Implementing the Advanced Switching Minimum Bandwidth Egress Link Scheduler*

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

Abstract

Advanced Switching (AS) is a new fabric-interconnect technology that further enhances the capabilities of PCI Express, which is the next PCI generation. On the other hand, 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.

One of the mechanisms that AS provides to support QoS is the minimum bandwidth egress link scheduler, or just MinBW scheduler. In this paper, we propose several implementations of the MinBW scheduler and compare their performance by simulation. These implementations fulfill all the properties that an AS MinBW scheduler must have, including the interaction with the AS link layer flow control.

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 [13] eliminates the legacy shared bus-based architecture of PCI and introduces an improved and dedicated point-to-point interconnect. The primary strength behind PCI Express is in its support for legacy PCI while addressing its inadequacies.

The need for Advanced Switching (AS) [1] essentially comes because computing and communication platforms begin to converge by exhibiting increasing overlap in terms of the functions they serve. 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. AS is an extrapolation of PCI Express, borrowing its lower two architectural layers from PCI Express, but diverging at the transaction layer and in the marketplaces it intends to serve. Whereas PCI Express has already begun to reshape a new generation of PCs and traditional servers, AS is intended to proliferate in multiprocessor, peer to peer systems in the communications, storage, networking, servers and embedded platform environments. Together, PCI Express and AS have the potential for building the next generation interconnects [9].

The provision of Quality of Service (QoS) in the environment where AS is foreseen to be used will be very important. AS networks will be required to carry not only traffic of applications, such as e-mail or file transfer, which does not require pre-specified service guarantees, but also traffic of other applications that requires different performance guarantees, like real-time video or telecommunications [10]. The best-effort service model, though suitable for the first type of applications, is not so for applications of the other type [12]. Even in the same application, different kinds of traffic (e.g. I/O requests, coherence control messages, synchronization and communication messages, etc.) can be considered, and it would be very interesting that they were treated according to their priority.

A key component for the support of QoS in any network 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. AS defines two egress link schedulers: The virtual channel arbitration table scheduler and the Minimum Bandwidth egress link scheduler (MinBW). 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 algo-

*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.

rithms 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 link layer flow control, like Internet or ATM.

In [8], we showed how to use the AS mechanisms to provide applications with QoS. Moreover, we compared the performance of our proposals using both the table and the MinBW schedulers. In order to do this we proposed a modification to the table scheduler that works with variable packet sizes and presented an implementation of the MinBW scheduler based on the Self-Clocked Weighted Fair Queuing (SCFQ) algorithm [3]. We called this algorithm SCFQ Credit Aware (SCFQ-CA).

In this paper, we focus on the implementation of the MinBW scheduler. We review the SCFQ-CA algorithm and propose two other implementations for the MinBW scheduler: the Weighted Fair Queuing Credit Aware (WFQ-CA) and the Deficit Round Robin Credit Aware (DRR-CA) algorithms. These two last algorithms are based on two well-known schedulers: the WFQ [2] and the DRR [14] algorithms, respectively. The SCFQ-CA, the WFQ-CA, and the DRR-CA schedulers fulfill all the properties that an AS MinBW scheduler must have, including the interaction with the AS flow control. We will see that in the SCFQ and DRR cases the adaptation of these well-known scheduling algorithms to the MinBW scheduler is more or less simple. However, in the case of the WFQ, the solution is not trivial. In the performance evaluation section we compare the performance of the three posibilities.

Proposing adapted scheduling algorithms for AS is quite significant if we take into account that PCI Express and AS are foreseen to be the *de facto* standard in a lot of interconnection environments. As far as we know, nobody has proposed previously a specific implementation for this key component of AS. Moreover, the three credit aware scheduling algorithms that we propose 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. To the best of our knowledge, the important issue of adapting well-known scheduling algorithms to environments that employ a link level flow control mechanism has not yet been treated.

The structure of the paper is as follows: Section 2 reviews 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 and future work is proposed.

2. AS mechanisms to provide QoS

AS permits us to employ Virtual Channels (VCs) and egress link scheduling to differentiate between traffic flows. Moreover, 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 implemented unicast VC with the highest identifier is called the Fabric Management Channel (FMC).

AS defines two schedulers to resolve between the up to 16 unicast 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 [1]:

- Work conserving: If at least one VC has a packet available to be sent, it should be transmitted.
- 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.
- Bandwidth metering, not packet metering: The MinBW scheduler allocates link bandwidth to each VC taking into account packet sizes.
- Fair redistribution of unused bandwidth: Bandwidth left over, after all the VCs have consumed their configured 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 having 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.

