Evaluating several implementations for the AS Minimum Bandwidth Egress Link Scheduler

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

***Abstract*— The relevance of the provision of QoS is taken into account in the definition of the new network technologies like for example Advanced Switching (AS). AS is a new fabric-interconnect technology that further enhances the capabilities of PCI Express, which is the next PCI generation.**

In this paper we discuss the aspects that must be considered for implementing a specific mechanism for the AS minimum bandwidth egress link scheduler, or just MinBW scheduler. We also propose several implementations for this scheduler, analyze their computational complexity, and compare their performance by simulation.

The main differentiating aspect from other interconnection technologies that must be taken into account when implementing the AS MinBW scheduler is that both the link-level flow control and the scheduling are made at a Virtual Channel (VC) level. This means that the scheduler must have the ability to enable or disable the selection of a given VC based on the flow control information.

***Index Terms*— Quality of Service (QoS), Advanced Switching, scheduling algorithms, performance evaluation.**

I. Introduction

CURRENT packet networks are 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 telephony [1]. For example, in some applications, if a packet experiences a latency higher than a certain value, the value to the application of the packet information may be greatly diminished or even worthless. Moreover, a larger delay bound implies increased burstiness of the session at the output of the scheduler, thus increasing the buffering needed at the switches to avoid packet losses [2]. In this line, the IEEE standard 802.1D-2004 [3] defines 7 traffic types at the Annex G with different Quality of Service (QoS) requirements.

Advanced Switching Interconnect, or just Advanced Switching (AS), is a new open-standard fabric-interconnect technology for communications, storage, and embedded environments based on PCI Express [4] that includes in its specification mechanisms to provide the applications with QoS. PCI Express is itself a technology that is replacing the extensively used PCI bus. PCI Express eliminates the legacy shared bus-based architecture of PCI and introduces an improved and dedicated point-to-point interconnect. AS is an extension of PCI Express that adds additional protocols to support reliable and efficient peer-to-peer communications and a rich set of other capabilities. Together, PCI Express and AS have the potential for building the next generation interconnects [5].

A key component for networks with QoS support is the output scheduling algorithm. One of the schedulers defined by the AS specification is the Minimum Bandwidth egress link scheduler, or just MinBW 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.

In [6], we showed how to use the AS mechanisms to provide applications with QoS based on bandwidth and latency requirements. We also presented an implementation of the MinBW scheduler, based on the Self-Clocked Weighted Fair Queuing (SCFQ) algorithm [7]. We called this algorithm SCFQ Credit Aware (SCFQ-CA). In this paper, we study more deeply the considerations that must be made when implementing the MinBW scheduler. Moreover, we review the SCFQ-CA algorithm and propose two additional schedulers for implementing it based on well-known algorithms: the Weighted Fair Queuing Credit Aware (WFQ-CA) and the Deficit Round Robin Credit Aware (DRR-CA) algorithms. We also present a discussion about the computational complexity of the three possibilities. Finally, we evaluate the performance of the three schedulers in a multimedia environment. In order to do so we employ the IEEE standard 802.1D-2004 [3] traffic types.

The discussion on the aspects that must be considered for implementing a specific mechanism for the MinBW scheduler and the proposal of adapted scheduling algorithms for this scheduler is quite significant if we take into account that PCI Express and AS are foreseen to be the *de facto* standard in 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

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.

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 II reviews the MinBW scheduler characteristics. In Section III, we propose our credit aware scheduling algorithms. In Section IV, we analyze the computational complexity of the new schedulers. In Section V, we review how to configure the MinBW scheduler to provide the applications with bandwidth and latency requirements. Details on the experimental platform and simulation scenario are presented in Section VI. Section VII presents and analyzes the simulation results. Finally, some conclusions and future work are given.

II. The AS Minimum Bandwidth Egress Link Scheduler

AS uses Virtual Channels (VCs) to aggregate flows with similar characteristics. The arbitration and the link-level flow control are both made at a VC level. This means that each VC has its own credit count for the credit-based flow control. 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. The table based scheduler faces the problem of not working properly with variable packet sizes, and thus, the MinBW scheduler is intended for a more precise allocation of bandwidth regardless of packet size.

