A Strategy to Manage Time Sensitive Traffic in InfiniBand*

Francisco J. Alfaro, José L. Sánchez
Dept. de Informática
Escuela Politécnica Superior
Universidad de Castilla-La Mancha
02071- Albacete, Spain
{falfaro, jsanchez}@info-ab.uclm.es

José Duato
Dept. de Informática de
Sistemas y Computadores
Universidad Politécnica de Valencia
46071- Valencia, Spain
jduato@gap.upv.es

Abstract

The InfiniBand Architecture (IBA) is becoming an industry standard both for communication between processing nodes and I/O devices and for interprocessor communication. It replaces the traditional I/O bus with a switch-based interconnect for connecting processing nodes and I/O devices. It is being developed by the InfiniBand^{SM} Trade Association (IBTA) to provide the levels of reliability, availability, performance, scalability, and quality of service (QoS) necessary for present and future server systems. For this, IBA provides a series of mechanisms that are able to guarantee QoS to the applications.

In [1, 2], we proposed a strategy to compute the InfiniBand arbitration tables. We only evaluated our proposal for traffic with bandwidth requirements. In this paper, we propose and evaluate a strategy to compute the InfiniBand arbitration tables for traffic with bandwidth and delay requirements, which is a more complex task. Performance results show that, with a correct treatment of each traffic class in the arbitration of the output port, both traffic classes (with just bandwidth requirements and with bandwidth and delay requirements) meet their QoS requirements.

1 Introduction

Although buses have the major advantage of simplicity, and have served the industry well up to this point, they do not provide the level of performance, scalability, reliability, and availability now being required for server systems. Several external interconnects, such as Fiber Channel, have been used to overcome these difficulties. However, they are still attached to the proccessing nodes through an industry-standard bus, making it impossible to avoid the bottlenecks and low availability characteristic of standard I/O buses [9].

This was the primary reason why a group of the most important IT companies joined together to develop a standard for communication between processing nodes and I/O devices as well as for interprocessor communication, known as InfiniBand [7]. The InfiniBand^{SM} Trade Association (IBTA) is a group of more than 200 companies founded in August 1999 to develop IBA. Membership is also open to Universities, research laboratories, and others. The IBTA is led by a Steering Committee whose members come from Dell, Compaq, HP, IBM, Intel, Microsoft, and Sun, co-chaired by IBM and Intel. It also consists of eight sponsoring members (3Com, Cisco Systems, Fujitsu-Siemens, Hitachi, Adaptec, Lucent Technologies, NEC, and Nortel Networks) and over 120 member companies. The first specification of the InfiniBand Architecture (release 1.0) was published in October 2000 [5], and more than 200 companies are currently supporting the InfiniBand initiative.

On the other hand, most of the current networking products have tried to achieve maximum throughput and minimum latency, leaving aside other aspects like guarantee of bandwidth, maximum latency deadline, interarrival delays, etc [10, 11]. Many current applications need those characteristics that not all the current networks are able to provide.

Therefore, it would be important for InfiniBand to be able to satisfy both the applications that only need minimum latency, and also those different applications that need other characteristics to satisfy their QoS requirements. InfiniBand provides a series of mechanisms that, when properly used, are able to provide QoS to the applications. These mechanisms are mainly the segregation of traffic according to categories and the arbitration of the output ports according to an arbitration table that can be configured to give priority to the packets with higher QoS requirements.

In [1, 2] we proposed a traffic classification able to provide QoS to each class of traffic. This traffic classification was based on the one previously proposed in [8]. We also proposed a strategy to compute the InfiniBand arbitration tables that had the significant advantage of not requiring recomputing the table entries for previously established connections when a new connection is established. We evalu-

*This work was partly supported by the Spanish CICYT under Grant TIC2000-1151-C07

ated our proposal only with traffic with bandwidth requirements. In this paper, we evaluate our proposal for traffic with delay requirements, which is a more complex task.

The structure of the paper is as follows: Section 2 presents a summary of the general aspects in the specifications of InfiniBand, including the most important mechanisms that InfiniBand provides to support QoS. In Section 3, traffic is classified into several categories. Section 4 presents a delay analisys for time sensitive traffic in InfiniBand. In Section 5, we explain our proposal to easily manage time sensitive traffic in InfiniBand, and performance is evaluated in Section 6. Finally, some conclusions are given and future work is proposed.

