A framework to provide Quality of Service over Advanced Switching

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

***Abstract*— Advanced Switching (AS) is a network technology that expands the capabilities of PCI-Express adding new features like peer-to-peer communication. Together, PCI Express and AS have the potential for building the next generation interconnects. Furthermore, 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.**

In this paper we propose a framework to provide QoS based on bandwidth, latency, and jitter over AS employing the mechanisms provided by AS. We also present several implementations for the output scheduling mechanism. Finally, we evaluate our proposals by simulation, comparing the performance of the schedulers that we propose and their implementation complexity.

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

I. Introduction

ADVANCED Switching (AS) [1] is an open-standard fabric-interconnect technology based on PCI Express [27]. PCI Express is already replacing the extensively used PCI bus. The PCI bus has served industry well for the last 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 eliminates the legacy shared bus-based architecture of PCI and introduces an improved and dedicated point-to-point interconnect.

AS is an extrapolation of PCI Express, borrowing its lower two architectural layers from PCI Express, and including an optimized transaction layer to enable essential communication capabilities like peer-to-peer communication. The need for AS essentially comes because computing and communication platforms begin to converge by exhibiting increasing overlap in terms of the functions they serve. In this way, 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 [23].

On the other hand, Quality of Service (QoS) is becoming an important feature for high-performance networks and parallel machines. The provision of QoS in computing and communication environments is currently the focus of much discussion and research in industry and academia. 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 telecommunications [24]. The best-effort service model, though suitable for the first type of applications, is not so for applications of the other type [26]. 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 [5].

AS provides mechanisms that can be used to support QoS. Specifically, an AS fabric permits us to employ Virtual Channels (VCs), egress link scheduling, and an admission control mechanism. Moreover, AS performs a link-level flow control in a per VC basis. This means that both the scheduling and the flow control are made at a VC level. These mechanisms allow us to aggregate traffic with similar characteristics in the same VC and to provide each VC with a different treatment according to its traffic requirements.

A key component for providing the VCs with their QoS requirements in AS, and in any other network 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. AS defines two egress link schedulers: The VC arbitration table scheduler and the Minimum Bandwidth egress link scheduler (MinBW). The main problem of the AS table scheduler is that it does not work properly with variable packet sizes. Regarding the MinBW scheduler, AS does not specify an algorithm or implementation for it, but some characteristics that it must respect.

In [22] we showed a very first approach to provide QoS in AS. We examined the AS mechanisms intended for providing QoS and showed how to provide QoS based on bandwidth and latency requirements with fixed packet sizes. In [21] we expanded the basic ideas presented there. Specifically, we showed how to modify the table scheduler in order to support variable packet sizes. Moreover, we proposed a new algorithm, the Self-Clocked Weighted Fair Queuing Credit Aware (SCFQ-CA) algorithm, that fulfills all the properties that the AS MinBW scheduler must have In [20] we proposed two new algorithms for implementing the MinBW scheduler: the Deficit Round Robin Credit Aware (DRR-CA) and the Weighted Fair Queuing Credit Aware (WFQ-CA). In [19] we reviewed the three MinBW schedulers and briefly analyzed their complexity.

In this paper we thoroughly review our previous proposals to provide applications with QoS based on bandwidth, latency, and jitter, and we present a full comprehensive version of all of them. Moreover, we review the four possible scheduling algorithms that we have proposed for the AS output scheduling mechanism, expanding the analysis of their implementation complexity. Finally, we deeply evaluate the performance of all

The authors are with the Computing Systems Department. University of Castilla-La Mancha. Albacete, Spain. {raulmm, falfaro, jsanchez}@dsi.uclm.es

This work has been jointly supported by the Spanish MEC and European Comission FEDER funds under grants “Consolider Ingenio-2010 CSD2006-00046” and “TIN2006-15516-C04-0X”, 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.

our proposals interacting among them. Specifically, we compare the performance of the modified table scheduler and the three MinBW implementation possibilities in a multimedia scenario.

The structure of the paper is as follows: Section II presents a summary of the general aspects in the AS specification including the most important mechanisms that AS provides to support QoS. In Section III, we show the problems of the AS table scheduler with variable packet sizes and also show how to modify this scheduler to solve those problems. In Section IV, we present the main considerations that must be made to implement the MinBW scheduler and we show three possible specific implementations. In Section V, we comment the complexity of the four schedulers that we take into account in this paper: the modified table scheduler and our three MinBW scheduler implementations. In Section VI, we present our proposal to use the AS mechanisms to provide applications with QoS and how to configure the table and the MinBW schedulers in order to provide different QoS requirements. Details on the simulation platform and the performance evaluation are presented in Section VII. Finally, some conclusions are given and future work is proposed.

II. Advanced Switching Review

AS is an extension of PCI Express that includes additional protocols to support reliable and efficient peer-to-peer communications. The AS transaction layer, which is built on top of the physical and link layers of PCI Express, provides a rich set of capabilities like peer-to-peer communications, protocol encapsulation, flexible topologies, multicasting, congestion management, redundant paths, and fail-over mechanisms.

A credit-based flow control protocol ensures that packets are only transmitted when there is enough buffer space at the other end to store them, making sure that no packets are dropped when congestion appears. This makes AS a lossless network. Flow control credits use a 64 bytes granularity. The maximum packet size of an AS packet is 2176 bytes. An AS fabric permits us to employ Virtual Channels (VCs), egress link scheduling, and an admission control mechanism to differentiate between traffic flows.

AS VCs provide a means of supporting multiple independent logical data flows over a given common physical channel.AS supports up to 20 VCs of three different types: Up to 8 bypassable unicast VCs, up to 8 ordered-only unicast VCs, and up to 4 multicast VCs. The bypassable VC with the highest identifier in each network element is called the Fabric Management Channel (FMC). Note that the link-level flow control is made at a VC level. This means that each VC has its own credit count for the credit-based flow control.

The arbitration is also made at a VC level. AS defines two schedulers to resolve between the up to twenty VCs competing for bandwidth onto the egress link: The table scheduler and the MinBW scheduler. A given implementation may choose any of them or may implement its own proprietary mechanism.

When implementing the egress link scheduler the interaction with the credit-based flow control must be taken into account. Packets from VCs that lack sufficient 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. While this situation persists, the bandwidth ordinarily given to that queue is considered excess bandwidth and must be redistributed among queues for which corresponding VC credits are available.

The VC arbitration table is a register array with fixed-size entries of 8 bits. Each 8-bit table entry contains a field of 5 bits with a VC identifier value and a reserved field of 3 bits. When arbitration is needed, the table is cycled through sequentially and a packet is transmitted from the VC indicated in the current table entry regardless of the packet size. If the current entry points to an empty VC, that entry is skipped. The number of entries may be 32, 64, 128, 256, 512, or 1024.

