Studying several proposals for the adaptation of the DTable scheduler to Advanced Switching

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

Abstract

Advanced Switching (AS) is a new fabric-interconnect technology that further enhances the capabilities of PCI Express. On the other hand, the provision of Quality of Service (QoS) in computing and communication environments is currently the focus of much discussion and research in industry and academia.

A key component for networks with QoS support is the egress link scheduling algorithm. AS defines two egress link schedulers: The virtual channel arbitration table scheduler and the minimum bandwidth egress link scheduler. The table-based scheduler is simple to implement and can offer good latency bounds with a fixed packet size. However, it does not work properly with variable packet sizes and faces the problem of bounding the bandwidth and latency assignments.

We have proposed a new table-based scheduler, which we have called Deficit Table (DTable) scheduler, that works properly with variable packet sizes. Moreover, we have proposed a methodology to configure this table-based scheduler that partially decouples the bandwidth and latency assignments.

In this paper we propose several possible modifications to the original AS table scheduler in order to implement the DTable scheduler. Moreover, we review our proposals to provide bandwidth and latency requirements over AS using the DTable scheduler. Finally, we evaluate our proposals in a multimedia scenario and compare the performance of the DTable scheduler with the performance of the minimum bandwidth scheduler.

1 Introduction

Advanced Switching (AS) [1] is a new high performance interconnection technology based on PCI Express [15]. The need for Advanced Switching (AS) essentially comes because computing and communication platforms begin to converge by exhibiting increasing overlap in terms of the functions they serve. While PCI Express, as the next generation of the PCI bus, is clearly the interconnect of choice for the computing industry, a common interconnect with the communications industry seems logical and necessary, in order to keep development cost down, performance up and to reduce time-to-market.

AS is an extrapolation of PCI Express, borrowing the physical and link layers from PCI Express, but diverging at the transaction layer and in the marketplaces it intends to serve. Whereas PCI Express has already begun to reshape a new generation of PCs and traditional servers, AS is intended to proliferate in multiprocessor, peer to peer systems in the communications, storage, networking, servers and embedded platform environments. Together, PCI Express and AS have the potential for building the next generation interconnects [11].

The provision of Quality of Service (QoS) in the environment where AS is foreseen to be used will be very important. AS networks will be required to carry not only traffic of applications such as e-mail or file transfer, which does not require pre-specified service guarantees, but also traffic of other applications that require different performance guarantees, like real-time video or telephony [12]. The best-effort service model, though suitable for the first type of applications, is not so for applications of the other type [14].

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. An ideal scheduling algorithm implemented in a high performance network with QoS support should satisfy two main properties: good end-to-end delay and simplicity. AS defines two egress link

schedulers: The virtual channel arbitration table scheduler and the Minimum Bandwidth egress link scheduler (MinBW). The table-based scheduler is simple to implement and can offer good latency bounds with a fixed packet size. However, it does not work properly with variable packet sizes and faces the problem of bounding the bandwidth and latency assignments [10].

In [6] we proposed a new table-based scheduler, which we called Deficit Table (DTable), which works properly with variable packet sizes. We also proposed a methodology to configure this scheduler that allows us to decouple, at least partially, the bandwidth and latency assignments. Moreover, in [8] we proposed a method to increase the flexibility of the decoupling methodology without decreasing the bandwidth nor the latency performance. This method consists in using different specific Maximum Transfer Units (MTUs) for the different flows[1]. However, we also showed that the specific flow MTU must be assigned taking into account the characteristics of the traffic flow. A too small specific MTU may decrease performance of the flow. Note that, with a smaller MTU, the same amount of information may require more packets, and thus, we have a bandwidth overhead due to the packet headers and a higher latency due to the need of proccessing a higher number of packets.

In [10, 9] we examined the AS mechanisms intended for providing QoS and showed how to provide QoS based on bandwidth and latency requirements. Specifically, in [9] we showed how to use a limited version of the DTable scheduler to implement the AS table scheduler in order to support variable packet sizes. The original DTable scheduler considers a different weight per table entry, however, in this version we used the same weight to all the table entries. The resulting scheduling mechanism does not require to modify the interface provided in the AS specification for configuring the table scheduler and only requires simple hardware modifications of the original AS table scheduler. However, if we want to take advantage of the decoupling configuration methodology, we need to be able to assign different weights to the table entries.

In this paper we proposed three different ways of implementing a full version of the DTable scheduler to substitute the original AS table scheduler, but modifying as little as possible the AS specification. Moreover, we review the methods we have proposed to provide QoS over AS, now employing a full version of the DTable scheduler. We also take into account the possibility of using several MTUs. Finally, we evaluate the performance of our proposals by simulation. In order to do this, we consider a multimedia scenario with different types of traffic. Moreover, we are going to compare the performance of the DTable scheduler with the performance of the MinBW scheduler. The AS specification does not offer a particular algorithm to implement the MinBW scheduler, but only the properties it must respect. In [7] we proposed the Self-Clocked Weighted Fair Queuing Credit Aware (SCFQ-CA) and the Deficit Round Robin Credit Aware (DRR-CA) scheduling algorithms as implementations for the MinBW scheduler. Therefore, we are going to compare the DTable scheduler with these two implementations of the MinBW scheduler.

The structure of the paper is as follows: In Section 2, we review the DTable scheduler and our methodology to decouple the bandwidth and latency assignments. Section 3 presents a summary of the general aspects in the AS specification including the most important mechanisms that AS provides to support QoS. In Section 4 we propose how to provide QoS requirements over AS using a full implementation of the DTable scheduler. In Section 5, we propose how to fully implement the DTable scheduler in AS. Details on the experimental platform and the performance evaluation are presented in Section 6. Finally, some conclusions are given.

2 The Deficit Table scheduler

In [6] we proposed a new table-based scheduling algorithm that works properly with variable packet sizes. We called this algorithm Deficit Table scheduler, or just DTable scheduler, because it is a mix between the already proposed table-based schedulers and the Deficit Round Robin (DRR) algorithm [17]. The scheduler works in a similar way than the DRR algorithm but instead of serving packets of a flow in a single visit per frame, the service is distributed throughout the entire frame.