2 InfiniBand

The InfiniBand Architecture (IBA) defines a system area network for connecting processing nodes to each other and to I/O units. The architecture is independent of the host operating system and processor platform. An IBA network is divided into subnets interconnected by routers, each subnet consisting of one or more switches, processing nodes and I/O devices. IBA supports any topology defined by the user, including irregular ones, in order to provide flexibility and incremental expansion capability.

Processing and I/O nodes use channel adapters to connect to the fabric. A channel adapter is a programmable DMA engine that provides InfiniBand transport level functions. The IBA defines two kinds of channel adapter: Host Channel Adapter and Target Channel Adapter, that are intended for use by processing nodes and I/O devices, respectively.

IBA links are bidirectional full-duplex point-to-point communication channels, and may be either copper cable, optical fiber or printed circuit on a backplane. The signaling rate on the links is 2.5 GHz in the 1.0 release, the later releases possibly being faster. The physical links may be used in parallel to achieve greater bandwidth. Currently, IBA defines three link bit rates: 2.5 Gbps, 10 Gbps, and 30 Gbps (referred to as 1x, 4x, and 12x, respectively).

IBA's basic unit of communication is a message. Messages are segmented into packets for transmission on links and through switches. The packet size is such that after headers are considered, the Maximum Transfer Unit (MTU) of data may be 256 bytes, 1KB, 2KB or 4KB.

The IBA transport mechanisms provide both connection-oriented and datagram services, and these services could be reliable (acknowledged) or unreliable. Obviously, for supporting QoS, the applications must use reliable connections in order to be able to do resource allocation.

IBA management is defined in terms of managers and agents. While managers are active entities, agents are passive entities that respond to messages from managers. Every IBA subnet must contain a single master subnet manager, residing on an endnode or a switch that discovers and initializes the network.

The interested reader is referred to the InfiniBand Specifications [5] for more details on InfiniBand. Other interesting papers that are good summaries of the official specifications are [9, 3].

2.1 IBA support for QoS

In this section we are going to explain the mechanisms that IBA provides for supporting QoS. Basically, IBA has three mechanisms to support QoS: service levels, virtual lanes, and virtual lane arbitration for transmission over links.

IBA defines a maximum of 16 service levels (SLs), but it does not specify what characteristics the traffic of each service level should have. IBA provides fields for marking packets with a class of service. By allowing the traffic to be segregated by category, we will be able to distinguish between packets from different SLs and to give them a different treatment based on their needs.

IBA ports support virtual lanes (VLs), providing a mechanism for creating multiple virtual links within a single physical link. A VL is an independent set of receive and transmit buffers associated with a port. IBA switches may implement between one and 16 VLs ($VL_0 \ldots VL_{15}$). All ports support VL_{15}, which is reserved exclusively for subnet management, and must always have priority over data traffic in the other VLs. Since systems can be constructed with switches supporting different numbers of VLs, the number of VLs used by a port is configured by the subnet manager. Also, packets are marked with a service level (SL), and a relation between SL and VL is established at the input of each link by means of the SLtoVLMappingTable.

When more than two VLs are implemented, an arbitration mechanism is used to allow an output port to select which virtual lane to transmit from. This arbitration is only for data VLs, because VL_{15}, which transports control traffic, always has priority over any other VL. The priorities of the data lanes are defined by the VLArbitrationTable.

The VLArbitrationTable has two tables, one for scheduling packets from high-priority VLs and another one for low-priority VLs. However, IBA does not specify what is high and low priority. The arbitration tables implement weighted round-robin arbitration within each priority level. Up to 64 table entries are cycled through, each one specifying a VL and a weight, which is the number of units of 64 bytes to be transmitted from that VL. This weight must be in the range of 0 to 255, and is always rounded up as a whole packet.

A LimitOfHighPriority value specifies the maximum number of high-priority packets that can be sent before a low-priority packet is sent. If no high-priority packets are

ready for transmission at a given time, low-priority packets can also be transmitted.

3 Traffic classification

In [1, 2] we proposed a classification of traffic based on the one previously proposed in [8]. This classification can be extended to consider two priority levels for best-effort traffic. So, the classification of traffic that we are going to use is:

- DBTS: Dedicated Bandwidth Time Sensitive traffic, which requires a given minimum bandwidth and must be delivered within a given latency in order for the data to be useful.
- DB: Dedicated Bandwidth traffic, which requires a given minimum bandwidth but is not particularly sensitive to latency. This includes traffic that must have a bounded latency but where the actual value of the bound is not critical.
- PBE: Preferential Best Effort. This kind of traffic does not have any QoS guarantees but it will have priority over the usual best effort traffic. This category reflects the case for a massively accessed database or a WEB server where no guarantees can be given, but this traffic must have priority over the usual best effort traffic like FTP transfers.
- BE: Best Effort. This traffic tends to be bursty in nature and is largely insensitive to both bandwidth and latency.
- CH: Challenged. This traffic is intentionally degraded so as not to interfere with best effort traffic.