The AS specification states that variants of WFQ such as SCFQ, and variants of Weighted Round Robin (WRR) [7] such as DRR exhibit the desired properties of the MinBW scheduler. The problem with 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 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. These new algorithms are based on the WFQ, SCFQ, and DRR scheduling algorithms.

3.1. Weighted Fair Queuing Credit Aware

The WFQ algorithm [2] is an approximation of the Generalized Processor Sharing (GPS) model [11]. GPS is an ideal 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 considered.
- 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 aspect that must be taken into account is that the WFQ algorithm uses 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. However, this is not a trivial task because events still may happen during that time. An event is anything that changes the scheduler state, namely the arrival or departure of a packet, or the arrival of a credit flow control message that changes a queue from inactive to active.

Figure 1 shows an example of how the $V(t)$ is calculated. The figure shows 7 events occurring in the system and two "gaps" (shadowed boxes) in the time line due to the transmission of control packets. The t line represents the real time of the system. The t' line represents the time 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.

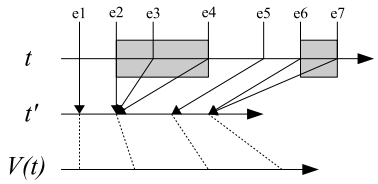

Figure 1. Time line in the WFQ-CA implementation of the MinBW scheduler.

3.2. Self-Clocked Weighted Fair Queuing Credit Aware

The SCFQ algorithm [3] 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, and thus simplifying the scheduling process.

3.3. Deficit Round Robin Credit Aware

The DRR algorithm [14] is a variation of the Weighted Round Robin (WRR) algorithm [7]. In the WRR, a list of flow weights is visited sequentially, each weight indicating the number of packets from the flow in question that can be transmitted. The WRR algorithm faces a problem if the average packet size of the different flows is different. In that case, the bandwidth that the flows obtain may not be proportional to the assigned weights. Therefore, the WRR algorithm does not work properly with variable packet sizes. However, today network technologies usually use variable packet sizes.

In order to handle properly variable packet sizes the DRR algorithm associates each queue with a deficit counter, which is set to zero at the start. The scheduler visits and serves a fixed amount of data (referred to as *quantum*) from each flow. When a packet is transmitted, the flow's quantum is reduced by the packet size. For each flow, the scheduler transmits as many packets as its quantum allows. The unused quantum is saved in the deficit counter, representing the amount of quantum that the scheduler owes the flow. 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 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 active queue is selected when any of the following conditions occurs:
  - There are no more packets from the current queue or there are not enough flow control credits for transmitting the packet that is at the head of the queue. In this case, the current queue becomes inactive, and its deficit counter becomes zero.
  - The remaining quantum is less than the size of the packet at the head of the current queue. In this case, its deficit counter becomes equal to the accumulated weight in that instant.

A well-known problem of the WRR and DRR algorithms, which is shared by the DRR-CA algorithm, is that the latency and fairness depend on the frame length. The frame length in these algorithms is defined as the sum of all the weights in the WRR algorithm or the quantums in the DRR algorithm. The longer the frame is, the higher the latency and the worse the fairness. In order for DRR to exhibit lower latency and better fairness, the frame length should therefore be kept as small as possible. Unfortunately, given a set of flows, it is not possible to select the frame length arbitrarily. According to the implementation proposed in [14], DRR exhibits $O(1)$ complexity provided that each flow is allocated a quantum no smaller than the Maximum Transfer Unit (MTU). As observed in [6], removing this hypothesis would entail operating at a complexity which can be as large as $O(N)$. This restriction affects not only the weight assigned to the smallest flow, but to the rest of the flows in order to keep the proportions between them.

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. As stated before, the DRR algorithm exhibits O(1) complexity. However, variants of WRR, like DRR and DRR-CA, have lower worst-case fairness and latency tuning characteristics compared to sorted-priority algorithms [16].

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 [8]. 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.

In order 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 SC. The control SC will be assigned to the FMC in order to achieve the maximum priority.

Providing the traffic of VC with minimum bandwidth requirements using 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 [11] 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 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 unused bandwidth would be redistributed by the MinBW scheduler among the best-effort SCs.

5. Performance Evaluation

In [8], we evaluated our proposals for providing QoS over AS comparing the performance of the table scheduler and the MinBW scheduler using the SCFQ-CA algorithm. In this paper we explore and evaluate different alternatives of the MinBW scheduler. 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 of the switch is 145 ns, which is based on the AS StarGen's *Merlin* switch [15].

5.2. Traffic model

The IEEE standard 802.1D-2004 [4] defines 7 traffic types at the Annex G that can be easily adjusted to our traffic classification. 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 req.
VI: Video	QoS: Bandwidth and latency req.
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. Table 2 shows the percentage of traffic of each SC that each node injects regarding the link bandwidth.