The MinBW scheduler is intended for a more precise allocation of bandwidth regardless of the packet size. This scheduler consists of two parts: The first mechanism, or outter scheduler, provides the FMC with absolute priority, ahead of the other VCs, but with its bandwidth limited by a token bucket. The second mechanism, or inner scheduler, distributes bandwidth amongst the rest of the VCs according to a configurable set of weights. AS does not state a specific algorithm for the inner scheduler, but it must respect certain properties: *Work conserving*, *bandwidth metering, not packet metering*, *minimum bandwidth guarantee*, *fair redistribution of unused bandwidth*, and *memoryless* [1].

The AS specification states that variants of Weighted Fair Queuing (WFQ) [7] such as Self-Clocked Weighted Fair Queuing (SCFQ) [10], and variants of Weighted Round Robin (WRR) [16] such as Deficit Round Robin (DRR) [34] exhibit the desired properties of the inner MinBW scheduler. The AS specification also states that commonly employed scheduling 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.

Moreover, fabric management software may regulate access to the AS fabric, allowing new packet flows entry to the fabric only when sufficient resources are available. Fabric management software may track resource availability by monitoring AS fabric congestion and tracking active packet flows and their bandwidth.

III. Modifying the AS arbitration table to manage variable packet sizes

The main problem of the AS table-based scheduler is that it does not work in a proper way with variable packet sizes, as is common in actual traffic. If the average packet size of the different flows is different, the bandwidth that the flows obtain may not be proportional to the number of table entries. In [18] we showed in detail this problem and proposed a new table-based scheduling algorithm, the Deficit Table (DTable) scheduler, which works properly with variable packet sizes.

The DTable scheduler defines an arbitration table in which each table entry has associated a flow identifier and an *entry weight*. Moreover, each flow has assigned a *deficit counter* that is set to 0 at the start. When scheduling is needed, the table is cycled through sequentially until an entry assigned to an active flow is found. A flow is considered active when its queue has at least one packet and the link-level flow control allows that flow to transmit packets. When a table entry is selected, the *accumulated weight* is computed. The accumulated weight is equal to the sum of the deficit counter for the selected flow and the current entry weight. The scheduler transmits packets from the selected flow until the accumulated weight is smaller than the size of the packet at the head of the selected flow or the selected flow becomes inactive. In the first case, the unused accumulated weight is saved in the deficit

counter, representing the amount of weight that the scheduler owes the queue. In the second case, the remaining accumulated weight is discarded and the deficit counter is set to zero. Each time a packet is transmitted, the accumulated weight is reduced by the packet size.

In order to keep the computational complexity low, the minimum value that a table entry can have associated is the Maximum Transfer Unit (MTU) of the network. This is the smallest value that ensures that there will never be necessary to cycle through the entire table several times in order to gather enough weight for the transmission of a single packet. Note that this consideration is also made in the DRR algorithm definition [34].

In order to adapt the AS table scheduler into the DTable scheduler we must add the deficit counter mechanism and a way to associate each table entry with a weight. Note that the AS specification only considers a VC identifier assigned to each table entry. Adding the deficit counters associated to the VCs would require simple hardware modifications of the original AS table scheduler. However, this modification does not change the interface provided in the AS specification to configure the table scheduler. Note that these counters are set to zero at the beginning and are modified dynamically by the scheduler itself during the scheduling process, and thus they do not require any user configuration.

In order to be able to also assign a weight to each table entry without modifying the AS specification, we propose to use a fix value for all the entries. This weight is the Maximum Transfer Unit (MTU). This modified version of the table scheduler would only require quite simple hardware modifications. If we could change the AS specification, we could employ other approaches to assign each table entry with a weight, for example: To use the 3-bit reserved field of each table entry, to modify the arbitration table structure to dedicate two bytes instead of one to each table entry, and to use the same weight for all the entries of a VC. These different approaches would provide different properties regarding, among other, the amount of memory required to store the arbitration table and the bandwidth assignation granularity. However, in this paper we focus on improving the AS technology without modifying its specificaton. Therefore, we employ the fix value for all the entries option.

Summing up, in this section we have proposed a modification of the original AS table scheduler that works properly with variable packet sizes. However, this modification does not change the structure of the AS arbitration table. In this paper we will refer to this modified version as DTable. Note, however, that a full implementation of the original DTable scheduler would consider a different weight per table entry.

IV. Implementing the inner MinBW scheduler

Analyzing the properties of the inner scheduler of the MinBW cited in Section II, 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 with data in queue ($V \leq N$) during that interval of time.

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. And thus, as stated before, 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 [33].

The problem of the well-known scheduling algorithms that AS states as appropriate 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. The reason is that they were originally proposed for networks that do not have link layer flow control, for example Internet or ATM.

In this section, we present three new *credit aware* algorithms for the inner MinBW scheduler based on the well-known WFQ, SCFQ, and DRR scheduling algorithms. The resulting credit aware algorithms fulfill all the properties that the inner MinBW scheduler must have, and thus they can be used to implement this scheduler.

A. Weighted Fair Queuing Credit Aware

The WFQ algorithm [7] is an approximation of the Generalized Processor Sharing (GPS) model [25]. 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 Credit Aware (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 VC 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 VC. Packets are transmitted in an increasing order of timestamp, but only those VCs with enough credits to transmit the packet at their head are taken into account. When a VC 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.

B. Self-Clocked Weighted Fair Queuing Credit Aware

The SCFQ algorithm [10] 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. 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.

The SCFQ Credit Aware (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 VC queue, it is stamped with its service tag only if it is at the head of the VC 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 VC 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 VC queue i. This permits us to wait to stamp a packet until it reaches the VC head [32]. This allows the scheduler to require only a service tag per VC, instead of a tag per each individual packet like in the WFQ case [31]. Therefore, the SCFQ-CA does not require the restamping process of the WFQ-CA algorithm, and thus simplifies the scheduling process.

C. Deficit Round Robin Credit Aware

The DRR algorithm [34] is a variation of the Weighted Round Robin (WRR) algorithm [16] that works in a proper way with variable packet sizes. In order to handle properly variable packet sizes the DRR algorithm associates each flow with a *quantum* and a *deficit counter*. The quantum assigned to a flow is proportional to the bandwidth assigned to that flow. The deficit counter is set to 0 at the beginning. The scheduler visits sequentially each flow and serves packets in the same way than the deficit mechanism of the DTable scheduler.

The DRR Credit Aware (DRR-CA) algorithm that we propose works in the same way as the DRR algorithm, except in the following aspects: A VC 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 VC. When a packet is transmitted, the next active VC is selected when any of the following conditions occurs:

- There are no more packets from the current VC or there are not enough flow control credits for transmitting the packet that is at the head of the VC. In this case, the current VC 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 VC. 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 is defined as the sum of all the quantum values. 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 [34], DRR exhibits $O(1)$ complexity provided that each flow is allocated a quantum no smaller than the MTU. As observed in [14], removing this 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.