The MinBW 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 specify an algorithm or implementation for the MinBW scheduler, but it must respect certain properties: *Work conserving*, *bandwidth metering, not packet metering*, *minimum bandwidth guarantee*, *fair redistribution of unused bandwidth* and *memoryless* [8].

Analyzing these properties we can state that they refer to an ideal fair queuing model. In a fair queuing 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.

Attending to the specification, several well-known scheduling algorithms exhibit the desired properties of the fair queuing part of the MinBW scheduler: variants of Weighted Fair Queuing (WFQ) [9] such as Self-Clocked Weighted Fair Queuing (SCFQ) [7], and variants of Weighted Round Robin (WRR) [10] such as Deficit Round Robin (DRR) [11]. The AS specification also states that commonly employed schedulers algorithms, such as simple round robin or WRR, do not exhibit the desired properties of the MinBW scheduler and are, thus, not suitable for a MinBW scheduler implementation.

However, the AS specification also states that, when implementing the egress link scheduler, the interaction with the credit-based flow control must be taken into account. If the credits for a given VC have been exhausted, the VC scheduler must treat the corresponding queue as if it were empty. This means that the scheduler must have the ability to enable or disable the selection of a given VC based on the flow control information. Moreover, the scheduler is not allowed to 'save' bandwidth of inactive VCs for future use. Note that these requirements do not appear in technologies with a port-based link-level flow control mechanism like for example Gigabit Ethernet [12].

The problem of the previously stated well-known scheduling algorithms is that they were designed without taking into account the existence of a flow control mechanism, and thus, they do not consider the possibility of disabling a queue based on the flow-control information. In the next section, we propose several implementations for the MinBW scheduler that fulfill all the properties that an AS MinBW scheduler must have, including the interaction with the AS flow control.

III. Implementing the fair queuing mechanism of the MinBW scheduler

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

A. Weighted Fair Queuing Credit Aware

The WFQ algorithm [9] is an approximation of the Generalized Processor Sharing (GPS) model [13]. GPS is a fair queuing 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 head 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 aspect that must be taken into account in order to implement WFQ as part of the MinBW algorithm is that this algorithm uses the real time to calculate the virtual time. Note that the real time includes the time used to transmit the packets from the FMC VC, which are out of the control of the fair queuing MinBW mechanism. The WFQ-CA algorithm fits this problem by not taking into account the time employed in sending packets from the FMC VC 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 packets from the FMC VC. 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.

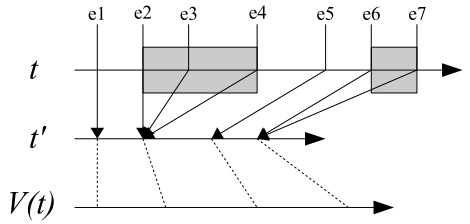

Fig. 1. Time line in the WFQ-CA implementation of the MinBW scheduler.

B. Self-Clocked Weighted Fair Queuing Credit Aware

The SCFQ algorithm [7] defines fair queuing in a self-contained manner and avoids using a hypothetical queuing 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 queuing system. 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. 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.

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

C. Deficit Round Robin Credit Aware

The DRR algorithm [11] associates each queue with a *quantum* and a *deficit counter*. The quantum assigned to a queue is proportional to the bandwidth assigned to that queue. The deficit counter is set to 0 at the start. The scheduler visits sequentially each queue. For each queue, the scheduler transmits as many packets as the quantum allows. When a packet is transmitted, the quantum is reduced by the packet size. 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 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.

IV. Complexity and latency considerations

An ideal scheduling algorithm implemented in a high performance network with QoS support should have good properties, but also have a simple computational complexity in order to achieve a good performance. "Sorted-priority" algorithms, like WFQ and SCFQ, are known to offer very good delay [2]. However, this family of algorithms suffers from two major problems. The first problem is that these algorithms require processing at line speeds for tag calculation and tag sorting. In other words, each time a packet arrives at a node, its time tag is calculated and the packet is inserted at the appropriate position in the ordered list of packets waiting for transmission. This means that these algorithms require

at least the complexity of a search algorithm in the list of queued packets: $O(log(N))$, where N is the maximum number of packets at the queue, or if the buffers are not shared, $O(log(J))$, where J is the number of active flows. The second problem that may happen in the sorted-priority approach is that, since the time tag is an increasing function of the time and depends on a common-reference virtual clock, which in turns reflects the value of the time tag of previously served packets, the virtual clock cannot be reinitialized to zero until the system is completely empty and all the sessions are idle. In other words, it is impossible to reinitialize the virtual clock during the busy period, which, although statistically finite (if the traffic is constrained), can be extremely long, especially given that most communication traffic has been shown to exhibit self-similar patterns which lead to heavily tailed buffer occupancy distributions.