Table 2. Injected traffic and scheduler configuration.

	Injected traffic		MinBW C.
SC	Bandw. %	Traffic pattern	Weight
NC	1	self-similar	-
VO	20.3125	64KB/s CBR connect.	0.265625
VI	20.3125	750 KB/s MPEG-4 traces	0.203125
CL	20.3125	750 KB/s CBR connect.	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
Total	61.9 - 138.1		0.8125

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 [5]. 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.

Note that the traffic model that we use is this performance evaluation is based on a multimedia environment. AS is intended to be used in very different kind of environments, and probably in some of them the multimedia traffic is not the most suitable one. However, we use a wide range of traffic behaviors, and thus the results obtained with this kind of traffic can be generalized to other AS environments with other kind of traffic with QoS requirements.

5.3. Scheduler configuration

Table 2 also 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 the three QoS SCs considered, but we have assigned a 33% weight more to VO SC due to its higher latency requirements [11].

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

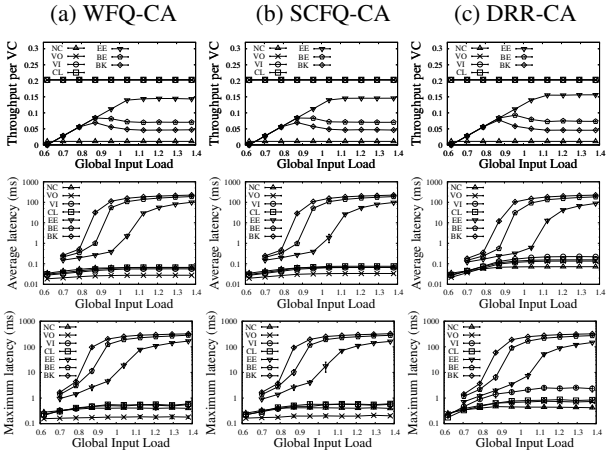

Figure 2. Performance of the three scheduler implementations of the MinBW.

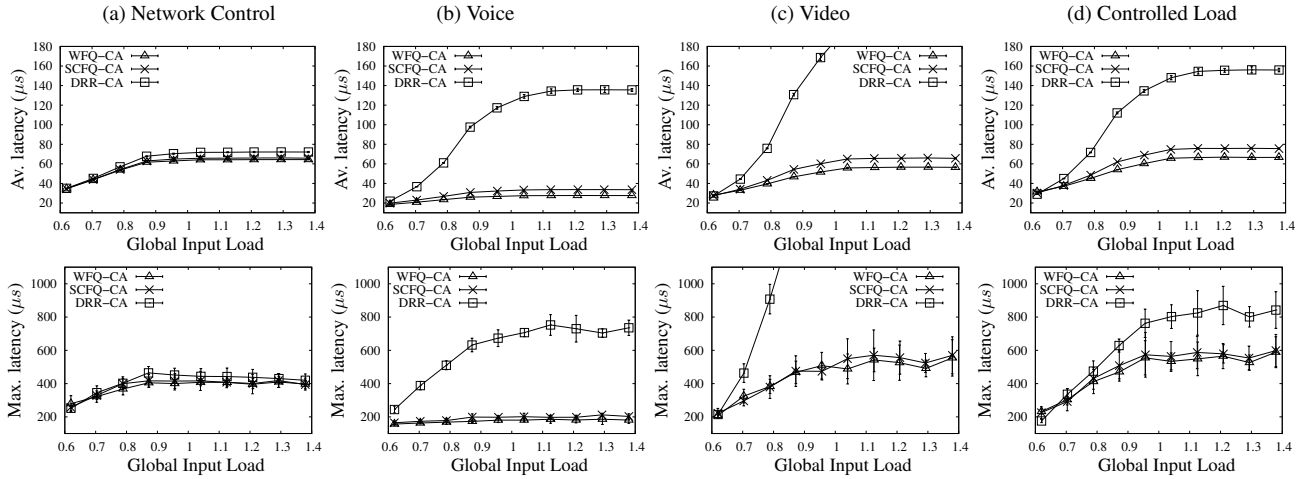

Figure 3. Average and maximum latency performance of the control and QoS SCs.

5.4. Simulation results

Figure 2 shows the performance of our three scheduler implementations of the MinBW. Figures 3 and 4 show 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 90% confidence level of ten different simulations performed at a given input load. For each simulation we obtain the normalized 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 the normalized throughput results per VC. It can be seen that the NC and the QoS SCs obtain all the bandwidth they inject. However, when the network load is high (around 85%), the best-effort SCs do not yield a corresponding result. These SCs obtain a bandwidth proportional to their priority.

