University of Castilla-La Mancha

A publication of the **Department of Computer Science**

Implementing the Advanced Switching Minimum Bandwidth Egress Link Scheduler

by

Raúl Martínez, Francisco J. Alfaro, José L. Sánchez

Technical Report **#DIAB-06-02-02** February 2006

This work has been submitted to the
Workshop on High Performance Switching and Routing (HPSR'2006)

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete - 02071 - Spain
Phone +34.967.599200, Fax +34.967.599224

Contents

Implementing the Advanced Switching Minimum Bandwidth Egress Link Scheduler[1]

Raúl Martínez, Francisco J. Alfaro, José L. Sánchez
Dept. de Sistemas Informáticos
Universidad de Castilla-La Mancha
02071 - Albacete, Spain
{raulmm, falfaro, jsanchez}@info-ab.uclm.es

[1]This work was partly supported by the Spanish CICYT under Grant TIC2003-08154-C06-02, by the Junta de Comunidades de Castilla-La Mancha under Grant PBC-05-005-1, and by the Spanish State Secretariat of Education and Universities under FPU grant.

Abstract

Advanced Switching (AS) is a new fabric-interconnect technology which provides the advanced features of existing proprietary fabrics in an open standard. AS is intended to proliferate in multiprocessor, storage, networking, servers, and embedded platform environments.

The provision of Quality of Service (QoS) in computing and communication environments is currently the focus of much discussion and research in industry and academia. AS provides some mechanisms which correctly used permit us to provide QoS. One of these mechanisms is the Minimum Bandwidth (MinBW) egress link scheduler, which is intended to provide the different traffic classes with a different treatment. In this paper, we propose three possible implementations of the MinBW scheduler and compare their performance by simulation.

1 Introduction

The PCI bus has served industry well for the last 10 years and is currently used extensively. However, the processors and I/O devices of today and tomorrow demand much higher I/O bandwidth than PCI 2.2 or PCI-X can deliver. The reason for this limited bandwidth is the parallel bus implementation. PCI Express [1] eliminates the legacy shared bus-based architecture of PCI and introduces an improved and dedicated point-to-point interconnect. While PCI Express is clearly the interconnect of choice for the computing industry, a common interconnect with the communications industry seems logical and necessary, in order to keep development cost down, performance up and to reduce time-to-market. Advanced Switching (AS) [2] is a new open-standard fabric-interconnect technology for multiprocessor, communications, storage, blade server, and embedded platforms built on the same physical and link layers as PCI Express technology.

Multiservice packet networks are required to carry not only traffic of different applications, such as e-mail or file transfer, which does not require pre-specified service guarantees, but also other applications that require different performance guarantees, like real-time video or telephony. The best-effort service model, though suitable for the first type of applications, is not so for applications of the other type. Therefore, multiservice packet networks need to enable Quality of Service (QoS) provisioning. A key component for networks with QoS support is the output scheduling algorithm, which selects the next packet to be sent and determines when it should be transmitted, on the basis of some expected performance metrics.

Among the mechanisms that AS provides to support QoS, one of them is the Minimum Bandwidth (MinBW) egress link scheduler. However, the AS specification does not offer a particular algorithm to implement this scheduler, but only the properties it must respect. Furthermore, one of the features added by the AS link layer is a credit-based flow control. Flow control protocol ensures that packets are only transmitted when there is enough buffer space at the other end to store them, thereby guaranteeing that no packets are dropped when congestion appears. The problem of most well-known scheduling algorithms is that they were designed without taking into account the existence of a flow control mechanism. The reason is that they were originally proposed for networks that do not have flow control mechanisms at the link layer, for example Internet or ATM. As far as we know, there are no studies that specifically propose scheduling algorithms for flow controlled networks.