V. Complexity considerations about the schedulers

An ideal scheduling algorithm implemented in a high performance network with QoS support should provide a good end-to-end delay (also called latency). The end-to-end delay is defined as the sum of the transmission delay, the propagation delay, and the queuing delay experienced at each network node. The last component is by far the most significant. In some applications if a packet experiences a latency higher than a certain value, the value 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 [36]. Thus, a good scheduling algorithm should guarantee acceptable queuing delay.

However, the end-to-end delay that a scheduler is able to provide is not the only parameter that must be taken into account when deciding which is the best scheduler in a high performance network with QoS support. The second main property that a scheduling mechanism should satisfy is a low complexity. This is because in order to achieve a good performance, the processing overheads must be some orders of magnitude smaller than the average packet transmission time. This means that the time needed to decide the next packet to be transmitted must be very small, if we consider the high speed of high performance networks. Moreover, a low complexity is required in order to be able to implement the scheduler in a small silicon area (note that high-performance switches are usually implemented in a single chip).

In this section we comment the complexity of the four schedulers that we have presented in this paper: the DTable scheduler and the three possible implementations of the MinBW scheduler.

A. The WFQ-CA and the SCFQ-CA schedulers

"Sorted-priority" algorithms, like WFQ and SCFQ, are known to offer very good delay [36]. 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. 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.

When considering the complexity 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 much more 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. Only when a packet from a given VC is transmitted, the next packet in the same VC may be inserted in the sorted list of eligible packets (if they have enough credits to be transmitted). Therefore, in AS the maximum number of packets that the scheduler must consider is twenty, which is the maximum number of VCs. Note that, in those environments where the scheduling is made at a flow level, the maximum number of packets that must be considered would be extremely higher.

Moreover, if we compare the complexity of the WFQ-CA and SCFQ-CA algorithms, apart from the complexity of the emulation of the GPS system, which is inherent to the WFQ algorithm, the WFQ-CA algorithm adds the complexity of the restamping process, which may be a very costly process.

B. The DRR-CA scheduler

The complexity of the DRR algorithm is quite small. Provided that each flow is allocated a quantum no smaller than the MTU and if a list of active flows is maintained, the algorithm can cycle through the list knowing that it is always possible to transmit at list one packet from each flow or, which is the same, that there will never be a need to cycle through the entire table several times in order to gather enough weight for the transmission of a single packet. Each time a packet is transmitted, the algorithm must compute if more packets from the same flow can be transmitted or it must change to the next active flow. However, this computation can be performed with simple adder and subtractor units. Note that the WFQ-CA and the SCFQ-CA schedulers require complex divisor units to calculate the time tags.

In the case of the DRR-CA algorithm the number of queues is equal to the number of VCs instead of the number of flows, and thus the complexity is even smaller. The only added complexity remains in taking into account the flow control in order to consider active or inactive a VC. This is clearly the simplest algorithm considered in this paper.

C. The DTable scheduler

In the case of the AS table scheduler, a list of active VCs would not be as simple to maintain as in the DRR case, because VCs must be visited not in a sequential way but in the order indicated by the table scheduler. Therefore, in this case the table must be looked over searching for the next active entry and skipping those entries that refer to a VC without packets or credits to transmit. Although the checking of each entry can be made with very simple computational units, in the worst case all the table must be looked over in order to find the next active entry. This kind of mechanism probably requires very little silicon area to be implemented, but may last too much time. In order to make the process faster several entries of the table can be read simultaneously at the expense of increasing the silicon area requirements. However, this algorithm has not the problem of the increasing tag value and does not need floating point units like in the case of the SCFQ and WFQ algorithms.

The deficit mechanism added to the AS table scheduler to implement the DTable scheduler only requires simple integer units, like adders and subtractors, to be implemented. Moreover, the memory requirements for this algorithm over the original table scheduler are the memory needed to store the deficit counter for each VC. The MTU is 2176 bytes (34 credits of 64 bytes) in a generic case for AS, and thus, the maximum deficit counter value is 33. We need at least 6 bits to represent this number. Therefore, in the more general case of 20 VCs this means $20 * 6 = 120$ bits per egress link. Taking all these things into account the DTable algorithm is probably simpler than the WFQ-CA and SCFQ-CA algorithms but more complex than the DRR-CA algorithm. Table I summarizes the different complexity issues that we have considered to performe this comparison among the four schedulers.

VI. Providing QoS over AS

As was stated in Section II, AS provides several mechanisms that can be used to provide QoS. However, the AS specification does not indicate how to use these mechanisms. In this section, we propose a way of using some of the above-presented AS mechanisms in order to provide QoS.

A. Traffic classification

As stated before, the AS output scheduling mechanism is applied not over flows but over VCs. Therefore, in order to provide QoS over AS, a set of Service Classes (SCs) with different requirements must be specified. When various flows obtain access to the AS fabric, they will be assigned a SC depending on their characteristics. If there are enough VCs we will devote a separate VC to the aggregated traffic of each existing SC. However, if the network that we are using does not have as many VCs as SCs we have defined, more than one SC should be assigned to the same VC and the scheduler should provide to each VC the most restrictive QoS requirements of the SCs that it has assigned.

From a networking perspective the general QoS provisioning parameters are throughput, latency, jitter, and loss-rate. Throughput is the effective number of data units transported per time unit, while latency is the time interval between the departure of a packet from the source to its arrival at the destination. Jitter represents the variance in latency and can be calculated as the difference between the latencies of consecutive packets belonging to a given flow. Finally, loss-rate is the percentage of packets that is not delivered to their destination.

The degree of sensitivity to each of these parameters varies widely from one application to another. For example, multimedia

TABLE I
COMPLEXITY COMPARISON SUMMARY.

Algorithm	Tag calculation	Tag shorting	Tag overflow	Computation units	Other considerations
WFQ-CA	High	Yes	Yes	+, /, comparison	tag per packet, restamping process
SCFQ-CA	Low	Yes	Yes	+, /, comparison	tag per VC
DRR-CA	-	No	No	+, -	-
DTable	-	No	No	+, -	Arbitration table searching process

applications are usually sensitive to latency and jitter, but many of them can tolerate packet losses to some extent. However, the severity of the effect of loss on the quality of these applications is also influenced by parameters such as the compression and encoding techniques used, loss pattern, transmission packet size, and the error recovery technique implemented [39]. For a further discussion about different applications and their requirements, see [9].