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 it stores at least one packet and the link-level flow control, if exists,

[1] In this paper we will use the term *flow* to refer both to a single flow or to an aggregated of several flows with similar characteristics.

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 as many packets from the active flow as the accumulated weight allows. When a packet is transmitted, the accumulated weight is reduced by the packet size.

The next active table entry is selected if the flow becomes inactive or the accumulated weight becomes smaller than the size of the packet at the head of the queue. In the first case, the remaining accumulated weight is discarded and the deficit counter is set to zero. In the second case, the unused accumulated weight is saved in the deficit counter, representing the amount of weight that the scheduler owes the queue. The weights are usually expressed in flow control credits.

In [6] we have also proposed a methodology to configure the DTable scheduler to decouple, at least partially, the bounding between the bandwidth and latency assignments. With this methodology we set the maximum distance between any consecutive pair of entries assigned to a flow depending on its latency requirement. By fixing this separation, it is possible to control the maximum latency of a network element crossing. 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. Therefore, given a maximum number of hops, we can control the maximum end-to-end latency. Note that setting the maximum distance between any consecutive pair of entries assigned to a flow entails assigning that flow a certain number of entries.

Moreover, we can assign the flows with a bandwidth that depends on the total weight assigned to the flow entries. The proportion of the egress link bandwidth that can be actually assigned to a flow depends not only on the number of table entries assigned, but also on two table configuration parameters. We have called these parameters w and k.

The w parameter determines the maximum weight M that can be assigned to a single table entry in function of the General MTU of the network ($GMTU$): $M = GMTU \times w$. The minimum weight that a table entry can have associated should ensure 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 [17]. Therefore, in [6] we set this minimum value to the GMTU. However, in [8] we proposed to use different MTUs for the different flows. This means that each flow has a specific MTU equal or lower than the GMTU. Therefore, we can assign a table entry a minimum weight equal to its flow's specific MTU instead of the GMTU.

The k parameter determines the total weight that can be distributed between all the table entries. We call this value the *bandwidth pool*: $pool = N \times GMTU \times k$. Note that the maximum value for the bandwidth pool is $N \times M$. The total number of weight units from the bandwidth pool that the table entries of a flow have assigned fixes the bandwidth that the flow has actually assigned.

Therefore, supposing an arbitration table with N entries, the minimum bandwidth $min\phi_i$ and the maximum bandwidth $max\phi_i$ that can be assigned to the i^{th} flow depends on the number of table entries n_i that it has assigned, the w and k parameters, and the proportion between its specific MTU (MTU_i) and the $GMTU$:

$$min\phi_i = \frac{n_i \times MTU_i}{pool} = \frac{n_i \times MTU_i}{N \times GMTU \times k} = \frac{n_i}{N} \times \frac{MTU_i}{GMTU} \times \frac{1}{k} \quad (1)$$

$$max\phi_i = \frac{n_i \times M}{pool} = \frac{n_i \times GMTU \times w}{N \times GMTU \times k} = \frac{n_i}{N} \times \frac{w}{k} \quad (2)$$

Note that varying the w and k parameters affects the minimum and maximum bandwidth that can be assigned to all the flows. However, assigning a specific MTU to a flow only affects the minimum bandwidth of that flow.

When choosing the value of these parameters some considerations must be made. Note that the objective for this methodology is to decrease the minimum bandwidth and to increase the maximum bandwidth that can be assigned to a flow. In order to be able to assign a small amount of bandwidth the k parameter must be high. However, the highest k is, the smaller the maximum bandwidth that can be assigned, and thus, the flexibility to assign the bandwidth decreases. We can solve this by increasing the value of w. However, increasing the value of the w parameter increases the latency of the flows, because each entry is allowing more information to be transmitted, and thus, the maximum time between any consecutive pair of table entries is higher. Using different MTUs for the different flows allows us to assign a smaller amount of bandwidth to those flows with a smaller MTU than the general MTU. Moreover, the value of the k parameter can be

smaller. However, the specific flow MTU must be assigned taking into account the characteristics of the traffic flow. A too small specific MTU may decrease the latency performance of the flow [8].

Summing up, the DTable scheduler is a table-based scheduler that is able to deal properly with variable packet sizes and considers the possibility of a link-level flow control mechanism. Moreover, with our configuration methodology we can provide a flow with latency and bandwidth requirements in a partially independent way.

3 Advanced Switching revision

Advanced Switching (AS) is built on the same physical and link layers as PCI Express. Moreover, it includes an optimized transaction layer, providing a rich set of features and capabilities. Direct unicast communication between any two nodes, or multicast communication between a source node and multiple designated destination nodes, are supported by the AS architecture.

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. However, 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. The flow control credit unit is 64 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 uses VCs to aggregate flows with similar characteristics. 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. Moreover, each VC type has its own MTU. The allowed MTU values for the bypassable VC type are 192, 320, 576, 1088, and 2176 bytes. The allowed MTU values for the ordered VC type are 64, 96, 128, 192, 320, 576, 1088, and 2176 bytes.

In AS, the arbitration is made at 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.

The table scheduler employs an arbitration table that consists in a list of VC identifiers. 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 [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.

4 Providing QoS over AS with the DTable scheduler

In [10, 9] we examined the AS mechanisms and showed how to provide QoS to the applications. First of all, a set of Service Classes (SCs) with different requirements must be specified. The egress link scheduler must be properly configured to provide the different SCs with their requirements. Moreover, an admission control protocol must be used to provide QoS guarantees.

In order to define this set of SCs, we propose a traffic classification based on two network parameters: Bandwidth and latency. We distinguish three broad categories of traffic: Network control traffic, QoS traffic, and best-effort traffic. The network control traffic is high-priority traffic used to maintain and support the

network infrastructure. The QoS traffic has explicit minimum bandwidth and/or maximum latency requirements. The best-effort traffic is largely insensitive to both bandwidth and latency and is only characterized by the differing priority among each other.

When various flows obtain access to the AS fabric, they will be aggregated into the SCs depending on their characteristics. If there are sufficient VCs, we will devote a separate VC to each SC. The control SC will be assigned to the FMC. Note, however, that this VC does not have maximum priority when using this scheduler, so we will consider it as any other VC with traffic of high latency requirements.