For the DBTS and DB categories, we believe that a segregation of traffic could be made based on their characteristics. We have focused on the mean bandwidth requirements and have classified the connections into several classes according to their mean bandwidth requirements. By classifying connections into different categories, and by using appropriate values in the arbitration tables, we will be able to give a more adjusted treatment to each traffic class according to its needs. We can map each traffic class in one IBA SL. So, the categories that we distinguish for DBTS (SLs 0, 1, 2, and 3) and DB (SLs 4, 5, 6, and 7), are:

- SL 0 and SL 4. Connections with very low bandwidth. Specifically, mean bandwidth requirements are lower than or equal to 64 Kbps.
- SL 1 and SL 5. Connections with low bandwidth, higher than 64 Kbps, but lower than or equal to 1.55 Mbps.
- SL 2 and SL 6. Connections with high bandwidth, higher than 1.55 Mbps, but lower than or equal to 64 Mbps.
- SL 3 and SL 7. Connections with very high bandwidth, higher than 64 Mbps.

Each of these categories will use a different service level and, whenever possible, we are going to give a different treatment to each SL so that each one could get the required QoS. Note that it will not always be possible to use a different virtual lane for each one of these categories. If we have fewer virtual lanes than SLs we will put together some SLs in each virtual lane. Nevertheless, we distinguish these four categories for each kind of traffic and we will evaluate the QoS behavior of each one.

4 Time sensitive traffic in InfiniBand

Pelissier [8] proposed to use the table of high priority for DBTS traffic and the table of low priority for the rest of the traffic, but he did not specify how to compute the weights for the VLs. In [1, 2] we proposed a strategy to compute the InfiniBand arbitration tables for this classification. We used the table of low priority for the traffic classes DB, PBE, BE, and CH, while the DBTS traffic used the table of high priority. We evaluated our strategy only with traffic with bandwidth requirements. For this type of traffic we must guarantee enough bandwidth to satisfy the bandwidth requirements of each connection.

For DBTS traffic the problem is more complex. Besides bandwidth guarantees, each connection must have a guaranteed maximum latency. The arbitration at each output port in the path for the connection must give higher priority to this traffic. The packets of these connections must arrive at their destinations before the maximum deadline.

The computation of the maximum delay that a packet of a given connection suffers when crossing a switch depends on the switch architecture. Nowadays there is no widely accepted model for the InfiniBand switches. Some companies are designing or testing their switches, and there is very little information available. It seems that IBM is going to use a central queue which dynamically allocates buffer space [4]. However, other interesting architectures have input and output buffers. These architectures either have as many inputs in the crossbar as ports (multiplexed crossbar) or each virtual lane has its own input in the crossbar (full crossbar) [6]. In the next subsections we are going to obtain an expression for the maximum latency per switch for each one of these architecture models.

4.1 Maximum delay in a switch with multiplexed crossbar

This type of switch should have a structure similar to the one shown in Figure 1. Each input port has a single input in the crossbar. Each output port also has only one output from the crossbar. With this structure the maximum delay that a packet suffers when crossing a switch can be computed, step by step, in the following way:

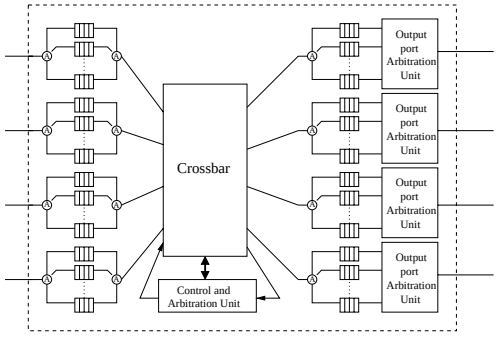

Figure 1. Possible structure of an InfiniBand switch with multiplexed crossbar

- Considering only the input buffer, and if the buffer has a FIFO behavior, all the packets that arrived before it in the buffer must leave before it. The number of packets depends on the buffer size (BS) and the maximum packet size (MTU): $\lceil \frac{BS}{MTU} \rceil$, where $\lceil \rceil$ means to round up (ceiling function).

