QoS in InfiniBand Subnetworks

Francisco J. Alfaro, José L. Sánchez, and José Duato

Abstract—The InfiniBand Architecture (IBA) has been proposed as an industry standard both for communication between processing nodes and I/O devices and for interprocessor communication. It replaces the traditional bus-based interconnect with a switch-based network for connecting processing nodes and I/O devices. It is being developed by the InfiniBandSM Trade Association (IBTA) in the aim to provide the levels of reliability, availability, performance, scalability, and quality of service (QoS) required by present and future server systems. For this purpose, IBA provides a series of mechanisms that are able to guarantee QoS to the applications. In previous papers, we have proposed a strategy to compute the InfiniBand arbitration tables. In one of these, we presented and evaluated our proposal to treat traffic with bandwidth requirements. In another, we evaluated our strategy to compute the InfiniBand arbitration tables for traffic with delay requirements, which is a more complex task. In this paper, we will evaluate both these proposals together. Furthermore, we will also adapt these proposals in order to treat VBR traffic without QoS guarantees, but achieving very good results. Performance results show that, with a correct treatment of each traffic class in the arbitration of the output port, all traffic classes reach their QoS requirements.

Index Terms—InfiniBand, QoS, clusters, arbitration, performance.

1 INTRODUCTION

Input-output (I/O) buses have become a bottleneck for disk accesses, especially in high-performance servers. While buses have the major advantage of simplicity, and have served the industry well up to this point, bus-based I/O systems do not use their underlying electrical technology well enough to provide high data transfer bandwidth out of a system to devices. Despite recent upgrades, the most popular I/O bus (PCI bus) offers limited bandwidth, concurrency and reliability, and this limitation is unacceptable for a large number of current applications and server systems.

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 [2]. The InfiniBandSM 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, cochaired 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 more than 120 member companies. The first specification of the InfiniBand Architecture (release 1.0) was published in October 2000 [12], and more than 200 companies are currently supporting the InfiniBand initiative.

In a first stage, instead of directly replacing the PCI bus with a switch-based interconnect to access I/O devices, these devices are attached to a Host Channel Adapter (HCA), which is connected to the PCI bus. In this way, from HCA the communication is switched, affording the desired reliability, concurrency, and security. Moreover, it is foreseen that the PCI bus could be replaced in the near future by other advanced technologies like PCI Express Advanced Switching [3].

On the other hand, most of the current networking products have tried to achieve maximum throughput and minimum latency, leaving aside other aspects like guarantees of bandwidth, maximum latency deadline, interarrival delays, etc. [14], [15]. Many current applications need those characteristics that not all the current networks are able to provide. The current network that can best provide QoS is probably ATM [9], [10]. However, ATM focuses more upon wide area networks, and can only be used with great difficulty in LAN environments. Besides, it is not suitable for connections between a processor and its devices since it introduces significant overhead.

The Internet Engineering Task Force (IETF) has two proposals for providing QoS on the Internet. The first one is referred to as Differentiated Services [7]. This approach distinguishes among different kinds of traffics and provides different treatment for these categories. The second one is referred to as Integrated Services [8]. The later provides QoS guarantees for the applications based on their requirements requested during the connection establishment phase.

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, properly used, are able to provide QoS for 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.

InfiniBand distinguishes up to a maximum of 16 service levels, but it does not specify which characteristics will bear the traffic of those service levels. Moreover, the output ports have a table to make the arbitration among the different...
flows. However, InfiniBand does not specify how the arbitration table should be filled in. It only specifies the form that this table must have, leaving to the manufacturers’ or users’ consideration, how to fill it in according to the characteristics that they want to obtain.

An overview of a possible implementation of Differentiated Services over IBA has been described in [13]. In this study, the traffic is classified in several categories and the author proposes that the arbitration tables of InfiniBand should deal with each category in a different way, but no attempt is made to indicate how to fill in those tables.

In [5], we proposed a traffic classification able to provide QoS for each class of traffic. This traffic classification was based on the one previously proposed in [13]. We also proposed a strategy to compute the InfiniBand arbitration tables that had the significant advantage of not requiring recomputing the entries in the table for previously established connections when a new connection is established. We evaluated our proposal only with traffic with bandwidth requirements. In [4], we extended our previous proposal for traffic with delay requirements, which is a more complex task. Furthermore, in [6], we tested our proposal with VBR traffic using video sequences. In this paper, we will evaluate together all these proposals using both CBR and VBR traffic.

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, a traffic classification is proposed. Section 4 presents a way to fill in the IBA Arbitration tables. Section 5 presents a delay analysis for time sensitive traffic in InfiniBand. In Section 5.2, we explain our proposal to easily manage time sensitive traffic in InfiniBand. In Section 6, we propose how to provide QoS for VBR traffic. In Section 7, we evaluate our proposals. Finally, some conclusions are given and future work is proposed.

## 2 INFINIBAND

The InfiniBand Architecture (IBA) Specification [12] describes a system area network (SAN) for connecting multiple independent processor platforms (i.e., host processor nodes), I/O platforms, and I/O devices. The IBA SAN is a communication and management infrastructure supporting both I/O and interprocessor communications for one or more computer systems. The architecture is independent of the host operating system and processor platform.

IBA is designed around a switch-based interconnect technology with high-speed point-to-point links. 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.

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. Each packet, even those for unreliable datagrams, contains two separate CRCs, one covering data that cannot change and another covering data that changes in switches or routers, which means that the packet must be recomputed during the trip.

IBA switches route messages from their source to their destination based on forwarding tables that are programmed during initialization and network modification. The forwarding table can be linear, specifying an output port for each possible destination address up to a switch-specific limit, indexed by that address; or random, initialized by storing {destination, output port} pairs. The number of ports of a switch is vendor-specific, but is limited to 256 ports. Switches can be cascaded to form large networks. Switches may also optionally support multicast routing.

