Improving InfiniBand Routing through Multiple Virtual Networks*

J. Flich, P. López, J.C. Sancho, A. Robles, and J. Duato

Department of Computer Engineering
Universidad Politécnica de Valencia, Spain
{jflich,plopez,jcsancho,arobles,jduato@gap.upv.es}

Abstract. InfiniBand is very likely to become the de facto standard for communication between nodes and I/O devices (SANs) as well as for interprocessor communication (NOWs). The InfiniBand Architecture (IBA) defines a switch-based network with point-to-point links whose topology is arbitrarily established by the customer. Often, the interconnection pattern is irregular. Up*/down* is the most popular routing scheme currently used in NOWs with irregular topologies. However, the main drawbacks of up*/down* routing are the unbalanced channel utilization and the difficulties to route most packets through minimal paths, which negatively affects network performance. Using additional virtual lanes can improve up*/down* routing performance by reducing the head-of-line blocking effect, but its use is not aimed to remove its main drawbacks. In this paper, we propose a new methodology that uses a reduced number of virtual lanes in an efficient way to achieve a better traffic balance and a higher number of minimal paths. This methodology is based on routing packets simultaneously through several properly selected up*/down* trees. To guarantee deadlock freedom, each up*/down* tree is built over a different virtual network. Simulation results, show that the proposed methodology increases throughput up to an average factor ranging from 1.18 to 2.18 for 8, 16, and 32-switch networks by using only two virtual lanes. For larger networks with an additional virtual lane, network throughput is tripled, on average.

1 Introduction

InfiniBand [8] has been recently proposed as an standard for communication between processing nodes and I/O devices as well as for interprocessor communication. Several companies, including the leading computer manufacturers, support the InfiniBand initiative. InfiniBand is designed to solve the lack of high bandwidth, concurrency and reliability of existing technologies (PCI bus) for system area networks. Moreover, InfiniBand can be used as a platform to build networks of workstations (NOWs) or clusters of PCs [13] which are becoming a cost-effective alternative to parallel computers. Currently, clusters are based on different available network technologies (Fast or Gigabit Ethernet [21],

* This work was supported by the Spanish CICYT under Grant TIC2000-1151 and by Generalitat Valenciana under Grant GV00-131-14.

Myrinet [1], ServerNet II [7], Autonet [20], etc...). However, they may not provide the protection, isolation, deterministic behavior, and quality of service required in some environments.

The InfiniBand Architecture (IBA) is designed around a switch-based interconnect technology with high-speed point-to-point links. Nodes are directly attached to a switch through a Channel Adapter (CA). An IBA network is composed of several subnets interconnected by routers, each subnet consisting of one or more switches, processing nodes and I/O devices.

Routing in IBA subnets is distributed, based on forwarding tables (routing tables) stored in each switch, which only consider the packet destination node for routing [9]. IBA routing is deterministic, as the forwarding tables only store one output link per every destination ID. Moreover, virtual cut-through switching is used [10]. IBA switches support a maximum of 16 virtual lanes (VL). VL15 is reserved exclusively for subnet management, whereas the remaining VLs are used for normal traffic. Virtual lanes provide a mean to implement multiple logical flows over a single physical link [3]. Each virtual lane has associated a separate buffer. However, IBA virtual lanes cannot be directly selected by the routing algorithm. In IBA, to route packets through a given virtual lane, packets are marked with a certain Service Level (SL), and SLtoVL mapping tables are used at each switch to determine the virtual lane to be used. According to the IBA specifications, service levels are primarily intended to provide quality of service (QoS). However, virtual lanes can be also used to provide deadlock avoidance and traffic prioritization.

2 Motivation

Usually, NOWs are arranged as switch-based networks whose topology is defined by the customer in order to provide wiring flexibility and incremental expansion capability. Often, due to building constraints, the connections between switches do not follow any regular pattern leading to an irregular topology. The irregularity in the topology makes the routing and deadlock avoidance quite complicate. In particular, a generic routing algorithm suitable for any topology is required. However, the IBA specification does not establish any specific routing algorithm to be used.

