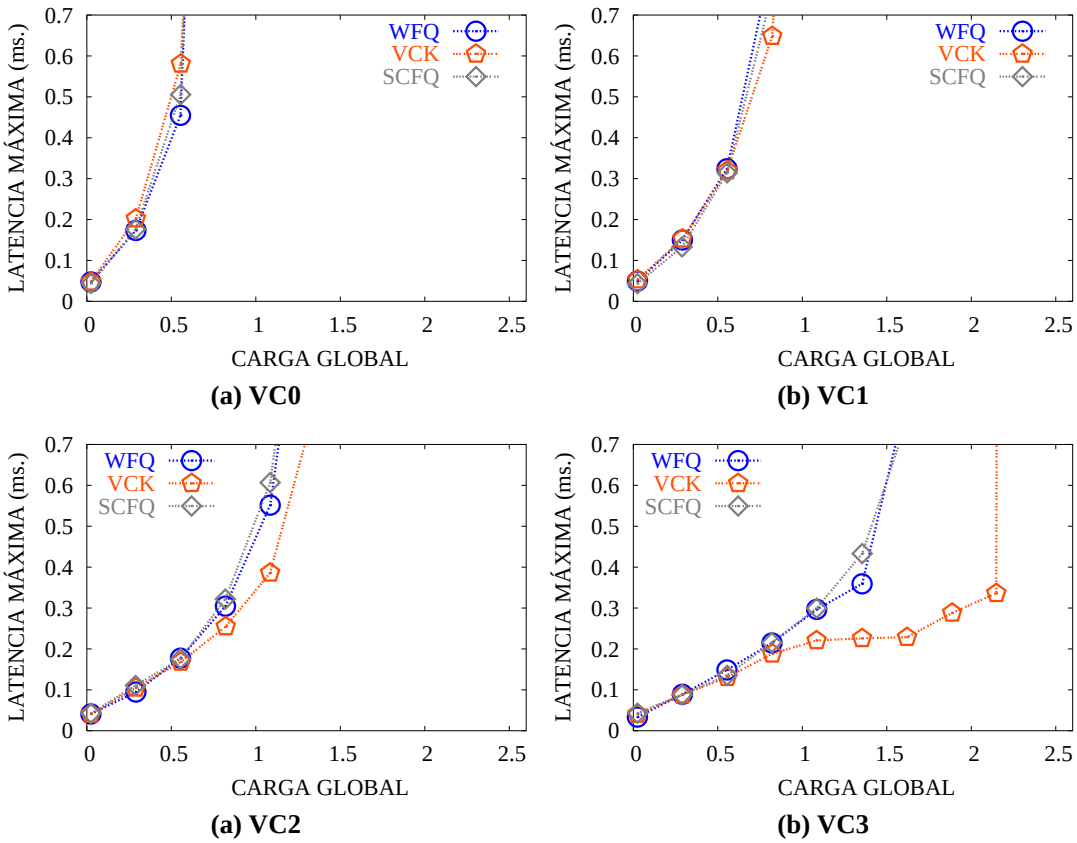
Para el caso de la latencia se aprecia lo mismo que para la productividad, con un cierto matiz. Al aumentar la inyección de paquetes en VCK-CA y conseguir que los canales se saturen, ocurren dos situaciones diferentes, los canales de baja prioridad se comportan como en los demás algoritmos. Nótese que en VC1 la pendiente de la recta que forma es mucho más vertical que en los otros algoritmos, esto nos conduce a pensar que en situaciones de mucho tráfico e incluso momentos de saturación, el algoritmo VCK-CA ofrecerá una peor latencia que la ofrecida por SCFQ-CA o WFQ-CA. Este último parece ser el que mejor comportamiento muestra para redes con mucho tráfico, pero en el fondo no hay mucha diferencia con SCFQ-CA. Sin embargo para los canales de alta prioridad el algoritmo VCK obtiene menor latencia que sus homónimos, los cuales no recuperan la distancia perdida como pasaba en el VC1. Para apoyar estas afirmaciones, más abajo se muestran las latencias máximas de los CVs para cada planificador, señalando el peor de los comportamientos mostrados por los algoritmos de planificación para cada uno de los canales.




**(a) VC0** **(b) VC1**

**(a) VC2** **(b) VC3**

**Figura 5.11 Latencia de los algoritmos basados en prioridades.**




**(a) VC0** **(b) VC1**

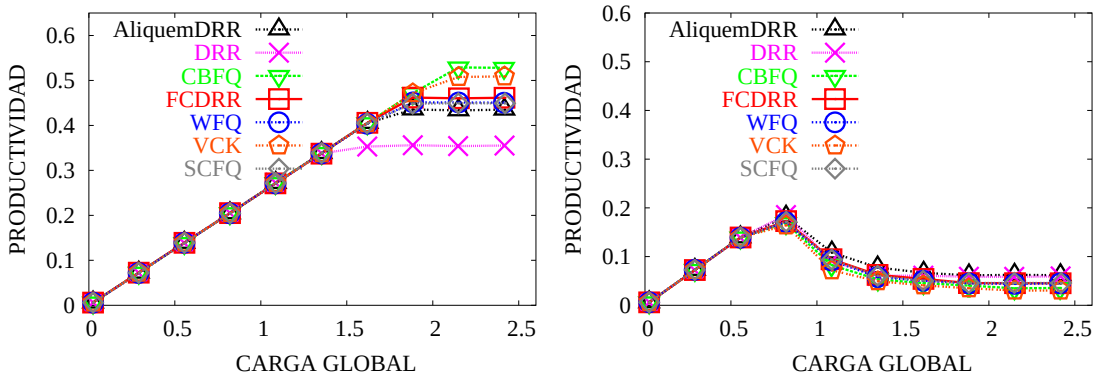
**(a) VC2** **(b) VC3**

**Figura 5.12 Latencia máxima de los algoritmos basados en prioridades.**

#### 5.2.2.3 Otras consideraciones.

Para finalizar, se muestran otras situaciones que merecen ser mencionadas. Comparando los dos tipos de algoritmos, se observa que los algoritmos basados en prioridades tienen un comportamiento muy homogéneo, y obtienen unas prestaciones muy parecidas entre sí. Sin embargo los algoritmos basados en "frames" muestran claras diferencias en su comportamiento. Por ejemplo: para el tráfico Burst 15, tamaño de buffer 2 y el CV de mayor prioridad, se observa fácilmente cómo DRR-CA se comporta de manera totalmente diferente a CBFQ-CA y FCDRR (Figura 5.13 (a)), aunque para otras combinaciones de los parámetros se observa peor. Nótese que mientras en DRR-CA la productividad es la menor de todo el conjunto, dicho parámetro obtiene su mejor valor en el algoritmo CBFQ-CA y FCDRR. En la Figura 5.13 se muestra el canal de menor prioridad, en él se invierte la tendencia, obteniendo DRR-CA y Aliquem DRR-CA las mejores productividades de todas las mostradas.

Siguiendo con los comentarios sobre el algoritmo DRR-CA, se puede decir que es un algoritmo de planificación que no muestra un buen comportamiento en presencia de un tamaño de buffer pequeño, pues no puede hacer una correcta distribución del ancho de banda. Esto se debe a que cada vez que se selecciona un CV para transmitir, este debe enviar una cantidad de información proporcional al ancho de banda asignado. Dado que la cantidad mínima de información a transmitir por un CV debe de ser la MTU, si hay una diferencia muy grande entre el ancho de banda asignado a los diferentes CVs, la cantidad a transmitir por los CVs con mayor ancho de banda asignado puede ser mayor que la cantidad de información que alberga el buffer de dicho CV, es decir, el buffer no tiene suficiente cantidad de datos pertenecientes al CV como para transmitir todo lo que debería y el planificador pasaría al siguiente CV. En resumen, esta situación puede producir que los CVs con mayor ancho de banda asignado no reciban todo el ancho de banda que tienen asignado. El ancho de banda desaprovechado, será utilizado por el resto de CVs, específicamente por los que tienen asignada una menor proporción de ancho de banda, que transmitirán más información de la que deberían.




**(a) VC3** **(b)VC0**

**Figura 5.13 Tráfico Burst 15 y tamaño de buffer 2.**

En lo referente al algoritmo que se ha propuesto, FCDRR, el único comentario que se puede hacer es que su comportamiento, tanto para el parámetro de productividad, como para la latencia media y máxima, es muy similar al comportamiento medio de los otros seis algoritmos de planificación, y no muestra en ninguna situación un funcionamiento anómalo.

### 5.2.3 Efectos del parámetro "q" en el algoritmo ALIQUEM DRR-CA.

Como se ha explicado en el apartado dedicado a dicho algoritmo, al no tener en cuenta el número de colas ("q") necesarias para la estructura de colas, es posible que un CV adelante a los demás canales al dar la vuelta completa a la estructura de colas en la que se apoya el algoritmo, y por tanto no se sirven los paquetes en el orden esperado. Tras realizar la prueba con q=8 y q=16, se observan las siguientes anomalías: la productividad es casi igual en todos los CVs utilizando el menor de los tamaños de buffer. Esto se debe a que el CV de menor prioridad da una vuelta completa a la estructura de colas, se posiciona en la cola actual o en una cola posterior y podrá enviar el siguiente paquete antes de lo esperado. Lo único que consigue detener a dicho canal es el disponer de pocos paquetes que enviar en el buffer o agotar sus créditos, y como resultado se desactiva el canal. Al aumentar el tamaño de buffer conseguimos que los canales dispongan de mayor número de paquetes, produciéndose menos desactivaciones por falta de paquetes, y a partir de cierto límite, el algoritmo asigna todo el ancho de banda al canal de menor prioridad, el cual es más posible que dé una vuelta completa a la estructura de colas, dándose la paradójica situación de que el canal de menor prioridad llega a consumir todo el ancho de banda. Al repetir las simulaciones con un parámetro q mayor (q=16) se observa que el funcionamiento del algoritmo es el esperado.

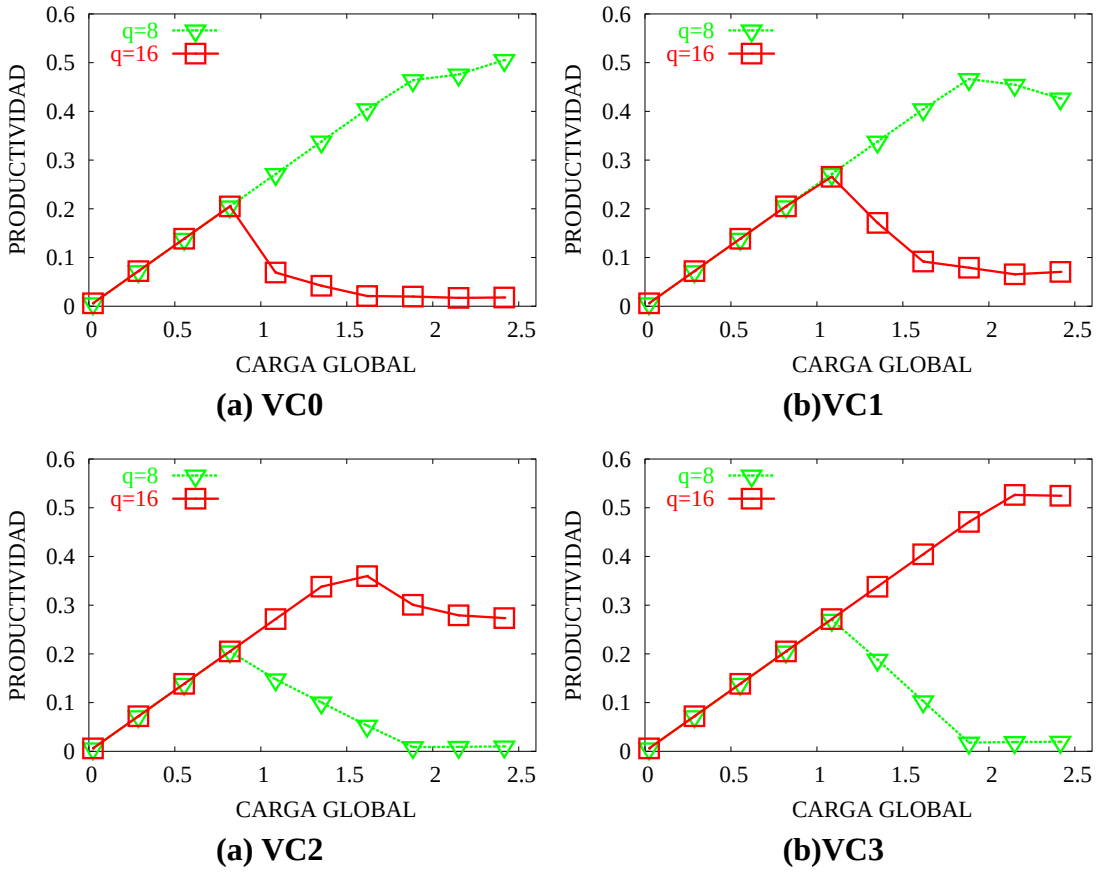



**Figura 5.14 Comparación parámetro "q"**

La gráfica para q con valor 8 está marcada como q=8 y q=16 indica que el parámetro q es 16. Puede verse que con q=8 los canales de baja prioridad y los de alta prioridad obtienen una productividad no muy diferenciada, por el contrario, con q=16 obtenemos un productividad proporcional al ancho de banda asignado y similar a la productividad mostrada en figuras anteriores para otros algoritmos de planificación.

# 6 CONCLUSIONES Y PROPUESTAS.

Finalmente, en este capítulo se presentan las conclusiones más interesantes del trabajo realizado. Para realizar una valoración más objetiva de los avances obtenidos, se retoman los objetivos iniciales planteados en el Capitulo 3. Luego se proponen una serie de trabajos futuros que han surgido durante la realización de este proyecto. No es atrevido decir que la envergadura de los mismos es igual o superior al presente estudio.

## 6.1 CONCLUSIONES.

En cuanto a los resultados del estudio hay que indicar que algunos de ellos eran totalmente esperados, simplemente por conocimiento previo o incluso por pura lógica. Sin embargo, se han obtenido datos sobre comportamientos que han resultado más complicados de explicar, y que podrían requerir de estudios más concretos para obtener información adicional que permitiera hacerlo. Se repasa a continuación los objetivos parciales que han permitido la realización de este estudio.

- **Revisión del estado del arte de las tecnologías de red y los algoritmos de planificación.**

  Este ha sido el apartado más tedioso del proyecto, pues aún partiendo de una buena bibliografía, el tiempo empleado para la lectura y aprendizaje de las tecnologías de red y algoritmos de planificación ha sido significativo. También se ha buscado información adicional para completar y aclarar conceptos que no quedaban muy claros. No obstante, la importancia de este objetivo era fundamental para la realización del proyecto. El Capitulo 2 da una visión general de los conocimientos obtenidos durante la revisión del estado del arte.

- **Estudio de las modificaciones necesarias en los algoritmos.**

  Una vez abordado el objetivo anterior y obtenidos los conocimientos necesarios sobre tecnologías de red y algoritmos de planificación, se puede estudiar cuáles son las necesidades que plantea el mecanismo de control de flujo. Este objetivo ha sido abordado en el Capitulo 4, en el cual se proponen las modificaciones necesarias tanto para algoritmos de planificación basados en “frames” como para algoritmos de planificación basados en prioridades.