Routing between different subnets (across routers) is done on the basis of a Global Identifier (GID) 128 bits long, modeled over IPv6 addresses. On the other hand, the addressing used by switches is with Local Identifiers (LID) which allow 48K endnodes on a single subnet, the remaining 16K LID addresses being reserved for multicast.

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 subnet must contain a single master subnet manager, residing on an endnode or a switch that discovers and initializes the network.

The IBA transport mechanisms provide both connection-oriented and datagram services, and these services could be reliable (acknowledged) or unreliable. Although QoS could be provided for any transport mechanisms used, applications should use reliable connections in order to be able to carry out resource allocation and provide them with a QoS guarantee.

Basically, IBA has three mechanisms to support QoS: service levels, virtual lanes, and virtual lane arbitration. IBA defines a maximum of 16 service levels (SLs), but it does not specify what characteristics the traffic of each service level should have. Therefore, it depends on the implementation or the administrator how to distribute the different existing traffic types among the SLs.

IBA provides fields for marking packets with a class of service. By allowing the traffic to be segregated by categories, 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 receiving and transmitting buffers associated with a port (Fig. 1). Each VL must be an independent resource for flow control purposes.

IBA ports have to support a minimum of two and a maximum of 16 virtual lanes (VL0 … VL15). All ports support VL15, 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 with 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 structure of the $VLArbitrationTable$ is shown in Fig. 2. 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. More specifically, the VLs of the High_Priority table can transmit a maximum of $LimitOfHighPriority \times 4,096$ bytes before a packet from the Low_Priority table can be transmitted. If no high-priority packets are ready for transmission at a given time, low-priority packets can also be transmitted.

### 3 Traffic Classification

As it has been previously stated, IBA does not specify the characteristics of traffic at the different SLs. In [13], a traffic classification was proposed, which was based on the QoS requirements of current applications. We extended this classification in [4] to consider two priority levels for best-effort traffic. So, the traffic classification that we assume is:

- **Dedicated Bandwidth Time Sensitive traffic (DBTS).** This kind of traffic requires a given minimum bandwidth and must be delivered within a given latency in order for the data to be useful. Examples of such data streams include video conferencing and interactive audio.
- **Dedicated Bandwidth traffic (DB).** This type of traffic requires a given minimum bandwidth, but is not particularly sensitive to latency. This includes traffic that must have a bounded latency, but the current value of the bound is not critical. An example of this class of traffic could be the playback of a video clip.
- **Preferential Best Effort (PBE).** 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.
- **Best Effort (BE).** This traffic tends to be bursty in nature and is largely insensitive to both bandwidth and latency. The majority of traffic in current networks belongs to this type, like file and printing services or Web browsing.
- **Challenged (CH).** This traffic is intentionally degraded so as not to interfere with best effort traffic. It is only transmitted if no other traffic is available in the network. An example of this traffic could be disk backup activities, which we do not want to interfere with other traffic.

Although this is a generic classification, some of the referred applications have sense in a SAN or a cluster based on InfiniBand. Thus, we will use this classification to split up the traffic into different SLs.

Moreover, 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 are able to give a more adjusted treatment to each traffic class according to its needs. In fact, we can map each traffic class in one set of IBA SLs. Therefore, the categories that we distinguish for DBTS traffic (SLs 0, 1, 2, and 3) and DB traffic (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. An example of this class would be a phone conversation.
- **SL 1 and SL 5.** Connections with low bandwidth, higher than 64 Kbps, but lower than or equal to 1.55 Mbps. An example of this class would be a video conference.
- **SL 2 and SL 6.** Connections with high bandwidth, higher than 1.55 Mbps, but lower than or equal to 64 Mbps. An example of this class would be digital television transmission.
• SL 3 and SL 7. Connections with very high bandwidth, higher than 64 Mb/s. An example of this class would be high-definition television (HDTV) transmission.

Each of these categories will use a different service level and, whenever possible, we will give a different treatment to each SL so that each one can receive 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 VLs than SLs, we will put together some SLs in some virtual lanes. Nevertheless, we distinguish these four categories for each kind of traffic in need of QoS, and we will evaluate the QoS behavior of each.

4 FILLING IN THE VL ARBITRATION TABLE

Of course, the classification of traffic into categories based on its QoS requirements, is just a first step to achieve the objective of providing QoS. A suitable filling in of the arbitration table is critical.

Pelissier [13] proposed using the table of high priority for DBTS traffic and the table of low priority for the remaining traffic, but he did not specify how to compute the weights for the VLs. We agree with this proposal. Besides, we propose a strategy to fill in the weights for the arbitration tables. In this section, we see how to fill in the table in order to provide the bandwidth requested by each application. In the next section, we will study how to provide latency guarantee.

The local communication management agents are responsible for accepting or rejecting the connection requests based on the local information available in each switch. This information includes the state of the output links and how much bandwidth they have already reserved. Every time an agent accepts a connection, it modifies the arbitration table to adjust it to the connection requirements. This strategy does not vary if the modification of the arbitration table is carried out by the Subnet Manager in a centralized way.

As the entries of the arbitration table are examined in a cyclic way, the time can be split into frames of fixed duration that are repeated. So, a certain number of slots per frame will be assigned to each connection, according to its bandwidth requirements.

According to the InfiniBand specifications, each arbitration table can consist of a maximum of 64 entries, each one with a maximum weight of 255. Assuming that each slot corresponds to the transmission of 64 bytes, which corresponds to a weight of one in an entry (according to InfiniBand specifications), the maximum number of slots per frame is 64 × 255. The number of slots that a connection needs will be computed according to this value. As we shall see, this will drastically simplify the computations when connections are dynamically established.

Note that the important thing is not the particular value of the weight for a certain entry in the table, but the ratio between weights. This ratio must agree with the ratio between the mean bandwidth requirements of the accepted connections. For example, if a table has only two used entries, they will consume all the bandwidth. The ratio between these two entries defines the bandwidth that the VL of each one of these entries will obtain. If both entries have a weight of 20, both of them will consume 50 percent of available bandwidth. However, if one entry has a weight of 30, while the other only has a weight of 10, the first one will consume 75 percent of available bandwidth, while the other will use the other 25 percent. Therefore, we will compute the number of slots (and their weights) according to the mean bandwidth of the request.