- IBA specifications do not define any mechanism to arbitrate at the crossbar input. So, we are going to assume that round-robin is applied to select the next packet to transmit. Note that when a packet arrives at a switch, it is stored in a VL based on its SL and the SLtoVLMappingTable of IBA. But when the packet crosses the crossbar, it goes to the same VL at the output port. So, when due to the behavior of the arbitration tables an output VL is full, no other packets can cross toward this VL. In this way, the behavior of the arbitration tables affects the behavior of the round-robin algorithm, adapting it to the priority policy that is being applied in the output port.

  Thus, to compute the number of packets that can cross before one packet of high priority, all possible packets from all the buffers must be considered. We consider neither the internal port nor the buffers for control traffic, because these can only contain control packets, and always have priority over the remaining ones. Thus, the maximum number of packets P to consider before a certain high-priority packet crosses is:

$$P = (EPorts \times VLsPort) \times \left\lceil \frac{BS}{MTU} \right\rceil$$

  where $EPorts$ is the number of external ports of the switch, and $VLsPort$ is the number of VLs per port.

- As all the packets of the same SL use the same VL, when the high-priority packet crosses the crossbar, all the packets that it will encounter on the way belong to the same SL. So, we must add to the previous expression one packet that could be crossing the crossbar and as many packets as the buffer of the output port can have:

$$P = \left[\left(EPorts \times VLsPort \right) \times \left\lceil \frac{BS}{MTU} \right\rceil \right] + \\ +1 + \left\lceil \frac{BS}{MTU} \right\rceil$$

- The time in the output buffers will depend on the output arbitration tables. We use the term *Scan* for the maximum separation between two entries corresponding to this VL in the high-priority arbitration table, or a complete round of the table, if the VL has only one entry in the high-priority table. So, we must include this value in the previous expression:

$$P = \left[\left[\left(EPorts \times VLsPort \right) \times \left\lceil \frac{BS}{MTU} \right\rceil \right] + \\ +1 + \left\lceil \frac{BS}{MTU} \right\rceil \right] \times Scan$$

- We must also consider a possible packet leaving the switch, and the effect of the Limit of High-Priority (LHP). This limit allows the switch to send $LHP \times 4$ Kbytes of high-priority packets before sending a packet of low priority. Thus, the number of low-priority packets that can be transmitted before a high-priority packet depends on the buffer size and on the MTU. Adding this value to the previous expression we obtain:

$$P = \left[\left[\left(EPorts \times VLsPort \right) \times \left\lceil \frac{BS}{MTU} \right\rceil \right] + 1 + \\ + \left\lceil \frac{BS}{MTU} \right\rceil \right] \times Scan + 1 + \left\lceil \frac{BS}{4096 \times LHP} \right\rceil \quad (1)$$

P is the maximum number of packets that can be sent before sending a given high-priority packet. Computing the maximum delay per switch that a certain high-priority packet suffers since it arrives at the switch until it leaves is easy, once we know the MTU and the link speed:

$$T_{MaxSw} = \frac{P \times MTU}{Link\ Speed}$$

4.2 Maximum delay in a switch with full crossbar

This type of architecture (Figure 2) can be used only when the number of VLs per port is small. In this case, each VL has a dedicated input in the crossbar. In this way,

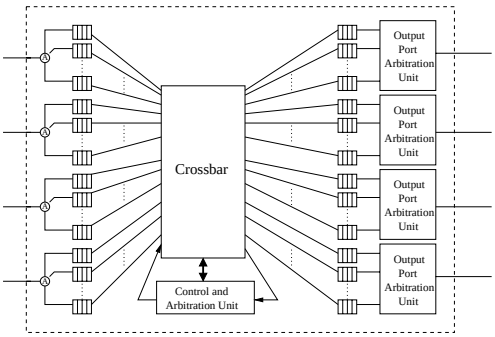

Figure 2. Possible structure of an InfiniBand switch with full crossbar.

two packets from two different VLs of the same input port could be crossing at the same time toward different output VLs. In this case, the expression (1) can be simplified. Only the packets that are in the same VL of the other input ports could cross the crossbar before a packet of high priority. The final expression is:

$$P = \left[\left[EPorts \times \left\lceil \frac{BS}{MTU} \right\rceil\right] + 1 + \left\lceil \frac{BS}{MTU} \right\rceil\right] \times Scan + \\ +1 + \left\lceil \frac{BS}{4096 \times LHP} \right\rceil \quad (2)$$