Up*/down* [20,1] is the most popular routing algorithm used in the NOW environment. Other routing schemes have been proposed for NOWs, such as the adaptive-trail [14], the minimal adaptive [19], the smart-routing [2] and some improved methodologies to compute the up*/down* routing tables (DFS) [17].

The adaptive-trail and the minimal adaptive routing algorithms are adaptive and allow the existence of cyclic dependencies between channels provided that there exist some channels (escape paths) without cyclic dependencies to avoid deadlocks. However, these escape paths must be selected at run-time, when the other channels are busy. Hence, both routings are not supported by IBA because routing in IBA is deterministic and it cannot select among virtual channels. On the other hand, the smart-routing algorithm is deterministic and it has been shown that it is able to obtain a very high performance [6]. However, its main drawback is the high time required to compute the routing tables. In fact, it has not been possible yet to compute the routing tables for medium and large network sizes. Therefore, in this paper, we will focus on the up*/down* routing.

The main advantage of using *up*/down** routing is the fact that it is simple and easy to implement (see Section 3). However, there exist several drawbacks that may noticeably reduce network performance. First of all, this routing scheme does not guarantee all the packets to be routed through minimal paths. This problem becomes more important as network size increases. In general, *up*/down** concentrates traffic near the root switch, often providing minimal paths only between switches that are allocated near the root switch [16,6]. Additionally, the concentration of traffic in the vicinity of the root switch causes a premature saturation of the network, thus obtaining a low network throughput and leading to an uneven channel utilization.

As stated earlier, IBA allows the use of several virtual lanes, which are closely related to service levels. Although these virtual lanes are primarily intended to provide QoS, they can also be used to improve network performance. In fact, we have recently proposed [18] the use of virtual lanes to reduce the head-of-line blocking effect when using the up*/down* routing algorithm in IBA. In particular, we were using virtual lanes as if were different virtual networks, because packets remained in a particular virtual network (virtual lane) while crossing the network. However, as we were using the same up*/down* tree on each particular virtual network, we were not addressing the main drawbacks of up*/down*. Notice that, if several virtual networks are available, we can use a different routing algorithm on each particular virtual network. We can take advantage of this fact to obtain more minimal paths and a better traffic balance.

In this paper, we take on this challenge by proposing an easy way to improve the performance of InfiniBand networks by using virtual lanes in an efficient way. In particular, we propose the use of different up*/down* trees on each virtual network.

Notice that according to IBA specifications, we need a number of service levels equal to the number of virtual lanes for our purposes. However, service levels are primarily intended for QoS. Therefore, any proposal aimed to improve performance in InfiniBand by using virtual lanes and service levels should limit as much as possible the number of resources devoted to this purpose.

The rest of the paper is organized as follows. In section 3, the up*/down* routing scheme and its implementation on IBA is described. Section 4 describes the proposed methodology to improve network performance. In section 5 the IBA simulation model is described, together with the discussion on the performance of the proposal. Finally, in section 6 some conclusions are drawn.

3 Up*/Down* Routing on IBA

Up/down** routing is the most popular routing scheme currently used in commercial networks, such as Myrinet [1]. It is a generic deadlock-free routing algorithm valid for any network topology.

Routing is based on an assignment of direction labels ("up" or "down") to the operational links in the network by building a BFS spanning tree. To compute a BFS spanning tree a switch must be chosen as the root. Starting from the root, the rest of the switches in the network are arranged on a single spanning tree [20].

After computing the BFS spanning tree, the "up" end of each link is defined as: 1) the end whose switch is closer to the root in the spanning tree; 2) the end whose switch has the lowest identifier, if both ends are at switches at the same tree level. The result of this assignment is that each cycle in the network has at least one link in the "up" direction and one link in the "down" direction. To avoid deadlocks while still allowing all links to be used, this routing scheme uses the following up*/down* rule: a legal route must traverse zero or more links in the "up" direction followed by zero or more links in the "down" direction. Thus, cyclic channel dependencies [1] [4] are avoided because a packet cannot traverse a link in the "up" direction after having traversed one in the "down" direction.