- **Aprendizaje y manejo del simulador de redes de interconexión y algoritmos de planificación.**

Para abordar este objetivo ha sido fundamental la colaboración de mi tutor Raúl Martínez. Él me ha facilitado toda la información y todos sus conocimientos sobre el simulador con el que se llevan a cabo todos los ensayos para obtener los planificadores "credit aware" y los resultados de este proyecto.

- **Estudio de las modificaciones necesarias y creación de los nuevos algoritmos de planificación.**

Una vez que disponemos de los conocimientos teóricos sobre los planificadores, las modificaciones necesarias para su adaptación y la plataforma sobre la que se van a llevar a cabo las simulaciones, la creación de los nuevos algoritmos de planificación en un lenguaje de programación es una tarea que requiere paciencia. La operación más compleja es comprobar el correcto funcionamiento de los mismos. Como resultado de este objetivo, el CD anexo contiene los ficheros necesarios para la ejecución del simulador y los ficheros de implementación de los algoritmos obtenidos.

- **Obtención y análisis de resultados.**

Para el último objetivo parcial se han creado los ficheros de configuración del simulador que se han utilizado para poder realizar las pruebas. Esta tarea se ha podido realizar en un tiempo aceptable gracias a las computadoras de la red Myrinet del Instituto de Investigación en Informática de Albacete, y sin ellas el tiempo invertido para obtener los resultados habría sido ocho veces mayor. Todos los resultados obtenidos están disponibles en el CD anexo.

Como puede deducirse de los apartados anteriores, todos los objetivos parciales se han llevado a cabo, y también el objetivo principal de obtener unos algoritmos de planificación con control de flujo a nivel de enlace de red basados en los algoritmos tradicionales. A continuación se describen las conclusiones a las que se ha podido llegar al analizar los resultados obtenidos.

- Los efectos de los parámetros de diseño (tamaño del buffer y tipo de tráfico) son más importantes que los efectos que produce el cambio del algoritmo de planificación. No obstante, el funcionamiento de cada planificador puede variar sustancialmente para algunas combinaciones de los parámetros de diseño.

- Los algoritmos basados en "frames" presentan un comportamiento muy homogéneo, y ante configuraciones iguales del planificador y del conmutador, se comportan casi de la misma forma.

- El algoritmo DRR-CA no funciona de manera adecuada cuando hay una gran diferencia en el ancho de banda asignado a los diferentes CVs y el tamaño de buffer es muy pequeño.

- Los algoritmos basados en prioridades, sin embargo, presentan un comportamiento más heterogéneo, y ante la misma configuración, los CVs obtienen diferentes prestaciones dependiendo del algoritmo de planificación utilizado.

- El parámetro de latencia media no es muy significativo, pues para los diferentes planificadores y la misma configuración, se obtienen valores muy similares. El parámetro de latencia máxima resulta de más ayuda para obtener diferencias entre los algoritmos de planificación.

## 6.2 TRABAJO FUTURO.

La metodología seguida para el desarrollo de este trabajo, así como las características de los resultados obtenidos han permitido observar detalles que invitan a plantear una continuidad al mismo.

Se indican a continuación algunas de esas líneas de trabajo que podrían dar continuidad al estudio que aquí se ha presentado:

- Realizar el estudio con más parámetros de diseño, por ejemplo el *threshold*, que es el límite inferior en el nivel del buffer de entrada a partir del cual se envían créditos de control de flujo al conmutador anterior.

- Realizar la implementación de otros algoritmos de planificación. En la literatura existen multitud de variaciones sobre los algoritmos tradicionales, dichas variaciones podrían ser determinantes para obtener un buen algoritmo de planificación.

- Repetir el estudio comparando las prestaciones obtenidas en los algoritmos "credit aware" con los algoritmos tradicionales. Esto permitiría mostrar las ventajas de estas nuevas implementaciones.

- Repetir el estudio comparando los algoritmos de planificación respecto a otros índices como el jitter o la equidad.

# 7 BIBLIOGRAFIA.


[Adv03] Advanced Switching Interrconnect Special Interest Group. Advanced Switching core architecture specification. Revision 1.0, December 2003.

[Adv05] Advanced Switching Interrconnect Special Interest Group. Advanced Switching core architecture specification. Revision 1.1, March 2005.

[Ascas] Advanced Switching core architecture specification. Technical report, disponible en http://www.asi-sig.org/specifications

[Alf04] F. J. Alfaro, J. L. Sánchez y J. Duato "QoS in Infiniband Subnetworks." IEEE Trans. Parallel and Distrib. Sys., vol. 15, no 9, Sept. 2004.

[Ben01] Brahim Bensaou, Danny H. K. Tsang and King Tung Chan "Credit-based Fair Queuing (CBFQ): A simple service-scheduling algorithm for packet-switched networks" ACM Transactions on Computer Systems, Oct 2001

[Ben96a] Jon C. R. Bennett and Hui Zhang "Why WFQ is not good enough for integrated services networks"

[Ben96b] Jon C. R. Bennett and Hui Zhang "WF2Q: Worst-Case Fair Weigthed Fair Queuing" IEEE INFOCOM 1996

[Ben97] Jon C. R. Bennett and Hui Zhang. "Hierarchical packet fair queueing algorithms". IEEE/ACM Trans. Ntew., Vol. 5, No 5, pp. 675-689, 1997.

[Ber98] Yoram Bernet. "A framework for Differentiated Services". Internet draft 2275, Internet Engineering Task Force, May 1998.

[Bla98] S. Blake, D. Back, M. Carlsom, E. Davies, Z. Wang, W. Weiss. "An Architecture for Differentiated Services". Internet Request for Comment RFC2475, Internet Engineering Task Force, December 1998.

[Bla00] U. Black. "QoS in Wide Area Networks". Prentice Hall Series in Advanced Communications Technologies. Prentice Hall 2000

[Bra94] R. Bramen, D. Clark, S. Shenker. "Integrated Services in the Internet Architecture: an Overview". Internet Request for Comment RFC1633. Internet Engineering Task Force, June 1994

[Cam00] M. Blanca Caminero y Francisco J. Quiles "Técnicas de planificación de conmutadores orientadas a garantizar QoS a tráfico multimedia" Dpto Informática Albacete, Febrero 2000

[Cha03] H.M. Chaskar, U. Madhow. "Fair scheduling with tunable latency: A round-robin aproach". IEEE/ACM Trans. Ntew., Vol. 11, No 4, pp. 592-601, 2003.

[Chr04] S. Christo. "Markets converge on advanced switching". RTC Magazine, May 2004. www.rtcmagazine.com/home/article.php?id100018

[Chr05] Stephen Christo. "ASI Vs. Ethernet: Evaluating Interconnect Choices" Penton Media, Inc., March 2005

[Chu01] G. Chuanxiong "SRR: an O(1) time complexity packet scheduler for flows in multi-service packet networks" ACM SIGCOMM, 2001

[Dem89] A. Demers, S. Keshav and S. Shenker "Analysis and simulation of a Fair Queuing algorithm" ACM SIGCOMM, 1989

[Dom00] M. Domene Rodriguez "Evaluación de prestaciones de un conmutador de red Myrinet para redes de estaciones de trabajo", PFC Universidad Politécnica Valencia.

[Fer98] P. Ferguson, G. Huston. "Quality of service: delivering QoS on the Internet and corporate networks". John Wiley & Son, 1998

[Fir01] V. Firoiu, J-Y. Le Boudec, D. Towsley, Z-L. Zhang. "Advances in Internet quality of service". Technical report, National Science Foundation 2001.

[Gir98] N. Giroux and S. Ganti. "Quality of Service in ATM networks" Prentice Hall PTR, 1998

[Gol94] S. J. Golestani "A Self-Clocked Fair Queuing Scheme for Broadband Applications" IEEE INFOCOMM 1994

[Gre92] A. G. Greenberg, N. Madras. "How fair is fair queuing".J. ACM, Vol. 39, No. 3, pp. 568-598, 1992.

[Hal01] Fred Halsall. "Multimedia Communications: Applications, Networks, Protocols and Standard". Addison-Wesley, 2001.

[Har05] Harpinder S. Matharu "Evaluating high speed industry standart serial interconnects" Embedded Computing Design, July 2005

[Inf00] InfiniBand Trade Association. InfiniBand architecture specification volume 1. Release 1.0, October 2000.

[Kan02] S. S. Kanhere, H. Sethu, A. B. Parekh. "Fair and efficient packet scheduling using elastic round robin". IEE Transactions on Parallel and Distributed Systems, 2002.

[Len04] L. Lenzini E. Mingozzi and G. Stea "Tradeoffs between low complexity, low latency and fairness with Deficit Round-Robin Schedulers" IEEE/ACM Transaction on networking, 2004

[Lie99] J. Liebeherr, E. Yilmaz. "Workconserving vs non-workconserving packet scheduling: An issue revisited". IEEE/IFIP International Workshop on Quality of Service (IWQOS), May 1999.

[Mar06] Raúl Martínez, Francisco J. Alfaro and José L. Sánchez "A framework for providing Quality of Service over Advanced Switching" 2006

[Par93] A. K. Parekh and R. G. Gallager "A generalized processor sharing approach to flow control in integrated services networks: The single-node case" IEEE/ACM Transaction on networking, 1993

[Par94] A. K. Parekh and R. G. Gallager "A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. IEEE/ACM Transaction on networking, 1994

[Par05] K. I. Park. QoS in Packet Networks. Springer, 2005.

[Raj05] Rajeev Kumar and Harpinder S. Matharu "A comparison of PCI Express, ASI and Serial RapidIO" november 2005

[Rei06] Sven-Arne Reinemo, Tor Skeie, Thomas Sødring and Olav Lysne "An overwiev of QoS capabilities in InfiniBand, Advanced Switching Interconnect and Ethernet." IEEE Communications Magazine, July 2006.

[She98] R. Sheifert. Gigabit Ethernet. Addison-Wesley, April 1998.

[Shr95] M. Shreedhar and G. Varghese "Efficient Fair Queuing using Deficit Round Robin" SIGCOMM, 1995

[Sti96] D. Stidialis and A. Varma "Latency-Rate Servers: A general model for analysis of traffic scheduling algorithms" IEEE/ACM Transaction on networking, 1996

[Wan01] Z. Wang. "Internet QoS: Architecture and Mechanisms for Quality of Service". Morgan Kaufmann, 2001.

[XL05] Jun Xu, Richard J. Lipton. "On fundamental tradeoff between delay bounds and computational complexity in packet scheduling algorithms". IEEE/ACM Trans Netw,, Vol 13, No. 1, pp. 15-28, 2005.

[Zha91] Lixia Zhang "Virtual Clock: A new traffic control algorithm for packet-switched networks" ACM Transactions on Computer Systems, May 1991

[Zha95] H. Zhang. "Service disciplines for guaranteed performance service in packet-switching networks". 1995.

[Zha02] Guansong Zhang "EL938 Report: Packet scheduling" Dpto of electrical and computer engineering New York, April 2002

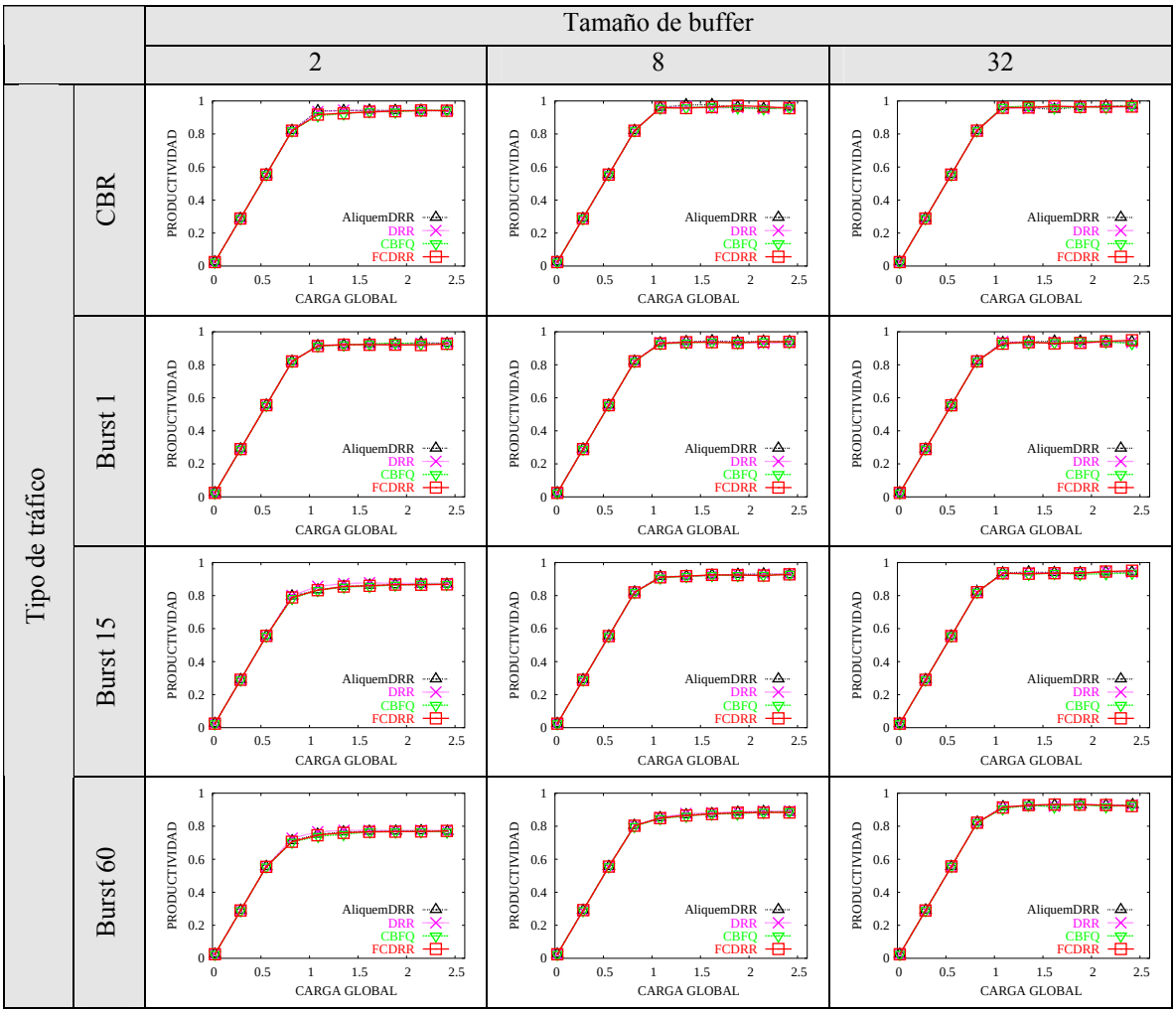
[Zha04] Q. Zhao, J. Xu. "On the computational complexity of maintaining gps clock in packet scheduling". IEEE INFOCOM, March 2004.