Regarding the latency performance, Figure 2 shows that 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 a different average latency according to their priority.

Figure 3 offers a clearer picture of the difference between the three schedulers for the control and QoS SCs, which are the SCs with stricter QoS requirements. As we can see, the best latency results are offered by the WFQ-CA algorithm. 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 assigned the same amount bandwidth). This is not the case for the 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, in order to fulfill its latency requirements, we have assigned it more bandwidth than it strictly requires.

As stated before, the DRR-CA algorithm provides latency performance far worse than the WFQ-CA and SCFQ-CA algorithms. However, the difference between these last two schedulers is no so big. Figure 4 shows the percentage of improvement on average and maximum latency of the WFQ-CA algorithm over the SCFQ-CA algorithm for the control and QoS SCs. As we can see the maximum improvement of the WFQ-CA algorithm is less than 20% for the average, and less than 15% for the maximum latency.

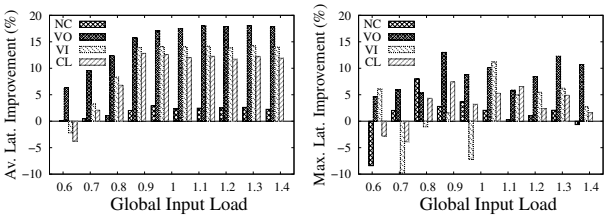

Figure 4. Latency improvement of the WFQ-CA algorithm over the SCFQ-CA algorithm.

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 propor-

tional to their priority. However, the three schedulers provide a different latency performance. The DRR-CA algorithm, which presents the lowest computational complexity, offers the worst latency results. Moreover, the latency of this scheduler depends on the frame length, which may be very long. Therefore, if we want to provide QoS based on latency requirements, the DRR-CA option may not be the most appropriate. On the other hand, if we want to provide QoS based only on bandwidth, the DRR-CA algorithm is probably the best option due to its computational simplicity. The WFQ-CA algorithm provides the best latency performance. However, this is the scheduler with the highest computational complexity among the three options. Note that when the switching node operates at high speed, a simple and fast scheduling algorithm is mandatory in order to achieve a good performance. On the other hand, the SCFQ-CA algorithm provides a quite good latency performance with an affordable computational complexity. Therefore, the SCFQ-CA option may be a better option if we want to provide QoS based on both latency and bandwidth requirements, without a very high computational complexity.

6. Conclusions

In this paper, we have reviewed the main characteristics of the new AS technology and claimed that in this environment the QoS provision will be very important. This is the reason because we have proposed and compared three scheduler implementations for the AS MinBW scheduler. These new algorithms, which are based on three well-known scheduling algorithms, accomplish all the properties that the AS MinBW scheduler must have, including the interaction with the AS flow control. In some cases, the adaptation of these traditional algorithms to AS is not a trivial task because several aspects must be taken into account. We have called these algorithms WFQ Credit Aware (WFQ-CA), SCFQ Credit Aware (SCFQ-CA), and DRR Credit Aware (DRR-CA). Note that these algorithms can be used not only to implement the AS MinBW scheduler, but also in any network technology with flow control. As far as we know, the important issue of adapting well-known scheduling algorithms to environments that employ a link level flow control mechanism has not yet been treated.

We have studied the computational complexity of the three proposals and their throughput and latency performance. Simulation results show that the three schedulers provide a similar throughput but a different latency 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. Therefore, if we want to provide only bandwidth requirements, the DRR-CA algorithm may be the best option. On the other hand, if we want to provide both latency and bandwidth requirements, the option that offers the best balance between latency performance and computational complexity is the SCFQ-CA algorithm.

As future work we are focusing our attention on obtaining analytical expressions for their latency characteristics.

References

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

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

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

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

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

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

[7] 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.

[8] R. Martínez, F. Alfaro, and J. Sánchez. Providing Quality of Service over Advanced Switching. *International Conference on Parallel and Distributed Systems (ICPADS)*, July 2006.

[9] D. Mayhew and V. Krishnan. PCI Express and Advanced Switching: Evolutionary path to building next generation interconnects. In *Hot Interconnects: 10th Symposium on High Performance Interconnects*, 2003.

[10] P. L. Montessoro and D. Pierattoni. Advanced research issues for tomorrow's multimedia networks. In *International Symposium on Information Technology (ITCC)*, 2001.

[11] 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.

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

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

[14] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round robin. In *SIGCOMM*, pages 231–242, 1995.

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

[16] D. Stiliadis and A. Varma. Latency-rate servers: a general model for analysis of traffic scheduling algorithms. *IEEE/ACM Transactions on Networking*, 1998.