However, it should be noted that the InfiniBand arbitration scheme will assign all the bandwidth to the set of established connections, even if these connections have requested a lower overall bandwidth. The reason is that once all the virtual channels in the corresponding table of priority have been serviced, the link arbitrator will repeat the cycle again. Therefore, all the link bandwidth will be consumed (assuming that there are packets ready to be transmitted) regardless of the values in the tables of priority. Thus, in order to simplify the computation of the values in the table, we decided to compute these values as if all the slots were in use (with respect to 64 × 255 slots per frame, as mentioned above). By doing so, the entries in the table do not need to be recomputed when a new connection is established because all of them have been computed with respect to the same reference. Note that each connection will receive more bandwidth than it requested unless the full link bandwidth has been reserved. But, this will happen anyway, regardless of how the values in the entries are computed. Therefore, our methodology provides bandwidth guarantee, having the significant advantage of not requiring to recompute the entries in the table for previously established connections when a new connection is established.

As each arbitration table only has 64 entries, if we devote a different entry to each connection, this could limit the number of connections that can be accepted. Moreover, a connection requiring very high bandwidth could also need slots in more than one entry in the table. For this reason, we propose grouping the connections with the same SL into a single entry of the table until completing the maximum weight for that entry, before moving to another free entry. In this way, the number of entries in the table is not a limitation for the acceptance of new connections, but only the available bandwidth.

The weight must therefore be computed according to the total mean bandwidth requested by the connections serviced by the corresponding entry of the table, and every time a new connection is accepted, only the entries in the table used by this connection need to be recomputed. Let us now compute the weight corresponding to a certain entry in the table.

We know that each frame consists of 64 × 255 = 16,320 slots of 64 bytes. For the different link rates of InfiniBand, we can compute its \( T_{frame} \), that is, the time it would take to transmit a frame assuming that all the entries in the table have assigned their maximum value. For the three link rates available in InfiniBand, \( T_{frame} \) is 3,342s36, 0.855584, and 0.278528 ms, respectively.

The bandwidth \( B \) assigned to a connection is the number of slots assigned to that connection divided by the total number of slots in the frame and multiplied by the link bandwidth. This is the percentage of the frame assigned to that connection multiplied by the link bandwidth.

\[
B = \frac{N_{Slots}}{N_{Slots per frame}} \times B_{link}.
\]

So, the number of slots for a connection with mean bandwidth \( B \) bps is
In order to guarantee that some bandwidth will be available for PBE, BE, and CH traffic, some slots must be reserved for these traffic classes. The amount of reserved bandwidth will depend on system requirements. The ratio between PBE and BE is not important at this point, and it will be established by the network administrator. For our tests, we will reserve the 20 percent of available bandwidth for PBE, BE, and CH traffics, but how this percentage is shared among them is not defined. However, for the CH traffic, almost no bandwidth is reserved, and we only set an entry in the low-priority table with weight equal to one. Therefore, this traffic will only be able to take out one packet in each frame. However, if we did not set any entry for CH traffic, the packets of this type could never be transmitted, so we must assign one entry to this traffic class, but with the minimum weight. Note that when no other traffic is available, CH traffic will be selected for packet transmission.

When DBTS or DB traffic do not use all their assigned bandwidth, this bandwidth will be used by PBE, BE, or CH traffic (if there is traffic of these categories available), according to their values in the arbitration tables. Note that the values in the arbitration table for PBE, BE, and CH traffic guarantee a minimum bandwidth, but do not limit the bandwidth available to these categories when no other traffic is available. This is a feature of the weighted round-robin arbitration policy used in InfiniBand.

### 5 Time Sensitive Traffic in InfiniBand

Due to its characteristics, DBTS traffic deserves special attention. As Pelissier proposed [13], this kind of traffic is going to use the table of high priority, while the rest of the traffic uses the table of low priority. For DBTS traffic, in addition to bandwidth guarantees, each connection must also have a guaranteed maximum latency. So, the packets of these connections must arrive at their destinations before their 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. There is very little information available about implementation details of commercial switches. However, whatever the switch architecture used, it is possible to compute the maximum delay that a packet of a certain SL will suffer when crossing a switch. As an example, in the next section, we will obtain an expression to compute this maximum crossing delay for a switch with multiplexed crossbar.

#### 5.1 Maximum Delay in a Switch with Multiplexed Crossbar

This type of switch should have a structure similar to the one shown in Fig. 3. 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.

First, the input buffer must be considered. Assuming a FIFO behavior, packets that have arrived at the buffer before a certain packet must leave before it. So, the number of packets leaving the buffer before a certain one depends on the buffer size (BS) and the maximum packet size (MTU): \( \left\lfloor \frac{BS}{MTU} \right\rfloor \).

IBA specifications do not define any arbitration mechanism 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, in order to compute the number of packets that can cross before a certain packet, all possible packets from all the buffers must be considered. We do not consider the internal port nor the buffers for control traffic, because these can only contain control packets, always having priority over those remaining. Thus, the maximum number of packets to consider before a certain packet crosses is:

\[
(EPorts \times VLSport) \times \left\lfloor \frac{BS}{MTU} \right\rfloor,
\]

where \( EPorts \) is the number of external ports, and \( VLSPort \) is the number of VLs per port.