In [3], we proposed several methods of using the AS mechanisms to provide applications with QoS based on bandwidth and latency requirements. In this paper, we focus on the implementation of the MinBW scheduler and its interaction with the AS credit-based flow control. We review several well-known scheduling algorithms

(Weighted Fair Queueing (WFQ), Self-Clocked Weighted Fair Queuing (SCFQ), and Deficit Round Robin (DRR)) and propose and compare three new schedulers based on these algorithms. We have called these algorithms: WFQ Credit Aware (WFQ-CA), SCFQ Credit Aware (SCFQ-CA), and DRR Credit Aware (DRR-CA). These new algorithms fulfill all the properties that an AS MinBW scheduler must have, including the interaction with the AS flow control. Moreover, the three credit aware scheduling algorithms are actually appropriate not only for being used in AS, but also for being used in any network that employs a link level flow control.

The structure of the paper is as follows: Section 2 presents the most important AS mechanisms to support QoS. In Section 3, we propose our credit aware scheduling algorithms. In Section 4, we review how to configure the MinBW scheduler to provide the applications with bandwidth and latency requirements. Details on the experimental platform and the performance evaluation are presented in Section 5. Finally, some conclusions are given.

2 AS mechanisms to provide QoS

An AS fabric permits us to employ an admission control mechanism, virtual channels, and egress link scheduling to differentiate between traffic flows. Fabric management software may regulate the access to the AS fabric, allowing new packet flows entry to the fabric only when sufficient resources are available.

AS supports up to 16 unicast VCs and up to 4 multicast VCs. The unicast VC with the highest identifier in each network element is called the Fabric Management Channel (FMC), which is used by the control traffic. Each VC has its own credit count for the credit-based flow control.

AS defines two schedulers to resolve between the up to twenty VCs competing for bandwidth on the egress link: The table scheduler and the MinBW scheduler. A given implementation may choose either of them or may implement its own proprietary mechanism. In any case, when implementing the egress link scheduler, the interaction with the credit-based flow control must be taken into account. Packets from VCs that lack enough credits must not be scheduled. Thus, if the credits for a given VC have been exhausted, the VC scheduler must treat the corresponding queue as if it were empty.

The MinBW egress link scheduler consists of two parts: The first is a mechanism to provide the FMC with absolute priority, ahead of the other VCs, but with its bandwidth limited by a token bucket. The second is a mechanism to distribute bandwidth amongst the rest of the VCs according to a configurable set of weights.

AS does not state a specific algorithm for the MinBW scheduler, but it must respect the following properties [2]:

- Work conserving: If at least one VC has a packet available to be sent over the egress link, it should be transmitted.

- Bandwidth metering, not packet metering: The MinBW scheduler allocates link bandwidth to each VC taking into account packet sizes.

- Minimum bandwidth guarantee: Egress link bandwidth is allocated among the VCs in proportion to a set of configurable weights that represent the fraction of egress link bandwidth assigned to each VC.

- Fair redistribution of unused bandwidth: Bandwidth left over, after all the VCs have consumed their configured minimum bandwidth, must be redistributed among those VCs that have credits and packets to be transmitted in proportion to their bandwidth allocations.

- Memoryless: During the time that a VC has no packets to transmit, or credits to do so, it does not consume bandwidth and the scheduler must not save that VC's minimum bandwidth allocation for future use.

In the next section, we review several scheduling algorithms and propose three new algorithms based on them, which fulfill all these properties.

3 Implementation of the MinBW scheduler

The MinBW properties reviewed in the previous section refer to an ideal fair-queueing model. In a fair-queueing system, supposing a service rate R, N flows, with the i^{th} flow assigned a weight ϕ_i, during a given interval of time, the flow i receives a fair share bandwidth (B_i) proportional to its weight

$$B_i = \frac{\phi_i}{\sum_{j=1}^{V} \phi_j} * R$$

where V is the set of flows ($V \leq N$) with data in queue during that interval of time.

As stated in the AS specification, there are several well-known scheduling algorithms that fit this model in a proper way to be used to implement the MinBW algorithm. The problem of these algorithms is that they were designed for networks without a flow control mechanism. Therefore, one of the main issues when implementing the MinBW scheduler is its interaction with the AS credit-based flow control. A given implementation of a scheduler is not allowed to select packets from a VC lacking transmission credits, nor it is allowed to 'save' this bandwidth for future use.

In this section, we present three new scheduling algorithms based on well-known algorithms (WFQ, SCFQ, and DRR) that take into account the AS credit-based flow

control and fulfill all the properties that the AS MinBW scheduler must have and, therefore, can be implemented in this new technology.

