# Providing QoS in InfiniBand for Regular and Irregular Topologies *

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

Luis Orozco
School of Information
Technology and Engineering
University of Ottawa
161 Louis Pasteur,
Ottawa, Ontario, Canada
lbarbosa@uottawa.ca

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

**Abstract**

*The InfiniBand Architecture (IBA) is becoming an industry standard for communication between processing nodes and I/O devices or for interprocessor communication. 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.*

*In [1] we proposed a new strategy to address these issues. We have evaluated this new strategy only for irregular topology networks [4]. In this paper we evaluate our proposal for regular topologies (hypercube and mesh) and we compare the results obtained. In this way, we want to study the influence of the topology on the QoS mechanisms.*

***Keywords***: *InfiniBand, Arbitration, QoS, NOWs.*

## 1. INTRODUCTION

A group of the most important IT companies have joined to develop a standard for communication between processing nodes and I/O devices as well as for interprocessor communication, known as InfiniBand [6]. 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.

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.

*This work was partly supported by the Spanish CICYT under Grant TIC2000-1151-C07, and Junta de Comunidades de Castilla-La Mancha under Grant PBC-02-008..

In [1] we have proposed and evaluated a new strategy to compute the InfiniBand arbitration tables. In this paper, we evaluate our proposal for different topologies (hypercube and mesh).

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, our strategy to manage traffic in InfiniBand is explained. In Section 4, this proposal is evaluated for several topologies. Finally, some conclusions are given.

## 2. INFINIBAND

The InfiniBand Architecture (IBA) Specification describes a System Area Network (SAN) for connecting multiple independent processor platforms (i.e., host processor nodes), I/O platforms and I/O devices.

IBA has been 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. Routing in IBA subnets is performed, based on forwarding tables stored in each switch. IBA supports any topology defined by the user, including irregular ones, in order to provide flexibility and incremental expansion capability.

IBA links are 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. Physical links may be used in parallel to achieve greater bandwidth. Currently, IBA defines three link bit rates: 2.5, 10, and 30 Gbps corresponding to 1-bit, 4-bit, and 12-bit wide links, respectively. The width or widths that will be

supported by a link is vendor-specific.

The IBA transport mechanisms provide several types of communication services: reliable and unreliable connection and connectionless services. Obviously, for supporting QoS guarantee, the applications must use reliable connections in order to be able to do resource allocation.

IBA has three mechanisms that permit QoS to be supported: 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. Therefore, it depends on the implementation or administrator how to distribute the different existing traffic types among the SLs. IBA provides fields for marking packets with a class of service. Allowing the traffic to be segregated by category, IBA is able to distinguish among packets from different SLs and to treat them 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 represents a set of transmit and receive buffers in a port. Each VL must be an independent resource for flow control purposes. IBA ports must support a minimum of two and a maximum of 16 virtual lanes ($VL_0 \ldots VL_{15}$). 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 a SL to VL Mapping Table.

When more than two VLs are implemented, the priorities of the data lanes are defined by the VLArbitrationTable. This arbitration is only for data VLs, because $VL_{15}$, which transports control traffic, always has priority over any other VL. Each VLArbitrationTable has two tables, one for scheduling packets from high-priority VLs and another one for low-priority VLs. 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 $LimitOfHighPriority \times 4096$ bytes before a packet from the low-priority table could be transmitted. If no high-priority packets are ready for transmission at a given time, low-priority packets can be transmitted.

## 3. OUR PROPOSAL

In [1] we have proposed a new strategy to fill in the virtual lane arbitration tables of InfiniBand. We put all the traffic with requirements about latency or about bandwidth in the high-priority arbitration table. So, the low-priority arbitration table is only devoted for traffic that does not need any guarantee.

In [3] we studied how to provide bandwidth guarantee to the connections. We showed that a certain bandwidth requirement turns into a weight to put in the arbitration table for the VL that these connections is using. On the other hand, in [2] we studied how to provide latency guarantees. We showed that a maximum end-to-end latency can be turned in each switch crossed, into a maximum distance between two consecutive entries in the high-priority table for the VL used.

We classify the traffic in SLs based on the maximum latency required. For a certain connection requesting a maximum distance $d$ and a mean bandwidth, that turns in a weight $w$, the number of entries needed in each switch crossed is $max\{\frac{64}{d}, \frac{w}{255}\}$. However, not all possible values are permitted. We round up to distances power of 2 (for details see [1]). The available distances are 2, 4, 8, 16, 32, and 64, being the last one devoted to traffic without delay requirement or for traffic having big enough latency requirement that only needs an entry in the table.

We have developed an algorithm to maximize the number of requests to be allocated in the high-priority table that can be accepted. This algorithm achieves to meet a request in the table if there is enough available entries. It sets the requests in an optimal way being able later to put the most restrictive possible request.

Let be an arbitration table $T$, the sequence $t_0, t_1, \ldots, t_{62}, t_{63}$ represents the entries of this table. Each $t_i$ has an associated weight $w_i$ whose value can vary between 0 and 255. We say an entry $t_i$ is *free* if an only if $w_i = 0$.