Therefore, for practical implementation of sorted-priority algorithms, very high-speed hardware needs to be designed to perform the sorting, and floating-point units must be involved in the computation of the time tags. This, of course, can be done, but at a great cost and with very limited scalability. As stated before, the SCFQ algorithm avoids the emulation of a GPS system to maintain the *virtual time*. This reduces the computational complexity of the tag calculation. Therefore, the computational complexity of the SCFQ algorithm is lower than the complexity of the WFQ algorithm. However, the latency of the WFQ algorithm does not depend on the number of flows sharing the same egress link. In a SCFQ scheduler, however, the latency is a linear function of the maximum number of flows sharing the egress link [2].

On the other hand, a well-known problem of the WRR and DRR algorithms is that the latency depends 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 [11], DRR exhibits $O(1)$ complexity provided that each flow is allocated a quantum no smaller than the Maximum Transfer Unit (MTU). Moreover, the calculations made by this algorithm are quite simple and do not involve floating-point units. As observed in [14], removing the minimum quantum per flow hypothesis would entail operating at a complexity which can be as large as $O(N)$. Note that 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.

The credit aware versions that we propose of the SCFQ and DRR algorithms have a quite similar complexity than the original ones. However, the WFQ-CA version adds the complexity of the restamping process, which may be a very costly process. Furthermore, when considering the complexity and latency performance of the WFQ-CA and SCFQ-CA algorithms, it must be taken into account that in AS the scheduling is made at a VC level. This involves, for example, that the tag sorting process is quite simpler than in other environments, where each flow is considered separately. In AS, the scheduler must consider only the packets at the head of each active VC. When a packet from a given VC is transmitted, the next packet in the same VC must be inserted in the sorted list of eligible packets. Therefore, in AS the maximum number of packets that the scheduler must consider is twenty (20 VCs per port using a FIFO discipline). Note that, in those environments where the scheduling is made at a flow level, the maximum number of packets that must be considered can be extremely higher. Regarding the latency performance, it must be taken into account that the maximum number of active VCs competing for the egress link is also twenty.

Summing up, regarding the computational complexity, the WFQ-CA scheduler has the highest computational complexity of the three possibilities for implementing the MinBW scheduler that we propose in this paper. However, it is expected to offer the best latency performance. The SCFQ-CA algorithm is still rather complex, but simpler than the WFQ-CA algorithm, and is expected to offer a worse latency performance than the latter. Finally, the DRR-CA scheduler algorithm is the simplest option, but is expected to offer the worst latency performance.

V. Providing QoS with the MinBW scheduler

In [6], we showed how to use the AS mechanisms to provide applications with QoS. We distinguished three broad categories of traffic: Network Control traffic, which is high-priority traffic to maintain and support the network infrastructure; QoS traffic, which is traffic that has explicit minimum bandwidth, maximum latency, and/or jitter requirements; And Best-effort traffic, which is traffic largely insensitive to both bandwidth and latency and only characterized by the differing priority among each other.

First of all, in order to provide the applications with QoS, a certain amount of Service Classes (SCs) with different specific requirements must be specified. Generally, one network control traffic SC, several QoS SCs, and several best-effort SCs. The seven traffic types defined by the IEEE standard 802.1D-2004 [3] at the Annex G fit perfectly this classification. Table I shows each traffic type, which we consider as SCs, and its requirements.

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 egress link scheduler, in this paper the MinBW, must be properly configured in order to provide a different treatment to the VCs attending to the requirements of their associated SCs.

We propose to assign the network control traffic to the FMC VC in order to achieve the maximum priority. The rest of SCs will be assigned to normal VCs and will be scheduled by the fair queuing mechanism of the MinBW scheduler. Providing the traffic of VC with minimum bandwidth requirements using a fair queuing mechanism is as easy as assigning to that VC a weight equal to the proportion of the egress link bandwidth that it needs. Moreover, Parekh and Gallager [13] analyzed the performance of WFQ from the standpoint of worst-case

TABLE I
SCs SUGGESTED BY THE STANDARD IEEE 802.1D-2004.