In order to define the different SCs, we propose a traffic classification based on three network parameters: Bandwidth, latency, and jitter. In this way, this classification is similar to the one presented by Pelissier [28]. We distinguish between three broad categories of traffic:

- Network Control traffic: High-priority traffic to maintain and support the network infrastructure. One SC will be dedicated to this kind of traffic.
- QoS traffic: This traffic has explicit minimum bandwidth, maximum latency, and/or jitter requirements. Various QoS SCs can be defined with different specific requirements. This category can be divided into two groups:
  - Traffic which requires a given minimum bandwidth and must be delivered with a maximum latency and/or jitter in order for the data to be useful. Examples of such data streams include video conference, interactive audio, and video on demand.
  - Traffic which requires a given minimum bandwidth but is not particularly sensitive to latency or jitter.
- Best-effort traffic: This traffic accounts for the majority of the traffic handled by data communication networks today, like file and printing services, web browsing, disk backup activities, etc. This traffic tends to be bursty in nature and largely insensitive to both bandwidth and latency. Best-effort SCs are only characterized by the differing priority among each other.

B. Admission control

In a loss-less network like AS, congested packets are not thrown away and as such the loss-rate due to congestion is zero. This has the advantage of avoiding retransmissions that would severely affect the latency and jitter performance of the flows. On the case of applications with packet loss resilience, it would allow to reduce the overhead due to the encoding techniques used to minimize the impact of errors. On the other hand, loss-less networks have other problems, being the most important the formation of congestion (or saturation) trees [29]. This congestion trees may produce a dramatic network performance degradation, affecting not only the flows traversing the original point of congestion, but other flows that share common upstream links.

A common approach to avoid this problem is by using an admission control (AC) mechanism. The AC decides whether a new connection is accepted or rejected and ensures that the entry of additional traffic into a network cannot create congestion. Many AC schemes have been proposed. In [30] an implementation and comparison of a probe, a statistical, and a bandwidth broker approach is presented.

AS specification just cites AC as a possible mechanism to be used, but does not give any indication of how to implement it. We propose to use a bandwidth broker, which is an AC scheme that makes the decisions based on the bandwidth that is expected to consume the new flow. This solution assumes that both topology and routing information about network is available. Moreover, the flows must use the same path during all their life. This is possible in AS due to its source-based routing.

The bandwidth broker algorithm must maintain a graph of the network egress links reporting the available free bandwidth on each link. When a new connection tries to get access to the network, the bandwidth broker checks if there is enough bandwidth available all along the path of that connection. If all the links have enough bandwidth, the amount of required bandwidth is subtracted from the available bandwidth of those links and the new connection is accepted. If any of the links have not enough bandwidth to accommodate the new flow the connection is rejected.

One of the main problems of the bandwidth broker AC mechanism is the connection establishment procedure overhead. Applying this mechanism when trying to initiate every single flow can produce an excessive overhead. However, as stated before, AS defines the credit-based flow control and the scheduling mechanisms at the VC level. This provides a certain degree of isolation to the traffic traversing one VC regarding the traffic of the rest of VCs. Specifically, it allows devoting a certain minimum proportion of the link bandwidth to each VC. Therefore, this allows us to apply the bandwidth broker mechanism over a reduced subset of VCs in order to avoid the appearance of congestion trees within those VCs. Even in the case that congestion trees appear in the rest of VCs, the traffic of the brokered VCs will not be affected.

Therefore, we propose to apply the AC mechanism only to those VCs employed by the QoS SCs and not to the control SC or the best-effort SCs. Note, that the QoS SCs are the ones which actually have specific QoS requirements. In addition, the latency constraints of the control traffic are not so clear. Moreover, we can assume that the amount of control traffic that is going to traverse the network is going to be quite small. And thus, taking into account the maximum amount of expected control traffic, the scheduling algorithm can assign the network SC with an *a priory* amount of bandwidth.

Note that the AS source-based routing allows this AC approach to not need specific flow information in the switches in order to make sure that each flow uses always the same path through the network. Switches must only maintain the configuration of the output schedulers at a VC level. In this way, this AC approach is an end-to-end mechanism that can be implemented in a centralized manner, which has all the brokering information

in a single host, or in a distributed manner, like in [11]. As a first approximation, a centralized bandwidth broker[1] based on the average bandwidth value required per each flow is used in the performance evaluation section.

C. Scheduler configuration

The schedulers must be properly configured at the different network elements to distribute the bandwidth among the different VCs but also to provide the traffic traversing each VC with a differentiated treatment.

There are two possible ways of configuring the schedulers. The first possibility is to configure the schedulers in advance, assigning each VC with a specific weight in the case of the MinBW scheduler, or assigning each table entry with a given VC identifier, in the table scheduler case. This distribution would be made taking into account the requirements and expected amount of traffic of the SCs that traverse each VC [30].

The second possibility is to configure the schedulers in accordance with the connection requirements in a dynamic way. With this approach, the scheduler configuration may be modified both when a new connection is accepted and when a previously established connection ends [2].

In the following sections we will show how to configure the two normative AS schedulers, the MinBW scheduler and the table scheduler, to provide the flows aggregated in the different VCs with bandwidth and latency requirements. Note that the maximum jitter performance is intimately linked to the maximum latency performance, and thus, we can translate any maximum jitter requirement into a maximum latency requirement.

1) Configuring the MinBW scheduler: 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. The control SC will be assigned to the FMC in order to achieve the maximum priority, and thus no bandwidth will be assigned explicitly to this SC.

Parekh and Gallager [25] analyzed the performance of a queuing network with fair queuing service discipline and derived upper bounds on the end-to-end delays when the input traffic streams conform to the leaky bucket characterization. As a first approximation, we are not going to conform the traffic to a given pattern, but 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. Therefore, not all the bandwidth can be distributed among the SCs, thereby requiring a certain bandwidth to be left unassigned. Moreover, a certain amount of bandwidth must be reserved to the control SC according to its expected traffic. 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.

2) Configuring the Table scheduler: The main advantage of the table is that it allows us to configure not only the number of table entries assigned to each queue or VC, but also the distribution of the entries assigned to each queue. In [3], we explained how to configure this kind of arbitration table (in that case for InfiniBand) to provide bandwidth and latency guarantees.

In order to provide traffic of a given VC with a minimum bandwidth, the number of table entries assigned to that VC must accomplish with the proportion of desired egress link bandwidth. In order to provide maximum latency requirements to the traffic of a VC, the maximum separation between any consecutive pair of entries devoted to that VC must be fixed. Controlling the maximum separation between any consecutive pair of entries assigned to the same VC, it is possible to control the latency of that VC. This is because this distance determines the maximum time that a packet at the head of a flow queue is going to wait until being transmitted. If for example, we assign a VC a maximum separation of 2, that VC is going to be given the possibility to transmit after any other VC has been selected to do so and thus, the waiting time at the egress queue is going to be very short.

This way of assigning the entries of the table faces the problem of bounding the bandwidth and latency assignments. If one sets a maximum separation between two consecutive table entries of a VC, a certain number of them are being assigned, and hence a minimum bandwidth, to the VC in question. This can be a problem because the most latency-restrictive traffic does not usually require a high bandwidth reservation. This is usually the case of, for example, the control traffic. However, in this case, we can prevent this problem by assigning the control SC the bandwidth that, as it has been previously said, should be left unassigned in the MinBW case.