3.1 Weighted Fair Queuing Credit Aware (WFQ-CA)

The WFQ algorithm [4] is an approximation of the Generalized Processor Sharing (GPS) model [5]. GPS is a fair-queueing model based on a fluid model that provides perfect instant fairness in bandwidth allocation. This ideal model assumes that several packets from different queues can be simultaneously transmitted. WFQ is a packet-by-packet algorithm that tries to emulate the GPS model by stamping each packet that arrives at the egress link with its departure time (*virtual finishing time*) in a corresponding GPS system. The packets are then transmitted in an increasing order of timestamp.

Let F_i^k be the virtual finishing time of the k^{th} packet from flow i,

$$F_i^k = max\{F_i^{k-1}, V(t)\} + \frac{L_i^k}{\phi_i}$$

where L_i^k is the length of the k^{th} packet and $V(t)$ is the *virtual time* of the WFQ system. The WFQ algorithm tracks the set of queues which are active in each instant and the real time of the system to calculate $V(t)$.

The WFQ-CA algorithm that we propose works in the same way as the WFQ algorithm, except in the following aspects:

- When a new packet arrives at a queue, it is stamped with its *virtual finishing time* if there are enough credits to transmit the packet that is at the head of the queue.
- Packets are transmitted in an increasing order of timestamp, but only those queues with enough credits to transmit the packet at their heads are taken into account.
- When a queue is inactive because of lack of credits and receives enough credits to be able to transmit again, its packets are restamped, from the head to the tail, as if they had arrived in that instant.

Another problem that the WFQ algorithm faces as an implementation of the MinBW scheduler comes from using the real time to calculate the virtual time. Note that the real time includes the time used to transmit control packets, which are out of the control of the WFQ algorithm. The WFQ-CA algorithm fits this problem by not taking into account the time employed in sending control packets for calculating the virtual time. Figure 1 shows an example with 7 events occurring in the system and two "gaps" (shadowed boxes) in the time line. Note that an event happens whenever

a queue changes its active status or a control message begins or ends transmission. A queue can change from active to inactive, or viceversa, when a packet arrives at an empty queue, when a packet is transmitted, or when a credit flow control message is received.

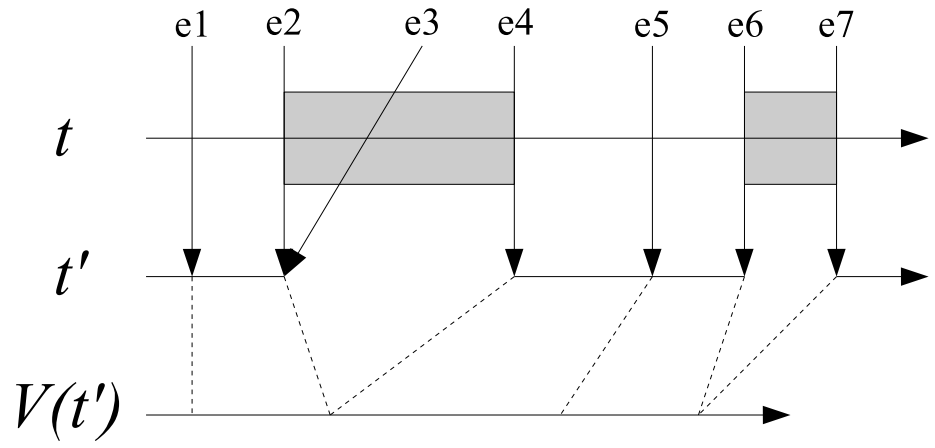

Figure 1: Time line in the WFQCA implementation of the MinBW scheduler.

In Figure 1, the t line represents the real time of the system, t' represents the time line that is actually being used to calculate $V(t)$ and when the events are considered to happen. Note that the events that happen during a gap time are considered to happen at the beginning of that gap.