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. We only implement an AC for the QoS traffic and not for the network control and the best-effort traffic. The AC mechanism would allow a new QoS connection to be established if there is enough bandwidth all along its path.

As stated before, when using the DTable scheduler, 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. In order to provide traffic of a given VC with a minimum bandwidth, the amount of weight units from the bandwidth pool assigned to those VC table entries must accomplish with the proportion of desired egress link bandwidth.

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. Secondly, QoS traffic may be bursty (for example a video transmission) and may require, during short periods of time, more bandwidth than average. Therefore, we propose that 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. Moreover, we propose to assign the FMC not only the expected amount of the control SC traffic, but also the rest of the best-effort bandwidth and the amount of bandwidth that is expected that the network is not going to be able to provide. Note that the bandwidth left over by the FMC would be redistributed by the DTable scheduler among the rest of VCs.

When we know the maximum distance between two consecutive table entries, and thus, the number of entries, and the amount of bandwidth that we want to assign to each VC, we must choose the w and k parameters that best fit that distribution. Note that the latency performance depends on the w parameter. Therefore, an option is to establish the maximum distance between any consecutive pair of entries of the SCs taking into account the maximum w value allowed by the hardware implementation of the DTable scheduler. In this way, when configuring the DTable scheduler, we can choose a smaller w value to improve the latency performance, but, in any case, the maximum latency requirements are going to be guaranteed.

Moreover, we can limit the MTU of some VCs in order to have a smaller minimum bandwidth for those VCs and for being able to use smaller k values. We can assign each VC a different MTU at a communication library level, but this would entail to add complexity to the AS communication protocols. On the other hand, we can take advantage of the AS characteristics to simplify the process. As stated before, AS allows us to establish two different MTUs for the two unicast VC types. Therefore, we can have two sets of VCs with two different MTUs and we can assign the SCs to the VCs taking into account this. Note that those SCs that have high latency requirements, and thus require more table entries, usually have small bandwidth requirements and use small packets. Therefore, we can assign these SCs to the VCs with the smallest MTU.

There are two possible ways of configuring the schedulers at the network elements. The first possibility is to configure the schedulers in advance, defining a set of SCs with a different minimum bandwidth and maximum latency reservation [16]. 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. This second possibility can be implemented in two ways. In the first one, we can modify both the distribution of the table entries and the weights assigned to them. In the second one, we fix the distribution of the table entries, and thus the maximum latency performance properties of each VC, and modify the bandwidth assignation in a dynamic way. To do this, we distribute the weight units from the bandwidth pool among the VCs in a dynamic way taking into account the minimum and maximum bandwidth that the decoupling configuration methodology allows us to assign to each VC. In this last case

the reconfiguration of the arbitration table is much faster than if we modify also the distribution of the table entries.

5 Implementing the DTable in AS

The AS arbitration table consists in a list of VC identifiers without any weight assigned to each entry as it is the case in the DTable scheduler. As stated before, 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. This is the reason of the AS table scheduler problem with variable packet sizes: 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 [9] we proposed to convert the AS table scheduler into the DTable scheduler. To do this we proposed to assign each table entry with the same weight, the MTU. Moreover, we internally assign each VC with a deficit counter. These little changes require simple hardware modifications of the original AS table scheduler but they do not require to modify the interface provided in the AS specification for configuring the table scheduler. This simple modification solves the problem of the AS table scheduler with variable packet sizes but it does not allow to decouple the bandwidth and the latency assignments using our configuration methodology because this methodology relies on using different weights for the table entries assigned to the different VCs.

In this paper we propose several possibilities to fully implement the DTable scheduler in AS. Our objective is to be able to assign the table entries with different weights but modifying as little as possible the AS specification. The AS arbitration table is a register array with fixed-size entries of 8 bits. Each entry contains a field of 5 bits with a VC identifier value and a reserved field of 3 bits. Figure 1 shows an example of the original arbitration table with 64 entries. We propose three possibilities to assign each table entry with a weight: to use the 3-bit reserved field, to modify the arbitration table structure, and to use the same weight for all the entries of a VC.

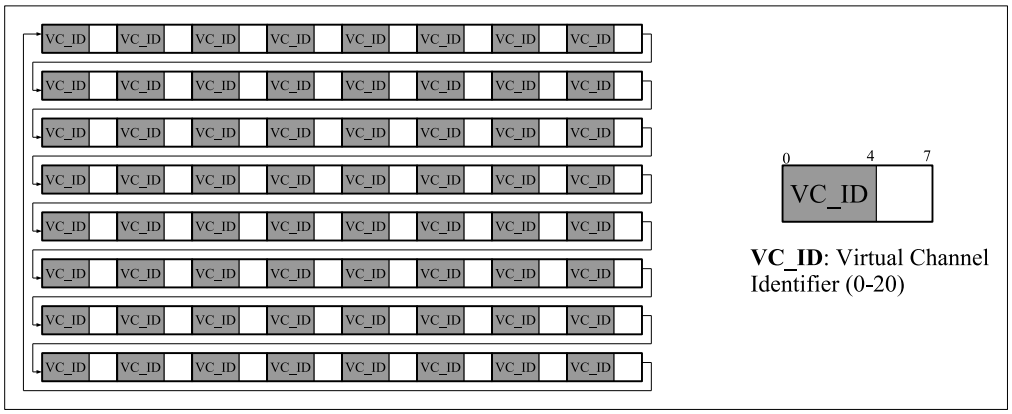

Figure 1: Example of the original AS arbitration table with 64 entries.

5.1 Using the 3-bit reserved field

The easiest possibility to implement the DTable scheduler in AS is to employ the 3-bit reserved field to assign a weight to each entry. The problem of this implementation is that this field only allows us to specify a weight between 0 and 7, and thus, several considerations must be made.

First of all, note that the weight (the number of weight units) assigned to each table entry in the DTable scheduler represents the amount of information that each table entry allows to be transmitted. This weight could be expressed for example in bytes (1 weight unit = 1 byte). However, in networks with a credit-based flow control, which is the case of AS, the flow control allows to transmit information with a granularity equal to the flow control credit size. Note that in order to transmit a packet of a given size, both the flow control and the scheduling must allow that packet to be transmitted. Therefore, expressing the table entries weight in flow control credits (1 weight unit = 1 flow control credit) is the logical option. However, as we will see, this is not always possible.

