Providing Quality of Service over Advanced Switching*

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

Abstract

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

The provision of Quality of Service (QoS) in computing and communication environments is currently the focus of much discussion and research in industry and academia. AS provides some mechanisms, which correctly used permit us to provide QoS. In this paper we examine these mechanisms and show how to provide QoS based on bandwidth and latency requirements.

Furthermore, we propose a new algorithm based on the Self-Clocked Weighted Fair Queuing (SCFQ) algorithm, which we call SCFQ Credit Aware (SCFQ-CA), as an implementation of the AS minimum bandwidth egress link scheduler. Finally, we show that the AS table-based scheduler does not work properly with variable packet sizes, and we propose a modification of the table scheduler, based on the Deficit Table (DTable) scheduler, to solve this drawback.

Keywords: *Quality of Service (QoS), Advanced Switching, scheduling, application requirements, interconnection networks, aggregated flows, performance evaluation.*

1. Introduction

The PCI bus has served industry well for the last ten 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 [16] eliminates the legacy shared bus-based architecture of PCI and introduces an improved and dedicated point-to-point interconnect. Advanced Switching (AS) [1] is a new open-standard fabric-interconnect technology for communications, storage, and embedded environments based on PCI Express.

Multiservice packet networks are required to carry not only traffic of different applications, such as e-mail or file transfer, which does not require pre-specified service guarantees, but also other applications that require different performance guarantees, like real-time video or telephony. The provision of QoS in computing and communication environments is currently the focus of much discussion and research in industry and academia. 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 to provide a different treatment to the different traffic classes. In [12], we examined these mechanisms and showed a first approach to provide QoS in AS. In this paper, we expand largely the basic ideas presented there and propose several methods of using the AS mechanisms to provide applications with QoS based on bandwidth and latency requirements.

A key component for networks with QoS support is the egress link 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).

AS does not specify an algorithm or implentation for the MinBW scheduler, but some charateristics that it must respect. In this paper, we propose a new algorithm based on the Self-Clocked Weighted Fair Queuing (SCFQ) algorithm [6] that fulfills all the properties that the AS MinBW scheduler must have, including the interaction with the AS credit-based flow control. We have called this algorithm *SCFQ Credit Aware* (SCFQ-CA).

Moreover, we show that the AS table-based scheduler does not work properly with variable packet sizes. In [11] we proposed a new table-based scheduler that is able to deal properly with variable packet sizes. We called this algorithm *Deficit Table* scheduler or just DTable scheduler. In this paper we show how to adapt the DTable scheduler to

*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.

implement the AS table scheduler. The resulting scheduling mechanism requires simple hardware modifications of the original AS table scheduler. However, it does not require to modify the interface provided in the AS specification for configuring the table scheduler.

The structure of the paper is as follows: Section 2 presents a summary of the general aspects in the AS specification including the most important mechanisms that AS provides to support QoS. In Section 3, we propose our implementation of the MinBW scheduler. In Section 4, we show the problems of the table scheduler with variable packet sizes and show how to modify this scheduler to solve those problems. In Section 5, we present our proposal to use the AS mechanisms to provide applications with QoS. Details on the experimental platform and the performance evaluation are presented in Section 6. Finally, some conclusions are given and future work is proposed.

2. Advanced Switching

Advanced Switching (AS) is built on the same physical and link layers as PCI Express. Moreover, it includes an optimized transaction layer to enable essential communication capabilities, including protocol encapsulation, enhanced fail-over, high availability, and congestion and system management.

The physical layer consists in a dual-simplex channel, which is implemented as a transmit pair and a receive pair. A data clock is embedded using the 8b/10b encoding scheme, with an initial frequency of 2.5 Gb/s, but the bandwidth of a link may be linearly scaled by adding signal pairs to form multiple lanes.

The link layer is responsible for data integrity and adds a sequence number and a CRC to the transaction layer. 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.