For a table $T$ and a request of distance $d = 2^i$, we define the sets $E_{i,j}$ with $i = \log_2 d$ and $0 \leq j < d$, as

$$E_{i,j} = \left\{ t_{j+n\times 2^i} \quad n = 0, \ldots, \frac{64}{2^i} - 1 \right\}$$

Each $E_{i,j}$ contains the entries of the table $T$ separated between two consecutive ones with a distance $d$, that are able to attend a request of distance $d = 2^i$ starting with the entry $t_j$. We say a set $E_{i,j}$ is free if $\forall t_k \in E_{i,j}$, $t_k$ is free. Other properties derived from this definition are available in [1].

For a new request of distance $d = 2^i$, our algorithm inspects all possible sets $E_{i,j}$ and selects the first one that is free (so, it has all its entries free). The order in which the sets are inspected is based on the application of the bit-reversal permutation to the values in the interval $[0, d-1]$. Specifically, for a new request of maximum distance $d = 2^i$, the algorithm selects the first $E_{i,j}$ free in the sequence

$$E_{i,{}_iR_0}, E_{i,{}_iR_1}, \ldots, E_{i,{}_iR_{d-1}}$$

where ${}_iR_j$ is the bit-reversal function applied to $j$ codified with $i$ bits.

For example, the order to inspect the sets for a request of distance $d = 8 = 2^3$ is $E_{3,0}$, $E_{3,4}$, $E_{3,2}$, $E_{3,6}$, $E_{3,1}$, $E_{3,5}$, $E_{3,3}$, and $E_{3,7}$. Note that this algorithm first fills in the even entries and the odd entries later. In this way, if we have available entries, we can always attend a request of distance 2, the most restrictive one. The same can be applied to other longer distances.

In [1] we have proved several theorems showing that with this algorithm we can always meet a new request if there are enough available entries. This is due to the fact that our algorithm always selects the sequences in the optimal way to be able to attend later to the most restrictive possible request.

## 4. PERFORMANCE EVALUATION

In this section, we evaluate the behavior of our proposal via simulation [3]. We have used irregular and regular topologies. The regular ones have hypercube and mesh topology. The irregular ones have been randomly generated. In all cases, switches have 8 ports, 4 of them with a host attached, and the other 4 links are used for interconnection between switches. Both of input and output ports have 16 VLs in order to permit that each SL has its own VL. Each VL is big enough to store four whole packets. Each switch has a multiplexed crossbar.

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

We have used 10 SLs for traffic needing QoS. Each one with different maximum distance and bandwidth requirements. The SLs used are shown in Table 1. For the most demanded distances a division has been done in base on the mean bandwidth of the connections. We have used CBR traffic, randomly generated between the bandwidth range of each SL. The connections of each SL request a maximum distance between two consecutive entries in the high-priority table and a mean bandwidth in the range shown in the Table 1. Note, this is equivalent to request a maximum deadline and compute the maximum distance between two consecutive entries in the virtual line arbitration tables.

Table 1: Features of the SLs.

| SL | Maximum Distance | Bandwidth Range (Mbps) |
|---|---|---|
| 0 | 2 | 0.064 - 1.55 |
| 1 | 4 | 0.064 - 1.55 |
| 2 | 8 | 0.064 - 1.55 |
| 3 | 16 | 0.064 - 1.55 |
| 4 | 32 | 0.064 - 1.55 |
| 5 | | 1.55 - 64 |
| 6 | 64 | 0.008 - 0.064 |
| 7 | | 0.064 - 1.55 |
| 8 | | 1.55 - 64 |
| 9 | | 64 - 255 |

Each request is verified at each node in its path, and it is only accepted if there is available resources. Connections of the same SL are grouped in the same sequence of entries computing the total weight of the sequence based on the accumulated bandwidth of the connections sharing it. When the connection can not be settled in a previously established sequence (or there is not a previous one), the algorithm look for an empty sequence of entries in the high-priority arbitration table with the correct distance between its entries.

Although the results for BE and CH traffic are not the main focus of this paper, we have reserved 20% of available bandwidth for these kind of traffic, that would be attended by the low-priority table. So, connections would just be established until the 80% of the available bandwidth.

### 4.1 Simulation results

We can see in Table 2 the injected and delivered traffic (in bytes/cycle/node), the average utilization (in %) and the average bandwidth reserved (in Mbps) in host interfaces and switch ports. Note that the maximum utilization reachable is 80%, because the other 20% is reserved for BE and CH traffic. So, we are close to the maximum utilization. Obviously, we could achieve a higher utilization establishing more connections, but we have already made a lot of tries for each SL. Other connections that we could establish would be of SLs of small mean bandwidth because the network is already very loaded.