Table 1: Configuration aspects when using the 3-bit option and the same MTU for all the VCs.

$GMTU$ (in weight units)	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Minimum number of credits per weight unit	5	5	6	7	9	12	17	34
w parameter	1	1.14	1.33	1.6	2	2.66	4	8
Maximum weight per entry (in weight units)	8	8	8	8	8	8	8	8
Minimum k	1	1	1	1	1	1	1	1
Maximum w/k	1	1.14	1.33	1.6	2	2.67	4	8
Maximum k	1	1.14	1.33	1.6	2	2.66	4	8

Table 2: Configuration aspects when using the 3-bit option and two different MTUs.

$GMTU$ (in weight units)	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
MTU' (in weight units)	1	1	1	1	1	1	1	1
Minimum number of credits per weight unit	5	5	6	7	9	12	17	34
Maximum MTU' (bytes)	320	320	320	576	576	576	1088	2176
w parameter	1	1.14	1.33	1.6	2	2.67	4	8
Maximum weight per entry (in weight units)	8	8	8	8	8	8	8	8
Minimum k	0.34	0.36	0.38	0.4	0.44	0.5	0.63	1
Maximum w/k	2.94	3.17	3.5	4	4.54	5.32	6.35	8
Maximum k	1	1.14	1.33	1.6	2	2.66	4	8
$MTU'/GMTU$	1/8	1/7	1/6	1/5	1/4	1/3	1/2	1

As stated before, the entry weight must represent at least the value of the GMTU. In AS, the GMTU can be up to 34 flow control credits (2176 bytes). Obviously, it is not possible to represent directly a value of at least 34 with just 3 bits. Therefore, when using the 3-bit reserved field to assign a weight to each entry, each weight unit will represent a weight equivalent to a certain number of flow control credits. The maximum weight per entry, the bandwidth pool, and the actual value assigned to each table entry are expressed in weight units. However, the scheduler is still going to consider flow control units to arbitrate. This means that the deficit counter and the accumulated weight are still going to be expressed in flow control credits.

Therefore, when an entry is selected its weight must be translated into its value in flow control credits. As stated before, the MTU is the smallest weight that can be assigned to a table entry. Therefore, a weight of 0 is not going to be used, and thus, we propose to consider the weight 0 as 1, the weight 1 as 2, etc. This allows us to specify a weight between 1 and 8 with the 3-bit field. Therefore, when a new table entry is selected, the accumulated weight is computed as: $accumulated\_weight = deficit\_counter + ((entry\_weight + 1) \times credits\_per\_weight\_unit)$. When configuring the DTable scheduler we must specify, apart from the VC identifier and weight of each table entry, the number of flow control credits that represents each weight unit.

Moreover, in order to calculate the minimum and maximum bandwidth that can be assigned to a VC, we must have a value in weight units for the GMTU and the specific MTUs, if applicable. These values indicate the minimum weight in weight units that can be assigned to a table entry depending on its associated VC. Therefore, during the configuration phase, we must choose a value between 1 and 8 weight units to represent the different MTUs. Note that the real MTU expressed in flow control credits and its equivalent value in weight units determine the minimum number of credits that a weight unit must represent. For example, if we choose to represent a MTU of 34 flow control credits with 3 weight units, each weight unit must represent at least 12 flow control credits ($12 \times 3 = 36 \geq 34$).

Table 1 shows several configuration aspects for the different values of the GMTU expressed in weight units using the same GMTU for all the VCs. This table shows the minimum number of flow control credits per weight unit if we have a GMTU of 34. The table also shows the value of the w parameter (calculated with the expresion $w = M/GMTU$), the maximum weight per entry, which is 8 in all the cases, the minimum and maximum values of the k parameter (the minimum value is 1 in all the cases and the maximum is the value of the w parameter), and the maximum value for the w/k fraction. Note that this fraction indicates the maximum bandwidth that can be assigned per table entry. In the same way, the maximum k value indicates the minimum bandwidth that can be assigned per table entry.

Let us now suppose a scenario where we have two VCs with high latency requirements, but low bandwidth

Table 3: Number of bits assigned to the weight.

Number of bits	**6**	**7**	**8**	**9**	**10**	**11**
Entry weight	1-64	1-128	1-256	1-512	1-1024	1-2048
Maximum w	1.88	3.76	7.53	15.06	30.12	60.24
Maximum granularity	1/(N×64)	1/(N×128)	1/(N×256)	1/(N×512)	1/(N×1024)	1/(N×2048)

Table 4: Specific MTU.

MTU' (bytes)	**64**	**96**	**128**	**192**	**320**	**576**	**1088**	**2176**
MTU'/$GMTU$	0.03	0.04	0.06	0.09	0.15	0.26	0.5	1
Minimum k	0.27	0.28	0.29	0.32	0.36	0.45	0.63	1

requirements, and other VCs with low latency requirements. In this case, a possibility is to assign a maximum distance of 2 (50% of entries) to the most prioritary VC and a maximum distance of 4 (25 % of entries) to the other high-latency requiring VC, and to use a small specific MTU (MTU') for them to decrease the minimum bandwidth that can be assigned to them. The remaining VCs will use the generic MTU. Table 2 shows several configuration aspects for the different values of the GMTU expressed in weight units in this new scenario (75% of entries with a MTU of MTU' and 25% of entries with an MTU of $GMTU$, where $MTU' \leq GMTU$).

Table 2 shows the same information as Table 1 and also the fraction $MTU'/GMTU$. This fraction indicates the reduction of the minimum bandwidth per table entry of the entries that use MTU'. Note, that we have chosen one weight unit for MTU' because it is the smallest possible. This entails using a maximum real MTU' as indicated in the table (considering only the possible MTUs in AS stated in Section 3).

Table 2 also shows the advantage of using two MTUs instead of only one. For example, the minimum value for the k parameter is smaller, and thus, we have a higher flexibility. Note that the minimum value for the k parameter is calculated from the minimum bandwidth pool ($pool = GMTU \times k$, $k = pool/GMTU$), which is 0.75×MTU' + 0.25×GMTU. Moreover, those VCs using MTU' have a smaller minimum bandwidth. However, we have an important disadvantage because of the 1-8 range of the table entry weight: When we increase the w parameter, the flexibility increases, but the advantage of using two MTUs decreases because we have been limited the value of the specific MTU.