AS supports unicast and multicast traffic. For unicast traffic the AS transaction layer provides source-based routing. The maximum packet size of an AS packet is 2176 bytes. An AS fabric permits us to employ VCs, egress link scheduling, and an admission control mechanism to differentiate between traffic flows.

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 each VC has its own credit count for the credit-based flow control.

AS defines two schedulers to resolve between the up to twenty VCs competing for bandwidth 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 table scheduler provides an implementation of the Weighted Round Robin (WRR) algorithm [10]. The VC arbitration table is a register array with fixed-size entries of 8 bits. Each table entry, which contains a VC identifier value, corresponds to a slot of a WRR arbitration period. 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 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* [1].

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.

3. Implementation of the MinBW scheduler

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

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

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

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

The AS specification states that variants of Weighted Fair Queuing (WFQ) [5] such as SCFQ [6], and variants of Weighted Round Robin (WRR) [10] such as Deficit Round Robin (DRR) [19] exhibit the desired properties of the MinBW scheduler. In order to provide QoS, we have discarded the variants of WRR because they generally produce worse latency and fairness properties than variants of WFQ [21]. Therefore, we have chosen the SCFQ algorithm to implement the MinBW for performance evaluation.

The SCFQ algorithm is a variant of the WFQ algorithm which has a lower computational complexity. It defines fair queueing in a self-contained manner and avoids using a hypothetical queueing system as reference to determine the fair order of services. Instead, it uses a virtual time function which depends on the progress of the work in the actual packet-based queueing system. Each packet that arrives at the egress link is stamped with a service tag. The packets are then transmitted in an increasing order of timestamp.

Therefore, when a packet arrives, the SCFQ algorithm uses the service tag of the packet currently in service as the virtual time to calculate the new packet tag. Let F_i^k be the service tag of the k^{th} packet from flow i,

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

where L_i^k is the length of the k^{th} packet and $F_{current}$ is the service tag of the packet currently in service. 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.

The SCFQ Credit Aware (SCFQ-CA) algorithm that we propose works in the following way:

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

This new algorithm fulfills all the properties that the AS MinBW scheduler must have. Moreover, this scheduling algorithm is actually appropriate not only for being used in AS, but also for being used in any network that employs a link level flow control.

4. Modifying the table to work with 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. Figure 1 shows the performance of various table schedulers when there are four traffic flows in the network, all of them with the same data rate but a different packet size, and with all the flows having the same number of assigned table entries (the same bandwidth reservation). The simulated architecture is the same that it has been used for the performance evaluation in Section 6. As it is shown in Figure 1(a), when using the AS table scheduler, due to the fact that each flow has a different packet size, the flows obtain a very different bandwidth. The reason of this is that, when a table entry is selected, a packet from the VC indicated in that entry is transmitted regardless of the packet size. Therefore, although the same number of packets from each flow is going to be transmitted, the amount of information is not going to be the same.

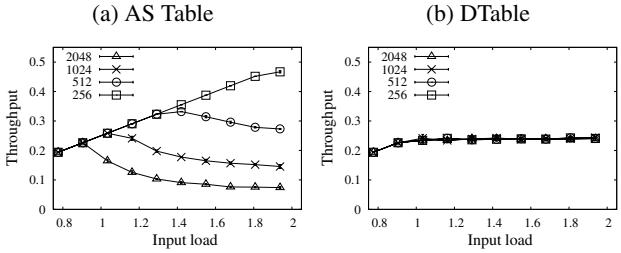

Figure 1. AS Table and DTable performance with different packet sizes.