Type	SC	Description
Control	Network control (NC)	Traffic to support the network infrastructure.
QoS	Voice (VO)	Traffic with a limit of 10 ms for latency and jitter.
QoS	Video (VI)	Traffic with a limit of 100 ms for latency and jitter.
QoS	Controlled load (CL)	Traffic with explicit bandwidth requirements.
Best-effort	Excellent-effort (EE)	Preferential best-effort traffic.
Best-effort	Best-effort (BE)	LAN traffic as we know it today.
Best-effort	Background (BK)	Traffic that should not impact other flows.

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.

In order 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 FMC VC. 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 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.

Finally, an admission control protocol for the QoS SCs must be used to provide QoS guarantees. Note that we use VCs to isolate the traffic with specific QoS requirements from the best-effort traffic. However, the requirements of the QoS flows can be guaranteed only if they do not exceed the amount of bandwidth that they have reserved.

VI. SIMULATION SCENARIO

In [6], 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.

A. 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 combined input-output buffer architecture, with a crossbar to connect the buffers. Virtual output queuing 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].

B. Traffic model

We are going to evaluate the behavior of the three credit aware algorithms we have proposed in using the 7 traffic types defined in the IEEE standard 802.1D-2004 as guidelines to generate the workload. In this way, we consider 7 SCs and each one will be assigned to a different VC, the NC SC being assigned to the FMC.

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 II shows the percentage of traffic of each SC that each node injects regarding the link bandwidth.

The packets are generated according to different distributions, as can be seen in Table II. 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 [16]. 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.

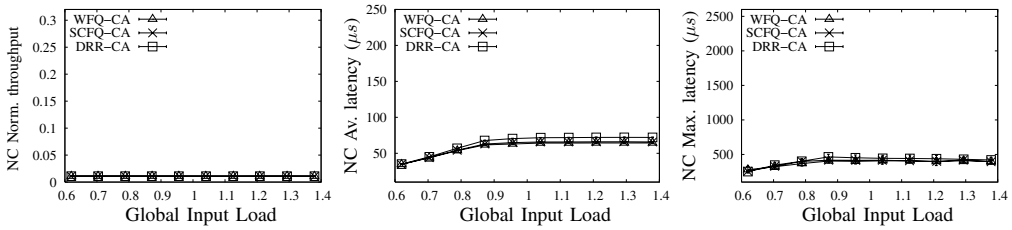

Fig. 2. Performance comparison of the NC SC.

Note that the traffic model that we use in 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.

C. Scheduler configuration

Table II 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 bandwidth 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 [13].

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 given VC is selected. The rest of VCs are assigned a proportional quantum.

TABLE II
INJECTED TRAFFIC AND MINBW CONFIGURATION.

	Injected traffic		**MinBW C.**
SC	**Bandwidth %**	**Traffic pattern**	**Weight**
NC	1	self-similar	-
VO	20.3125	64KB/s CBR	0.265625
VI	20.3125	750 KB/s MPEG-4 traces	0.203125
CL	20.3125	750 KB/s CBR	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

VII. SIMULATION RESULTS

In this section, we show the performance that our three scheduler implementations provide to the different SCs. We perform simulations at different input loads. For each simulation we obtain the normalized average throughput, the average packet latency, and the maximum packet latency of each flow. Moreover, we obtain the maximum jitter for the connection oriented flows (VO, VI, and CL SCs). 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 and the maximum jitter of all the flows of the same SC. Note that the maximum latency and the maximum jitter shows the behavior of the flow with the worst performance. Figures 2, 3, and 4 show the average values and the confidence intervals at 90% confidence level of ten different simulations performed at a given input load of these statistics.

Figure 2 shows the normalized throughput, average latency, and maximum latency performance of the three schedulers for the NC SC. This SC obtains all the bandwdith that it injects and a low latency. All the schedulers provide a similar performance for this SC because in the three cases this SC is assigned to the FMC, and thus, to the strict priority mechanism.