As all the packets of the same SL use the same VL, when a certain packet crosses the crossbar, all the packets that are in its 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 contain:

\[
\left( EPorts \times VLSPort \right) \times \left\lfloor \frac{BS}{MTU} \right\rfloor + 1 + \left\lfloor \frac{BS}{MTU} \right\rfloor.
\]
The wait in the output buffers will depend on the output arbitration tables and, so, on the distribution of the entries in the table for the VL used. We use the term \( \text{Scan} \) for the maximum distance between two consecutive entries corresponding to this VL in the arbitration table, or a complete round of the table, if the VL has only one entry in the table. So, this value must be included in the previous expression:

\[
\frac{[\text{EPorts} \times \text{VLSPort}] \times [\frac{\text{BS}}{\text{MTU}}]}{1 + [\frac{\text{BS}}{\text{MTU}}]} \times \text{Scan}.
\]

Moreover, if the packet considered uses a high-priority VL placed in the high-priority arbitration table (DBTS traffic uses it), then the effect of the Limit of High-Priority (LHP) must be considered. This limit allows the switch to send \( \text{LHP} \times 4 \) 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:

\[
P = \left[\frac{[\text{EPorts} \times \text{VLSPort}] \times [\frac{\text{BS}}{\text{MTU}}]}{1 + [\frac{\text{BS}}{\text{MTU}}]} + \frac{\text{BS}}{4,096 \times \text{LHP}}\right] \times \text{Scan} + \frac{\text{BS}}{4,096 \times \text{LHP}}.
\]

(1)

\( P \) is the maximum number of packets that can be transmitted before sending a packet of a high-priority SL (used by DBTS traffic). As we know the MTU and the link speed, it is easy to compute the maximum delay per switch that a packet of a high-priority SL suffers since it arrives at the switch until it is transmitted.

\[
T_{\text{MaxSw}} = \frac{P \times \text{MTU}}{\text{Link Speed}}.
\]

A similar study could be done whatever the architecture of the switch. So, it is always possible to compute the maximum time that a packet of a high-priority SL can be waiting to leave a switch.

5.2 A Simple Strategy to Manage Time-Sensitive Traffic in InfiniBand

The main problem of (1) is that the \( \text{Scan} \) term varies when new connections are accepted. We would be forced, therefore, to recompute all the entries in the arbitration table every time 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 test that it is still valid when a new connection is going to be accepted. This seems very complex and not viable.

We propose to put together all the SLs for time sensitive traffic in the same VL. Note that, in the arbitration tables, only VLs are considered. Thus, with this proposal, only one entry of the high-priority arbitration table is needed. This is because the high-priority table is only used by DBTS traffic, and all the SLs of high priority share the same VL. The weight of this unique high-priority entry is not important because this traffic will always have priority over the other traffics.

Taking into account how the LimitOfHighPriority works, this does not allow a fine-grain distribution of bandwidth between low and high priority traffics. Therefore, with this strategy, no bandwidth can be guaranteed to the low-priority traffic. We propose to distribute the available bandwidth between high-priority and low-priority traffic when the connections are accepted. As the administrator of the cluster should know the proportion of each type of traffic, (s)he could configure the connection establishment so that this proportion could be satisfied.

This strategy faces the requirement that the hosts must comply with the requested bandwidth. If the hosts have a bad behavior, and they use more bandwidth than the requested one when the connections were established, no guarantee can be given to low-priority traffic. InfiniBand does not provide support in the switches to guarantee that a host will not consume more bandwidth than the requested one during the connection establishment phase. Thus, we assume trusted hosts that will not try to consume more bandwidth than the one allocated to them. Note that this is not an Internet scenario, but in a SAN or a cluster based on InfiniBand, the assumption about trusted host is reasonable.

Thus, whatever the switch architecture used, the expression for computing the maximum number of packets that can be sent before transmitting a packet of a high-priority SL will be simplified by setting \( \text{Scan} = 1 \). For the architecture studied in the previous section (switch with multiplexed crossbar), the simplified expression is:

\[
P = \left(\frac{\text{EPorts} \times \text{VLSPort}}{1 + \frac{\text{BS}}{\text{MTU}}} + \frac{\text{BS}}{4,096 \times \text{LHP}}\right) \times \text{Scan} + \frac{\text{BS}}{4,096 \times \text{LHP}}.
\]

(2)

Therefore, with this simple strategy, a final expression can be computed independently of the \( \text{Scan} \) term. So, the maximum delay per switch that a certain high-priority packet suffers since it arrives at the switch until it leaves can be easily computed. Thus, the maximum delay requested by a connection during its establishment period can be studied in a distributed way by the local agents, or in a centralized way by the subnet manager.

6 QoS for VBR Traffic

In the previous sections, we have proposed a methodology to provide QoS guarantee for the applications. This method is based on mean bandwidth and maximum delay guarantee. In order to achieve this guarantee, our methodology needs to know the mean bandwidth and maximum latency requirements of the applications. However, VBR traffic cannot be treated in the same way. Due to the variability of VBR traffic, it is not possible to provide this traffic with any guarantee. A reservation based on the mean bandwidth is not able to guarantee QoS for this kind of traffic.

Although it is not possible to provide VBR traffic with QoS guarantee, it is possible to support different service types as Differentiated Services do [7]. So, our proposal is to treat VBR traffic as DB traffic, using the mean bandwidth of each connection to allocate slots in the low-priority arbitration table. As the mean bandwidth may not be enough, we will test different values to admit connections. Specifically, the values...
used in our tests are 110 percent, 120 percent, 130 percent, 140 percent, and 150 percent of the mean bandwidth.

Moreover, for video transmission, VBR traffic also has latency requirements. Each video frame must arrive at its target by 33.3 ms. However, as explained before, it is not possible to provide a guarantee for this kind of traffic. Although the network can try to meet this requirement, it cannot offer a guarantee.

In this way, an SL and a VL is granted to VBR traffic. This kind of traffic has its own buffer and, due to the treatment given by the arbitration table, the variability of this kind of traffic will never affect other traffic types. Guarantees given to the other traffic types can then be satisfied. However, the variability of VBR traffic could affect the connections that use the same VL. But, as all these connections are of type VBR, none of them need guarantees and they must be satisfied with the received treatment.

7 Performance Evaluation

In this section, we will evaluate the behavior of our proposals using simulation. We have developed a flit-level simulator that models the InfiniBand behavior. First, we will evaluate the strategy proposed in Section 4 to fill in the arbitration table to provide the applications with bandwidth guarantee. These results are partially shown in [5]. Next, we will test that the proposal presented in Section 5.2 is also able to provide QoS for the connections with latency requirements. These results are partially shown in [4]. Finally, we are going to test the behavior of VBR traffic using the strategy proposed in Section 6. These results are partially shown in [6]. Before showing the results obtained, we will explain the network and traffic models used in our experiments.