The bandwidth assignation granularity depends on the bandwidth pool. The maximum bandwidth pool is the maximum weight per table entry multiplied by the number of table entries, and thus, the maximum granularity is 1/(8×N).

Summing up, this possibility limits the possible values for the w parameter and the specific MTUs. This limits the flexibility of the table configuration. However, the implementation of this option is quite simple.

5.2 Modifying the arbitration table format

Other possibility is to modify the structure of the arbitration table in order to dedicate a higher number of bits to the entry weight. Specifically, we propose to use two bytes per table entry, and use 5 bits for the VC identifier and up to 11 for the entry weight.

Depending on the actual number of bits assigned to the weight we can assign a different maximum value to the w parameter. Table 3 shows the maximum w value depending on the number of bits used for the weight. It also shows the allowed weight range per entry and the maximum bandwidth assignation granularity. Note that at least 6 bits are required to represent a MTU of 34 credits. Table 4 shows the fraction $MTU'/GMTU$ and the minimum k value depending on the specific MTU' value. As in Table 2 we consider a table with 75% of entries of a MTU of MTU' and 25% of entries with an MTU of $GMTU$.

This possibility allows a higher flexibility in the assignation of the w parameter and the specific MTUs values. However, it requires the double of memory to store the arbitration table than the previous option for the same number of entries. Moreover, it requires processing two bytes per entry instead of only one.

5.3 Using only one weight per VC

The third possibility that we propose is to associate the same weight to all the entries assigned to a VC. Therefore, we only need to specify a table weight per VC instead of per table entry. In order to change as

little as possible the AS specification a possibility is to specify the weight assigned to the entries of each VC employing the MinBW configuration structure, which provides 12 bits to specify a weight per each VC. This allows us to specify a weight between 1 and 4096, and thus, the maximum w value is around 120 ($M = GMTU \times w$, $w = 4096/34$). When a new table entry is selected, the accumulated weight is computed as: $accumulated\_weight = deficit\_counter + VC\_weight$.

This possibility also allows a higher flexibility in the assignation of the w parameter and the specific MTU values than the 3-bit option. The disadvantage of this possibility is that we cannot assign the weight units from the bandwidth pool in a totally free way between the table entries. We have to assign the weights in exact fractions of the number of entries of each VC. Therefore, the bandwidth assignation granularity is different for each VC and depends on the number of entries assigned to that VC (n_i). In this way, the maximum granularity is $n_i/4096$.

6 Performance evaluation

In this section, we evaluate the performance of our proposals to provide latency and bandwidth requirements over AS with a full implementation of the DTable scheduler. We consider a realistic multimedia scenario with different types of traffic. This allows us to apply our proposals of using different MTUs taking into account the traffic characteristics. We also compare the flexibility in the bandwidth assignation of the three possibilities that we propose to implement the DTable scheduler. Moreover, we compare the performance of the DTable scheduler with the performance of the MinBW scheduler implemented with the DRR Credit Aware (DRR-CA) and the SCFQ Credit Aware (SCFQ-CA) schedulers [7]. For this performance evaluation, we have developed a detailed simulator that allows us to model the network at the register transfer level, following the AS specification. We have chosen a multimedia scenario to test our proposals, but they are equally valid for other environments.

6.1 The SCFQ and DRR algorithms

As stated before, the design of a traffic scheduling algorithm involves an inevitable trade-off among latency and computational complexity. Many scheduling algorithms have been proposed. Among them, the "sorted-priority" family of algorithms are known to offer very good latency [19]. However, their computational complexity is very high, making their implementation in high-speed networks rather difficult. The Self-Clocked Weighted Fair Queuing (SCFQ) algorithm [3] is an example of "sorted-priority" algorithm. In order to avoid the complexity of the sorted-priority approach, the Deficit Round Robin (DRR) algorithm [17] has been proposed. The main problem of this algorithm is that its delay depends on the frame length. Depending on the situation, the frame can be very long, and thus, the latency would be very high.

In [7] we proposed the SCFQ Credit Aware (SCFQ-CA) and the DRR Credit Aware (DRR-CA) algorithms. These new algorithms, which are based on the SCFQ and DRR algorithms, accomplish all the properties that the AS MinBW scheduler must have, including the interaction with the AS flow control. In this performance evaluation we compare the DTable scheduler with the MinBW scheduler implemented with these two algorithms. We have chosen the SCFQ-CA algorithm as an example of "sorted-priority" algorithm with a good latency performance and the DRR-CA algorithm because of its very low computational complexity. In order to simplify the comparison among the three schedulers, we have not reflected the different complexity into different scheduling time.

6.2 Simulated architecture

We have used a perfect-shuffle Bidirectional Multi-stage Interconnection Network (BMIN) with 64 end-points connected using 48 8-port switches (3 stages of 16 switches). In AS any topology is possible, but we have used a MIN 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.

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

Table 5: 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.

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 [18].

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. We are going to use the maximum MTU allowed in AS: 2176 bytes (34 flow control credits). The buffer capacity is 17408 bytes (8×MTU) per VC both at the input and at the output ports of the switches.

6.3 Simulated scenario

The IEEE standard 802.1D-2004 [4] defines 7 traffic types at the Annex G, which are appropriate for this study. We will consider each traffic type as a SC. Table 5 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. As stated before, the NC SC is assigned to the FMC.

We suppose a scenario in which the goal is to dedicate around 20-25% of the egress link bandwidth to best-effort traffic, around 5-10% bandwidth to voice traffic (we expect to have a lot but low-bandwidth requiring connections), around 5-10% of bandwidth to controlled load, and 40-50% of bandwidth to video traffic (a lot and high-bandwidth requiring connections). This percentages are intended to represent a multimedia scenario with a realistic combination of traffic from applications with very different requirements. Moreover, we expect that the maximum network throughput to be around 85-95%. We also consider that control and voice traffic uses small packet sizes, as it is usually the case. In [20] 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 voice traffic.