Unfortunately, *up*/down** routing cannot be applied to InfiniBand networks in a straightforward manner because it does not conform to IBA specifications. The reason for this is the fact that the *up*/down** scheme takes into account the input port together with the destination ID for routing, whereas IBA switches only consider the destination ID. Recently, we have proposed two simple and effective strategies to solve this problem [15,12]. In particular, we will use the destination renaming technique proposed in [12].

4 New Strategy for IBA Routing

The basic idea proposed in this paper is the following. For a given network, several up*/down* trees can be easily computed by considering different root nodes. Moreover, for a given source-destination pair, some trees may allow shorter paths than others. Indeed, some trees may offer minimal paths while others not. At first, the idea is to use two of these trees to forward packets through the network. To avoid conflicts between them, every tree will use a different virtual lane (i.e., there will be two virtual networks). Hence, for a given source-destination pair, the tree which offers the shortest path can be used. This will mitigate the non-minimal path problem of basic up*/down* routing. Additionally, as there are now two different trees with two different roots and the paths are distributed between them, network traffic is better balanced.

In order to implement this scheme on IBA, we need to use one virtual lane for each of the virtual networks that correspond to each up*/down* tree. This is easily done by assigning a different SL to each virtual network and mapping each SL to a given virtual lane with the SLtoVL mapping table. Hence, we need only two SLs to implement the proposed mechanism with two up*/down* trees on IBA. Notice that there are up to 15 available SLs, so the remaining ones can be used for other purposes. Additionally, for every source-destination pair,

[1] There is a channel dependency from a channel c_i to a channel c_j if a packet can hold c_i and request c_j. In other words, the routing algorithm allows the use of c_j after reserving c_i.

a path from one of the two up*/down* trees will be used. Hence, all the packets sent from the source host to the destination host will use the SL assigned to the selected tree. Thus, every source node needs a table that returns the SL that must be used to forward a packet to a given destination host.

It must be noticed that this routing scheme based on the use of multiple virtual networks is deadlock-free, as packets are injected into a given virtual network and remain on it until they arrive at their destinations (i.e., packets do not cross from one virtual network to another).

Let us explain now how both trees are selected. Ideally, both up*/down* trees should complement each other, in such a way that those paths that are non-minimal in one tree are minimal in the another tree. Another idea is to select the two trees that better balance network links, following some heuristic. In practice, though, this problem translates into how to select the two root nodes. We propose a simple and intuitive strategy. The first root node is the node which has the lowest distance to the rest of nodes (as Myrinet does [1]). The second root is the node that is farthest from the first one. In this way, if the whole set of paths are equally distributed between both trees, as the root nodes are very far from each other, we could expect that many paths from one tree do not interfere with the paths from the another tree (i.e., they do not use the same physical links). As a consequence, a better traffic balance should be expected by using this straightforward strategy. More complex strategies can also be used. However, this issue is beyond the scope of this paper.

Once we have computed the full set of paths for each of the up*/down* trees, the final set of paths needs to be selected. In this set, only one path for every source-destination host has to be included[2]. Again, several strategies can be used. We propose here the straightforward one of randomly selecting one path from the set of all shortest paths (this set is the union of the shortest paths of both trees). Notice that as up*/down* is a partially adaptive routing algorithm, one of the trees may supply more paths than the another one. Hence, with the proposed strategy to select the final paths, this tree will have more chance of being selected than the another tree. Other more complex strategies could select paths trying to better balance link utilization.

The proposed routing scheme based on the use of multiple virtual networks can be easily extended to use more than two (say n) different up*/down* trees. The possibility of having more trees may lead to reduce even more average distance (by supplying shorter paths or even minimal ones) and to achieve a better link utilization. This may be specially noticeable in large networks. The only drawback of using more than two trees is that every new tree needs a different virtual network, and thus, consumes a new SL/VL in InfiniBand. While this may not be a problem for some applications, it can be a limiting factor for other applications that require traffic prioritization and quality of service.