# 8 ANEXO.

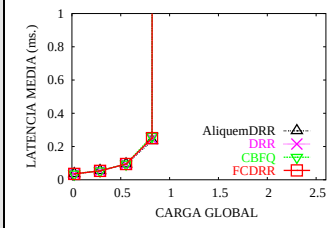
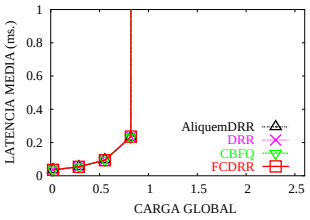
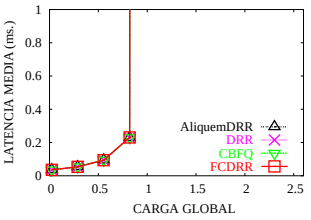
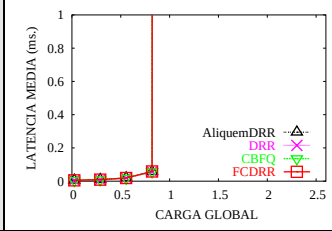
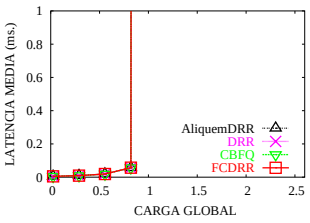
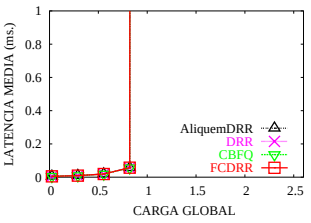
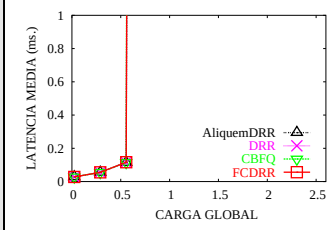
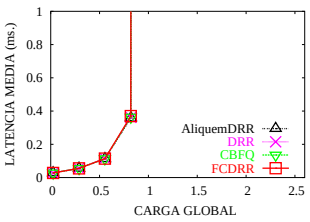
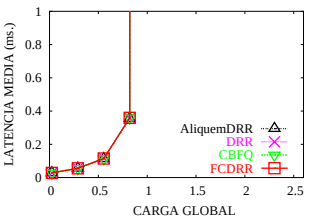
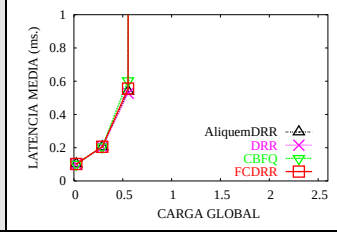
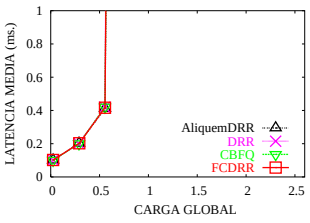
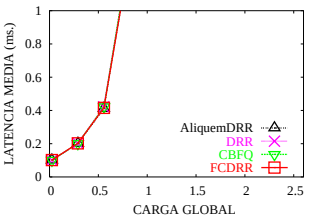
Las gráficas obtenidas tras las simulaciones se muestran a continuación. Para mostrarlas de una forma más clara se dividen en algoritmos basados en "frames" y algoritmos basados en prioridades, dentro de cada uno de estos apartados hay una tabla que muestra el comportamiento de los algoritmos según los parámetros de diseño vistos en la Sección 5.2 y luego se muestran las gráficas obtenidas para cada uno de los CVs estudiados. Para cada canal se muestra la productividad, latencia media y latencia máxima.

**Algoritmos basados en frames:**

Productividad global:

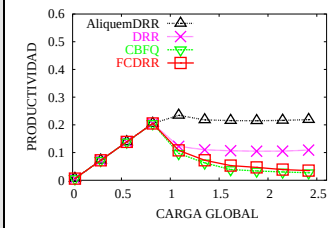
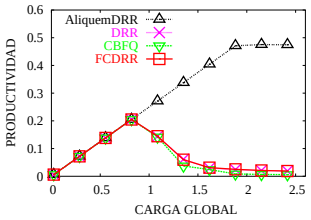
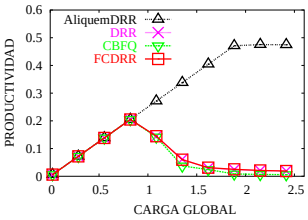
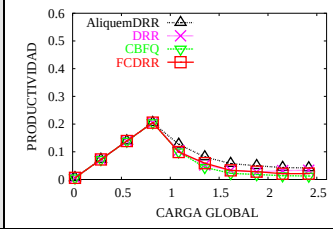
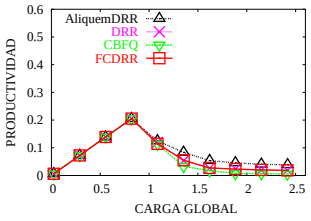
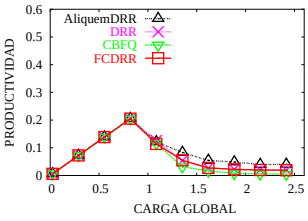
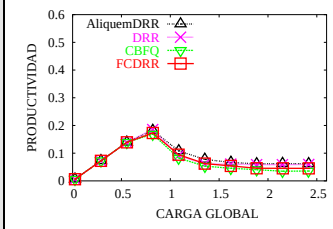
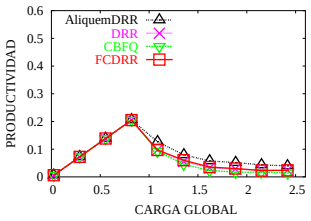
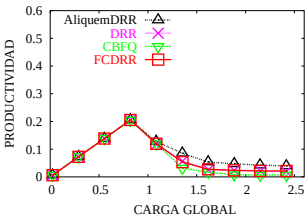
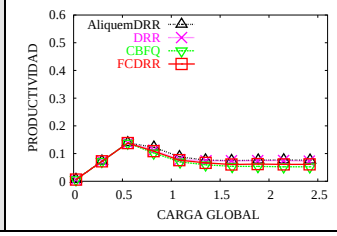
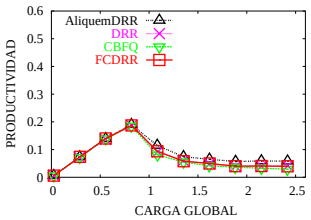
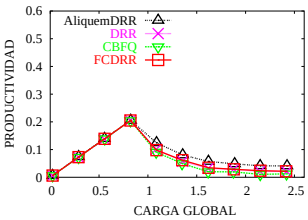


## Latencia global:

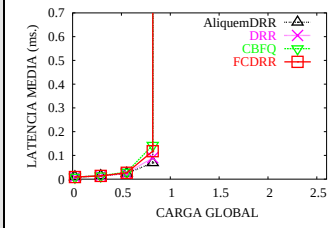
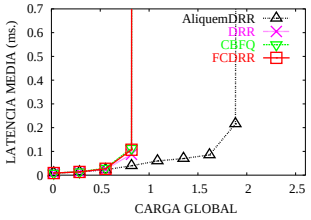
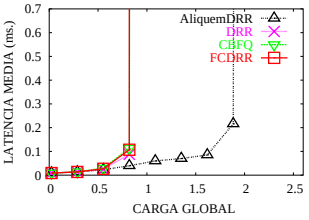
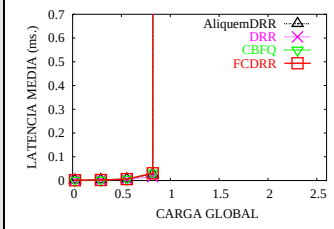
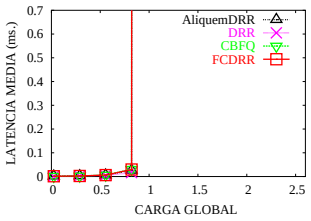
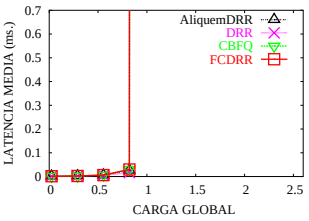
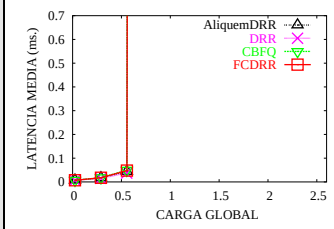
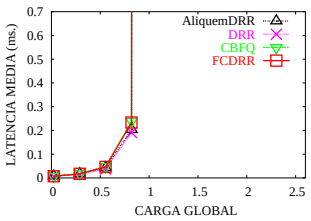
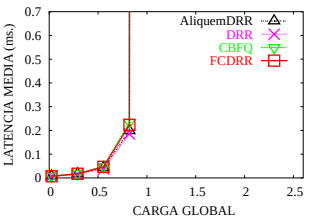
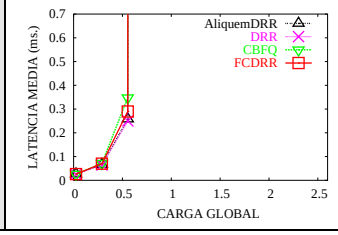
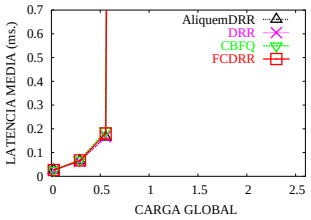
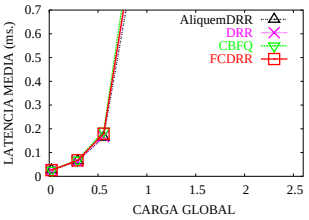
| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

Cada una de las tablas posteriores mostrará las gráficas de productividad, latencia media y máxima para cada CV.

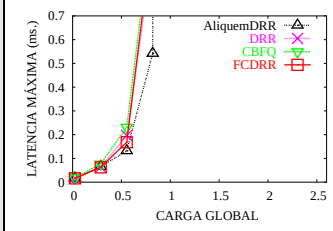
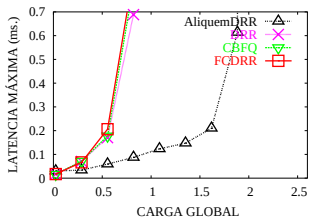
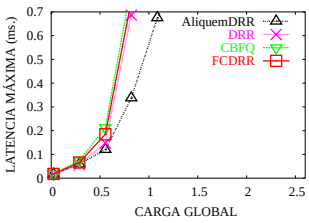
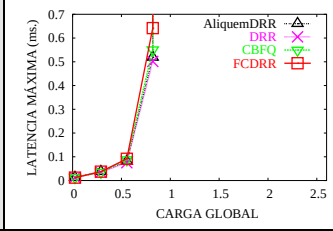
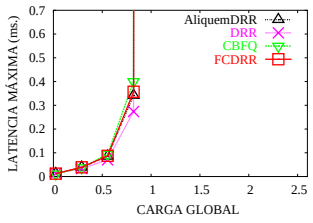
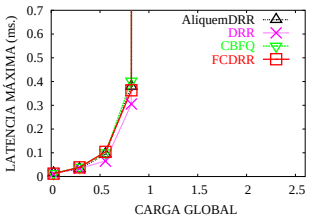
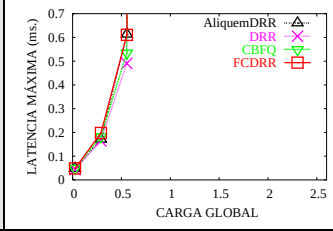
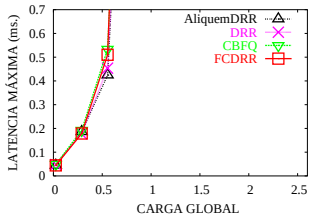
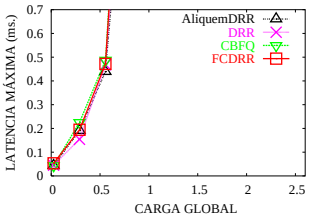
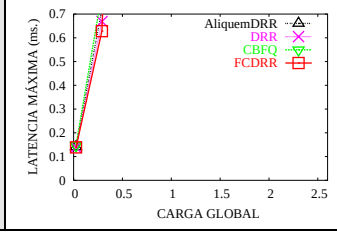
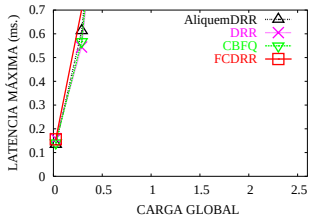
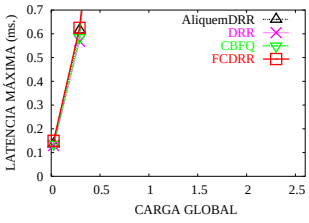
Productividad VC0:

| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

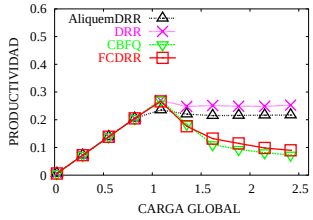
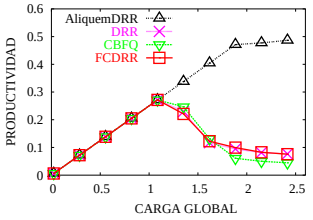
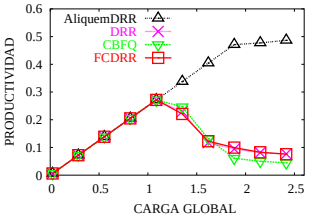
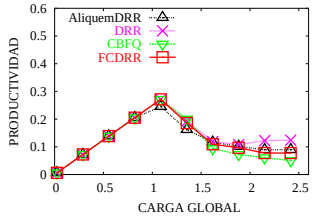
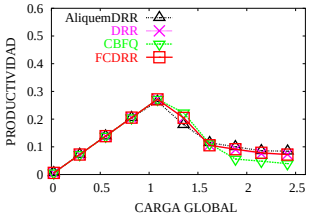
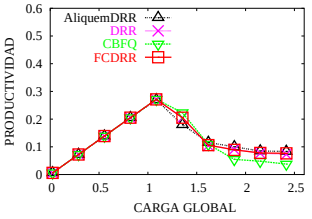
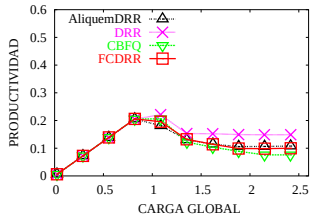
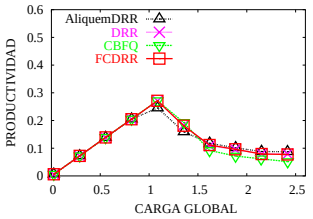
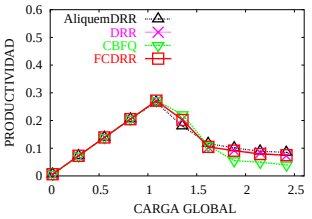
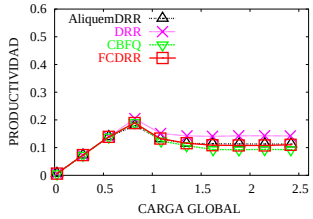
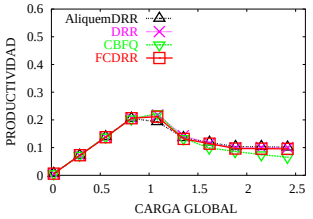
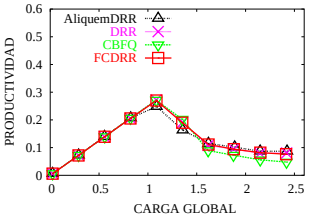
Latencia VC0:

| Tipo de tráfico | Tamaño de buffer = 2 | Tamaño de buffer = 8 | Tamaño de buffer = 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