3.2 Self-Clocked Weighted Fair Queuing Credit Aware (SCFQ-CA)

The SCFQ algorithm [6] defines fair queueing in a self-contained manner and avoids using a hypothetical queueing system as reference to determine the fair order of services. This objective is accomplished by adopting a different notion of virtual time. Instead of linking virtual time to the work progress in the GPS system, it uses a virtual time function which depends on the progress of the work in the actual packet-based queueing system. This approach offers the advantage of removing the computation complexity associated to the evaluation of $V(t)$ that may make WFQ unfeasible in high-speed interconnection technologies.

Therefore, when a packet arrives, SCFQ uses the service tag (finish time in WFQ) of the packet currently in service as the $V(t)$ to calculate the new packet tag. Thus, in this case the virtual finishing time is computed as

$$F_i^k = max\{F_i^{k-1}, F_{current}\} + \frac{L_i^k}{\phi_i}$$

The SCFQ-CA algorithm that we propose works in the following way:

- When a new packet arrives at a queue, it is stamped with its service tag only if it is at the head of the queue and there are enough credits to transmit it.
- Packets are transmitted in increasing order of service tag.
- When a packet is transmitted, if there are enough credits to transmit the next packet, this packet is stamped with its service tag.
- When a queue is inactive because of lack of credits and receives enough credits to transmit again, the packet at the head of the queue is stamped with its service tag.

Note that $F_{current} \leq F_i^{k-1}$ if there is at least one packet waiting, or being transmitted, in the queue i. This permits us to wait to stamp a packet until it reaches the queue head, avoiding the restamping process of the WFQ-CA algorithm.

3.3 Deficit Round Robin Credit Aware (DRR-CA)

The DRR algorithm [7] is a variation of the Weighted Round Robin (WRR) algorithm [8] that works properly with variable packet sizes. In the WRR algorithm, a list of queue weights is visited sequentially, each weight indicating the number of packets from the queue in question that can be transmitted. DRR associates each queue with a deficit counter, which is set to 0 at the start. The scheduler visits and serves a fixed amount of data (referred to as *quantum*) from each queue. When a packet is transmitted, the quantum is reduced by the packet size. For each queue, the scheduler transmits as many packets as the quantum allows. The unused quantum is saved in the deficit counter, representing the amount of quantum that the scheduler owes the queue. At the next round, the scheduler will add the previously saved quantum to the current quantum. When the queue has no packets to transmit, the quantum is discarded, since the flow has wasted its opportunity to transmit packets.

The DRR-CA algorithm that we propose works in the same way as the DRR algorithm, except in the following aspects:

- A queue can be selected for transmitting (is considered active) only if it has at least one packet to transmit and if there are enough credits to transmit the packet at the head of the queue.
- When a packet is transmitted, the next packet in its queue is also transmitted if there are enough quantum and credits. If there are not enough credits, the deficit counter is set to 0 and the scheduler chooses the next active queue for transmitting.

3.4 Implementation considerations

To choose a given MinBW implementation we need to consider not only the latency and fairness properties, but also the computational complexity of the different algorithms. Sorted-priority algorithms, like WFQ and SCFQ (also WFQ-CA and SCFQ-CA), require processing at line speeds for tag computation and tag sorting. Even the SCFQ (or SCFQ-CA) algorithm, which has a lower computation tag complexity, has to sort the VC tags (O(log N), where N is the number of VCs). Note that the WFQ-CA algorithm has the additional complexity of the restamping process. According to the implementation proposed in [7], DRR exhibits $O(1)$ complexity provided that each flow is allocated a quantum no smaller than its maximum packet size. On the other hand, variants of WRR, like DRR and DRR-CA, have lower worst-case fairness and latency tuning characteristics compared to sorted-priority algorithms and the SCFQ (or SCFQ-CA) algorithm offers worse characteristics than WFQ (or WFQ-CA) [9].

4 Providing QoS with the MinBW scheduler

To provide QoS requirements in AS, a set of Service Classes (SCs) with different requirements must be specified [3]. The egress link scheduler, in this paper the MinBW, must be properly configured to provide the different SCs with their requirements. Moreover, an admission control protocol must be used to provide QoS guarantees.