If we have to compute n up*/down* trees, our criteria is to select the n root nodes that satisfy the following two conditions (and in this order): i) the distance

[2] Notice, though, that IBA allows different paths from a given source-destination pair with the virtual addressing feature, but this is beyond the scope of this paper.

between all pair of roots is roughly the same (i.e., the roots are equidistant and far away from each other), and ii) the distance between any pair of the n roots is the longest one. Again, we use a simple algorithm that tries to balance network traffic by randomly selecting the final path to be used.

Finally, notice that the proposal of routing through multiple virtual networks can be generalized to use not only up*/down* and other tree-based routing algorithms but also any deadlock-free routing algorithm. As once a packet enters a virtual network it does not switch to another virtual network, the resulting routing algorithm is deadlock-free.

5 Performance Evaluation

In this section we will evaluate the proposed strategy. For this purpose, we have developed a detailed simulator that allows us to model the network at the register transfer level following the IBA specifications [9]. First, we will describe the main simulator parameters and the modeling considerations we have used in all the evaluations. Then, we will evaluate the use of multiple up*/down* trees under different topologies and different traffic patterns.

5.1 Simulation Model

In the simulator, we will use a non-multiplexed crossbar on each switch. This crossbar supplies separate ports for each VL. We will use a simple crossbar arbiter based on FIFO request queues per output crossbar port, that will select the first request that has enough space in the corresponding output crossbar port. If there is sufficient buffer capacity in the output buffer, the packet is forwarded. Otherwise, the packet must wait at the input buffer. A round-robin arbiter is used to select the next VL that contains a packet to be transmitted over the physical link. The crossbar bandwidth will be set accordingly to the value of the injection rate of the links. Buffers will be used both at the input and the output side of the crossbar. Buffer size will be fixed in both cases to 1 KB.

The routing time at each switch will be set to 100 ns. This time includes the time to access the forwarding and SLtoVL tables, the crossbar arbiter time, and the time to set up the crossbar connection. Links in InfiniBand are serial. The link speed is fixed to 2.5 Gbps. Therefore, a bit can be injected every 0.4 ns. With 10/8 coding [9] a new byte can be injected into the link every 4 ns. We will model 20 m copper cables with a propagation delay of 5 ns/m. Therefore, the fly time (time required by a bit to reach the opposite link side) will be set to 100 ns.

For each simulation run, we assume that the packet generation rate is constant and the same for all the end-nodes. Once the network has reached a steady state, the packet generation rate is equal to the packet reception rate. We will evaluate the full range of traffic, from low load to saturation. Moreover, we will use a uniform distribution of packet destinations in all the evaluations. For some particular evaluations, we will also use the hotspot traffic pattern. Packet length is 32-bytes.

Every packet will be marked with the SL that corresponds to the up*/down* tree that the packet must use. In each switch, the SLtoVL table is configured to assign the same VL to each SL. Packets will continue through the same VL (same tree) until they reach the destination end-node. In all the presented results, we will plot the average packet latency[3] measured in nanoseconds versus the average accepted traffic[4] measured in bytes/ns/switch.

We will analyze irregular networks of 8, 16, 32 and 64 switches randomly generated. These network topologies will be generated taking into account some restrictions. First, we will assume that every switch in the network has 8 ports, using 4 ports to connect to other switches and leaving 4 ports to connect to hosts. Second, each switch will be connected to one switch by exactly one link. Ten different topologies will be generated for each network size. Results plotted in this paper will correspond to the most representative topology for every network size.

5.2 Simulation Results

In this section, we will study in detail the possible benefits of using different up*/down* trees through different virtual networks to increase network performance. In a first study, we will evaluate how network performance can be increased when using two up*/down* trees and selecting the farthest root nodes. This study will take into account different traffic patterns. In a second study, we will consider the use of three up*/down* trees and will evaluate its impact on network performance. In particular, we are interested in detecting under which circumstances it is worth using an additional up*/down* tree through a new virtual network. Finally, in the last study, we will evaluate how effective is the strategy used to select the root nodes (farthest nodes) and how the selection of an alternative pair of root nodes could affect to network performance

5.3 Using Two Virtual Networks

Figures 1.a, 1.b, and 1.c show evaluation results when using one and two up*/down* trees for 16, 32, and 64-switch networks, respectively. Packet size is 32 bytes and a uniform distribution of packet destinations is used. For comparison reasons, when using one up*/down* tree, two virtual lanes are equally used.