7.1 Network Model

We have used both regular and irregular networks. Regular networks have been chosen with hypercube and mesh topology, while irregular ones have been randomly generated. In all cases, each switch has eight ports, four of them devoted to interconnection between switches, and the other four ports with one host attached to each of them.

Each switch has a crossbar multiplexed among the VLs, with the same number of inputs as output ports in the switch. Therefore, for the networks analyzed in this paper, with 8-port switches, all the switches have crossbars of size $8 \times 8$. To prevent the crossbar from becoming a bottleneck, crossbar ports operate twice as fast as links, so that they are able to take out every incoming packet [1], [16].

According to the InfiniBand specifications, all SLs sharing the same VL receive the same treatment. So, in order to be able to provide a different VL for each SL, we need 11 data VLs. However, this is not a valid value in InfiniBand. Therefore, in our tests, and unless otherwise stated, each port has 15 data VLs both on the switches and on the hosts.

Each VL has its own buffer, with a size proportional to the packet sizes we use. IBA specifies that the MTU must be between 256 and 4096 bytes, so we consider these two values. For these packet sizes, the buffer sizes we have used are 1,024 and 16,384 bytes, respectively, which therefore, allow four entire packets to be stored in each case.

We have evaluated networks with sizes ranging from 8 to 64 switches. As each switch has four hosts attached, we have therefore used networks ranging from 32 to 256 hosts.

Moreover, we have made some preliminary tests using the three link rates specified in InfiniBand. The results obtained for the main parameters are similar. This is because the link rate has not influence on the parameters studied for the QoS. Instead, the most important parameter is the load reached by the network. When the load is close to the maximum network throughput, a correct resource reservation and an accurate arbitration are critical regardless of link bandwidth. Without a correct arbitration, packets may suffer delays that could affect their QoS requirements. Therefore, we will only include in the paper the results for the link rate of 2.5 Gbps, but the conclusions can be extended to the other link rates specified in InfiniBand.

7.2 Traffic Model

We have used both CBR and VBR traffic. Although the results for PBE, BE, and CH traffic are not the main focus of this paper, we have reserved slots for these traffic classes in the tables. VBR traffic is MPEG-4 video sequences. CBR traffic is randomly generated among the SLs considered in Section 3, and the requested mean bandwidth is uniformly distributed across the range for each SL. Specifically, we will establish connections on the following mean bandwidth ranges:

- SL 0 and SL 4: CBR connections with very low bandwidth ranging between 8 and 64 Kbps.
- SL 1 and SL 5: CBR connections with low bandwidth ranging between 64 Kbps and 1.55 Mbps.
- SL 2 and SL 6: CBR connections with high bandwidth ranging between 64 Kbps and 64 Mbps.
- SL 3 and SL 7: CBR connections with very high bandwidth ranging from 64 Mbps to 300 Mbps.
- SL 8: connections using VBR traffic.

Note that the SLs 0, 1, 2, and 3 are for DBTS traffic, and the SLs 4, 5, 6, 7, and 8 are for DB traffic. All of them use CBR traffic except SL 8, which uses VBR traffic. We can observe that the bandwidth range for each SL used for CBR traffic is very wide and therefore packets with very different characteristics are going to coexist in the same buffers and in the same SL.

For VBR traffic, we have used video sequences encoded according to the NTSC CCIR 601 format at 25 frames/sec. with Q factor equal to 1 and 8. The features of the traces used are given in Table 1.

In all cases, we will increase the traffic in the network by increasing the number of established connections. For CBR traffic, these connections are equally distributed among the different SLs used in the test. This way of establishing connections has the drawback that there may still be room in the network for more connections of smaller bandwidth. But, we will see in the next section that we can obtain quite high loads, and we will be able to compare the results obtained for each SL, because there are, more or less, the same number of established connections for each SL.

As we already mentioned, packets with 256 and 4,096 bytes have been considered. These sizes correspond to the minimum and the maximum packet size allowed in InfiniBand, packet headers included. Thus, for packet size equal to 256 bytes, the header represents a larger overhead for the network, but since it is smaller, it also has the advantage that the link takes up less time to transmit a packet.

For CBR traffic, when no more connections can be established, a transient period starts in order for the
network to reach a stationary state. This transient period lasts until 10,000 packets have arrived at their destinations. Once the transient period finishes, the steady state period begins. The measurements shown in this paper for CBR traffic have been collected during the steady state period. It is enough to receive a few packets. Thus, the period for collecting statistics ends when the connection with the lowest mean bandwidth has received 100 packets. At this moment, the other connections (with larger mean bandwidth) have received many more packets.

For simulations with VBR traffic, the transient period allows the network to be loaded. After that, the period for collecting statistics starts. This period lasts for 10 seconds, which is long enough for each video sequence to be transmitted at least once.

In the simulations, the deadline of each connection is computed following the expressions presented in this paper and the number of switches crossed by the connection. This is similar to generate connections with a delay requirement and to study in the target host (or in the Subnet Manager in a centralized way) whether the delay requirement can be satisfied or not, rejecting the connections in the last case.

7.3 Simulation Results for Bandwidth Guarantee

In this section, we show the performance evaluation results for the first of our proposals (Section 4). We want to test that our strategy of filling in the arbitration tables is able to provide bandwidth guarantee for the applications. These results are partially shown in [5].

We have reserved 20 percent of link bandwidth for PBE, BE, and CH traffic. How this percentage is shared among them falls beyond the scope of this work. So, for link speed of 2.5 Gbps, the maximum bandwidth available for CBR traffic is 2 Gbps.

Because, in this section, we only want to test that our strategy of filling in the arbitration tables is able to provide bandwidth guarantee, only DB traffic of type CBR (SLs 4, 5, 6, and 7) is used. As each switch or host port has 16 VLs, it is possible to devote one VL for each SL. So, each traffic class (SL) uses a different buffer. Moreover, in order to make a good use of available resources, the network should achieve a high performance.