To define this set of SCs, we propose a traffic classification based on two network parameters: Bandwidth and latency. We distinguish three broad categories of traffic:

- Network Control traffic: High-priority traffic to maintain and support the network infrastructure.
- QoS traffic: Traffic that has explicit minimum bandwidth and/or maximum latency requirements.
- Best-effort traffic: Traffic largely insensitive to both bandwidth and latency and only characterized by the differing priority among each other.

When various flows obtain access to the AS fabric, they will be aggregated into the SCs depending on their characteristics. If there are sufficient VCs, we will devote a separate VC to each existing SC. The control SC will be assigned to the FMC in order to achieve the maximum priority.

Providing minimum bandwidth requirements to a VC with the MinBW scheduler is as easy as assigning to that VC a weight equal to the proportion of the egress link bandwidth that it needs. Parekh and Gallager [5] analyzed the performance of WFQ from the standpoint of worst-case packet delay. On the basis of that study, we

assign a higher amount of bandwidth than is needed to those VCs with high latency requirements, in order to obtain a better average and maximum latency performance.

To distribute the link bandwidth between the VCs, several things must be taken into account. First of all, it is well-known that interconnection networks are unable to achieve a 100% global throughput. Moreover, a certain amount of bandwidth must be reserved to the control SC according to its expected traffic, which has strict priority in the MinBW scheduler. Therefore, not all the bandwidth can be distributed among the QoS and best-effort SCs, thereby requiring a certain bandwidth to be left unassigned. Secondly, QoS traffic may be bursty (for example a video transmission) and may require, during short periods of time, more bandwidth than average. Therefore, when configuring the MinBW scheduler, not all the bandwidth that is intended to be assigned to best-effort SCs will in fact be assigned to them, but rather only a small amount of bandwidth proportional to their relative priority. The rest of the best-effort bandwidth will also be added to this unassigned traffic. Note that the bandwidth unused by the control and QoS SCs would be redistributed by the MinBW scheduler among the best-effort SCs.

5 Performance Evaluation

In this section, we evaluate the behavior of the three MinBW implementations proposed. For this purpose, we have developed a detailed simulator that allows us to model the network at the register transfer level, following the AS specification. First, we will describe the main AS network model features. Secondly, the traffic model and the load used are described. Thirdly, the configuration of the egress link schedulers is specified. Finally, we present and analyze the results obtained.

5.1 Simulated architecture

We have used a perfect-shuffle multi-stage interconnection network with 64 end-points. In AS, any topology is possible, but we have used this topology because it is a common solution for interconnection in current high-performance environments. The switches have 8 ports and use a combined input-output buffer architecture, with a crossbar to connect the buffers. Virtual output queueing has been implemented to solve the head-of-line blocking problem at switch level, although all the queues of a VC share the same credit count.

In our tests, the link bandwidth is 2.5 Gb/s but, with the 8b/10b encoding scheme, the maximum effective bandwidth for data traffic is only 2 Gb/s. We are assuming some internal speed-up (x1.5) for the crossbar, as is usually the case in most commercial switches. AS gives us the freedom to use any algorithm to schedule the crossbar, and

we have implemented a Round Robin scheduler. The cut-through latency is 145 ns, which is based on the AS StarGen's *Merlin* switch [10].

5.2 Traffic model

The IEEE standard 802.1D-2004 [11] defines 7 traffic types at the Annex G, which are particularly appropriate for this study. We will consider each traffic type as an AS SC. Table 1 shows each SC and its requirements. In this way, the workload is composed of 7 SCs and each one will be assigned to a different VC, the NC SC being assigned to the FMC.

Table 1: SCs recommended by the standard IEEE 802.1D-2004.

SC	**Description**
NC: Network control	Control
VO: Voice	QoS: Bandwidth and latency requirements
VI: Video	QoS: Bandwidth and latency requirements
CL: Controlled load	QoS: Bandwidth requirements
EE: Excellent-effort	Best-effort: Most preferent
BE: Best-effort	Best-effort: Intermediate
BK: Background	Best-effort: Least preferent

Our intention is to evaluate the behavior of the three credit aware algorithms we have proposed, using an admission control mechanism for controlling the QoS traffic and a relatively small amount of control traffic (as is usually the case). The QoS SCs should meet their requirements, whatever the load of best-effort traffic. For that purpose, we constantly inject a fixed amount of control traffic (NC) and QoS traffic (VO, VI, and CL) all the time, and we start to inject best-effort traffic (EE, BE, and BK) at 0.7 normalized network input load, gradually increasing the amount. The amount of QoS traffic to be injected is the maximum allowed by the admission control, which is a simple one, based on average bandwidth. Table 2 shows the proportion of traffic of each SC that each node injects regarding the link bandwidth (2 Gb/s).