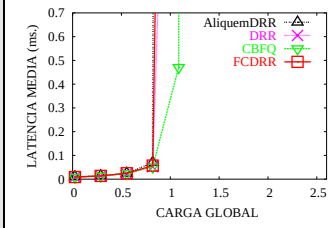
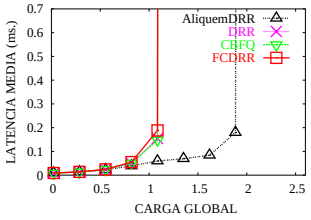
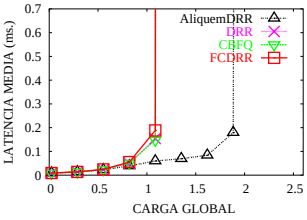
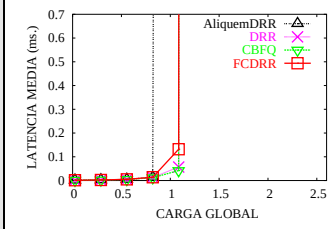
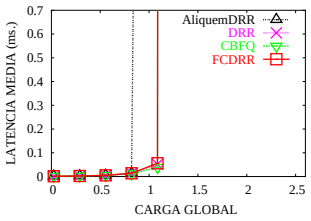
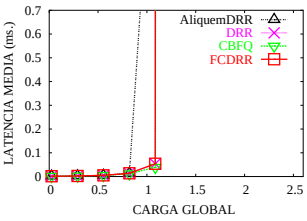
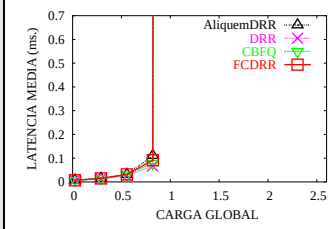
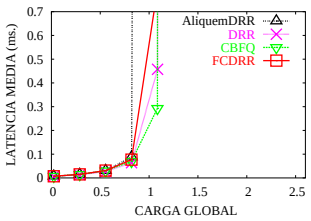
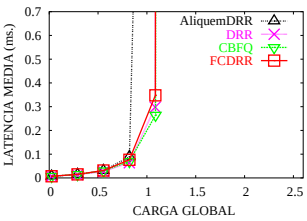
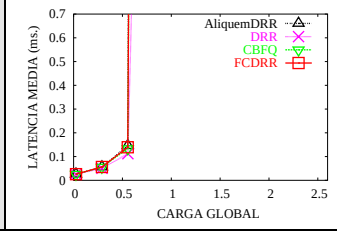
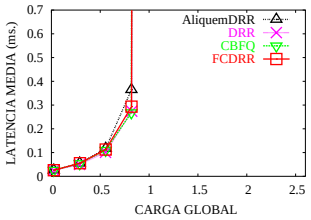
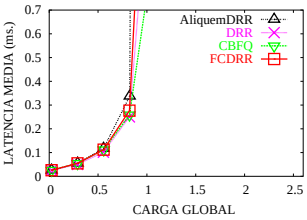
Latencia máxima VC0:

| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

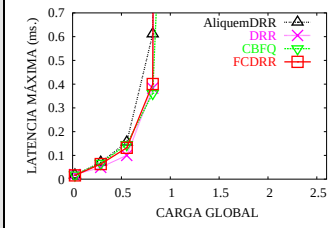
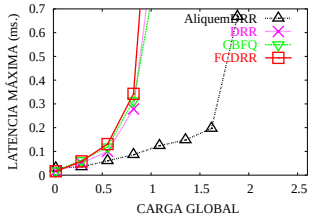
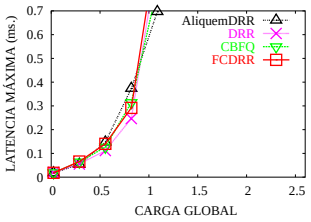
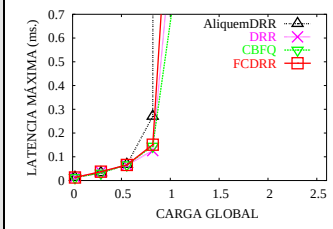
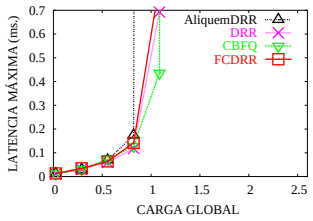
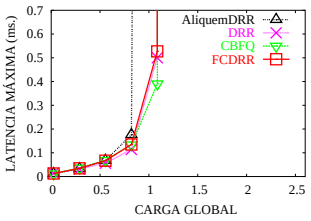
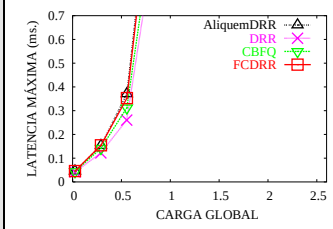
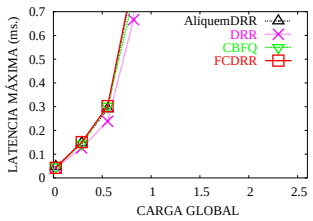
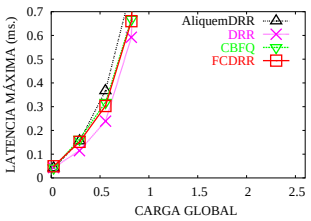
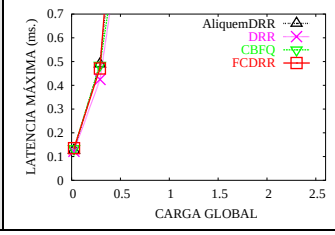
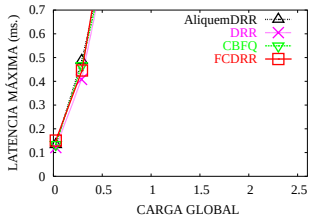
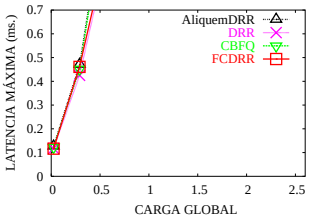
Productividad VC1:

| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

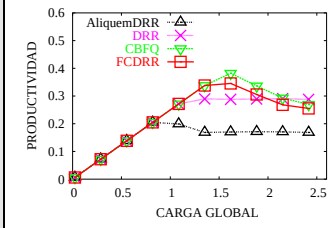
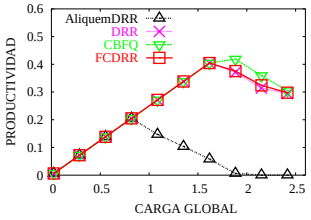
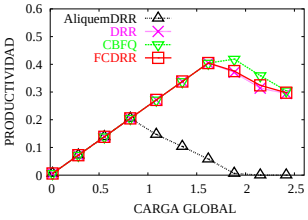
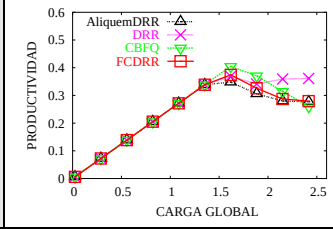
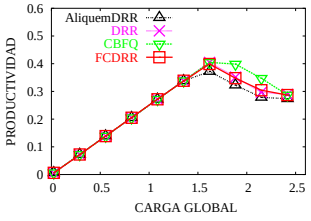
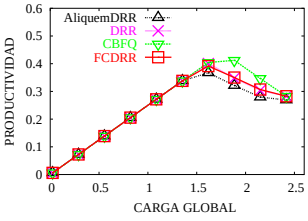
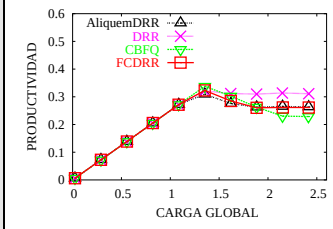
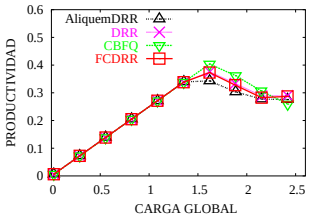
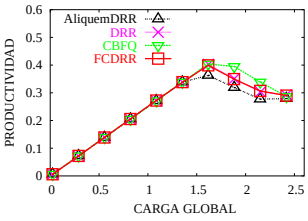
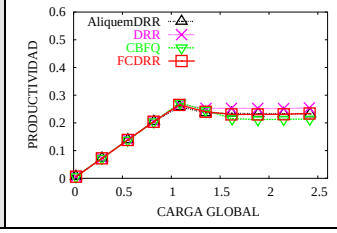
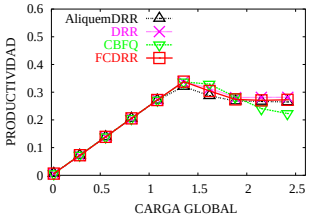
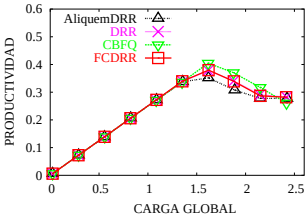
Latencia VC1:

| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

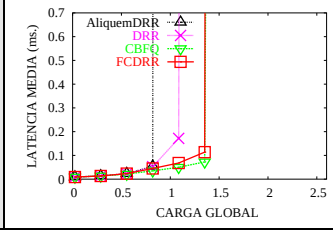
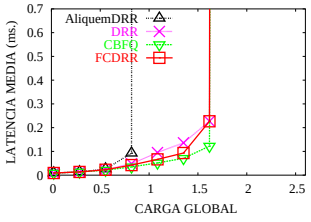
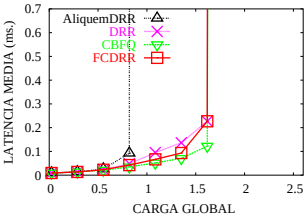
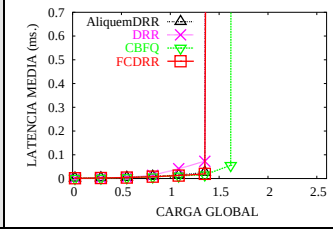
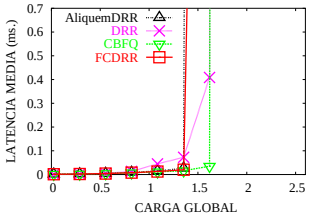
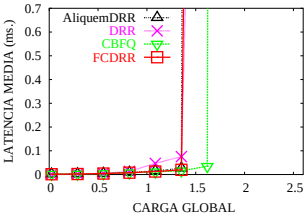
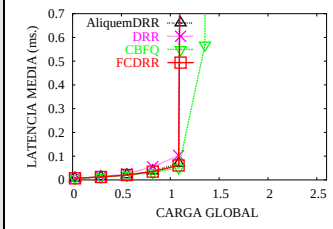
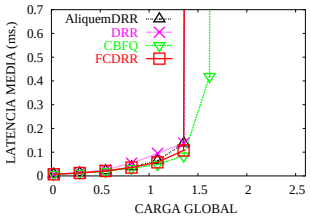
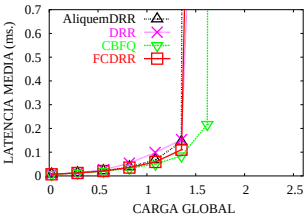
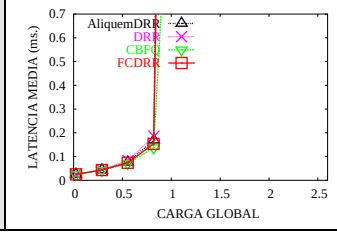
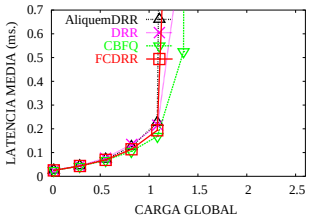
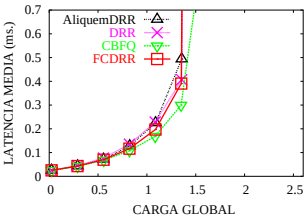
Latencia máxima VC1:

| Tipo de tráfico | Tamaño de buffer = 2 | Tamaño de buffer = 8 | Tamaño de buffer = 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

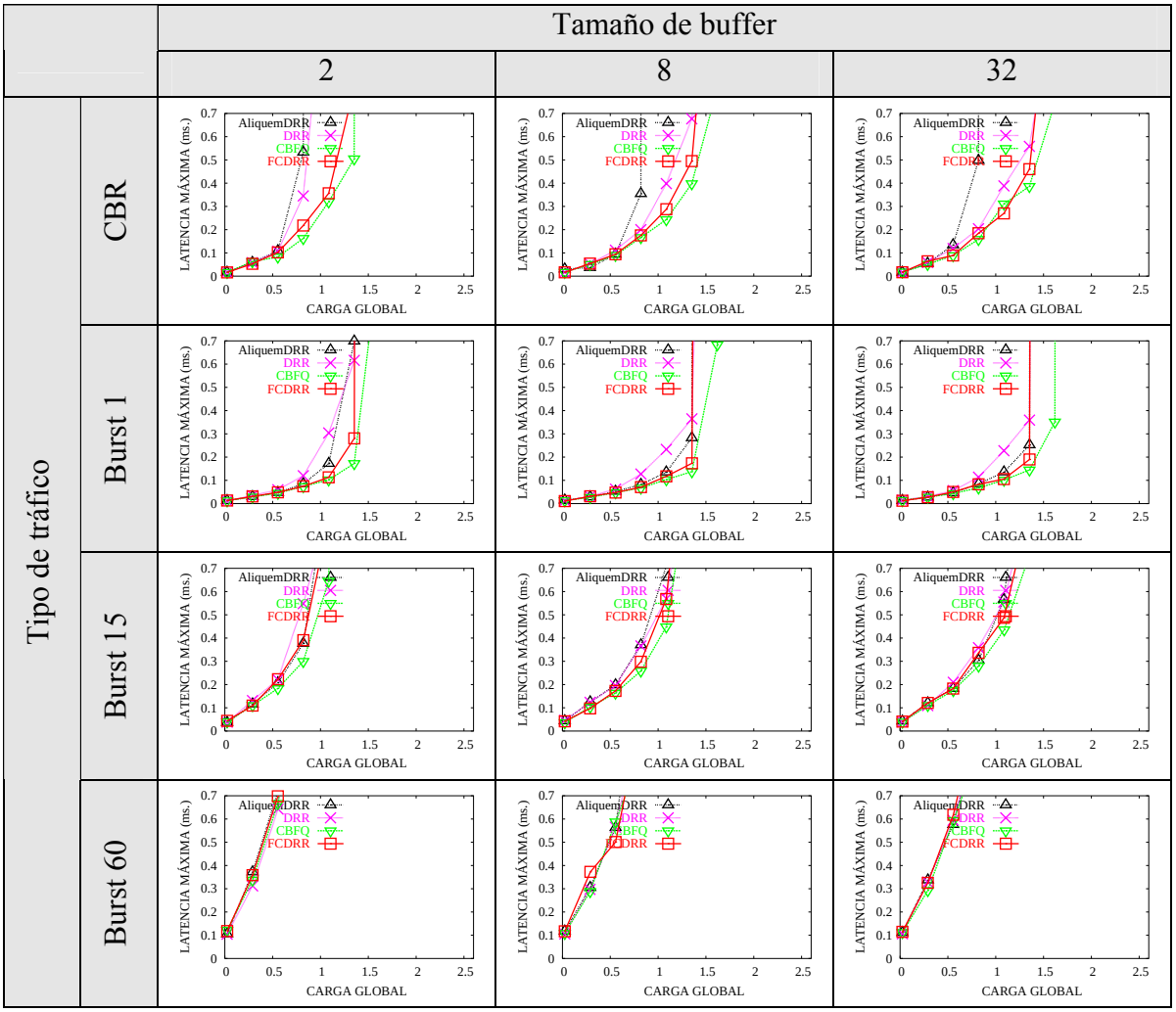
Productividad VC2:

| Tipo de tráfico | Tamaño de buffer = 2 | Tamaño de buffer = 8 | Tamaño de buffer = 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