VII. Performance Evaluation

In our previous work [22], [21], [20], [19], we have evaluated first approaches of our proposals and compared the performance of subsets of our proposed schedulers. In this paper, we evaluate thoroughly our proposals, comparing the performance of the four possible scheduling mechanisms that we have proposed for AS: the DTable scheduler and the three possible implementations of the MinBW scheduler. Specifically, we compare their throughput, average and maximum latency, and average and maximum jitter performance. 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.

A. Simulated architecture

We have used a perfect-shuffle Bidirectional Multi-stage Interconnection Network (BMIN) [8] with 64 end-points connected using 48 8-port switches. In AS any topology is possible, but we have used a BMIN because it is a common solution for interconnection in current high-performance environments [37]. The switch model uses a 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 [4]. However, all the queues of a VC share the same credit count.

[1]The use of a centralized or distributed AC does not affect the simulation results obtained in this paper.

TABLE II
SET OF SCS CONSIDERED.

	IEEE 802.1D-2004 traffic types suggestion		Simulated traffic pattern	
Type	**SC**	**Description**	**Traffic pattern**	**Packet size**
Control	Network control (NC)	Supports the network infrastructure.	Bursts1	up to 256B
QoS	Voice (VO)	Limit of 10 ms for latency and jitter.	64 Kb/s CBR connections	168B
QoS	Video (VI)	Limit of 100 ms for latency and jitter.	3 Mb/s MPEG-4 traces	up to 2176B
QoS	Controlled load (CL)	Explicit bandwidth requirements.	750 kb/s CBR connections	2176B
Best-effort	Excellent-effort (EE)	Preferential best-effort traffic.	Bursts60	up to 2176B
Best-effort	Best-effort (BE)	LAN traffic as we know it today.	Bursts60	up to 2176B
Best-effort	Background (BK)	It should not impact other flows.	Bursts60	up to 2176B

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 [15], [17]. The time header takes to cross the switch without any load is 150 ns, which is the same unloaded cut-through latency of the AS StarGen's *Merlin* switch [35].

B. Traffic model

The IEEE standard 802.1D-2004 [12] defines 7 traffic types at the Annex G, which are appropriate for this study. We will consider each traffic type as a SC. Table II shows each SC and its requirements. In this way, the workload is composed of 7 SCs and each one of them will be assigned to a different VC, the NC SC being assigned to the FMC.

The packets from each SC are generated according to different distributions, as can be seen in Table III. VO, VI, and CL SCs are composed of point-to-point connections of the given bandwidth. VO and CL SCs are generated following a Constant Bit Rate (CBR) distribution. In [38] several payload values for voice codec algorithms are shown. These values range from 20 bytes to 160 bytes. We have selected a payload of 160 bytes for the VO SC traffic. In the case of VI SC, MPEG-4 traces are used to generate the size of each frame. Each frame is injected into the network interfaces every 40 ms. If the frame size is bigger than the MTU, the frame is split into several packets which are injected all along the frame time. The traffic of the best-effort SCs is generated according to a Bursts60 distribution [6]. This traffic is composed of bursts of 60 packets heading to the same destination. The packets' size is governed by a Pareto distribution, as recommended in [13]. In this way, many small size packets are generated, with an occasional large size packet. The periods between bursts are modeled with a exponential distribution [13]. The Bursts60 pattern models worst-case real traffic scenarios. The NC SC is generated in the same way than the Burst60 traffic but with only one packet burst. For all the cases, the destination pattern is uniform in order to fully load the network.

Our intention is to show that with an AC mechanism for controlling the QoS traffic and a relatively small amount of control traffic (as is usually the case), the QoS requirements of the different SCs are met, whatever the load of best-effort traffic. For that purpose, we inject a fixed amount of control traffic (NC) and QoS traffic (VO, VI, and CL) all the time, and we gradually increase the amount of best-effort traffic (EE, BE, and BK). The amount of QoS traffic to be injected is the maximum allowed by the AC. Table III shows the proportion of traffic of each SC that each node injects regarding the link bandwidth.

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.

TABLE III
INJECTED TRAFFIC AND SCHEDULER CONFIGURATION.

	Injected traffic		Table C.		MinBW C.
SC	**Min.**	**Max.**	**# Entr.**	**Dist.**	**Weight**
NC	0.01	0.01	16	4	-
VO	0.1875	0.1875	16	4	0.25
VI	0.1875	0.1875	12	6	0.1875
CL	0.1875	0.1875	12	(6)	0.1875
EE	0	0.1425	5	(16)	0.078125
BE	0	0.1425	2	(32)	0.03125
BK	0	0.1425	1	(64)	0.015625
Total	0.5725	1	64		0.75

C. Scheduler configuration

The configuration of the DTable and the MinBW schedulers is shown in Table III. In order to compare the two schedulers we have assigned in both cases the same amount of bandwidth to each SC.

1) **Best-effort SCs.** We want to reserve 25% of link bandwidth to best-effort traffic. However, we have only assigned best-effort SCs 12.5% of bandwidth, because we do not need more to establish the preference between them.
2) **NC SC.** We have assigned the NC SC with 25% of bandwidth (rest of best-effort bandwidth + expected amount of control traffic + expected amount of lost network bandwidth). Note that the bandwidth assigned to the NC SC in the table case is left unassigned in the MinBW case.
3) **QoS SCs.** The remaining bandwidth has been distributed between the QoS SCs. We will inject the same amount of traffic of the three QoS SCs considered. However, in the MinBW case the way of providing a better latency to a SC is assigning a higher amount of bandwidth than is actually required to fulfill its bandwidth requirements [25]. Therefore, we have assigned 33% more bandwidth to VO SC due to its higher latency requirements.

For the sake of simplicity, a table of 64 entries has been used in the simulations. In order to fill in the table with the VC identifiers we have assigned the table entries minimizing the distance between any consecutive pair of entries for the NC, VO, and VI SCs, which are the SCs with latency requirements. Therefore, we have assigned a maximum distance of 4 to the NC and VO SCs, which have 16 entries each one, and a maximum

distance of 6 to the VI SC, which has 12 entries. For the CL SC and the best-effort SCs this is not necessary and we could have assigned the entries sequentially in the free gaps of the table, but to achieve better latency results for these SCs we have assigned their entries minimizing the distance between entries. Figure 1 shows the final distribution of the table entries among the VCs.