For each topology tested, we can see in Table 2 the injected and delivered traffic (in bytes/cycle/node), the average utilization (in percent) and the average bandwidth reserved (in Mbps) in host interfaces and switch ports, and the number of established connections. Note that the behavior is similar for all the networks and for both packet sizes. With our proposal, the network can reach a throughput close to 80 percent, which is the maximum available in this case, because the other 20 percent is reserved for other traffics. For small packet size, the network reaches a slightly higher throughput. This is due to the overhead introduced by packet headers, which is more significant for small packet size. Moreover, the smaller the packet size, more packets must be sent. Note that connections have been accepted until the load approaches 80 percent.

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 type of connection, and are related to their IAT (interarrival time). The results for each network and packet size are shown in Fig. 4. In all cases, we can observe that packets from SL 4 and SL 5 always arrive with jitter almost equal to zero (in the interval $\pm \frac{1}{8}$). For packets from SL 6 and SL 7, whose connections have lower IAT, the jitter has a Gaussian distribution never exceeding $\pm IAT$. Note also that the behavior is similar for all networks and packet sizes.

### Table 1: Features of the MPEG-4 Video Sequences Used

<table>
<thead>
<tr>
<th>Trace</th>
<th>Frames</th>
<th>Factor Compr.</th>
<th>Mean Bw (Mbps)</th>
<th>Peak Bw (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ayer</td>
<td>240</td>
<td>Q1</td>
<td>41.65</td>
<td>56.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q8</td>
<td>5.87</td>
<td>10.14</td>
</tr>
<tr>
<td>comp</td>
<td>300</td>
<td>Q1</td>
<td>43.99</td>
<td>67.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q8</td>
<td>6.16</td>
<td>10.90</td>
</tr>
<tr>
<td>flow</td>
<td>150</td>
<td>Q1</td>
<td>55.62</td>
<td>70.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q8</td>
<td>9.16</td>
<td>19.05</td>
</tr>
<tr>
<td>foot</td>
<td>40</td>
<td>Q1</td>
<td>49.03</td>
<td>61.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q8</td>
<td>6.91</td>
<td>10.61</td>
</tr>
<tr>
<td>hook</td>
<td>240</td>
<td>Q1</td>
<td>35.30</td>
<td>52.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q8</td>
<td>3.75</td>
<td>7.58</td>
</tr>
<tr>
<td>mart</td>
<td>240</td>
<td>Q1</td>
<td>32.70</td>
<td>53.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q8</td>
<td>2.22</td>
<td>6.75</td>
</tr>
<tr>
<td>mobi</td>
<td>150</td>
<td>Q1</td>
<td>50.38</td>
<td>76.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q8</td>
<td>5.82</td>
<td>20.50</td>
</tr>
<tr>
<td>tenn</td>
<td>150</td>
<td>Q1</td>
<td>44.62</td>
<td>75.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q8</td>
<td>3.99</td>
<td>19.42</td>
</tr>
</tbody>
</table>

### Table 2: Traffic and Utilization for Different Topologies and Packet Sizes for the Proposal of Section 4

<table>
<thead>
<tr>
<th>Topology</th>
<th>Irregular</th>
<th>Hypercube</th>
<th>Mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Small</td>
<td>Large</td>
<td>Small</td>
</tr>
<tr>
<td>Injected traffic (Bytes/Cycle/Node)</td>
<td>0.7691</td>
<td>0.7571</td>
<td>0.7675</td>
</tr>
<tr>
<td>Delivered traffic (Bytes/Cycle/Node)</td>
<td>0.7691</td>
<td>0.7571</td>
<td>0.7675</td>
</tr>
<tr>
<td>Average utilization for host interfaces (%)</td>
<td>79.32</td>
<td>75.86</td>
<td>79.15</td>
</tr>
<tr>
<td>Average utilization for switch ports (%)</td>
<td>79.90</td>
<td>77.50</td>
<td>79.85</td>
</tr>
<tr>
<td>Average reservation for host interfaces (Mbps)</td>
<td>1920.69</td>
<td>1892.67</td>
<td>1916.68</td>
</tr>
<tr>
<td>Average reservation for switch ports (Mbps)</td>
<td>1938.41</td>
<td>1933.69</td>
<td>1958.08</td>
</tr>
<tr>
<td>Established connections</td>
<td>2623</td>
<td>2631</td>
<td>2598</td>
</tr>
</tbody>
</table>
Finally, in order to test the scalability of our proposal, we have tested several configurations varying the network size from 8 to 64 switches. In all cases, each switch has four hosts attached. So, these configurations have from 32 to 256 hosts sending and receiving packets. As all the topologies and packet sizes tested obtain similar results, in this test, only the irregular topology and small packet size will be used.

The results obtained can be seen in Table 3. In all cases network size has not influence upon the results obtained. The utilization is always close to the maximum possible. So, our proposal to give bandwidth guarantee scales well and is able to provide for the established connections the bandwidth required during the establishment phase, regardless of the network size.

Therefore, all the connections established achieve the bandwidth previously requested, regardless of the topology, the network size or the packet size used. This is because our proposal carries out a correct resource reservation, and a good arbitration is performed in the output ports. Moreover, the network achieves a good throughput.

7.4 Simulation Results for Delay Guarantee
In the previous section, we have concluded that our proposal of Section 4 is able to provide bandwidth guarantee for each connection. In this section, we will show that, with the proposal of Section 5.2, it is also possible to provide delay guarantee when needed. As we have seen in the previous section, the results do not vary for regular or irregular networks, so we will only use now an irregular one. In this case, we will only use CBR traffic of type DBTS (SLs 0, 1, 2, and 3) and DB (SLs 4, 5, 6, and 7). Expression (2) has been considered to compute the maximum delay per switch. According to the proposal of Section 5.2, all DBTS SLs share the same VL, while the DB SLs share a different SL. Therefore, with this proposal, we only need each port to have four VLs. This approach is similar to the number of VLs that some implementations have [11].