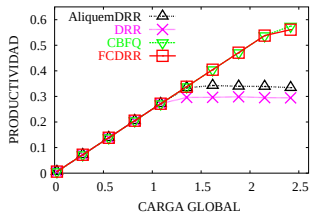
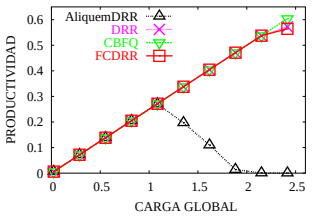
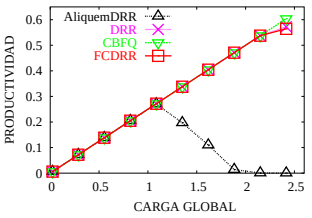
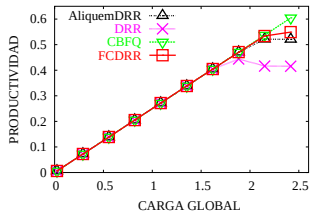
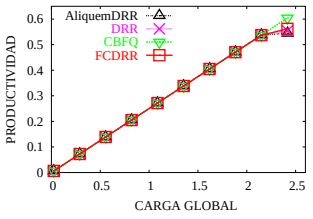
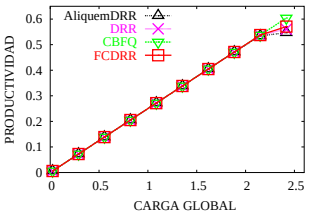
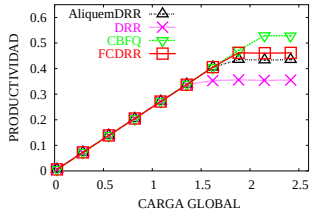
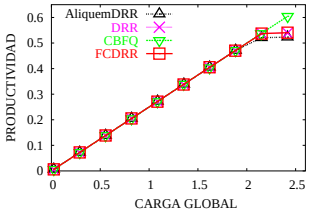
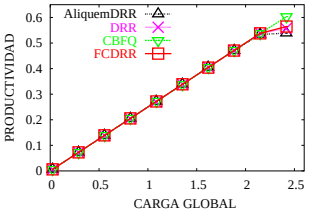
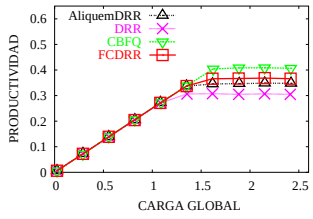
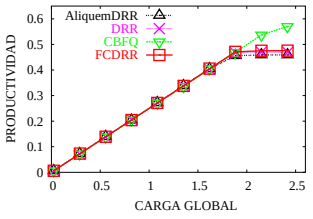
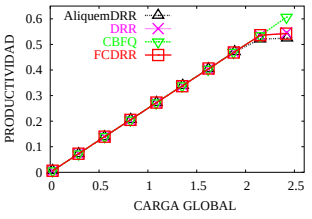
Latencia VC2:

| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

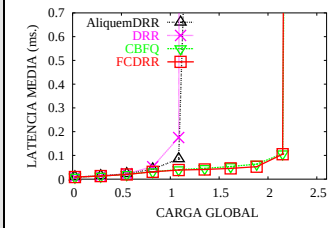
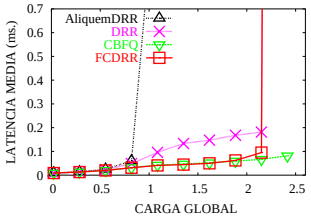
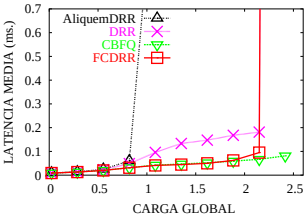
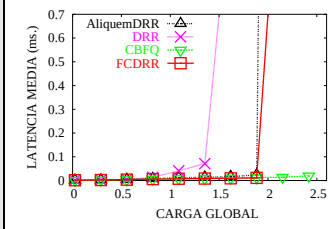
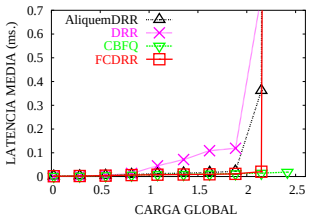
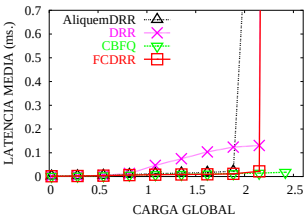
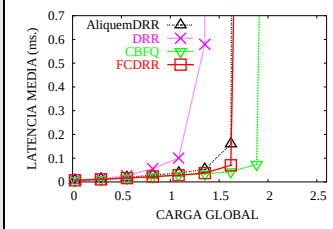
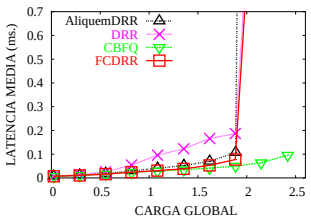
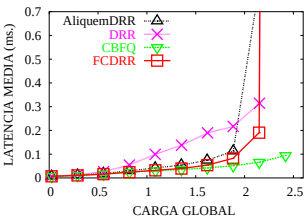
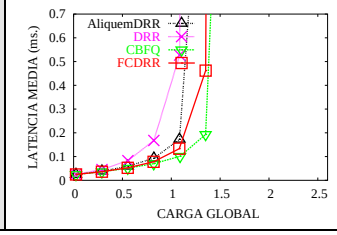
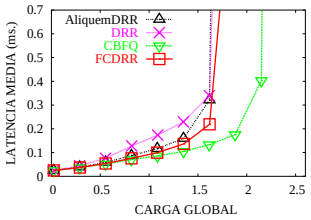
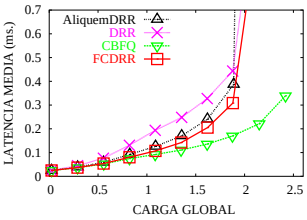
Latencia máxima VC2:



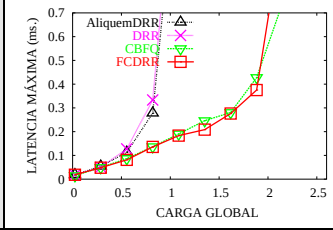
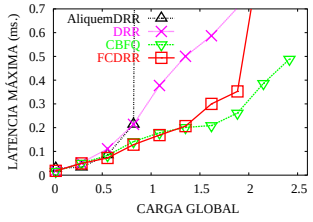
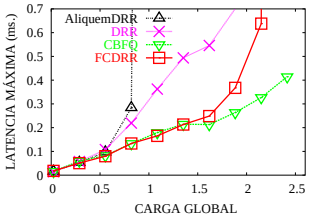
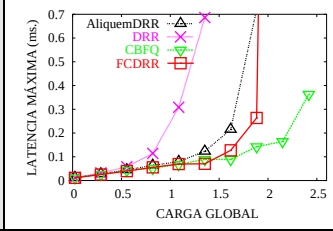
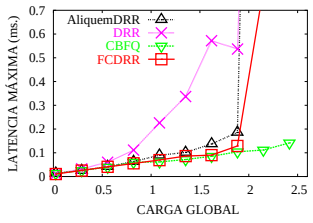
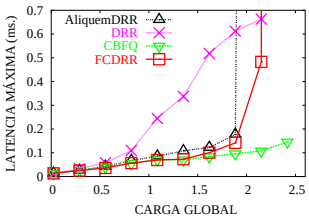
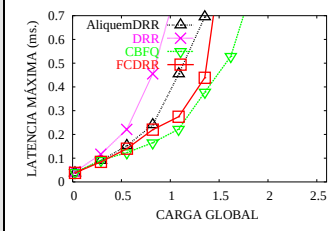
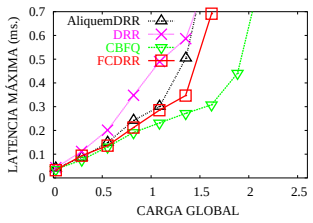
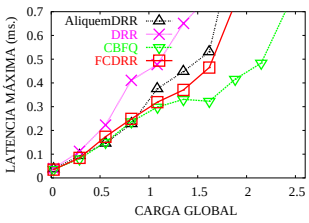
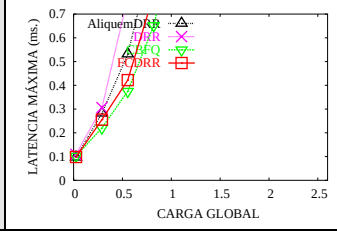
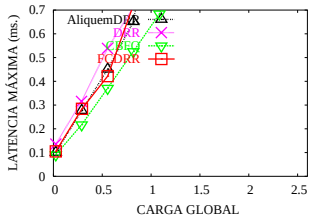
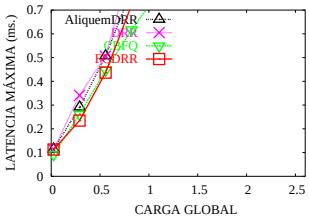
Productividad VC3:

| Tipo de tráfico | Tamaño de buffer = 2 | Tamaño de buffer = 8 | Tamaño de buffer = 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

Latencia VC3:

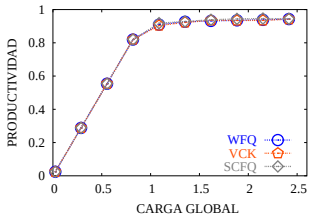
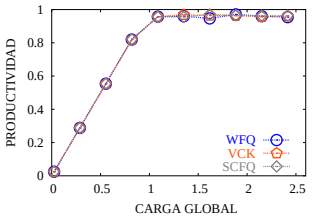
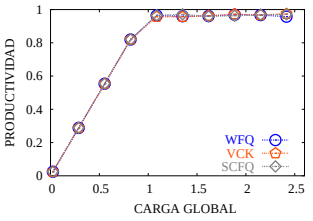
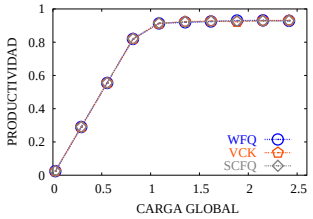
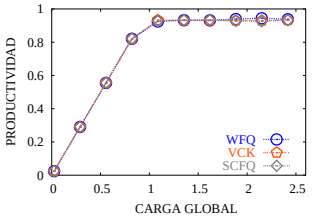
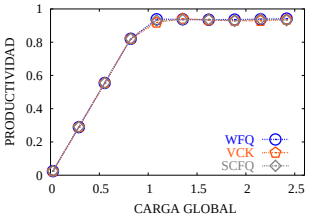
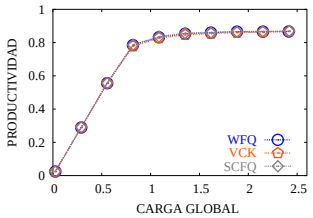
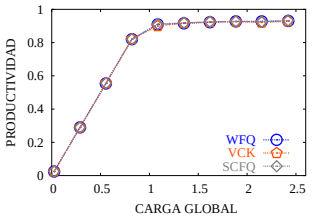
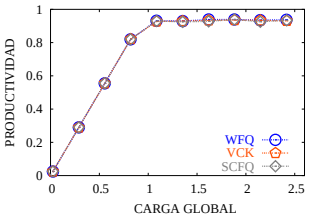
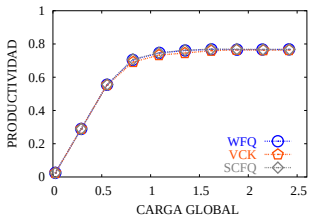
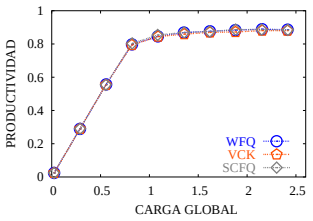
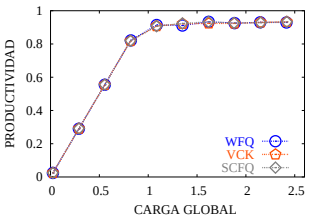
| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

Latencia máxima VC3:

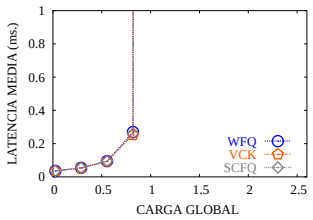
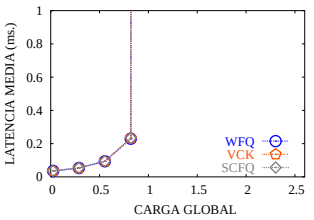
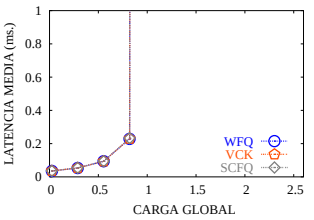
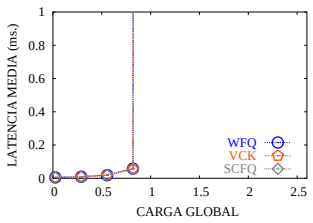
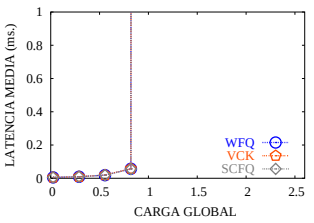
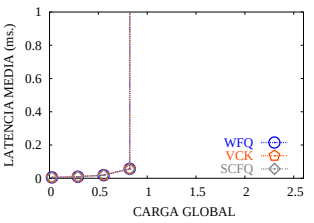
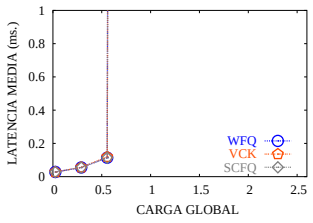
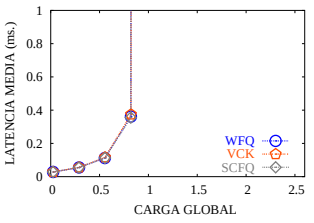
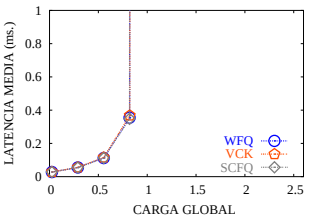
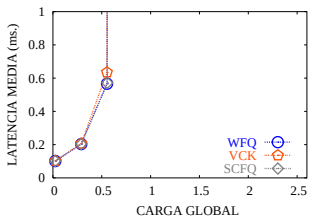
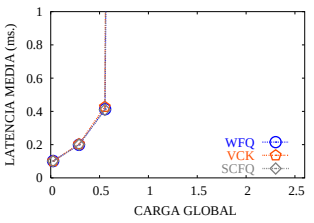
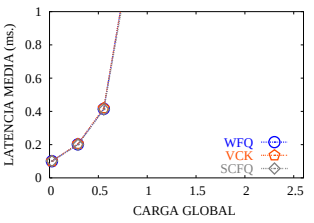
| Tipo de tráfico | Tamaño de buffer = 2 | Tamaño de buffer = 8 | Tamaño de buffer = 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