In [11] we proposed a new table-based scheduler that works properly with variable packet sizes. We called this new algorithm Deficit Table scheduler, or just DTable scheduler, because it is a mix between the table scheduler and the DRR algorithm. The DRR algorithm 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 DTable algorithm works like the DRR algorithm, but the amount of information assigned to each VC (the quantum) in the DRR algorithm is distributed among the table entries assigned to that VC. Each table entry has associated a VC identifier and an *entry weight*. The entry weight is the amount of information, in flow control credits, that is allowed to be transmitted by each entry. Moreover, the DTable scheduler assigns each VC with a deficit counter that works in the same way as in the DRR scheduler. The interaction with the AS credit-based flow control is also taken into account. Specifically, this scheduler works in the following way:

- A VC is active when it has at least one packet and there are enough credits to transmit the packet that is at the head of the VC queue.
- The table entries are cycled through until an entry that has a VC identifier from an active VC is found. We will call this VC the *selected VC*.
- 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 VC and the entry weight.
- Packets belonging to the selected VC are transmitted. The accumulated weight is reduced after sending each packet in an amount equal to the number of flow control credits required by the transmitted packet.
- The next table entry is selected when any of the following conditions occurs:
  - There are no more packets from the selected VC or there are not enough flow control credits for transmitting the packet that is at the head of the VC queue. In that case, the VC becomes inactive, and the deficit value for that VC becomes zero.
  - The accumulated weight is less than the size of the packet that is at the head of the queue. The deficit value becomes equal to the accumulated weight in that instant.

In order to apply to AS our proposed DTable scheduler it must be taken into account that the AS specification only consider the VC identifier assigned to each table entry. In order to be able to also assign a weight to each table entry the AS specification would require to be modified. Therefore, we propose to use a fix value for all the entries. This weight is the Maximum Transfer Unit (MTU). This value is known by the network elements and, besides, this is the smallest value that ensures that there is never a need to cycle through the entire table several times in order to gather enough weight for the transmission of a single packet. In this way, when an active table entry is selected at least one packet will be transmitted. This is also taken into account in the definition of the DRR algorithm [19]. This modified version of the table scheduler would only require quite simple hardware modifications. 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 16 unicast VCs this means $16 * 6 = 102$ bits per egress link.

The main advantage of the table (and DTable) scheduler over those schedulers that involve packet tag sorting like the WFQ-based schemes (including the SCFQ scheduler) is its simpler implementation [4]. Moreover, the table scheduler allows 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. As we will see in Section 5.2, we will use this property to provide different latency requirements to the AS VCs. Note that, with this algorithm each table entry allows us to transmit an amount of information up to twice the MTU minus one credit. This must be taken into account when the table is configured to provide a flow with latency requirements. Note also that, if all the table entries devoted to the same VCs are assigned consecutively, the behavior of this scheduler would be the same as the DRR algorithm.

The resulting application of the DTable algorithm to AS is a modification of the AS table scheduler that works properly with variable packet sizes, as can be seen in Figure 1(b). Moreover, the modification proposed does not change the structure of the AS arbitration table.

5. Providing QoS over AS

As was stated in Section 2, 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. First of all, 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. To define this set of SCs we propose a traffic classification based on two network parameters: Bandwidth and latency. In this way, this classification is similar to the one presented by Pelissier [17]. 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 and/or maximum latency requirements. Various QoS SCs can be defined with different bandwidth and latency requirements. This category can be divided into two groups:
  - Traffic which requires a given minimum bandwidth and must be delivered within a given deadline 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. An example could be a non-interactive playback of a video clip.
- 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.

AS supports the use of VCs, which provide a mean of supporting multiple independent logical data flows over a given common physical channel. Conceptually, this involves multiplexing various data flows onto a single physical link. Moreover, in AS, the egress link scheduling is performed at the VC level. This means that all the traffic that is transmitted through the same VC is considered as a single flow.

Therefore, if there are sufficient 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 has assigned. Note that the maximum number of SCs that we can define is 16, that is the maximum number of unicast VCs supported by AS.

In order to provide QoS guarantees, an Admission Control (AC) mechanism must be used. Without an AC it is only possible to obtain a scheme of priorities where some SCs would have a higher priority than others, but no guarantee could be given. The AC mechanism would allow a new connection to be established if it cannot create persistent congestion in its VC. In any case, no AC would be implemented for network control and best-effort traffic.