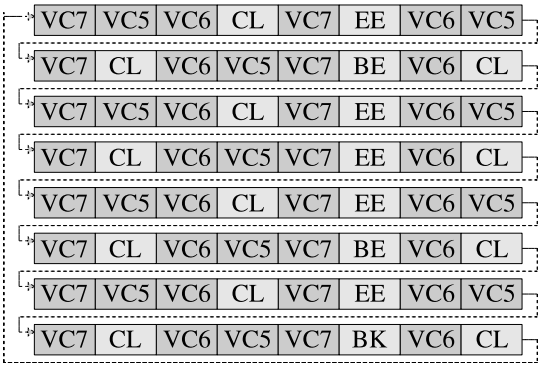

Fig. 1. Arbitration table configuration.

D. Simulation results

The figures of this section 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 average throughput, the average packet latency, the maximum packet latency, the average jitter, and the maximum jitter of each flow. No statistics on packet loss are given because, as it has been said, AS has a credit-based flow control mechanism to avoid dropping packets. 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 jitter, and the maximum latency and jitter of all the flows. Note that the maximum latency and jitter shows the behavior of the flow or flows with the worst performance.

Figure 2 gives a general overview of the performance when using the WFQ-CA variant of the MinBW scheduler. We do not show similar figures for the rest of schedulers due to lack of space. However, although the specific values are different, the general tendencies for the other mechanisms are the same. And thus, the comments that we are going to make based on this figure can be generalized to the rest of schedulers. If we compare the injection and the throughput results, we can see that that the NC and the QoS SCs obtain all the bandwidth that they inject. However, when the network load is high (around 75%), the best-effort SCs do not yield a corresponding result. From that input load, these SCs obtain a bandwidth proportional to their priority.

Regarding the latency performance, Figure 2 shows that the average and maximum latency of the control and QoS SCs grow with the load until they reach a certain value. Once this value is reached the latency remains more or less constant. This is because when the load is low the number of conflicts between packets from different VCs is also low, and thus the latency of all the SCs is also low. However, the average latency of best-effort SCs continually grows with the load. Furthermore, it can be seen that best-effort SCs obtain different average and maximum latency according to their different priority. In that sense, for example, the BK SC obtains a worse latency and starts to increase its latency sooner than the BE and EE SCs. Note that although the control

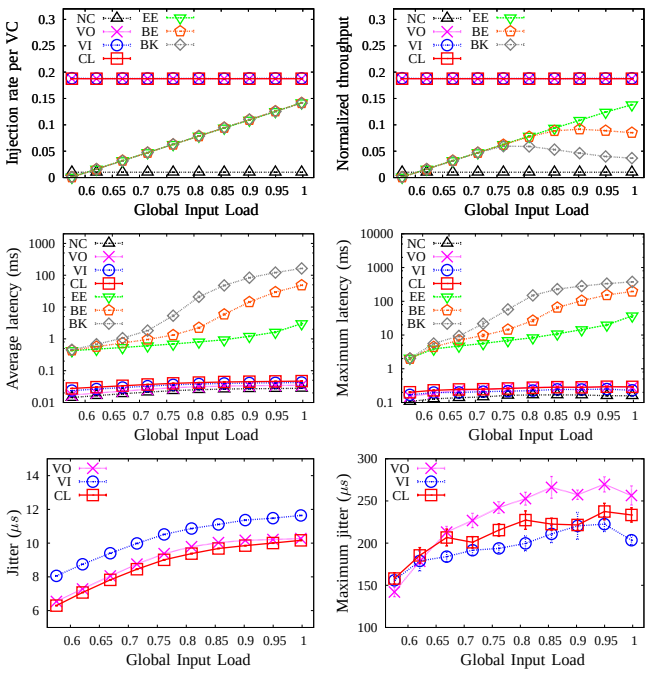

Fig. 2. Performance per VC with the WFQ-CA scheduler.

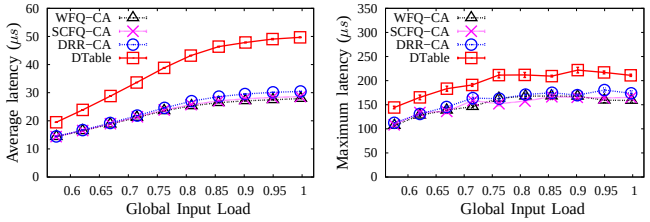

Fig. 3. Latency performance comparison for the NC SC.

and QoS SCs employ separate VCs, the growing best-effort traffic slightly affects their performance.

Regarding the jitter performance, Figure 2 shows only the performance of the QoS SCs because this is the connection oriented traffic and the jitter metric is only relevant for this kind of traffic. The tendency is similar to the latency case. The jitter slightly grow with the load until they reach a certain value.

Figures 3 and 4 show a more detailed comparison between the different schedulers for the control and QoS SCs. We do not show the performance comparison for the best-effort SCs because the relevant aspect of these SCs is not the actual latency and jitter values but that they obtain a differentiated performance among them. In this sense, the four schedulers provide this differentiation.

Regarding the control SC, Figure 3 shows that the three possibilities for the MinBW scheduler provide a similar performance, which is better than the provided by the DTable scheduler. This is because, as stated before, the MinBW scheduler employs a strict priority mechanism to schedule the FMC, which is the VC that we have assigned to the control SC. However, in the DTable scheduler case the control traffic does not have strict priority and it must compete with the other traffic.

Regarding the QoS SCs, Figure 4 shows that the DRR-CA variant of the MinBW provides the worst latency and jitter performance of the four schedulers. Specifically, the latency and jitter values offered by this scheduler are around the double than in the other cases. Note also that the performance of this scheduler