**Algoritmos basados en prioridades:**

Productividad global:

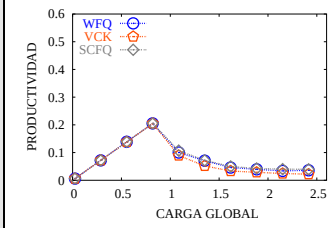
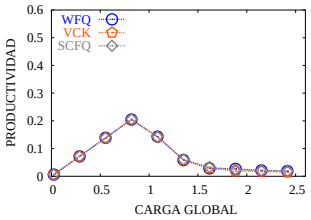
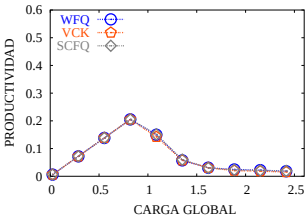
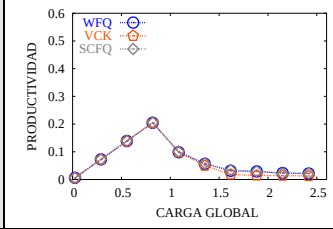
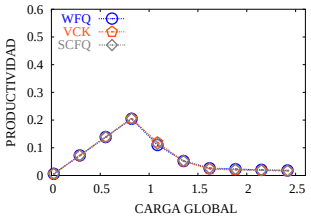
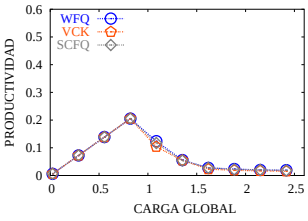
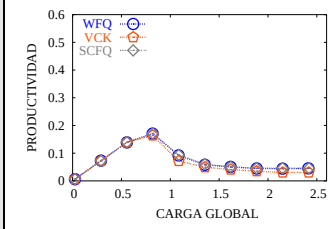
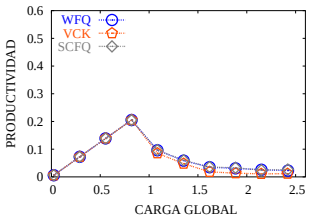
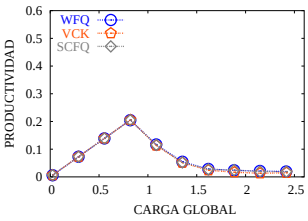
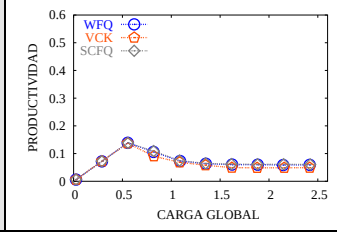
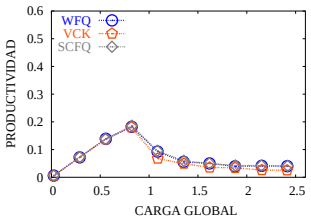
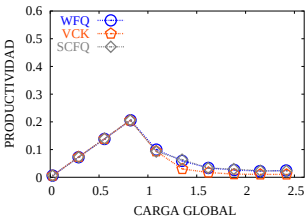
| Tipo de tráfico | Tamaño de buffer = 2 | Tamaño de buffer = 8 | Tamaño de buffer = 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

## Latencia global:

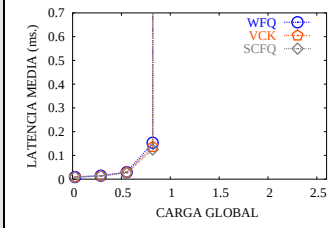
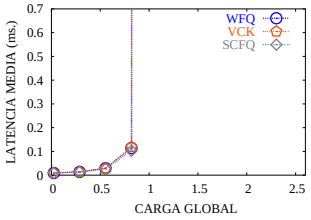
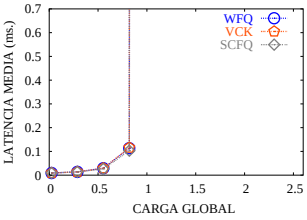
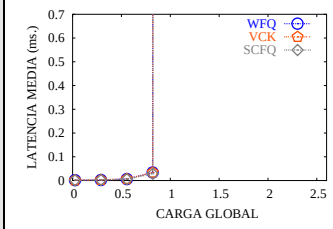
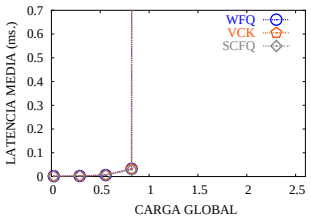
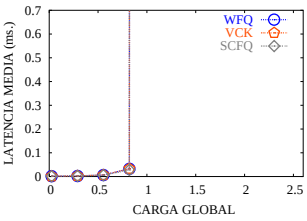
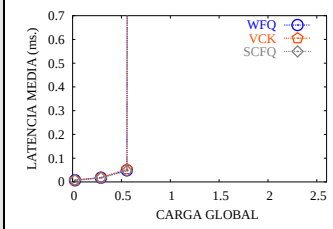
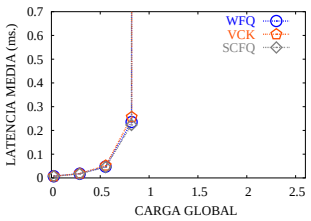
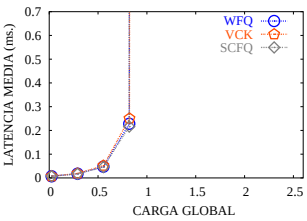
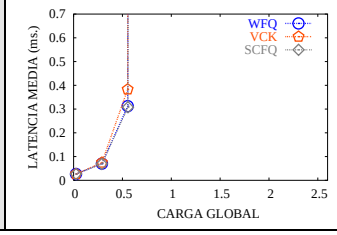
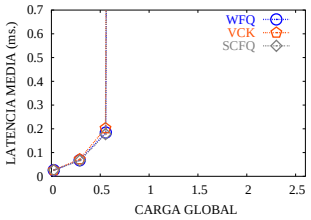
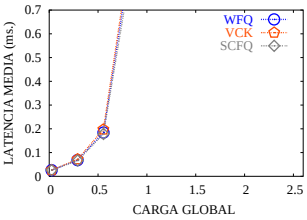
| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

Cada una de las tablas posteriores mostrará las gráficas de productividad, latencia media y máxima para cada CV.

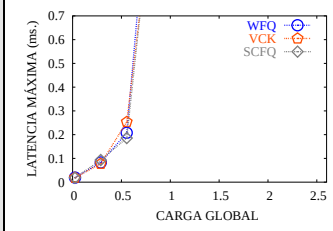
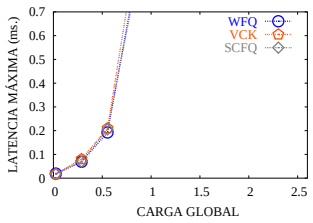
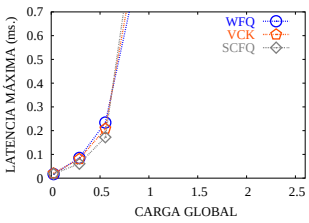
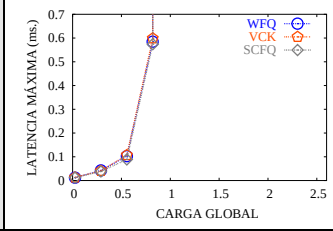
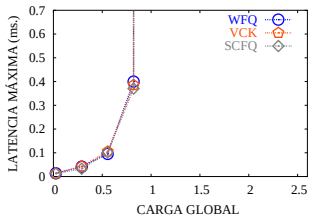
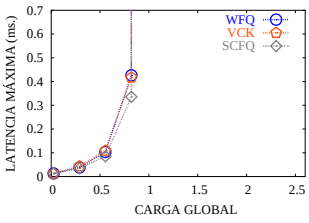
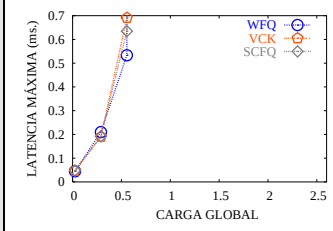
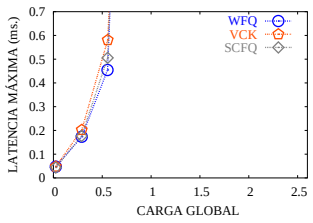
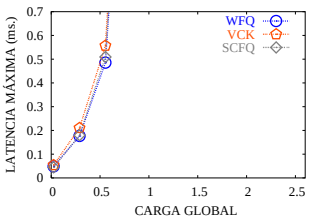
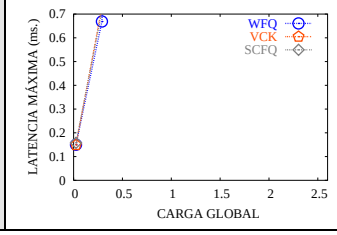
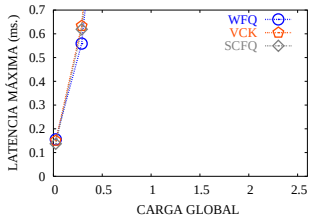
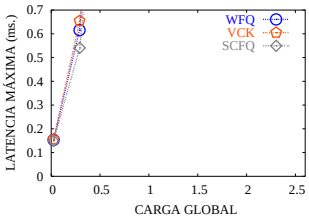
Productividad VC0:

| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

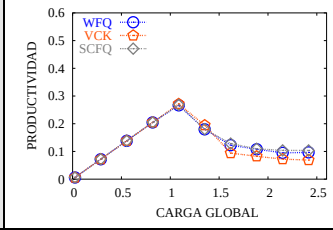
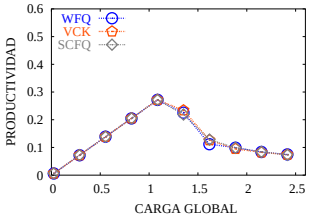
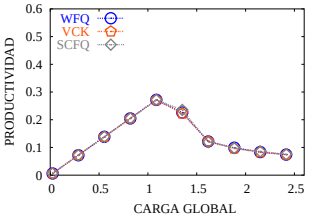
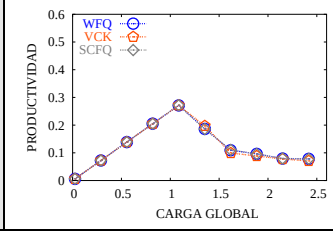
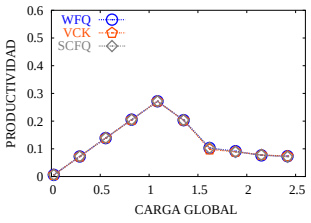
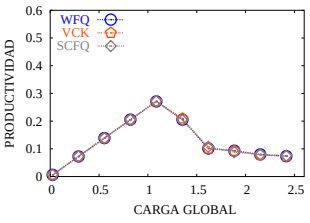
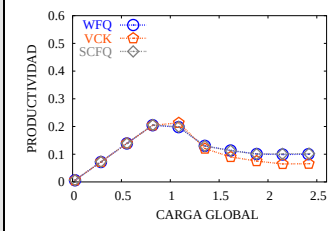
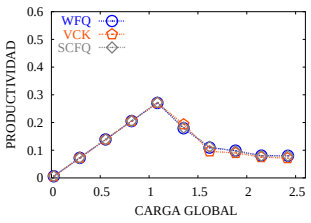
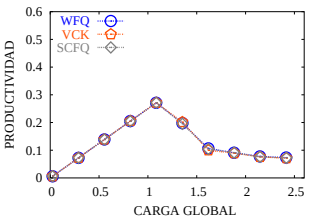
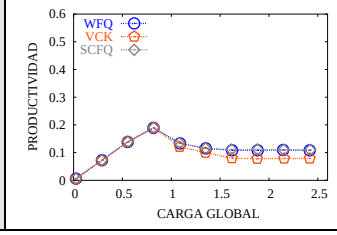
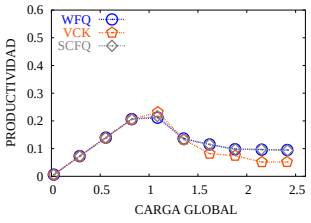
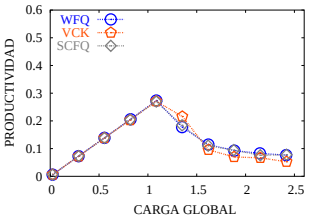
Latencia VC0:

| Tipo de tráfico | Tamaño de buffer: 2 | Tamaño de buffer: 8 | Tamaño de buffer: 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