The packets are generated according to different distributions, as can be seen in Table 2. VO, VI, and CL SCs are composed of point-to-point connections of the given bandwidth. In the case of the VI SC, the frames of the traces are split into packets and transmitted with an equal distribution through the video frame time (40 ms). The self-similar traffic is bursty traffic generated with on/off sources, governed by two Pareto distributions, as recommended by Jain [12]. The packet sizes that we have used are: Up to 64 bytes for NC traffic, 128 bytes for VO traffic, and up to 2176 bytes (the maximum packet size in AS) for the rest of SCs.

Table 2: Injected traffic and scheduler configuration.

	Injected traffic		MinBW Conf.
SC	**Bandw. %**	**Traffic pattern**	**Weight**
NC	1	self-similar	-
VO	20.3125	64KB/s CBR connections	0.265625
VI	20.3125	750 KB/s MPEG-4 traces	0.203125
CL	20.3125	750 KB/s CBR connections	0.203125
EE	0 - 25.4	self-similar	0.09375
BE	0 - 25.4	self-similar	0.03125
BK	0 - 25.4	self-similar	0.015625
	61.9 - 138.1		0.8125

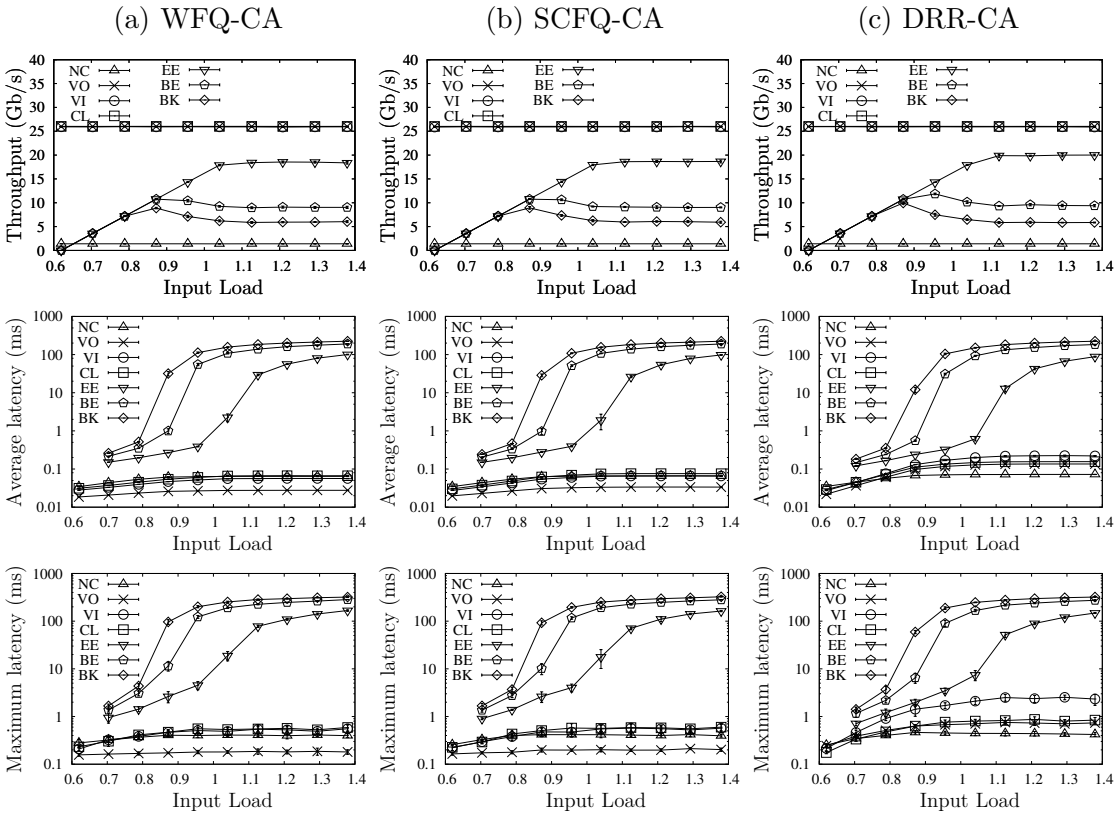

Figure 2: Performance of the three scheduler implementations of the MinBW.