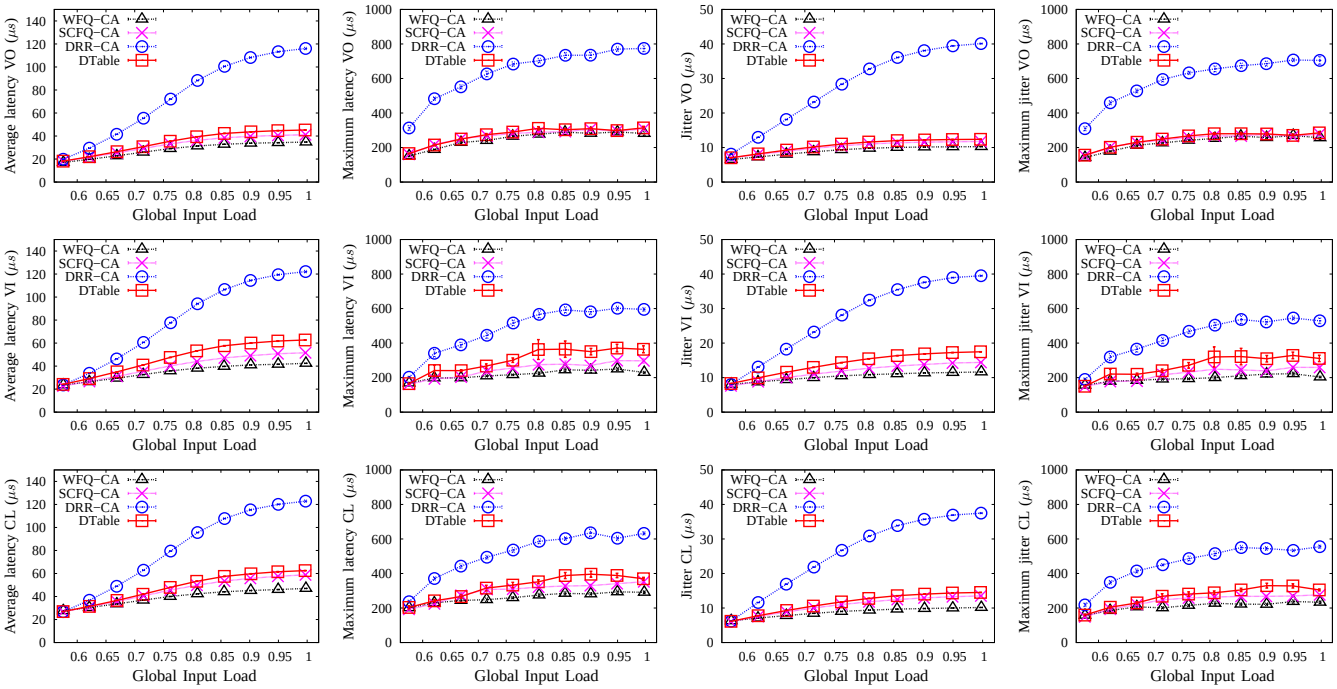

Fig. 4. Latency and jitter comparison for the QoS SCs.

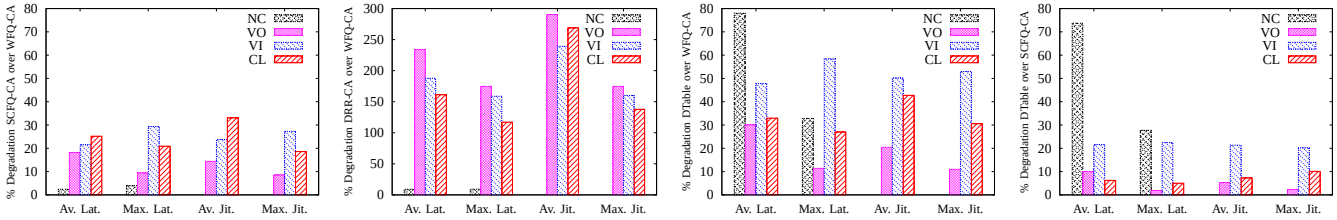

Fig. 5. Percentage of degradation in the performance at maximum traffic load.

depends on the frame length, and thus, in other scenarios, the performance could be even worse. If we compare the WFQ-CA and the SCFQ-CA MinBW variants, and the DTable scheduler, we can see that the WFQ-CA provides the best performance and the DTable the worst. However, the differences are actually very small. In some cases, we can even see that the different lines and/or confidence intervals are overlapped.

Figure 5 shows the percentage of degradation in the latency and jitter performance at maximum traffic load for the control and QoS SCs of the SCFQ-CA, the DRR-CA, and the DTable scheduler over the WFQ-CA scheduler, and the DTable scheduler over the SCFQ-CA scheduler. This figure shows that the SCFQ-CA scheduler entails only a maximum of 35% of degradation if compared with the WFQ-CA scheduler. This slight degradation may not justify the use of the WFQ-CA, which, as stated before, has much more computational complexity. On the other hand, the DRR-CA algorithm provides for this scenario up to 300% degradation. And thus, this scheduler despite its very low computational complexity does not seem appropriate for providing latency and jitter requirements. Finally, we can see that the DTable scheduler provides a degradation for the NC SC of 70%-80% on the average latency but only 30% on the maximum latency. Regarding the QoS SCs it provides a maximum degradation of 60% if compared with the WFQ-CA algorithm and 25% if compared with the SCFQ-CA algorithm. Therefore, although the performance of this scheduler is worse than both the WFQ-CA and the SCFQ-CA algorithms, it can be a good alternative for being used as the output scheduler mechanism if we take into account its relatively low complexity. Moreover, in this performance evaluation, in order to perform a fair comparison, we have considered the same scheduling time for all the schedulers. Note that if we would consider the computational complexity of each algorithm, the arbitration times would be probably different, and the difference between the WFQ-CA and the SCFQ-CA, and between the SCFQ-CA and the DTable schedulers would be probably even smaller.

VIII. Conclusions and Future Work

In this paper, we have proposed a framework to provide QoS requirements over AS. Specifically, we have proposed to define a reduced set of SCs with different bandwidth, latency, and jitter requirements. After that, we have proposed how to take advantage of the VCs, the link level flow control, the output scheduling, and the admission control mechanisms provided by AS in order to efficiently provide the different SCs with their requirements. In order to provide a differentiated treatment to the SCs we have proposed three specific implementations for the MinBW scheduler and the DTable scheduler as a modification to the table scheduler stated in the AS specification.

We have evaluated the performance of our proposals in a multimedia scenario using the IEEE standard 802.1D-2004 traffic types. Simulation results show that our proposals are able to

provide the SCs with their requirements in the test scenario: the network control SC obtains a good latency; the SCs with bandwidth requirements obtain the amount that they need; the SCs with latency or/and jitter requirements do not exceed the maximum allowed; the best-effort SCs obtain a different bandwidth and latency performance in accordance with their different priority. Although the simulation results obtained cannot prove that our proposals are going to be able to provide QoS requirements in every other environment or architecture. We believe that they provide a very good insight in their posibilities.

Moreover, analyzing the simulation results and the complexity of the different schedulers we can conclude that the WFQ-CA MinBW variant provides the best performance, but its complexity is excessive. The DRR-CA MinBW variant is not adequate to provide latency and jitter requirements. The SCFQ-CA variant is probably the best MinBW option due to its good performance and lower complexity than the WFQ-CA variant. Finally, the DTable scheduler is a good option due to its low complexity and relatively good performance.

As future work we are focusing our attention on offering a more accurate model to configure the schedulers and the admission control in order to provide the aggregated of flows with their requirements. This implies to choose how much bandwidth of each VC can be permited into the network by the admission control mechanism. On the other hand, with respect to the scheduler, it should be determined the weights to assign to each VC in the case of the MinBW and the maximum distance and number of entries in the DTable case.

Moreover, a deeper study on the complexity of the algorithms that we propose may allow us to offer hardware estimates. In fact, we are performing an study about the arbitration time and the silicon area that each algorithm would require. We are taking into account different values for some design parameters. We have considered the number of VCs and the MTU in all the cases. Moreover, for the DTable scheduler we have also considered the size of the table in terms of table entries and the parallelization grade, which is the number of table entries that we read each cycle. Preliminary results show that the cost of modifying the original AS table to handle in a proper way variable packet sizes is very small (around 10% increment in silicon area). Moreover, the DTable scheduler can be a simpler option, at least in terms of silicon area, when a small number of table entries is implemented (32-256) if compared with the SCFQ-CA scheduler.