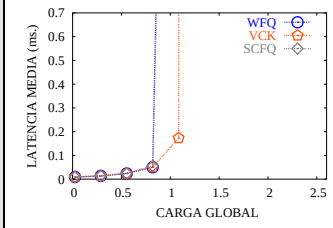
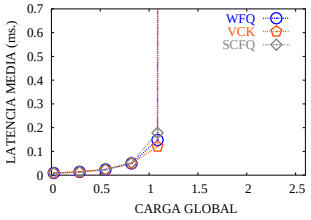
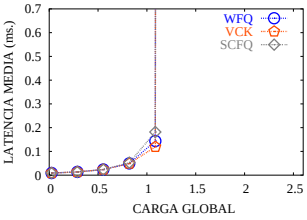
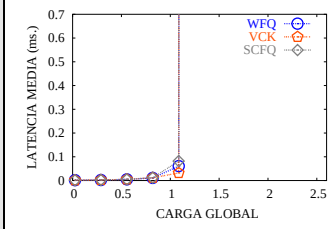
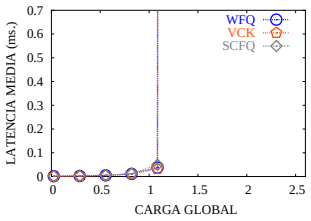
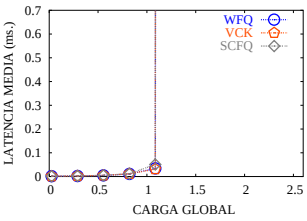
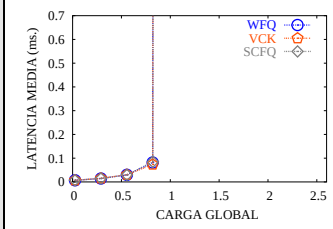
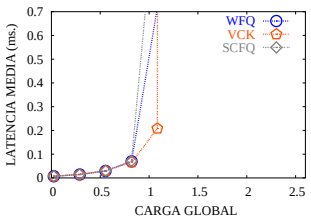
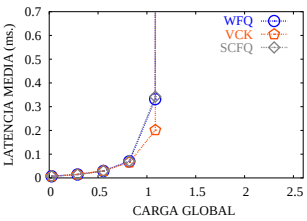
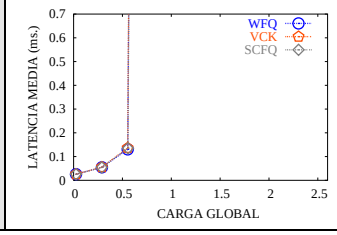
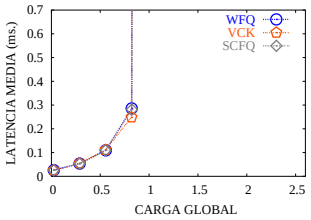
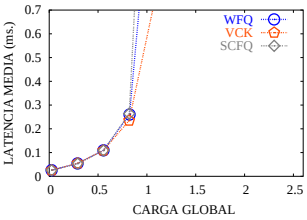
Latencia Máxima VC0:

| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

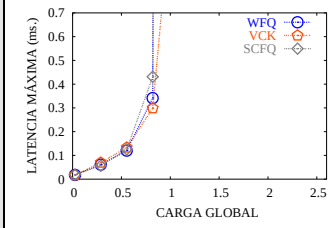
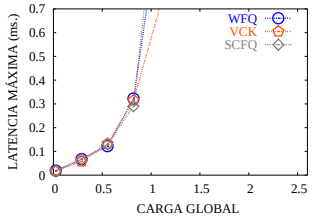
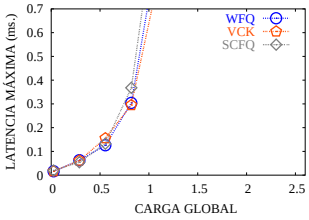
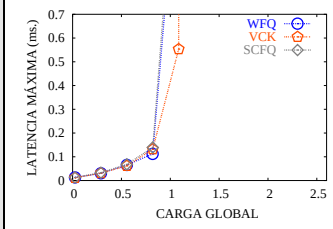
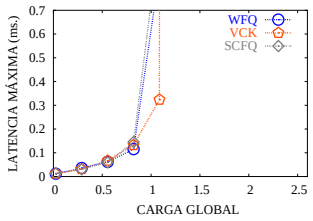
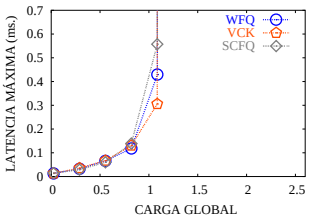
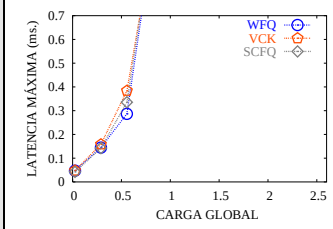
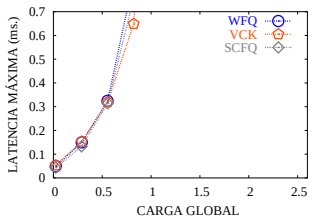
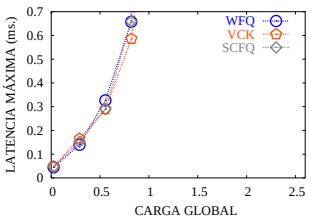
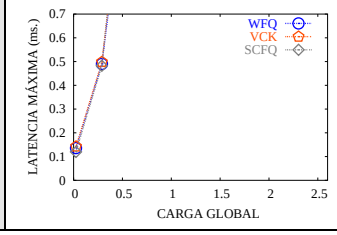
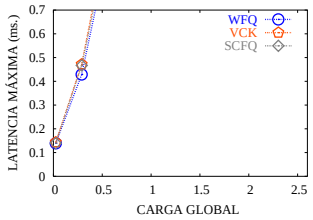
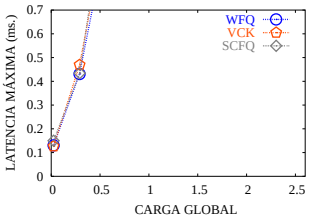
Productividad VC1:

| Tipo de tráfico | Tamaño de buffer = 2 | Tamaño de buffer = 8 | Tamaño de buffer = 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

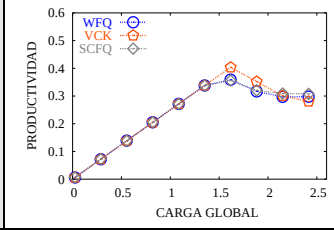
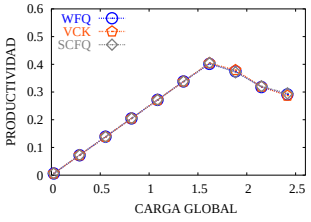
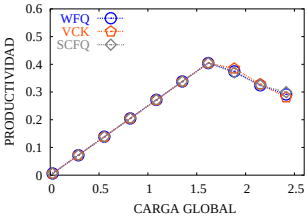
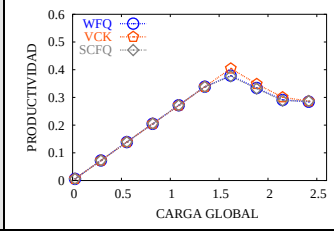
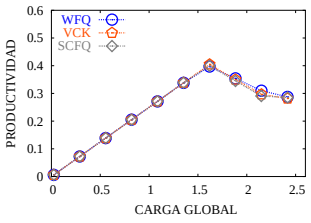
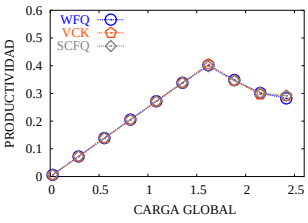
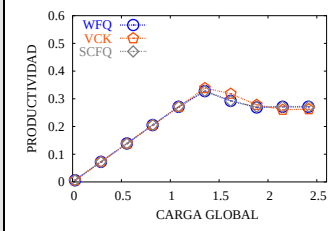
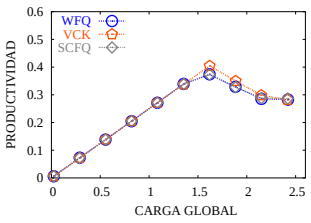
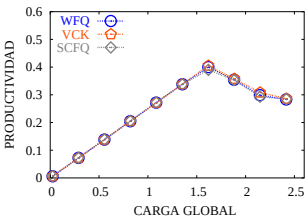
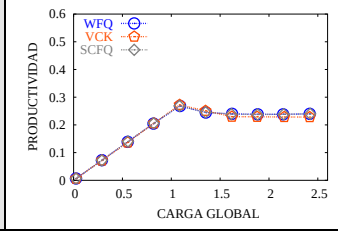
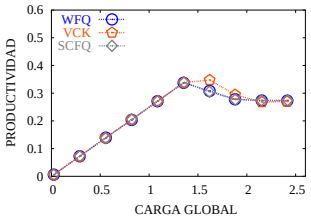
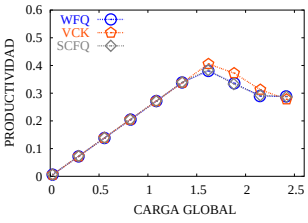
Latencia VC1:

| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

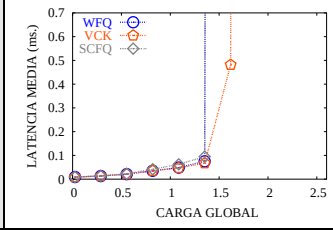
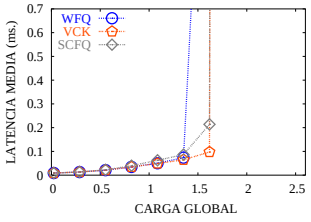
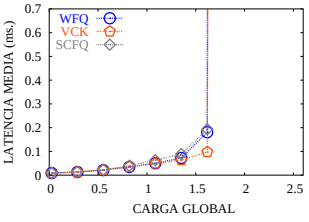
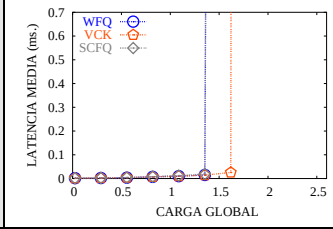
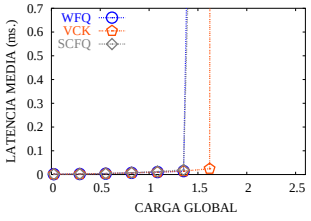
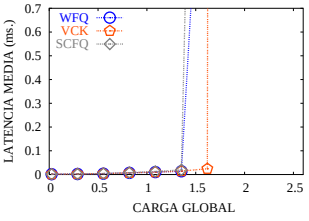
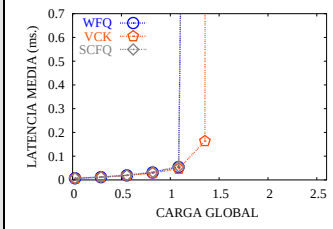
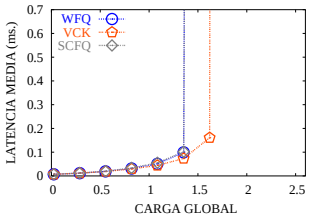
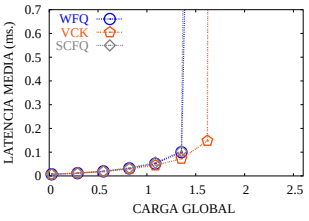
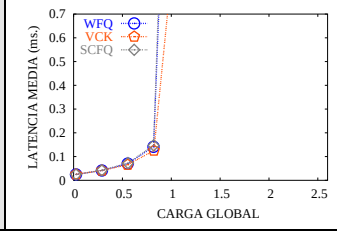
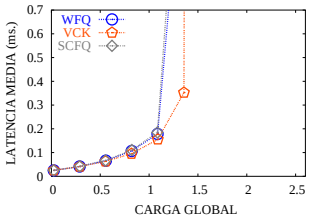
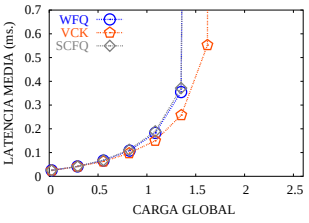
Latencia Máxima VC1:

| Tipo de tráfico | Tamaño de buffer = 2 | Tamaño de buffer = 8 | Tamaño de buffer = 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

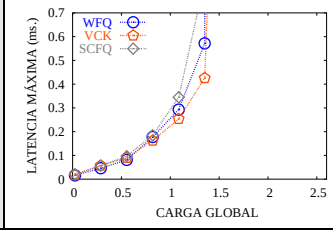
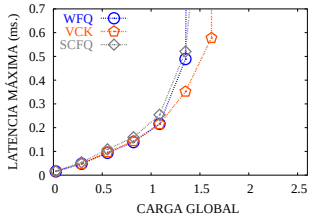
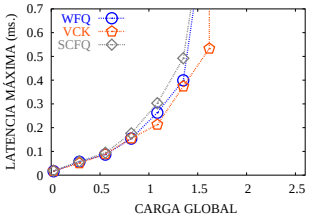
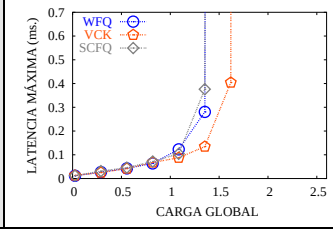
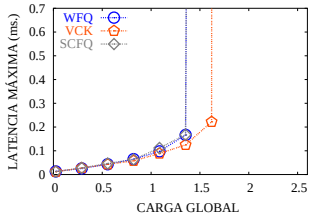
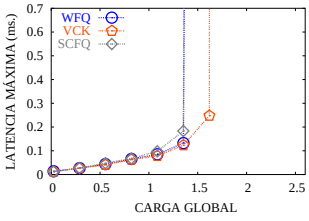
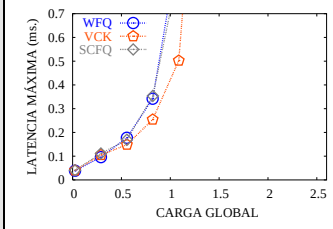
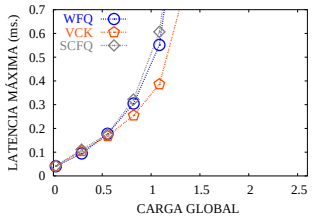
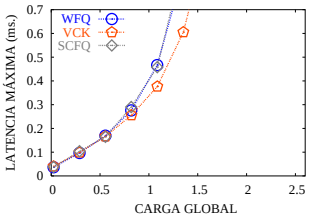
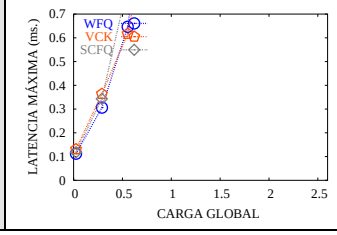
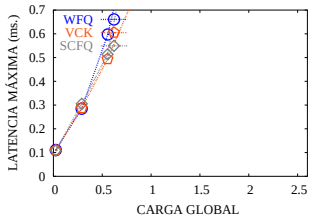
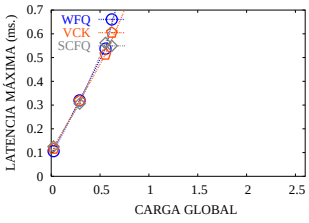
Productividad VC2:

| Tipo de tráfico | Tamaño de buffer = 2 | Tamaño de buffer = 8 | Tamaño de buffer = 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

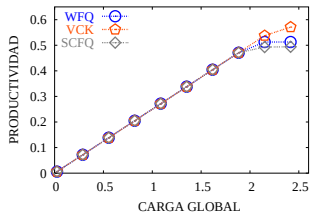
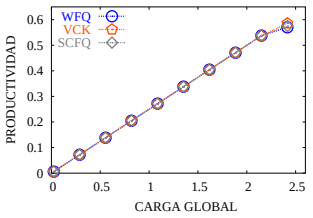
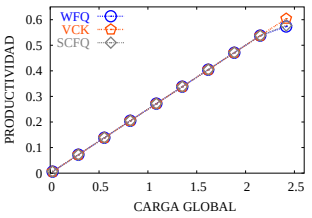
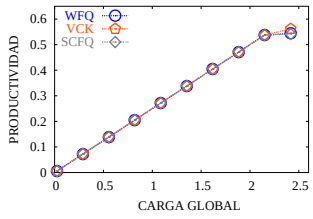
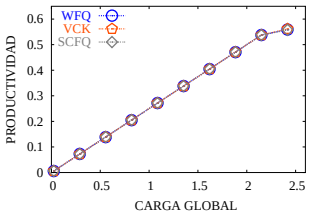
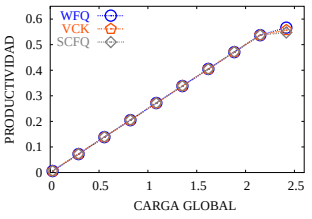
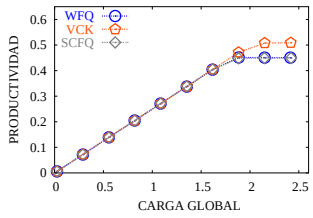
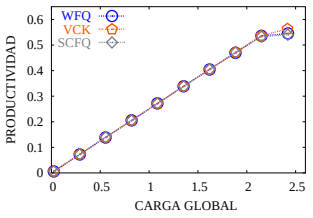
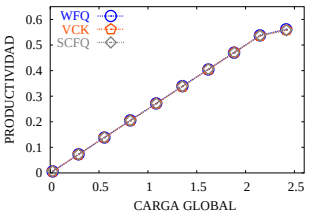
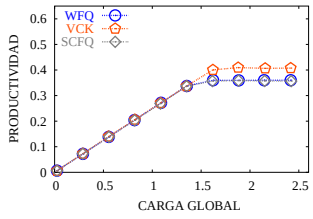
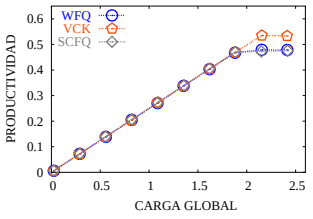
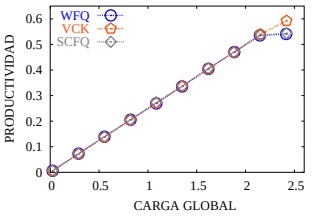
Latencia VC2:

| Tipo de tráfico | Tamaño de buffer = 2 | Tamaño de buffer = 8 | Tamaño de buffer = 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