5.3 Scheduler configuration

Table 2 shows the scheduler configuration. We want to reserve 20% of link bandwidth to best-effort traffic, but we have only assigned best-effort SCs a minimum bandwidth (14.0625%) to establish the preference between them. Thus, we have left 18.75% of bandwidth unassigned (rest of best-effort bandwidth + expected amount of control traffic + expected amount of lost network bandwidth). The remaining bandwith has been distributed between the QoS SCs. We will inject the same amount of traffic of

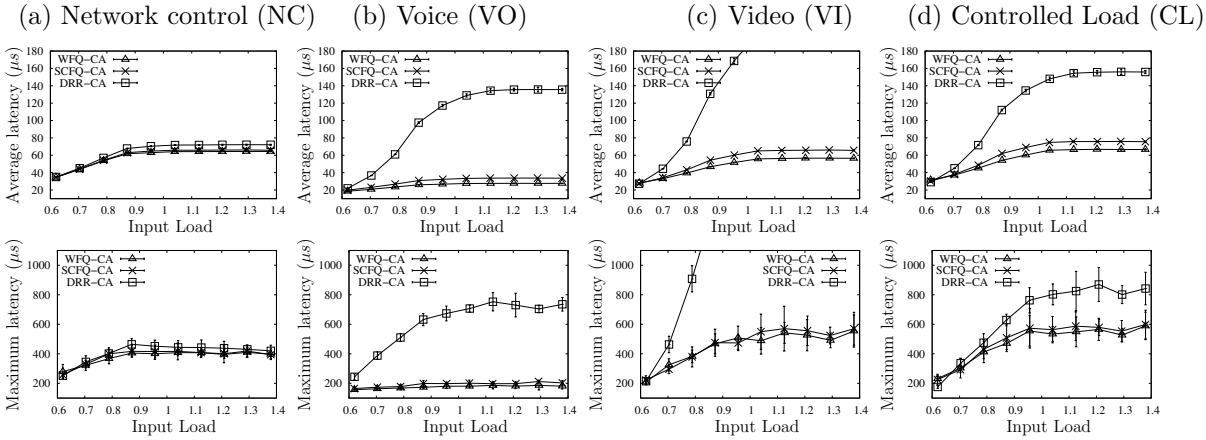

Figure 3: Latency results of the control and QoS SCs.

the three QoS SCs considered, but we have assigned a 33% weight more to VO SC due to its higher latency requirements [5].

In the case of the DRR-CA implementation of the MinBW scheduler, the VC that accommodates the BK SC is assigned a quantum that corresponds to 34 credits (the packet of maximum size), which ensures that at least one packet is going to be transmitted when a given VC is selected. The rest of VCs are assigned a proportional quantum.

5.4 Simulation results

Figure 2 shows the performance of our three scheduler implementations of the MinBW. Figure 3 shows a more detailed comparison between the schedulers based on the average and maximum latency of the control and QoS SCs. Figures show the average values and the confidence intervals at a 90% confidence level of ten different simulations performed at a given input load. For each simulation we obtain the average throughput, the average packet latency, and the maximum packet latency of each flow. We obtain statistics per SC aggregating the throughput of all the flows of the same SC, obtaining the average value of the average latency, and the maximum latency of all the flows. Note that the maximum latency shows the behavior of the flow with the worst performance.

Figure 2 shows some general results for the three schedulers. The NC and the QoS SCs obtain all the bandwidth that they inject. However, when the network load is high (around 80%), the best-effort SCs do not yield a corresponding result. These SCs obtain a bandwidth proportional to their priority.