Figure 3 shows the normalized throughput, average latency, maximum latency, and maximum jitter performance of the three schedulers for the QoS SCs. It can be seen that these SCs obtain all the bandwidth they inject. Note that we have used a control admission protocol in order to make sure that these SCs do not inject more traffic than they have reserved. The latency and jitter of these SCs grow slightly with the load until they reach a certain value. Once this value is reached the latency remains more or less constant. Note that the latency and jitter performance provided by the DRR-CA scheduler is rather worse than the provided by the other two schedulers. The average latency provided by the SCFQ scheduler is slightly worse than the provided by the WFQ-CA scheduler. However, the maximum latency and jitter performance are quite similar.

Another difference among 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 of 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.

Figure 4 shows the normalized throughput and average latency of the three schedulers for the best-effort SCs. Contrary to the rest of SCs that obtain all the bandwidth they inject,

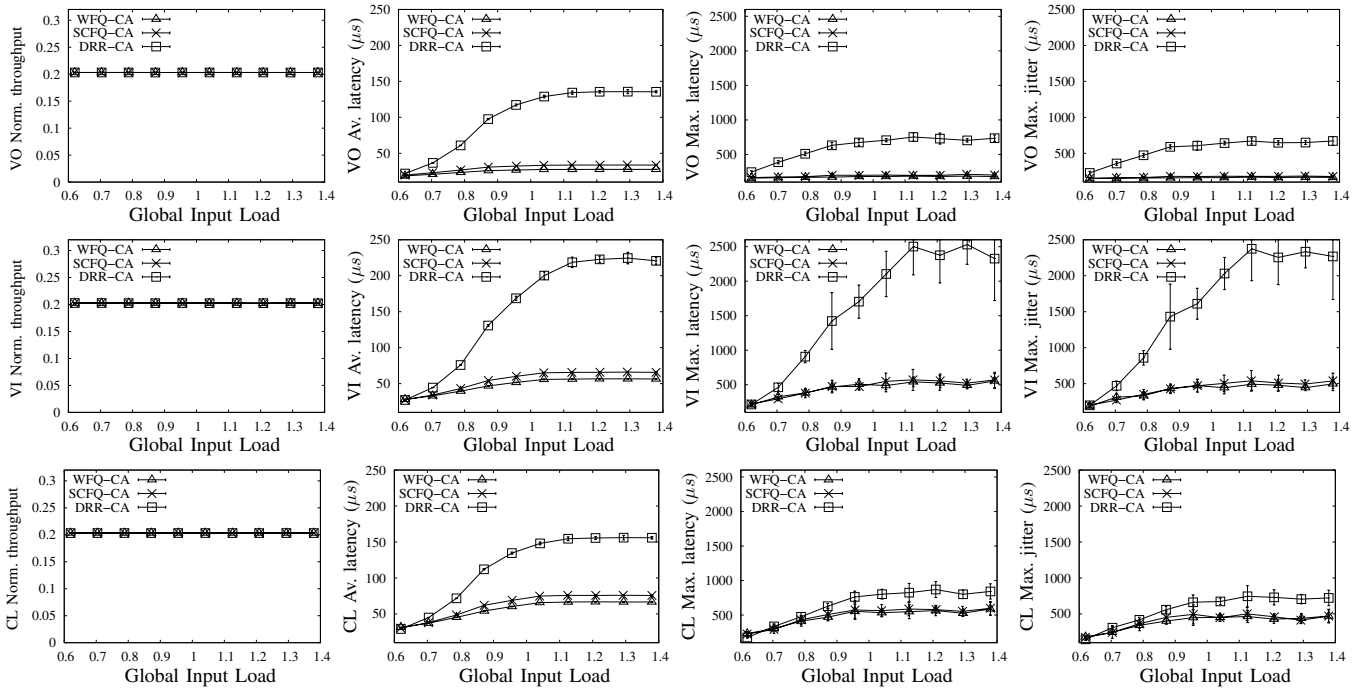

Fig. 3. Performance comparison of the QoS SCs (VO, VI, and CL).

these SCs do not yield a corresponding result when the network load is high (around 85%). This figure also shows that the average latency of these SCs grows with the load. Note, however, that the three best-effort SCs do not obtain the same performance. They obtain a different throughput and average latency according to their different priority.