Moreover, we are also working on obtaining mathematical expressions to characterize the QoS properties such as latency bounds of the different schedulers. We can find in the literature such expression for the well-known scheduling algorithms that are the base of the credit aware versions that we present in this paper. However, those expressions were obtained with studies that usually considered lossy networks. Therefore, a full study on the effect of the link-level flow control mechanism over the analytical models proposed for well-known scheduling algorithms should be performed. Furthermore, completely new expressions should be obtained for the DTable scheduler, because, as far as we know, there is no formal study on the properties of table-based schedulers.

REFERENCES

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

[2] F. J. Alfaro, J. L. Sánchez, and J. Duato. A new proposal to fill in the InfiniBand arbitration tables. In *IEEE Int. Conference on Parallel Processing (ICPP)*, pages 133 – 140, October 2003.

[3] F. J. Alfaro, J. L. Sánchez, and J. Duato. QoS in InfiniBand subnetworks. *IEEE Transactions on Parallel and Distributed Systems*, 15(9):810–823, September 2004.

[4] T. Anderson, S. Owicki, J. Saxe, and C. Thacker. High-speed switch scheduling for local-area networks. *ACM Transactions on Computer Systems*, 11(4):319–352, November 1993.

[5] L. Cheng, N. Muralimanohar, K. Ramani, R. Balasubramonian, and J. B. Carter. Interconnect-aware coherence protocols for chip multiprocessors. In *ISCA*, pages 339–351. IEEE Computer Society, 2006.

[6] N. Chrysos and M. Katevenis. Multiple priorities in a two-lane buffered crossbar. In *Proceedings of the IEEE Globecom 2004 Conference*, November 2004.

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

[8] J. Duato, S. Yalamanchili, and N. Lionel. *Interconnection networks. An engineering approach*. Morgan Kaufmann Publishers Inc., 2002.

[9] M. A. El-Gendy, A. Bose, and K. G. Shin. Evolution of the internet QoS and support for soft real-time applications. *Proceedings of the IEEE*, 91(7):1086–1104, 2003.

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

[11] G. Horn and T. Sdring. SH: A simple distributed bandwidth broker for source-routed loss-less networks. In *Computer, Networks and Information Security*. IASTED, 2005.

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

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

[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] M. Katevenis, G. Passas, D. Simos, I Papaefstathiou, and N. Chrysos. Variable packet size buffered crossbar (cicq) switches. In *IEEE Int. Conference on Communications (ICC 2004)*, pages 20–24, Paris, France, June 2004.

[16] 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*, October 1991.

[17] P. Krishna, N. Patel, A. Charny, and R. Simcoe. On the speedup required for work-conserving crossbar switches. *IEEE J. Sel. Areas in Communications*, 17(6):1057–1066, June 1999.

[18] R. Martínez, F. J. Alfaro, and J.L. Sánchez. Decoupling the bandwidth and latency bounding for table-based schedulers. *International Conference on Parallel Processing (ICPP)*, August 2006.

[19] R. Martínez, F. J. Alfaro, and J.L. Sánchez. Evaluating several implementations for the AS minimum bandwidth egress link scheduler. *15th International Conference on Computer Communications and Networks (ICCCN 2006)*, October 2006.

[20] R. Martínez, F. J. Alfaro, and J.L. Sánchez. Implementing the Advanced Switching minimum bandwidth egress link scheduler. *IEEE International Symposium on Network Computing and Applications (IEEE NCA06)*, July 2006.

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

[22] R. Martínez, F. J. Alfaro, J.L. Sánchez, and T. Skeie. A first approach to provide QoS in Advanced Switching. *International Conference on High Performance Computing (HiPC). Goa, India*, 2005.

[23] 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.

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

[25] 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.

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

[27] PCI SIG. *PCI Express base architecture specification. Revision 1.0a*, April 2003.

[28] J. Pelissier. Providing Quality of Service over Infiniband architecture fabrics. In *Proceedings of the 8th Symposium on Hot Interconnects*, August 2000.

[29] G. Pfister and A. Norton. Hot spot contention and combining in multistage interconnection networks. *IEEE Transactions on Computers*, C-34, 10:943–948, 1985.

[30] S.A. Reinemo, F.O. Sem-Jacobsen, T. Skeie, and O. Lysne. Admission control for diffserv based Quality of Service in cut-through networks. In *Proceedings of the 10th Int. Conference on High Performance Computing*, December 2003.

[31] J. Rexford, A. G. Greenberg, and F. Bonomi. Hardware-efficient fair queueing architectures for high-speed networks. In *INFOCOM (2)*, pages 638–646, 1996.

[32] J. W. Roberts. Virtual spacing for flexible traffic control. *International Journal of Communication Systems*, 7:307–318, 1994.

[33] R. Seifert. *Gigabit Ethernet: Technology and Applications for High-Speed LANs*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

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

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

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

[37] D. Tutsch and M. Brenner. MINSimulate - a multistage interconnection network simulator. In *17th European Simulation Multiconference: Foundations for Successful Modelling and Simulation (ESM'03)*, pages 211–216, 2003.

[38] A. Tyagi, J. K. Muppala, and H. de Meer. VoIP support on differentiated services using expedited forwarding. In *IEEE International Performance, Computing, and Communications Conference (IPCCC)*, February 2000.

[39] Y. Wang and Q. Zhu. Error control and concealment for video communication: A review. *Proceedings of the IEEE*, 86(5):974–997, May 1998.

Raúl Martínez Raúl Martínez received the MS degree in computer science from the University of Castilla-La Mancha in 2003 and the PhD degree from the University of Castilla-La Mancha in 2007. He is currently a researcher in the Intel Barcelona Research Center. His research interests include high-performance local area networks, QoS, design of high-performance routers, and computer architecture.

Francisco J. Alfaro Francisco J. Alfaro received the MS degree in computer science from the University of Murcia in 1995 and the PhD degree from the University of Castilla-La Mancha in 2003. He is currently an assistant professor of computer architecture and technology in the Computer Systems Department at the Castilla-La Mancha University. His research interests include high-performance local area networks, QoS, design of high-performance routers, and design of on-chip interconnection networks for multicore systems.

Francisco J. Alfaro José L. Sánchez received the PhD degree from the Technical University of Valencia, Spain, in 1998. Since November 1986 he is a member of the Computer Systems Department (formerly Computer Science Department) at the University of Castilla-La Mancha. He is currently an associate professor of computer architecture and technology. His research interests include multicomputer systems, QoS in high-speed networks, interconnection networks, parallel algorithms and simulation.