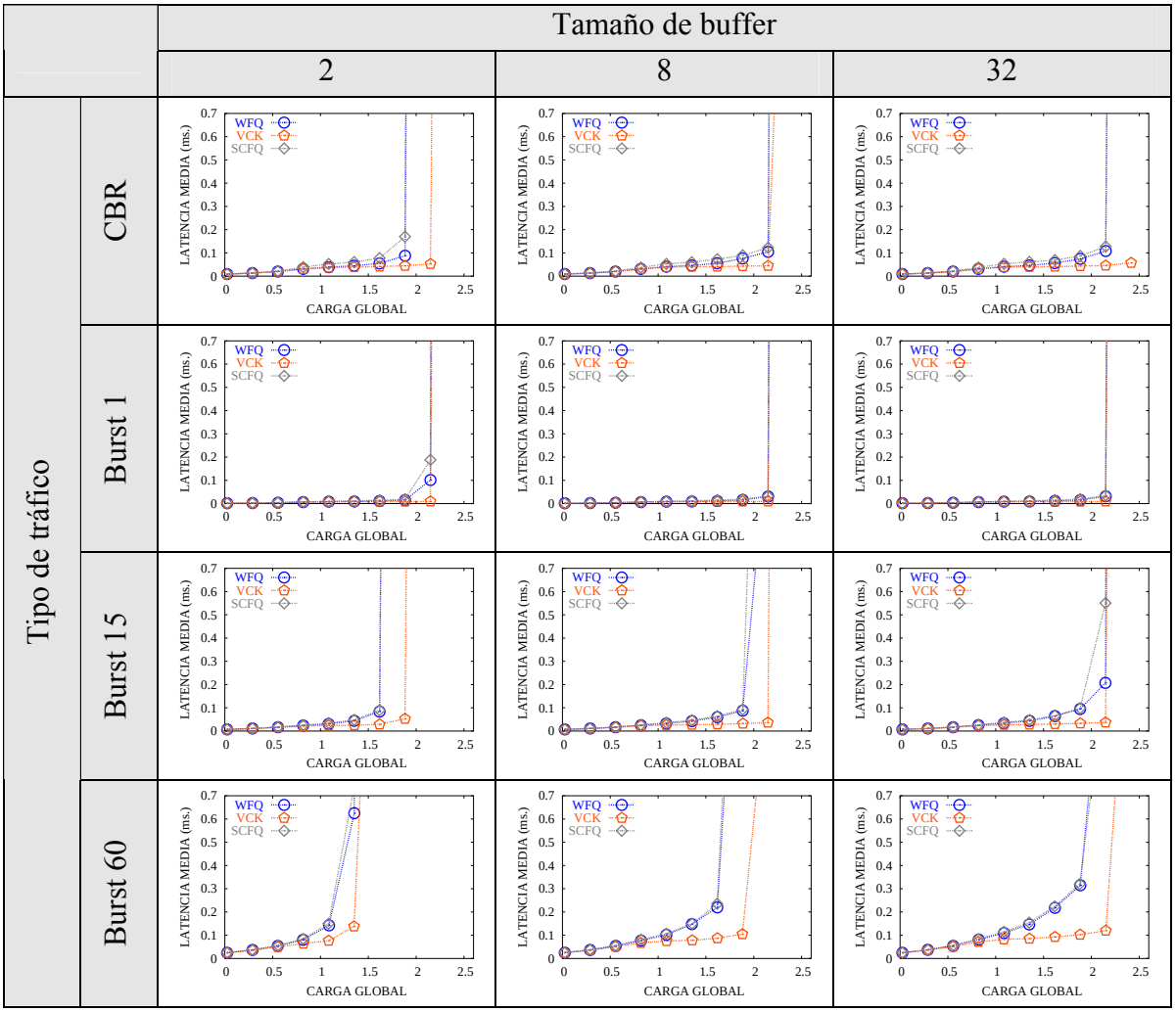
Latencia Máxima VC2:

| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

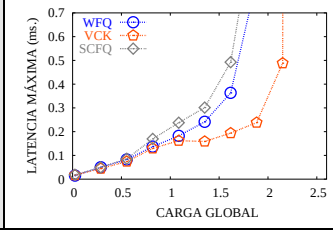
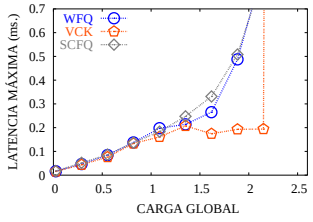
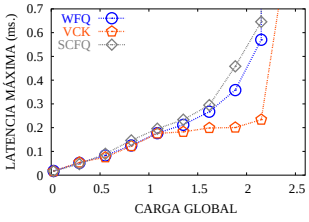
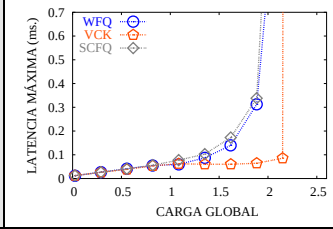
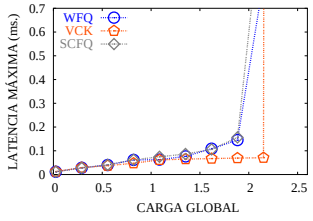
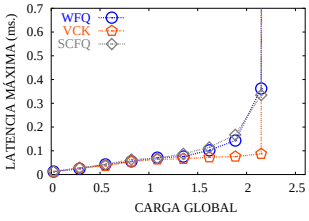
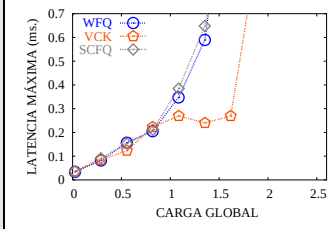
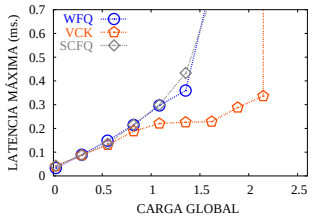
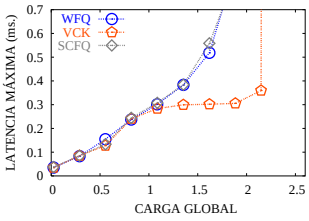
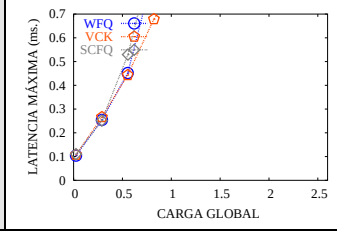
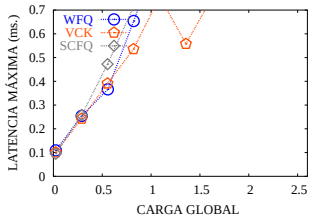
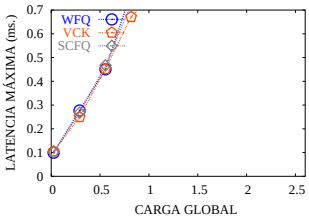
Productividad VC3:

| Tipo de tráfico | Tamaño de buffer: 2 | 8 | 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

Latencia VC3:

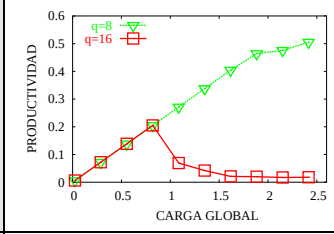
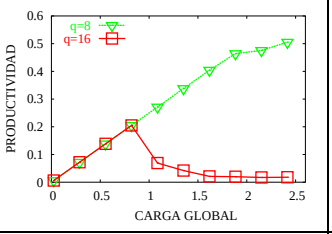
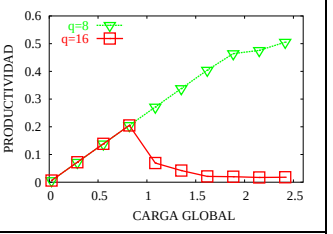
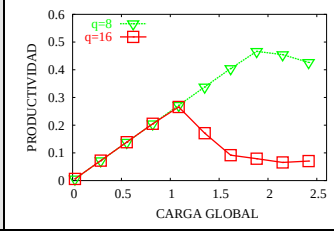
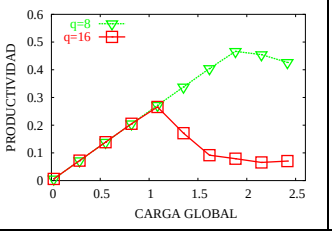
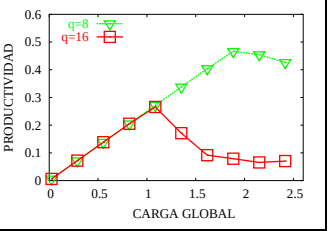
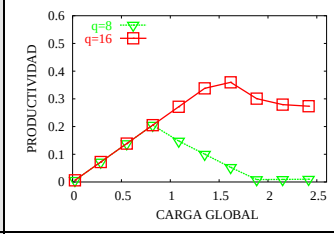
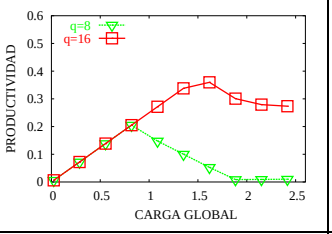
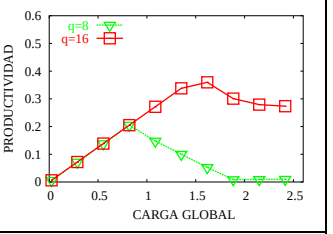
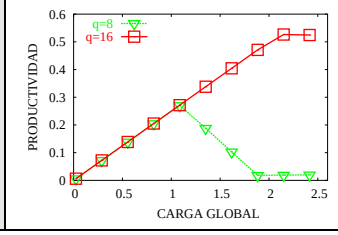
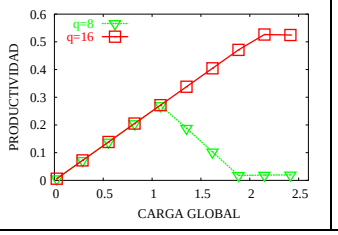
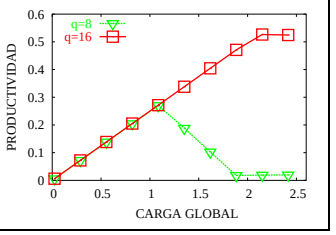


Latencia Máxima VC3:

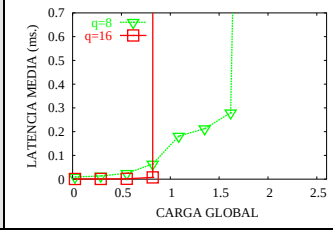
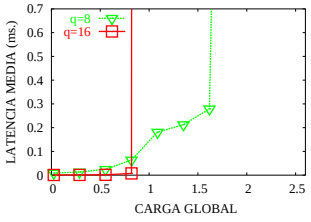
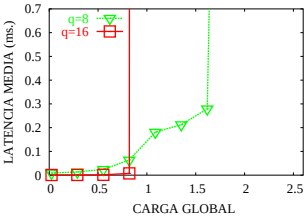
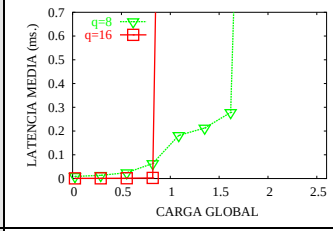
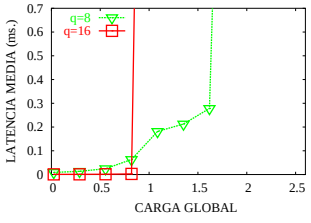
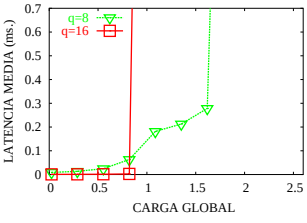
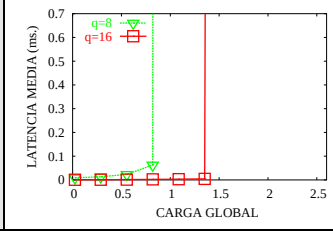
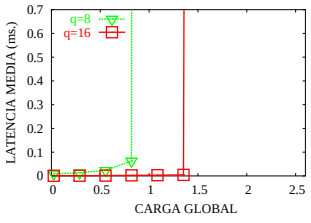
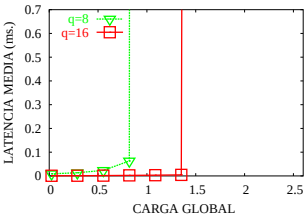
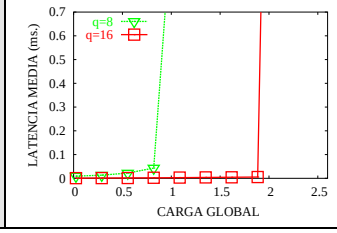
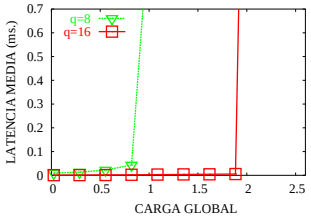
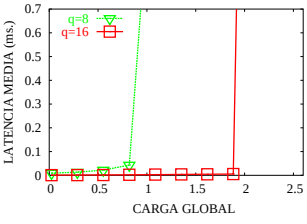
| Tipo de tráfico | Tamaño de buffer = 2 | Tamaño de buffer = 8 | Tamaño de buffer = 32 |
|---|---|---|---|
| CBR |  |  |  |
| Burst 1 |  |  |  |
| Burst 15 |  |  |  |
| Burst 60 |  |  |  |

**Efectos del parámetro q en el algoritmo Aliquem DRR**: En las siguientes tablas se muestra como afecta el parámetro q al algoritmo Aliquem DRR, para estas gráficas se ha utilizado tráfico CBR.

Productividad:

| | | Tamaño de buffer | | |
|---|---|---|---|---|
| | | 2 | 8 | 32 |
| Canal virtual | VC0 |  |  |  |
| | VC1 |  |  |  |
| | VC2 |  |  |  |
| | VC3 |  |  |  |

Latencia:

| Canal virtual | Tamaño de buffer: 2 | Tamaño de buffer: 8 | Tamaño de buffer: 32 |
|---|---|---|---|
| VC0 |  |  |  |
| VC1 |  |  |  |
| VC2 |  |  |  |
| VC3 |  |  |  |