The average and maximum latency of the NC SC and the QoS SCs grow with the load until they reach a certain value. Once this value is reached the latency remains more or less constant. The average latency of best-effort SCs grows with the load. Furthermore, it can be seen that best-effort SCs obtain different average latency according to their different priority.

Figure 3 offers a clearer picture of the difference between the three schedulers for the control and QoS SCs. As we can see, the best latency results are offered by the WFQ-CA algorithm. The SCFQ-CA algorithm offers slightly worse results than the WFQ-CA proposal. The DRR-CA algorithm offers clearly the worst latency results of the three schedulers. Note that the latency performance of the three schedulers is inverse to their complexity (see Section 3.4). Another difference between the schedulers is that the DRR-CA algorithm is affected negatively for the variable bit rate of video traffic (the VI SC obtains a worse latency than the CL SC having the same amount of assigned bandwidth). This is not the case for WFQ-CA and SCFQ-CA proposals. Note also that the control traffic obtains a worse latency than, for example, the voice traffic because control traffic must be emulated using self-similar traffic, which is more difficult to handle than the CBR traffic used for the voice traffic. Finally, the VO SC obtains a better latency than the VI and CL SCs because we have assigned it more bandwidth than it really requires to fulfill its bandwidth requirements.

Summing up, the three proposed algorithms are able to provide control and QoS SCs with the required throughput, and to provide best-effort SCs with a throughput proportional to their priority. The algorithms with the highest computational complexity, provide the best latency results. The DRR-CA algorithm offers the worst latency results, although it presents the lowest computational complexity. However, the SCFQ-CA algorithm, which is not as complex as the WFQ-CA algorithm, shows a very good latency performance.

6 Conclusions

In this paper, we have proposed and compared three new scheduler implementations based on three well-known scheduling algorithms. These new algorithms accomplish all the properties that the AS MinBW scheduler must have, including the interaction with the AS flow control. We have called these algorithms: WFQ Credit Aware (WFQ-CA), SCFQ Credit Aware (SCFQ-CA), and DRR Credit Aware (DRR-CA). These algorithms can be used not only to implement the egress link scheduler for the AS technology, but also in any network technology with flow control.

Simulation results show that the three schedulers provide a similar throughput performance. The WFQ-CA algorithm, which has the highest computational complexity, provides the best latency results. However, the SCFQ-CA algorithm shows a slightly worse latency performance than the WFQ-CA algorithm with a lower computational complexity. The DRR-CA proposal is the algorithm with the lowest computational complexity, but offers the worst latency results, and has very bad latency tuning characteristics. This is because the latency depends on the increment size in which the arbiter is configured.

Bibliography

[1] *PCI Express base architecture specification. Revision 1.0a*, PCI SIG, Apr. 2003.

[2] *Advanced Switching core architecture specification. Revision 1.1*, Advanced Switching Interconnect Special Interest Group, Mar. 2005.

[3] R. Martínez, F. Alfaro, J. Sánchez, and T. Skeie, "A first approach to provide QoS in Advanced Switching," *In the poster session of the International Conference on High Performance Computing (HiPC). Goa, India*, 2005, http://www.hipc.org/hipc2005/hipc2005posters.html.

[4] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulations of a fair queuing algorithm," in *SIGCOMM*, 1989.

[5] A. K. Parekh and R. G. Gallagher, "A generalized processor sharing approach to flow control in integrated services networks: The multiple node case," *IEEE/ACM Transactions on Networking*, 1994.

[6] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications." in *INFOCOM*, 1994.

[7] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," in *SIGCOMM*, 1995, pp. 231–242.

[8] M. Katevenis, S. Sidiropoulos, and C. Corcoubetis, "Weighted round-robin cell multiplexing in a general-purpose ATM switch chip," *IEEE Journal on Selected Areas in Communications*, Oct. 1991.

[9] D. Stiliadis and A. Varma, "Latency-rate servers: a general model for analysis of traffic scheduling algorithms," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, pp. 611–624, 1998.

[10] StarGen, *StarGen's Merlin switch*, 2004, http://www.stargen.com/products/merlin_switch.shtml.

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

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