4.3 Maximum delay in a switch with a central queue

In this architecture the packets are stored in a central buffer shared among all the ports and VLs. The output arbitration algorithm selects the packets to transmit directly from this buffer. As no crossbar has to be crossed, only the packets that can leave through the same output port need to be considered. These packets are the ones considered in the previous sections as stored in the output buffer. In this case, the final expression for the maximum number of packets is:

$$P = \left\lceil \frac{BS}{MTU} \right\rceil \times Scan + 1 + \left\lceil \frac{BS}{4096 \times LHP} \right\rceil \quad (3)$$

5 A simple strategy to manage time-sensitive traffic

The problem with the expressions (1), (2), and (3) is that the $Scan$ term varies when new connections are accepted. We are forced, therefore, to recompute the arbitration table every time that a connection is accepted. Moreover, the maximum time per switch is relative to the current values of the entries in the arbitration table. So, in each switch we must store the guaranteed maximum delay for each connection in order to be able to test that it is still valid when a new connection is going to be accepted. This seems to be very complex and not viable.

We propose to put together all the SLs for time sensitive traffic in the same VL. As in the arbitration tables only VLs are considered, the high-priority table is only used by DBTS traffic, and all the SLs of high priority share the same VL, only one entry of the high-priority arbitration table needs to be used. This entry will have a weight corresponding to the proportion of low and high-priority traffic required.

With this strategy, no bandwidth can be guaranteed to the low-priority traffic. We propose to distribute the available bandwidth when the connections are accepted. As the administrator of the cluster should know the proportion of each type of traffic (s)he can configure the connection establishment so that this proportion can be satisfied.

This strategy faces the problem that the hosts must comply with the requested bandwidth. If the hosts have a bad behavior, and they use more bandwidth than requested when the connections were established, no guarantee can be assured to low priority traffic. In this approach we are going to assume that the hosts do not try to use more bandwidth than requested.

For the architectures studied in the previous section, the expressions will be simplified by setting $Scan = 1$:

- multiplexed crossbar:

$$P = \left(EPorts \times VLsPort\right) \times \left\lceil \frac{BS}{MTU} \right\rceil + 1 + \\ \left\lceil \frac{BS}{MTU} \right\rceil + 1 + \left\lceil \frac{BS}{4096 \times LHP} \right\rceil \quad (4)$$

- full crossbar:

$$P = EPorts \times \left\lceil \frac{BS}{MTU} \right\rceil + 1 + \left\lceil \frac{BS}{MTU} \right\rceil + 1 + \\ \left\lceil \frac{BS}{4096 \times LHP} \right\rceil \quad (5)$$

- central queue:

$$P = \left\lceil \frac{BS}{MTU} \right\rceil + 1 + \left\lceil \frac{BS}{4096 \times LHP} \right\rceil \quad (6)$$

We are going to compute the maximum delay that a high-priority packet could suffers when crossing a switch with the previous strategy. We consider a switch with 8 ports, each one with 4 data VLs. Furthermore, the switch has an internal port and each external port has a VL for control traffic. Each VL can store 4 packets. IBA specifications indicate that the packet size must be between 256 and 4096 bytes. We are going to consider both packet sizes and so, in order for each buffer to have capacity of four packets, we are going to consider two buffer sizes: 1024 and 16384 bytes, respectively. Finally, we assume that LHP has a weight of one. Thus, 4096 bytes of high priority can be sent before a low-priority packet can be considered. The maximum delay per switch is shown in Table 1. The values have been obtained using the expressions (4), (5), and (6) for the three architectures considered. The three possible InfiniBand link rates and both packet sizes have also been considered.

Architecture	Packet Size	1x = 2.5 Gbps	4x = 10 Gbps	12x = 30 Gbps
Multiplexed crossbar	256 bytes	0.1106 msec	27.65 μsec	9.216 μsec
Multiplexed crossbar	4096 bytes	1.81 msec	0.452 msec	0.151 msec
Full crossbar	256 bytes	31.9488 μsec	7.9872 μsec	2.6624 μsec
Full crossbar	4096 bytes	0.5505 msec	0.1376 msec	45.875 μsec
Central queue	256 bytes	4.9152 μsec	1.2288 μsec	0.41 μsec
Central queue	4096 bytes	0.118 msec	0.295 μsec	9.83 μsec

Table 1. Maximum delay for a high-priority packet in a switch with 8 ports, each one with 4 data VLs.

6 Performance evaluation

In this section, we evaluate the behavior of our proposal by means of simulation. We have used the same flit-level simulator as in [1, 2]. First, we will explain the network and the traffic models that we have used.

6.1 Network model

In [1] we used both regular and irregular networks, and we concluded that the behavior was similar in both cases. In this paper, we are going to use only irregular networks. In all cases, each switch has 8 ports, 4 ports with one host attached to each of them, while the other 4 ports are used for interconnection between switches.