6.4 Scheduler configuration

The table scheduler must be properly configured to provide the SCs with their bandwidth and latency requirements. A table of 128 entries has been used. Table 6 shows the distribution of the table entries among the SCs. It shows the maximum distance between any consecutive pair of entries, the number of table entries, and the percentage of entries that this entails for each CS. We have assigned a distance of 2, 4, and 8 to the NC, VO, and VI SCs respectively, attending to their latency requirements. Note that this entails assigning 112 entries. We have distributed 8 entries among the best-effort SCs attending to the different priority among them. Finally, we have assigned the remaining 8 entries to the CL SC. For the CL SC and the best-effort SCs 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 any pair of consecutive entries.

Table 6 also shows the weight configuration that we have chosen to fit the bandwidth requirements stated in the previous section using the 3-bit implementation option of the DTable scheduler. As stated before, we must choose a value between 1 and 8 to represent the GMTU (in this case 34 flow control credits). We have chosen to represent the GMTU using 3 weight units. Therefore, each weight unit represents 12 flow control credits, the w value is 2.67, and the maximum specific MTU is 576 bytes (see Table 2). We are going to use this smaller MTU for the NC and VO SCs. Note that assigning a smaller MTU to these SCs is not going to decrease their performance because the control and voice traffic already use small packets. Note also that we

Table 6: Table entries distribution and weight configuration with the 3-bit option. $w = 2.67$, $M = 8$, $k = 0.76$, *pool*=292.

	Table entries distribution			**Weight configuration (3-bit option)**				
VC	**Max. distance**	**#entries**	**%entries**	**MTU**	$min\phi_i$	$max\phi_i$	ϕ_i	**Total weight**
NC	2	64	50	1	21.92	175.34	28.08	82
VO	4	32	25	1	10.96	87.67	10.96	32
VI	8	16	12.5	3	16.44	43.84	43.84	128
CL	16	8	6.25	3	8.22	21.92	8.91	26
EE	32	5	3.91	3	5.14	13.70	5.14	15
BE	64	2	1.56	3	2.05	5.48	2.05	6
BK	128	1	0.78	3	1.03	2.74	1.02	3
	Total	128	100	-	65.76	350.69	100	292

Table 7: Weight configuration with the modification of the table structure and the weight per VC options. Table configuration. $w = 2.68$, $M = 91$, $k = 0.76$, *pool*=3308.

				Modifying the table		**One weight per VC**	
VC	**MTU**	$min\phi_i$	$max\phi_i$	ϕ_i	**Total weight**	ϕ_i	**Total weight**
NC	9/5	17.41/9.67	176.06	28.08	928	27.09/29.02	896/960
VO	9/5	8.71/4.84	88.03	10.96	363	10.64/11.61	352/384
VI	34	16.45	44.01	43.84	1450	43.53/44.01	1440/1456
CL	34	8.22	22.01	8.91	295	8.71/8.95	288/296
EE	34	5.14	13.75	5.14	170	5.14	170
BE	34	2.05	5.50	2.05	68	2.05	68
BK	34	1.03	2.75	1.03	34	1.03	34
	Total	59.01/47.4	352.11	100	3308	98.19-101.81	3248-3368

could use a smaller specific MTU, but the minimum bandwidth of the NC and VO SCs would not decrease more. Moreover, we have chosen a k value of 0.76.

Table 6 shows the minimum and maximum bandwidth that this configuration entails (according to expressions 1 and 2) and the bandwidth ϕ_i that has been finally assigned to each SC. Moreover, it shows the total number of weight units from the bandwidth pool that each SC has been assigned to their entries. In order to choose these parameters we have considered mainly how to provide the VO SC with a small amount of bandwidth at the same time that we assign the VI SC a high amount of bandwidth.

Table 7 shows the weight configuration of the other two options that we propose as possible implementations of the DTable scheduler: to modify the structure of the arbitration table to use more bits for the weight and to use only one weight per VC. In order to compare these two cases with the 3-bit option, we have used the same values for the w and k parameters. Note, however, that the value of w is 2.68 instead of 2.67 because the value of M must be an integer ($M = GMTU \times w$, $91 = 34 \times 2.68$). These two options have the same minimum and maximum bandwidth per VC. Note that the values are slightly different from the 3-bit option because in these two cases the MTUs are expressed in flow control credits and not in weight units, which is an approximation of the real value. Note also that contrary to the 3-bit case, in these cases we could have assigned a smaller specific MTU to have smaller minimum bandwidths for the NC and VO SCs. Table 7 shows an example with a MTU of 320 bytes (5 flow control credits). The main difference between the options of modifying the table structure and using one weight per VC is the granularity in the bandwidth assignment. Table 7 shows that with one weight per VC the same minimum bandwidth values that in the 3-bit case cannot be assigned for all the VCs. This table shows the two nearest values that could be assigned instead. Note that the granularity depends on the number of table entries.

Note that the difference of the three DTable implementations is the way of specifying the weight of a given table entry. This is going to affect the flexibility to assign the bandwdith to the VCs, but not the performance that they provide. Therefore, we only show the performance results obtained with the 3-bit implementation. In order to compare the performance of the DTable and MinBW schedulers, both proposed implementations of the MinBW scheduler (DRR-CA and SCFQ-CA) have been configured to provide a minimum bandwidth equal to the stated for the DTable case in Table 6.

Table 8: Injected traffic and scheduler configuration.

SC	Injection rate	Traffic pattern	Packet size (bytes)
NC	0.01	Bursts60	up to 576
VO	0.119	64 Kb/s CBR	168
VI	0.438	750 Kb/s MPEG-4 traces	up to 2176
CL	0.089	750 Kb/s CBR	2176
EE	$0 \rightarrow 0.115$	Bursts60	up to 2176
BE	$0 \rightarrow 0.115$	Bursts60	up to 2176
BK	$0 \rightarrow 0.115$	Bursts60	up to 2176
Total	$0.656 \rightarrow 1.001$		

6.5 Traffic pattern

The packets are generated according to different distributions, as can be seen in Table 8. 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 the 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 [5]. 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. The packet sizes include the AS packet header of 8 bytes.