CBR traffic is randomly generated among the SLs considering the desired proportion of DBTS and DB traffic. We have considered three different traffic mixtures: 20 percent DBTS traffic and 60 percent DB traffic, 40 percent DBTS traffic and 40 percent DB traffic, and 60 percent DBTS traffic and 20 percent DB traffic. Note that the remaining 20 percent is reserved for PBE, BE and CH.

### Table 3

<table>
<thead>
<tr>
<th>Number of switches</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Injected traffic (Bytes/Cycle/Node)</td>
<td>0.7701</td>
<td>0.7691</td>
<td>0.7575</td>
<td>0.7356</td>
</tr>
<tr>
<td>Delivered traffic (Bytes/Cycle/Node)</td>
<td>0.7701</td>
<td>0.7691</td>
<td>0.7575</td>
<td>0.7356</td>
</tr>
<tr>
<td>Average utilization for host interfaces (%)</td>
<td>79.86</td>
<td>79.32</td>
<td>78.65</td>
<td>77.71</td>
</tr>
<tr>
<td>Average utilization for switch ports (%)</td>
<td>79.95</td>
<td>79.90</td>
<td>78.85</td>
<td>77.92</td>
</tr>
<tr>
<td>Average reservation for host interfaces (Mbps)</td>
<td>1952.67</td>
<td>1920.69</td>
<td>1896.68</td>
<td>1889.04</td>
</tr>
<tr>
<td>Average reservation for switch ports (Mbps)</td>
<td>1963.69</td>
<td>1938.41</td>
<td>1908.08</td>
<td>1895.06</td>
</tr>
<tr>
<td>Established connections</td>
<td>1361</td>
<td>2623</td>
<td>4928</td>
<td>9235</td>
</tr>
</tbody>
</table>
Table 4 shows the maximum delivered traffic for these traffic mixtures 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 percent) and the average bandwidth reserved (in Mbps) both in host interfaces and in switch ports. Note that the utilizations in this case are lower than the ones shown in Table 2. This is due to the mixture of DBTS and DB traffic and the reservation made for each one of these categories.

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 three loads considered. The thresholds are different for each connection and are related to their own deadline. This deadline is the maximum delay that can be guaranteed for each connection following the values computed in this paper, and is obviously different for each connection. For DBTS traffic, the deadline is a connection requirement in order for the connection to be accepted. However, for DB traffic, it is not necessary to guarantee a 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 multiplied by the number of hops for each connection.

In the figures, the deadline is referred to as $D$. The results for each SL and for both packet sizes are presented in Fig. 5. We can observe that, in all cases, packets arrive before their Deadline. Besides, for SLs of DBTS traffic (SLs 0, 1, 2, or 3), about 98 percent of packets arrive before the threshold Deadline. These are very good results, which show that our simple methodology is able to provide bandwidth and delay guarantees for the applications that requested it during the establishment phase.

Average packet jitter has also been measured. As mentioned previously, we have established several intervals to compute the jitter and bar graphs plot the percentage of packets received within each interval. These intervals are

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

Fig. 5. Distribution of packet delay for: (a), (b), and (c) small packet size and (d), (e), and (f) large packet size. (a) and (d) have 20 percent of DBTS traffic and 60 percent of DB traffic, (b) and (e) 40 percent of DBTS traffic and 40 percent of DB traffic, and (c) and (f) 60 percent of DBTS traffic and 20 percent of DB traffic.
different for each connection, and are related to its interarrival time (IAT). The results for each SL are shown in Fig. 6. The results correspond to the mixture of 60 percent DBTS traffic and 20 percent DB traffic, since this is the mixture with the highest delay requirements. The other distributions have even better behavior.

In all cases, the results of jitter for DBTS packets is better than for DB packets, which is due to the treatment that we give to the high-priority packets. Moreover, we can observe that packets from SL 0, SL 1, SL 4, and SL 5 always present a jitter almost equal to zero (in the interval $[\frac{-IAT}{C_0}, \frac{IAT}{C_1}]$). These SLs correspond to connections with low and very low bandwidth requirements and, thus, their IAT are larger than for the other SLs. For packets from SL2, SL 3, SL 6, and SL 7, the jitter has a Gaussian distribution never exceeding $\pm IAT$.

Finally, for a given threshold, we have selected the connections that have delivered the lowest and the highest percentage of packets before a certain 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 percent. In particular, we have selected the threshold $\text{Deadline}^{10}$. The results shown in Fig. 7 correspond to the mixture of 60 percent DBTS traffic and 20 percent DB traffic for small packet size, and the other distributions and packet size have a similar behavior. Each figure shows the behavior of the worst and the best connections in the SLs with the same bandwidth. 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). Note that there is not a significant difference between the best and the worst connection of each SL. This is because the arbitration treats all the connections belonging to the same SL in a similar way. These are very good results showing that our proposals are able to provide bandwidth and delay guarantee for the applications that have requested it during the establishment period.

7.5 Simulation Results for VBR Traffic

In the previous sections, we have shown our proposals are able to provide bandwidth and/or delay guarantee for the applications. We have previously stated that it is not possible to provide any kind of guarantee for VBR traffic. In this section, we are going to test our proposals with VBR traffic in order to give this kind of traffic a good treatment, but obviously, without guarantees.

In this section, we will use only VBR traffic of the video sequences shown in Table 1. These sequences are encoded according to the NTSC CCIR 601 format at 25 frames/sec, with Q factor equal to 1 and 8. All the connections for video of quality Q1 share the same VL. A second VL is used for the video of quality Q8.

In order to make the bandwidth reservation, we have tested several values that are always based on the mean bandwidth of each connection. Specifically, these values are 110 percent, 120 percent, 130 percent, 140 percent, and 150 percent of the mean bandwidth. This requirement is studied in each switch in a distributed way. Each time a connection is...
denied, another of the same video sequence is tried. After several tries without connection establishments, we have established the maximum number possible of connections. At this moment, a transient period begins, which permits the network to be loaded. After that, the period for collecting statistics starts. This period lasts for 10 seconds (when a video sequence finishes it is restarted), which is long enough for each video sequence to be transmitted at least once.