To evaluate our proposal we consider that the switch has a full crossbar, similar to that shown in Figure 2. Thus, in order to compute the maximum delay per switch, the expression (5) has been considered. Each port has 4 VLs, both at the input and at the output of a switch. This approach is similar to the number of VLs that could have some future implementations [4, 6]. Each VL has its own buffer, with a size proportional to the packet sizes we tried. IBA specifies that the MTU must be between 256 and 4096 bytes, so we are going to consider these two values. For these packet sizes, the buffer sizes we have used are 1024 and 16384 bytes, respectively. Therefore, these buffer sizes allow 4 entire packets to be stored.

With respect to network size, we have evaluated networks with sizes ranging from 8 to 64 switches (with 32 to 256 hosts, respectively) and, for all cases, the results are similar. Due to space limitation, we will only include here the results for the network with 16 switches and 64 hosts. For the same reason, only the results for the link rate of 2.5 Gbps will be shown.

6.2 Traffic model

We have used CBR traffic, both for DB and for DBTS traffic. Although the results for BE and CH traffic are not the main focus of this paper, we have reserved slots for these traffic classes in the tables.

CBR traffic is randomly generated among the SLs considering the proportion of DBTS and DB traffic desired. We have considered three different traffic mixtures: 20% DBTS traffic and 60% DB traffic, 40% DBTS traffic and 40% DB traffic and 60% DBTS traffic and 20% DB traffic. The connections of each SL request mean bandwidth according with the classification done in Section 3. When no more connections can be established we start a transient period in order for the network to reach a stationary state. Once the transient period finishes, the steady state period begins, where we will gather the data to be shown in the next section. The steady state period continues until the connection with a smaller mean bandwidth has received 100 packets.

In the simulations, the deadline of each connection is computed following the expressions in this paper and the number of hops in the connection. For DBTS traffic this is similar to generating connections with a delay requirement and then trying to establish the connections, while studying in the target host (or in the Subnet Manager in a centralized way) if the delay requirements can be satisfied, and denying them if it is not possible.

6.3 Simulation results

In this section, we show the performance evaluation results. In the simulations we have reserved 20% of link bandwidth for BE traffic. The rest of link bandwidth has been distributed between DBTS traffic and DB traffic in three ways: 20% DBTS and 60% DB, 40% DBTS and 40% DB, and finally, 60% DBTS and 20% DB.

Table 2 shows the maximum delivered traffic for these mixes and for both packet sizes considered. Note that the behavior is similar for all the mixes and for both packet sizes. This table also shows the average utilization (in %) and the average bandwidth reserved (in Mbps) both in host interfaces and in switch ports. Due to lack of space, in the rest of the paper only results for the mixture 60% DBTS – 20% DB (the mixture with higher delay requirements) will be shown.

In order to obtain more information about the distribution of packet delay, the percentages of packets that meet a certain deadline threshold have been computed for the highest load. The thresholds are different for each connection and are related to their deadline. This deadline is the maximum delay that can be guaranteed to each connection following the values computed in this paper, and is obviously different for each connection. For DBTS traffic the dead-

	Traffic Mixture					
	20% DBTS – 60% DB		40% DBTS – 40% DB		60% DBTS – 20% DB	
	Small	Large	Small	Large	Small	Large
Injected/Delivered traffic (Bytes/Cycle/Node)	0.7344	0.7364	0.7292	0.7448	0.7306	0.7411
Average utilization for host interfaces (%)	73.44	73.64	72.92	74.48	73.06	74.11
Average utilization for switch ports (%)	75.01	75.24	74.73	75.75	74.55	75.40
Average reservation DBTS for host interfaces (Mbps)	423.90	421.40	928.23	930.16	1420.86	1415.29
Average reservation DBTS for switch ports (Mbps)	445.06	440.04	949.25	950.60	1432.48	1431.59
Average reservation DB for host interfaces (Mbps)	1428.55	1421.63	911.26	933.87	422.16	439.49
Average reservation DB for switch ports (Mbps)	1447.09	1442.96	935.94	945.32	448.01	455.59

Table 2. Some simulation results

line is a connection requirement in order for the connection to be accepted. For DB traffic it is not necessary to guarantee a maximum delay to accept the connection, but we are also going to study it. In both cases, this maximum delay will be computed as the maximum delay per switch times the number of hops for each connection.