AS specification just cites admission control 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 [14], which is an AC scheme that makes the decisions based on the bandwidth that is expected to consume the new flow. The flows must be connection oriented, which means that are going to use the same path during all their life. 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 switches and the new connection is accepted. If any of the links have not enough bandwidth to accommodate the new flow the connection is rejected.

The bandwidth broker can be implemented in a centralized manner, which has all the information in a single host, or in a distributed manner, like in [7]. 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.

Finally, the schedulers must be properly configured at the different network elements to provide the different SCs with a differentiated treatment. Therefore, 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.

5.1. Providing QoS requirements with 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 [15] analyzed the performance of a queueing network with fair queueing 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,

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

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.

5.2. Providing QoS requirements with the Table scheduler

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 two consecutive table entries devoted to that VC must be fixed. By fixing this separation, it is possible to control the maximum latency of a network element crossing, and therefore, given a maximum number of hops, the maximum end-to-end latency. In order to choose the maximum separation, the maximum time must be studied that a packet can spend crossing a network element as well as the time it takes to be transmitted to the next element once it has been chosen by the scheduler [3]. Note that the control SC does not have maximum priority when using this scheduler, so we will consider it as any other SC with high latency requirements.

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. We have presented a proposal that partially solves this bounding in [11]. 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.

This way of configuring the table scheduler is equally valid for the DTable scheduler and for the original AS table scheduler, if a fixed packet size is being used. The only thing that must be taken into account to provide latency requirements is the maximum amount of information that a table entry allows to be transmitted.

5.3. Other considerations

There are two possible ways of configuring the schedulers. The first possibility is to configure the schedulers in advance, defining a set of SCs with a different minimum bandwidth and maximum latency reservation [18]. This distribution would be made taking into account the expected use of each SC. 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]. This allows more flexibility and a more accurate use of the resources. Note that the second possibility is only feasible when an AC mechanism is used.

6. Performance Evaluation

In [12], we evaluated a first and simpler aproximation of our proposals using fixed packet sizes, comparing the performance of the MinBW and the original table schedulers. In this paper, we evaluate thoroughly our proposals with variable packet sizes, comparing the performance of the MinBW scheduler implemented using our SCFQ-CA algorithm and the DTable scheduler.

6.1. Simulated architecture

We have used a perfect-shuffle Bidirectional Multi-stage Interconnection Network (BMIN) 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. 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. However, 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

Table 1. 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.

Table 2. Injected traffic and scheduler configuration.

	Injected traffic				Table C.		MinBW C.
SC	**Min.**	**Max.**	**Traffic pattern**	**Packet size**	**# Entr.**	**Dist.**	**Weight**
NC	0.01	0.01	self-similar (Bursts60)	up to 64B	16	4	-
VO	0.1875	0.1875	64 Kb/s CBR connects.	128B	16	4	0.25
VI	0.1875	0.1875	750 Kb/s MPEG-4 traces	up to 2176B	12	6	0.1875
CL	0.1875	0.1875	750 Kb/s CBR connects.	2176B	12	(6)	0.1875
EE	0	0.1425	self-similar (Bursts60)	up to 2176B	5	(16)	0.078125
BE	0	0.1425	self-similar (Bursts60)	up to 2176B	2	(32)	0.03125
BK	0	0.1425	self-similar (Bursts60)	up to 2176B	1	(64)	0.015625
Total	0.5725	1			64		0.75

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, so we have implemented a round robin scheduler.

The time that a packet header takes to cross the switch without any load is 145 ns, which is based on the unloaded cut-through latency of the AS StarGen's *Merlin* switch [20].

6.2. Traffic model

The IEEE standard 802.1D-2004 [8] defines 7 traffic types at the Annex G, which are appropriate for this study. We will consider each traffic type as a SC. Table 1 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 is assigned to the FMC.

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 gradually increase the amount of best-effort traffic (EE, BE, and BK), starting at 0.62 input load. The amount of QoS traffic to be injected is the maximum allowed by the AC. Table 2 shows the proportion of traffic of each SC that each node injects regarding the link bandwidth.