For each configuration, and for both packet sizes tested, we show in Table 5 the number of established connections.

Note that different configurations are able to accept a different number of connections but the values for all of them are similar. Other interesting results are shown in this table. As all the different configurations tested have a similar behavior, in the rest of this paper, we will only show results for the case in which the mean bandwidth of the connection alone is considered.

In order to see that all the sequences achieve their QoS requirements, we have computed the percentage of video frames that arrive at their targets by a given threshold. The thresholds, which are of width \(0.3\) msec, are close to the maximum of \(33.3\) msec. Fig. 8 shows how to compute the delay for each frame.

Table 5
Some Simulation Results for VBR Traffic

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>VBR Factor Used</th>
<th>VBR Factor Used</th>
<th>VBR Factor Used</th>
<th>VBR Factor Used</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bw x 110%</td>
<td>Bw x 120%</td>
<td>Bw x 130%</td>
<td>Bw x 140%</td>
</tr>
<tr>
<td>Small</td>
<td>1095</td>
<td>1091</td>
<td>1087</td>
<td>1079</td>
</tr>
<tr>
<td>Large</td>
<td>1126</td>
<td>1127</td>
<td>1111</td>
<td>1127</td>
</tr>
<tr>
<td>Small</td>
<td>0.793</td>
<td>0.781</td>
<td>0.795</td>
<td>0.789</td>
</tr>
<tr>
<td>Large</td>
<td>0.810</td>
<td>0.810</td>
<td>0.798</td>
<td>0.807</td>
</tr>
<tr>
<td>Small</td>
<td>79.32</td>
<td>78.08</td>
<td>79.52</td>
<td>78.93</td>
</tr>
<tr>
<td>Large</td>
<td>80.92</td>
<td>80.59</td>
<td>79.86</td>
<td>80.67</td>
</tr>
<tr>
<td>Small</td>
<td>79.32</td>
<td>78.08</td>
<td>79.52</td>
<td>78.93</td>
</tr>
<tr>
<td>Large</td>
<td>80.92</td>
<td>80.59</td>
<td>79.86</td>
<td>80.67</td>
</tr>
</tbody>
</table>

![Fig. 8. Distribution of the packets of a frame in the time and how to compute the delay for a frame.](image)

\[
\text{Delay}_n = t_{n+1} - (n \times 33.3 + t_0) \leq 33.3
\]

\[n = 0, \ldots, \infty\]
Q8. In all cases, all the frames arrive in time to the decoder. Note that frames with Q8 compression factor have a better behavior than the frames with a Q1 compression factor. This is because, having a greater compression factor, fewer packets are used to transmit the whole frame and, so, the last packet of the frame is sent sooner. This is also the reason why when a large packet size is used the frames arrive sooner than when the small packet size is used.

We have also computed the delay of each packet transmitted. In Fig. 10, the percentage of packets whose delays are lower than a set of thresholds has been computed. These thresholds are related to the maximum deadline (referred in the figures as D) for each connection. Although this deadline is not a requirement for this kind of traffic in order to accept the connection, it is a good performance index that can show if the packets transmitted are suffering delay. Remember that this deadline is different for each connection because each connection could cross a different number of switches. In all cases, all the packets arrive at their targets before the threshold $D_{10}$, all of them achieving their delay requirements.

Finally, we have studied the influence of mixing connections with very different mean bandwidth on the same virtual lane. Given a deadline, we have selected the connections that have delivered the lowest and the highest percentage of packets before that deadline. We have selected a very tight deadline so that the percentage of packets meeting the deadline was lower than 100 percent. In particular, we have selected a deadline equal to $D_{30}$. In the figures, these connections will be referred to as the worst and the best connection, respectively. Fig. 11 shows the best and the worst connection for both packet sizes and both Q factors. Note that there is not a lot of difference between the one considered the best and the connection considered the worst. In all cases, all the connections have a similar behavior. This is because the arbitration performed is able to give each connection a similar treatment based on its requirements.

Therefore, all video frames arrive at their targets on time to be decoded and shown. All packets of each video frame receive a similar treatment. Besides, all the connections have a behavior similar and there are not a lot of differences between the two connections considered to be the best and the worst. Thus, our proposal is able to provide QoS for VBR traffic but, obviously, without giving guarantees.

8 CONCLUSIONS

In [5], we have proposed a methodology to compute the virtual lane arbitration tables for InfiniBand, but we only have considered it for DB traffic. In [4], we have tested our proposal to also include time sensitive traffic (DBTS). Here, we have put all together and we have studied the influence of the mixture of both kind of traffic.

Our proposal has the significant advantage of not requiring the entries in the table to be recomputed 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 and VBR traffic with different mean bandwidth and deadline requirements. We have seen that the maximum utilization that can be reached is almost the maximum available for both small and large packet sizes, regardless of the network size or topology. We have shown that all traffic types 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 deadlines. Although these kinds of applications do not have a delay requirement in order to establish the connections, these are a very good result because they show that these connections also suffer low delay.

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 \([-IAT, IAT]\), 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.

In the last section, the proposals have been evaluated with VBR traffic of video sequences. We have shown that, although it is not possible to provide any kind of guarantee for this kind of traffic, it does, however, satisfy its requirements. In our tests, all video frames have arrived at their targets early enough before their deadlines expire.

Therefore, the proposals made are able to provide bandwidth and/or delay guarantee for this kind of traffic with a constant behavior (CBR traffic). However, with a correct configuration, the VBR traffic also satisfies its requirements, but without any kind of guarantees.

However, our methodology faces the problem that all hosts must respect the requested bandwidth. If the hosts use more bandwidth than requested, no QoS can be guaranteed for low priority traffic. We are currently studying different solutions to this problem as well as exploring other alternative designs with the aim of improving 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