Our intention is to show that with an AC mechanisms for controlling the QoS traffic and a relatively small amount of control traffic (as it 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 traffic of each QoS SC (VO, VI, and CL) equal to the minimum bandwidth that they have been assigned, representing the maximum injection allowed by the AC mechanism. Moreover, we inject a fixed amount of control traffic (NC), and gradually increase the amount of best-effort traffic (EE, BE, and BK). Table 8 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.

6.6 Simulation results

Figure 2 shows the throughput and bandwidth performance of the three schedulers considered: The DTable scheduler, the MinBW scheduler implemented with the DRR-CA algorithm, and the MinBW implemented with the SCFQ-CA algorithm. The figure shows the average values and the confidence intervals at 90% confidence level of twenty different simulations performed at a given input load. For each simulation we obtain the normalized average throughput, the average packet injection latency, and the maximum packet injection latency of each flow. No statistics on packet loss are given because, as has been said, we assume a credit-based flow control mechanism to avoid dropping packets. We obtain statistics per VC aggregating the throughput of all the flows of the same VC, 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.

Regarding the throughput performance, Figure 2 shows that the NC and the QoS SCs obtain all the bandwidth that they inject. However, when the network load is high (around 85%-90%), 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 NC SC and the QoS SCs grow with the load until they reach a certain value. Once this value is reached the latency remains more or less constant. On the other hand, the average and maximum latency of best-effort SCs grow with the load. Furthermore, it can be seen that best-effort SCs obtain a different average and maximum latency according to their priority.

Figures 3 and 4 show the percentage of improvement on average and maximum latency of the DTable scheduler over the MinBW scheduler implemented with the SCFQ-CA algorithm and the MinBW scheduler

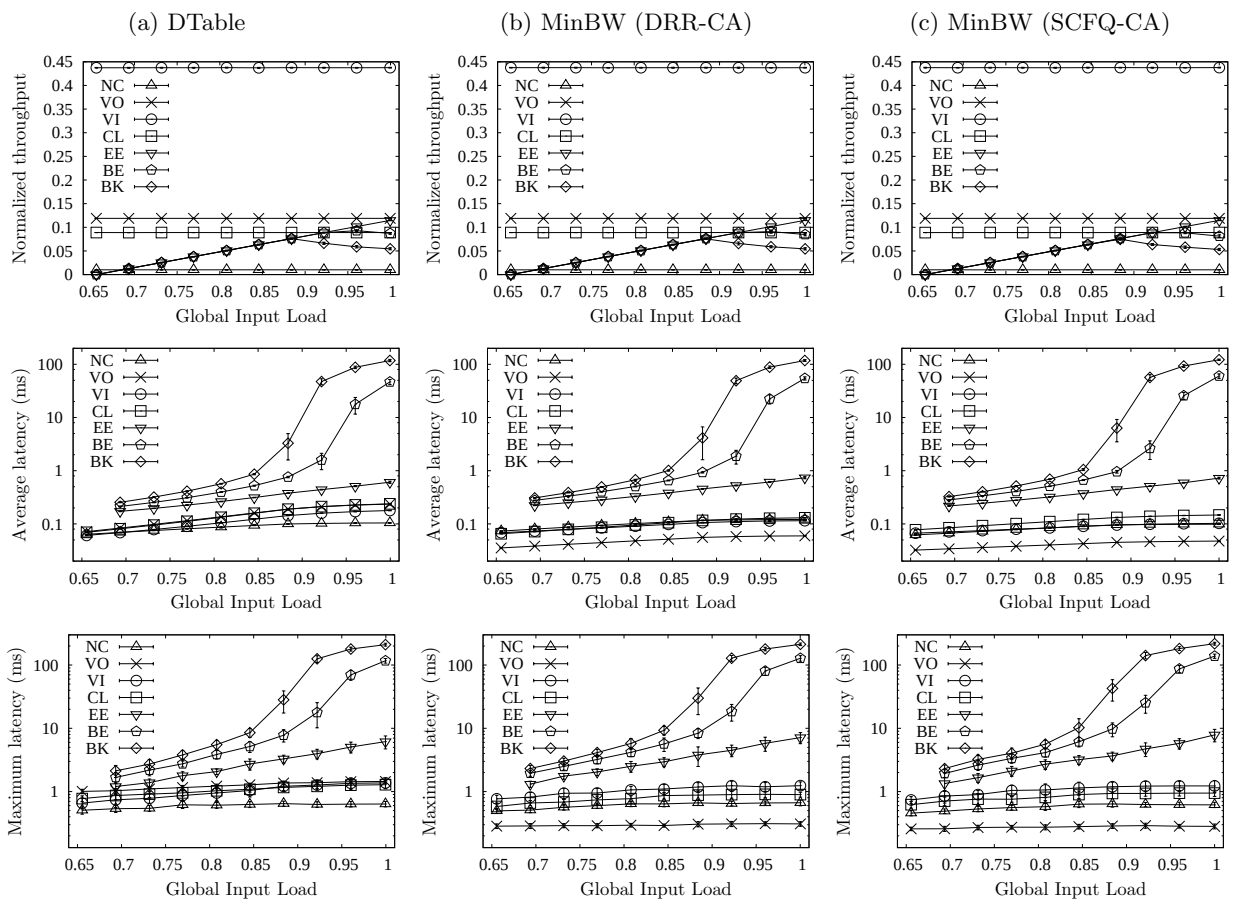

Figure 2: Throughput and latency performance per SC of the three schedulers.

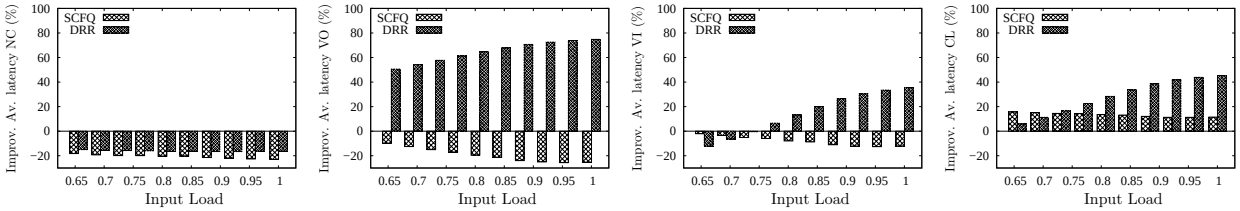

Figure 3: Average latency improvement of the DTable scheduler over the MinBW scheduler.