In the figures the deadline is referred to as D. The results for each SL are presented in Figure 3, for large packet size (for small packet size the results are similar). We can observe that in all cases, packets arrive before $Deadline$. Besides, for SLs of DBTS traffic (SLs 0, 1, 2 or 3) about 98% of packets arrive before the threshold $\frac{Deadline}{10}$. These are very good results, which show that our simple methodology is able to provide bandwidth and delay guarantees.

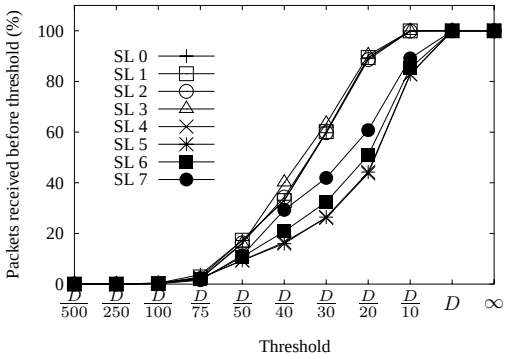

Figure 3. Distribution of packet delay.

Average packet jitter has also been measured. We have established several intervals to compute the jitter and bar graphs plot the percentage of packets received within each interval. These intervals are different for each connection, and are related to its inter-arrival time (IAT). The results are shown for each SL in Figure 4. As the results are very similar for both packet sizes considered, and due to the lack of space, only results for small packet size will be shown. The results correspond to the distribution 60% DBTS traffic and 20% DB traffic, since this is the mixture with higher delay requirements. The other distributions have an even better behavior.

In all cases, we can observe that packets from SL 0, SL 1, SL 4, and SL 5 always arrive with jitter almost equal to zero (in the interval $[\frac{-IAT}{8}, \frac{IAT}{8}]$). These SLs correspond

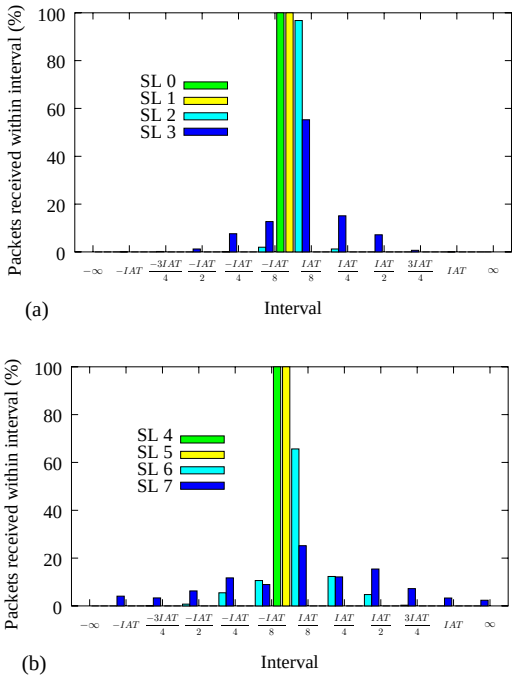

Figure 4. Average packet jitter.

to connections with low and very low bandwidth requirements, and thus, their IAT is larger than for the rest of SLs. For packets from SL2, SL 3, SL 6, and SL 7 the jitter has a Gaussian distribution never exceeding $\pm IAT$. Note that for these SLs, the behavior of DBTS packets (SL 2 and SL 3) is better than for DB packets (SL 6 and SL 7), and this is due to the treatment that we give to the high-priority packets.

Finally, for a given threshold, we have selected the connections that have delivered the lowest and the highest percentage of packets before that threshold. In the figures these connections will be referred to as the worst and the best connections, respectively. We have selected a threshold so that the percentage of packets meeting it was lower than 100%. In particular we have selected the threshold $\frac{Deadline}{10}$. The results shown in Figure 5 correspond to the distribution 60% DBTS traffic and 20% DB traffic for small packet size, and the other distributions and packet size have a similar behavior. In each figure we have shown the behavior of the worst and the best connections in the SLs with the same band-

width. Note that the connections of DBTS traffic (SLs 0, 1, 2, or 3) always have a better behavior than the connections of DB traffic (SLs 4, 5, 6, or 7).