The destination pattern is uniform in order to fully load the network. The packets are generated according to different distributions, as can be seen in Table 2. VO, VI, and CL SCs are composed of point-to-point connections of the given bandwidth. VO and CL SCs are generated following a Constant Bit Rate (CBR) distribution. In the case of VI SC, a video trace is 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. The traffic of the NC, EE, BE, and BK SCs is generated according to a Bursts60 distribution [13]. 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 [9]. In this way, many small size packets are generated, with an occasional large size packet. The periods between bursts are modeled with a Poisson distribution. The Bursts60 pattern models worst-case real traffic scenarios.

6.3. Scheduler configuration

The configuration of the DTable and the MinBW schedulers is shown in Table 2. In order to compare the two schedulers we have assigned the same amount of bandwidth to each SC in both cases.

We want to reserve 25% of link bandwidth to best-effort traffic, but we have only assigned best-effort SCs a minimum bandwidth (12.5%) to establish the preference between them. 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. For both schedulers, 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 [15]. 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.

6.4. 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, and the maximum packet latency 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 the maximum latency of all the flows. Note that the maximum latency shows the behavior of the flow with the worst performance. Note also that in the case of the VI SC this means the worst latency performance of a video frame (packets from the same frame have the same generation time).

Figure 2 gives a general overview of the performance of both schedulers. Regarding the throughput performance, the figure shows that the NC and the QoS SCs obtain all the bandwidth that they inject. However, when the network load is high (around 85%), 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. Note that the two schedulers fulfill the maximum latency requirements [8], 10 ms for voice traffic and 100 ms for video traffic. 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.

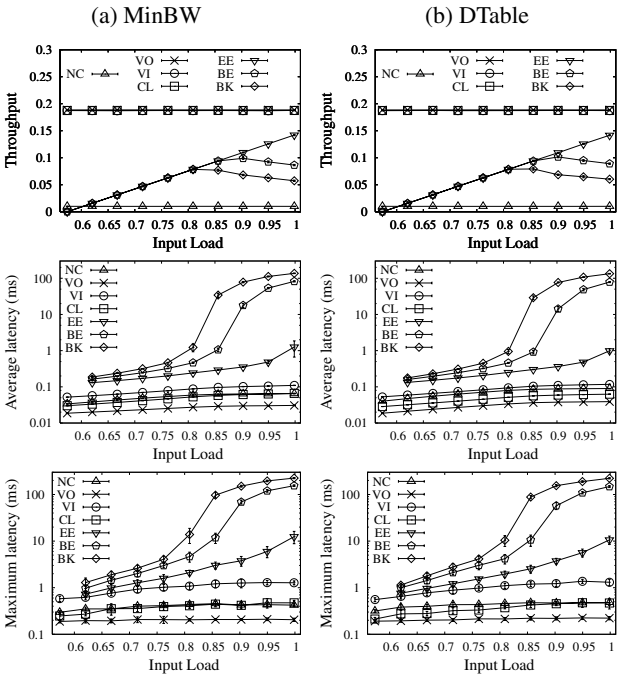

Figure 2. Throughput and latency results for the MinBW and DTable schedulers.

Figure 2 also shows some interesting results about the effect of the scheduler configuration and the traffic pattern in the latency performance of the NC and QoS SCs. If the different SCs would have been emulated using the same pattern traffic, attending to the configuration of both schedulers, we could have expected to obtain a better latency for the NC SC than the rest of SCs, or a similar latency for VI and CL SCs. However, we have obtained a worse latency for the NC SC than for the VO SC, and a worse latency for the VI SC than for the CL SC. The reason of this is that the pattern distributions that have been used to emulate the control and video traffic is bursty. The bursts make that the average and maximum latency grow because all the packets from the same burst are injected at the same time in the network interface. Thus, each packet must wait the previous packets before being injected in the egress link. Note that in the case of the VO and CL SCs, which have the same CBR distribution, the VO SC obtains a better latency than the CL SC. The reason is that the VO SC has been assigned more bandwidth than it theoretically requires (in the MinBW scheduler case) or the table entries of the VO SC have a smaller maximum distance between them (in the table scheduler case).