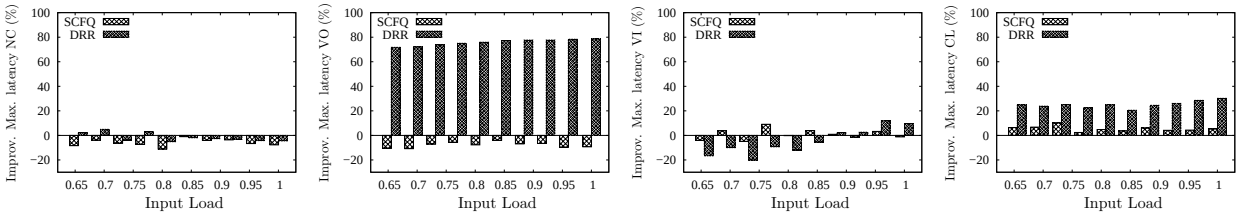

Figure 4: Maximum latency improvement of the DTable scheduler over the MinBW scheduler.

implemented with the DRR-CA algorithm for the NC and QoS SCs. These figures offer a clearer picture of the difference between the three schedulers. These figures show that both implementations for the MinBW scheduler obtain a better latency performance for the NC SC. This is because this SC is assigned to the FMC VC, which has maximum priority over the rest of VCs with the MinBW scheduler. However, we can say that the DTable scheduler obtains a very good performance if we consider that with this scheduler the FMC is treated like any other VC. Note that with this scheduler the average latency is only around 20% worse and

the maximum latency around 10% worse than with the MinBW scheduler. This is because we have assigned a maximum distance between any consecutive pair of entries of 2.

Regarding the QoS SCs Figures 3 and 4 show that the DTable scheduler provides a quite better latency performance than the DRR-CA scheduler for the VO SC. However, it provides a slightly worse latency than the SCFQ-CA scheduler. Again this difference is smaller for the maximum latency statistic.

Figures 3 and 4 also show that the improvement of the DTable scheduler over the DRR-CA scheduler for the VI SC depends in high degree on the network load. When the load is low, the average latency performance provided by the DTable scheduler is slightly worse than the latency provided by the DRR-CA. However, the DTable scheduler provides a better latency when the load is high. The SCFQ provides in any case a slightly better average latency for this SC. The maximum latency performance of the VI SC presents a similar behaviour, however, the variability of the maximum latencies is higher and the differences among the three schedulers are not so clear.

Regarding the CL SC, Figures 3 and 4 show that the DTable scheduler provides a better average and maximum latency performance than both MinBW implementations.

Summing up, all the schedulers provide the same bandwidth performance. The differences among the three schedulers appear only in the latency performance. The DTable scheduler provides a better latency performance for the QoS SCs and only slightly worse latency performance for the NC SC than the MinBW implemented with the DRR-CA algorithm. It provides, however, for all the SCs a slightly worse latency than the MinBW implemented with the SCFQ-CA algorithm. However, the complexity of the DTable scheduler is quite lower than the complexity of the SCFQ-CA algoritm and only slightly higher than the DRR-CA algorithm. Moreover, as stated before, we have not reflected the different complexity of the DTable and the SCFQ schedulers into different scheduling times. If so, the comparison would be even better for the DTable scheduler. Therefore, the DTable scheduler provides a good trade-off between latency and complexity.

7 Conclusions

AS provides a table-based scheduler that does not work properly with variable packet sizes and faces the problem of bounding the bandwidth and latency assignments. In this paper we propose three possible modifications to the original AS table scheduler in order to fully implement the DTable scheduler. The objective of these modifications have been to be able to assign each table entry with a weight, but modifying as little as possible the AS specification. The three possibilities that we propose are: to use the 3-bit reserved field, to modify the arbitration table structure, and to use the same weight for all the entries of a VC. The difference among the three DTable implementations is the way of specifying the weight of a given table entry. This is going to affect the flexibility to assign the bandwdith to the VCs, but not the performance that they provide.

Using the 3-bit reserved field is probably the simplest possibility. However, it limits the possible values for the w parameter and the specific MTUs. The possibility of using the same weight for all the entries of a VC allows us to use higher values for the w parameter and to choose freely the values for the specific MTUs. However, the bandwidth assignation granularity is different for each VC and depends on the number of entries assigned to that VC. The possibility of modifying the arbitration table structure does not present these problems, however, it requires a higher amount of memory to store the arbitration table and needs to process two bytes, instead of one, per table entry.

Moreover, we have review our proposals of providing QoS over AS using the DTable scheduler. Specifically, we have proposed how to distribute the table entries among the VCs, how to assign the table weights, and how to assign specific MTUs to the VCs. We have also proposed to take advantage of the fact that AS defines two unicast VC types that can have different MTU values to simplify the assignation of different MTUs to the VCs.

Finally, we have evaluated our proposals in a multimedia scenario and have compared the performance of the DTable scheduler with two different implementations for the MinBW scheduler: the SCFQ-CA and the DRR-CA algorithms. The simulation results show that all the schedulers provide the same bandwidth performance, but a different latency performance. The DTable scheduler provides a better latency performance than the MinBW implemented with the DRR-CA and only slightly worse than the MinBW implemented with the SCFQ-CA with only a slightly higher computational complexity than in the DRR-CA case.

Summing up, we have shown that modifying slightly the AS specification it is possible to solve in a high

degree the problems of the original AS table scheduler. The resulting scheduler would allow us to provide a good latency performance with a small computational complexity.

References

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

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

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

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

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

[7] 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.

[8] R. Martínez, F.J. Alfaro, and J.L. Sánchez. Improving the flexibility of the deficit table scheduler. Technical Report DIAB-06-06-1, Departamento de Informática Universidad de Castilla-La Mancha, June 2006. Also submitted to the International Conference on High Performance Computing (HiPC) 2006. Available at `https://www.info-ab.uclm.es/trep.php`.

[9] 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.

[10] 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.

[11] 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.

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

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

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

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

[16] 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.

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

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

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

[20] 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.