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. 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 and jitter provided by this scheduler depends on the frame length, which may be quite longer than in this scenario. Therefore, if we want to provide QoS based on latency or jitter 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. Therefore, if we want to provide QoS based on bandwidth, latency and jitter requirements, the SCFQ is probably the best option among the three proposed implementations of the MinBW scheduler. This scheduler provides a good latency performance with an affordable computational complexity.

VIII. Conclusions and Future Work

In this paper, we have reviewed the main aspects that must be taken into account to implement the AS MinBW scheduler. In AS, the link-level flow control and the scheduling are both made at a VC level. This means that the scheduler must have

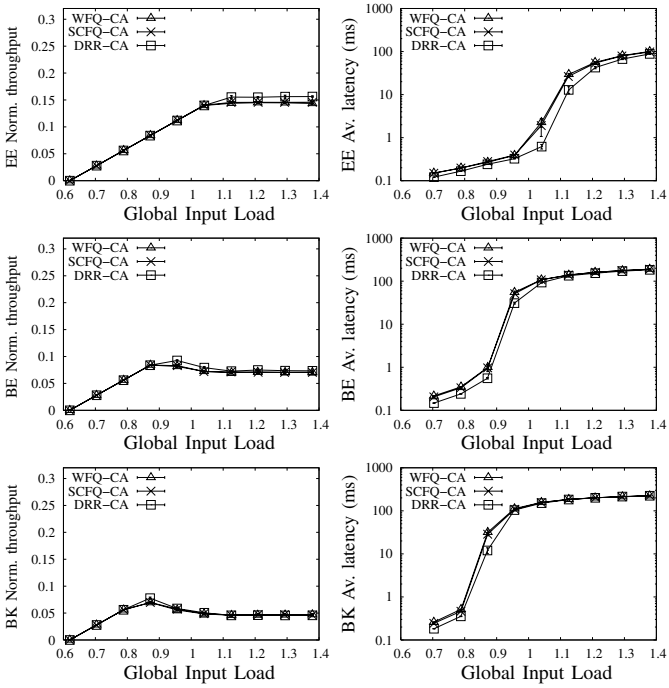

Fig. 4. Performance comparison of the best-effort SCs (EE, BE, and BK).

the ability to enable or disable the selection of a given VC based on the flow control information.

Moreover, we have proposed three new versions of well-known scheduling algorithms that accomplish all the properties that the AS MinBW scheduler must have, including the interaction with the AS flow control. Note that these algorithms can be used not only to implement the AS MinBW scheduler, but also in any network technology with a queue-based link-level flow control that takes into account each queue separately.

We have called these algorithms WFQ Credit Aware (WFQ-

CA), SCFQ Credit Aware (SCFQ-CA), and DRR Credit Aware (DRR-CA). 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. Moreover, we have studied the computational complexity of the three proposals.

We have evaluated the performance of our proposals in a multimedia scenario using IEEE standard 802.1D-2004 [3] traffic types. Simulation results show that if we want to provide QoS based on bandwidth, latency and jitter requirements, the SCFQ is probably the best option among the three proposed implementations of the MinBW scheduler. This scheduler provides a good latency performance with an affordable computational complexity.

A possible limitation of this paper is that we have studied the computational complexity of the algorithms that we propose in a rather general way. A deeper study may allow to offer estimates about the silicon area required to implement the schedulers, and the arbitration time that they would require. Moreover, as well as future work, we think that a study about the effect of a link-level flow control over the analytical models proposed for the well-known scheduling algorithms would be positive. It would be also positive to propose more credit aware versions of other well known scheduling algorithms. Finally, the comparison of the QoS scheme that we propose with other schemes like the proposed by the Differentiated Services [17] model would be interesting.

REFERENCES

[1] P. L. Montessoro and D. Pierattoni, "Advanced research issues for tomorrow's multimedia networks," in *International Symposium on Information Technology (ITCC)*, 2001. [Online]. Available: http://csdl.computer.org/comp/proceedings/itcc/2001/1062/00/10620336abs%.htm

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

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

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

[5] 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.

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

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

[8] *Advanced Switching core architecture specification. Revision 1.0*, Advanced Switching Interconnect Special Interest Group, Dec. 2003.

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

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

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

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

[13] 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.

[14] 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.

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

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

[17] S. Blake, D. Back, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," Internet Engineering Task Force, Internet Request for Comment RFC 2475, Dec. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2275.txt