Figures 3, 4, and 5 show a more detailed comparison between both schedulers. Regarding the control and QoS SCs,

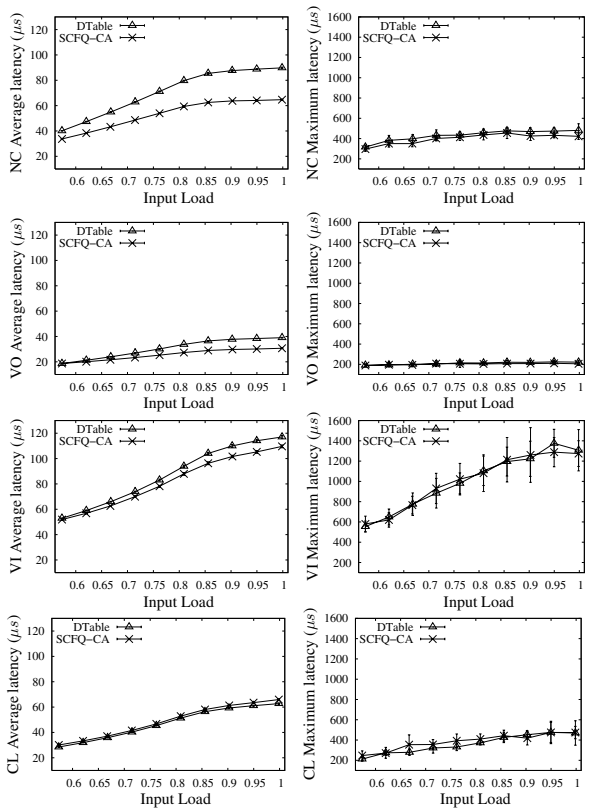

Figure 3. Control and QoS SCs performance comparison between the MinBW and the DTable schedulers.

Figure 3 shows that the MinBW scheduler provides a better average latency for the NC, VO, and VI SCs. The CL SC obtains a slightly better average latency when using the DTable scheduler. Regarding the maximum latency performance, Figure 3 shows that the MinBW scheduler provides a slightly better maximum latency for the NC and VO SCs than the DTable scheduler. However, the maximum latency for the VI and CL SCs are quite similar in both cases.

Regarding the best-effort SCs, Figure 4 shows that both schedulers provide a quite similar latency performance for the EE SC. However, the DTable scheduler provides a better latency for the BE and BK SCs when the input load is high.

Figure 5 shows the percentage of improvement of the MinBW scheduler over the DTable scheduler for three different input load levels. This improvement has been calculated using the average values, and thus the confidence intervals have not been taken into account. However, some general conclusions can be extracted from this figure: The MinBW scheduler provides a better latency performance for the NC, VO, and VI SCs. This improvement is higher for the average latency than for the maximum latency and it affects specially the NC SC. This is due to the fact that this SC has the maximum priority with the MinBW scheduler, which is not the case when using the DTable scheduler. On

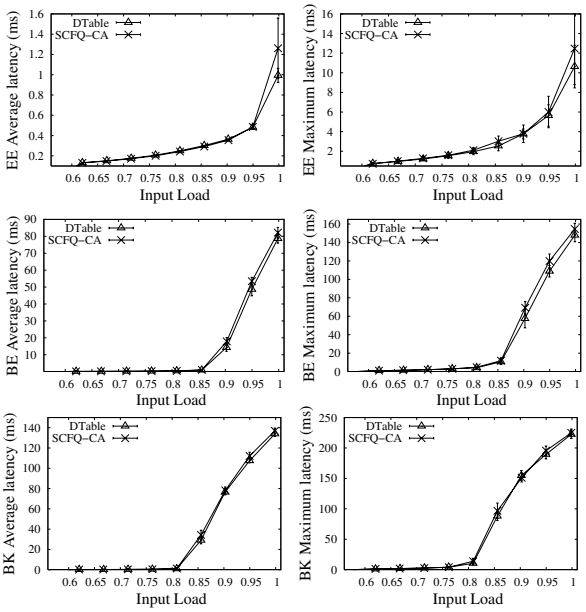

Figure 4. Best-effort SCs performance comparison between the MinBW and the DTable schedulers.

the other hand, the DTable scheduler offers a better latency performance for the CL and best-effort SCs when the load is high, because this scheduler does not provide the NC SC with absolute priority, and thus it treats better the other SCs.