We can not observe significant differences among the different topologies. Note the important thing in order to maximize the load of the network is to distribute the connections among all the links. However, this task depends on the routing algorithm. So, these results could be improved using other routing algorithms for these topologies distributing more the load among the different available paths.

| | Topology | | | | | |
|---|---|---|---|---|---|---|
| | Irregular | | Hypercube | | Mesh | |
| | Small | Large | Small | Large | Small | Large |
| Injected traffic (Bytes/Cycle/Node) | 0.7262 | 0.7607 | 0.7224 | 0.7417 | 0.7412 | 0.7644 |
| Delivered traffic (Bytes/Cycle/Node) | 0.7262 | 0.7607 | 0.7224 | 0.7417 | 0.7414 | 0.7644 |
| Average utilization for host interfaces (%) | 72.63 | 76.07 | 72.24 | 74.17 | 72.51 | 76.44 |
| Average utilization for switch ports (%) | 73.49 | 75.13 | 74.48 | 76.51 | 73.68 | 75.76 |
| Average reservation for host interfaces (Mbps) | 1849.81 | 1905.32 | 1840.20 | 1857.72 | 1846.87 | 1895.45 |
| Average reservation for switch ports (Mbps) | 1872.23 | 1881.63 | 1897.09 | 1917.70 | 1829.58 | 1940.23 |

Table 2: Traffic and utilization for different topologies and packet sizes.

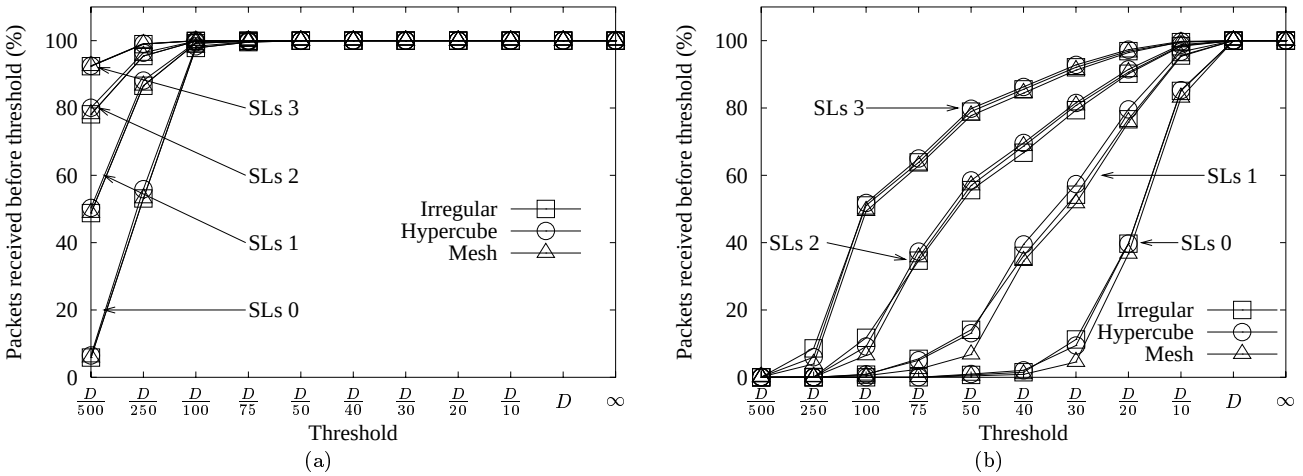


Fig. 1: Distribution of packet delay for (a) small packet size and (b) large packet size.

We have also computed the percentages of packets meeting a certain deadline threshold. These thresholds are different for each connection and are related to their requested maximum deadline. This maximum deadline ($D$) is the maximum delay that has been guaranteed to each connection. These results are shown in Figure 1 for small and large packet size, but only for SLs 0, 1, 2, and 3 (the most restrictive ones). The other SLs have even better behavior. We can observe that even for large packet size, all packets arrive to their targets before their deadline. Again, results are similar for all topologies tested.

## 5. CONCLUSIONS

In previous works we had proposed a new strategy to compute the InfiniBand arbitration tables. This strategy was tested in [4] for networks with irregular topology. In this paper we have tested our proposal for different topologies.

We have concluded that the important thing is not the topology used, but the routing algorithm. Due to the arbitration and the resource reservation done, when the connections are accepted they always meet their requirements regardless the topology used. In order to be able to admit more connections, and so achieve a higher load of the network, we should use routing algorithms doing a better use of available paths.

## References


[1] F. Alfaro, J. Sánchez, M. Menduiña, and J. Duato. Formalizing the Fill-In of the Infiniband Arbitration Table. Technical Report DIAB-03-02-35, Departamento de Informática Univ. de Castilla-La Mancha, Mar. 2003.

[2] F. J. Alfaro, J. L. Sánchez, and J. Duato. A Strategy to Manage Time Sensitive Traffic in InfiniBand. In *Proceedings of Workshop on Communication Architecture for Clusters (CAC'02)*, Apr. 2002. Held in conjunction with IPDPS'02, Fort Lauderdale, Florida.

[3] 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 (IPDPS'02)*, April 2002.

[4] F. J. Alfaro, J. L. Sánchez, L. Orozco, and J. Duato. A New Proposal to Fill in the InfiniBand Arbitration Tables. Submitted to the International Conference on Parallel Computing (ICPP'03), Oct. 2003.

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

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