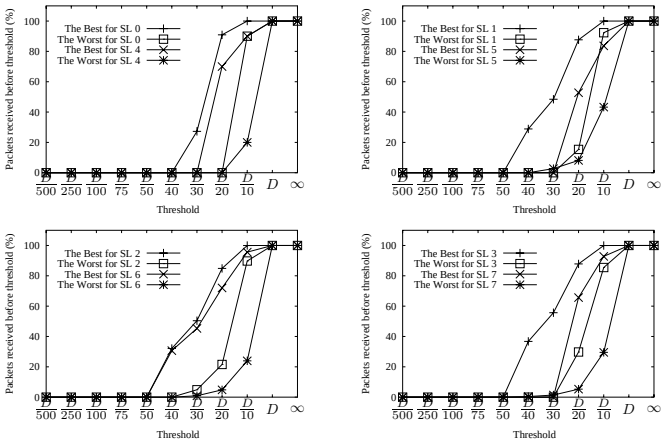

Figure 5. The best and the worst connection.

7 Conclusions

In [1, 2] we proposed a methodology to compute the virtual lane arbitration tables for InfiniBand, but we only considered it for DB traffic. In this paper, we have also included time sensitive traffic (DBTS). Our proposal has the significant advantage of not requiring recomputing the table entries for previously established connections when a new connection is established. This is because for DB traffic all the entries have been computed with respect to the same reference, and for DBTS traffic we give it higher priority while controlling the distribution between DBTS and DB traffic when establishing connections.

We have tested the behavior of the proposed methodology by simulation using CBR traffic with different mean bandwidth and deadline requirements. We have shown that all types of traffic can satisfy their QoS requirements. In particular, we have evaluated the percentage of packets that arrive before their deadline. Even for low-priority SLs (of DB traffic) their packets arrive before their deadlines.

We have also shown that all the connections sharing the same SL have very similar behavior, despite the fact they have very different mean bandwidth and delay requirements. In particular, we have seen that the best and the worst connections of each SL have a similar behavior. We have also compared the results for the connections of DBTS and DB traffic with similar bandwidth requirements. The results have shown that the worst high-priority connection always has a better behavior than the best low-priority connection.

We have also evaluated the jitter of the connections in each SL. The behavior of the jitter for DBTS connections is also better than for DB connections. Although for SLs with low bandwidth requirements all the packets arrive within the interval $[\frac{-IAT}{8}, \frac{IAT}{8}]$, for SLs with higher bandwidth requirements the jitter has a Gaussian distribution with all the packets within the interval $[-IAT, IAT]$. For these SLs with higher bandwidth requirements we can also observe that the SLs of DBTS traffic have a better behavior than the SLs of DB traffic.

The methodology proposed in this paper and in [1, 2] faces the problem that all hosts must respect the requested bandwidth. If the hosts use more bandwidth than they requested, no QoS can be guaranteed to low priority traffic. We are currently studying different solutions to this problem as well as exploring other alternative designs with the aim of improving the QoS support. Of course, we intend to test our proposal in an InfiniBand commercial product as soon as one becomes available, in order to verify the practicality of our model.

References

[1] F. Alfaro, J. Sánchez, and J. Duato. A Strategy to Compute the InfiniBand Arbitration Tables. Technical Report DIAB-01-02-20, Departamento de Informática Universidad de Castilla-La Mancha, Oct. 2001. `http://raap.info-ab.uclm.es/diab-01-02-20.pdf`.

[2] F. J. Alfaro, J. L. Sánchez, J. Duato, and C. R. Das. A Strategy to Compute the InfiniBand Arbitration Tables. In *Proceedings of International Parallel and Distributed Processing Symposium*, April 2002.

[3] P. by ISSG Technology Communications. Infiniband architectural technology. Technical Report TC000702TB, Compaq Computer Corporation, July 2000.

[4] IBM Corporation. IBM InfiniBand product advance summary datasheet. `http://www.chips.ibm.com/products/infiniband`, Aug. 2001.

[5] InfiniBand Trade Association. *InfiniBand Architecture Specification Volume 1. Release 1.0*, Oct. 2000.

[6] $InfiniBridge\ MT21108$. Product overview, Mellanox Technologies Inc., 2001. `http://www.mellanox.com/products/whitepaper.html`.

[7] $InfiniBand^{TM}\ Trade\ Association$. *http://infinibandta.com*.

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

[9] G. Pfister. *High Performance Mass Storage and Parallel I/O*, chapter 42: An Introduction to the InfiniBand Architecture, pages 617–632. IEEE Press and Wiley Press, 2001.

[10] M. Schwartz and D. Beaumount. Quality of service requirements for audio-visual multimedia services. *ATM Forum*, ATM94-0640, July 1994.

[11] K. H. Yum, E. J. Kim, and C. R. Das. QoS Provisioning in Clusters: An Investigation of Router and NIC Design. In *International Symposium on Computer Architecture (ISCA)*, July 2001.