Summing up, the simulation results show that with a correct configuration, both schedulers are able to provide the SCs with their requirements: the network control SC obtains a good latency; the SCs with bandwidth requirements obtain the amount that they need; the SCs with latency requirements do not exceed the maximum allowed; finally, the best-effort SCs obtain a different bandwidth and latency performance in accordance with their different priority. The latency performance of the different SCs depends on the scheduler configuration, but also on the pattern traffic. The MinBW scheduler provides a better average latency and a slightly better maximum latency than the DTable scheduler for those SCs with latency requirements. However, the DTable scheduler also has a correct performance and it has a lower implementation complexity (see Section 4).

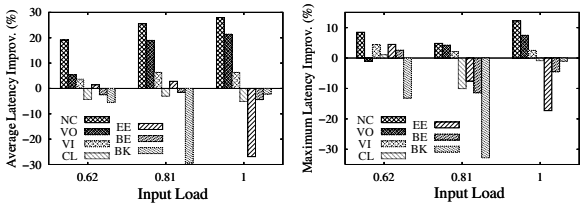

Figure 5. Latency improvement of the MinBW scheduler over the DTable scheduler.

7. Conclusions

In this paper, we have proposed several methods of using the AS mechanisms to provide applications with QoS. Specifically, we have shown how to provide QoS based on bandwidth and latency requirements. For that purpose, we have proposed a traffic classification to segregate the traffic into different SCs based on these requirements. These SCs must be processed appropriately by the egress link scheduler to obtain their QoS requirements. AS defines two egress link schedulers: The table scheduler and the MinBW scheduler. This paper shows how to configure each of them.

We have proposed the SCFQ-CA algorithm as a specific implementation of the MinBW scheduler adapting the original SCFQ algorithm taking into account the AS credit-based flow control. We have proposed to use the DTable scheduler with the same weight for all the table entries as an implementation of the table scheduler. This modified version of the table scheduler works properly with variable packet sizes and fits the AS specification.

The simulation results show that both schedulers, the MinBW (implemented with the SCFQ-CA) and the DTable scheduler, are able to provide the SCs with their requirements. The simulation results also show a better latency performance when using the MinBW, but the advantage of the DTable scheduler over the SCFQ-CA algorithm and other WFQ variant is its simpler implementation.

Acknowledgments

The authors would like to thank Tor Skeie and Sven-Arne Reinemo from the Simula Resarch Laboratory of Norway, their useful comments which have helped to improve the quality of this paper.

References

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

[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, Oct. 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, Sept. 2004.

[4] H. M. Chaskar and U. Madhow. Fair scheduling with tunable latency: A round-robin approach. *IEEE/ACM Transactions on Networking*, 11(4):592–601, 2003.

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

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

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

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

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

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

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

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

[13] M. K. N. Chrysos. Multiple priorities in a two-lane buffered crossbar. In *Proceedings of the IEEE Globecom 2004 Conference*, Nov. 2004.

[14] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the internet. Internet Request for Comment RFC 2638, Internet Engineering Task Force, July 1999.

[15] 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.

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

[17] J. Pelissier. Providing quality of service over Infiniband architecture fabrics. In *Proceedings of the 8th Symposium on Hot Interconnects*, Aug. 2000.

[18] S. Reinemo, F. 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*, Dec. 2003.

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

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

[21] D. Stiliadis and A. Varma. Latency-rate servers: a general model for analysis of traffic scheduling algorithms. *IEEE/ACM Transactions on Networking*, 6(5):611–624, 1998.