We can observe that for the 16-switch network the use of two up*/down* trees increases network throughput by a factor of 1.63. Moreover, when network size increases, benefits are much more noticeable. Network throughput is increased by a factor of 3 for the 32-switch network.

The benefits of using an additional up*/down* tree are mainly due to the better traffic balance achieved over the network. In Figures 2.a and 2.b we can

[3] Latency is the elapsed time between the generation of a packet at the source host until it is delivered at the destination end-node.

[4] Accepted traffic is the amount of information delivered by the network per time unit.

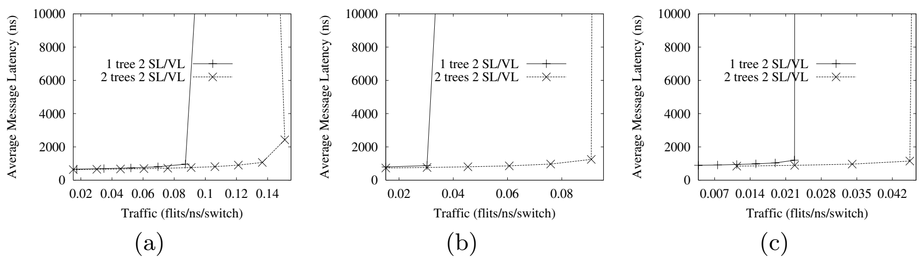


Fig. 1. Average packet latency vs. traffic. Network size is (a) 16, (b) 32, and (c) 64 switches. Packet size is 32 bytes. Uniform distribution of packet destinations.

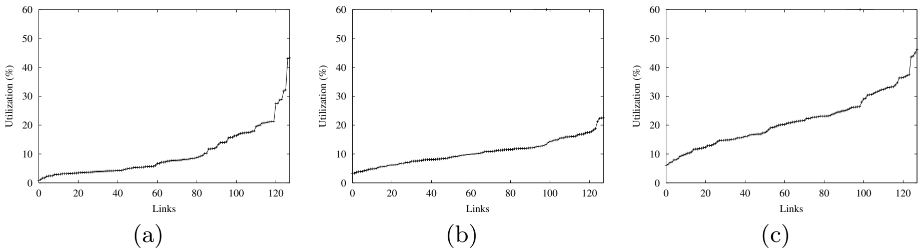


Fig. 2. Link utilization. (a) One up*/down* tree and (b, c) two up*/down* trees. Traffic is (a, b) 0.03 flits/ns/switch and (c) 0.09 flits/ns/switch. Network size is 32 switches. Packet size is 32 bytes. Uniform distribution of packet destinations.

observe the utilization of all the links connecting switches for the 32-switch network when using one and two up*/down* trees, respectively. Links are sorted by link utilization. Network traffic is 0.03 bytes/ns/switch (the routing algorithm with one up*/down* tree is reaching saturation). We can see that when using one up*/down* tree (Figure 2.a) there is a high unbalance of the link utilization. Most of the links have a low link utilization (50% of links with a link utilization lower than 10%), whereas few links exhibit a high link utilization (link utilization higher than 30%). The links with a high link utilization correspond to the links connected to the root node. These links cause a premature saturation of the network.

However, when using two up*/down* trees (Figure 2.b) we can observe that, for the same traffic point, the traffic is better distributed over the links. Almost all the links exhibit a link utilization lower than 20%. This is due to the fact that there are now two root nodes in the network and, on average, the traffic is equally distributed over both root nodes. Figure 2.c shows the link utilization of the routing based on two up*/down* trees at its saturation point (traffic is 0.09 bytes/ns/switch). We can observe that even at this traffic point the routing exhibits a good traffic distribution over the links.

For other random topologies, Table 1 shows minimum, maximum, and average factor of throughput increases for 10 different topologies generated for each network size. We can see how, on average, using two up*/down* trees helps to obtain a higher network throughput. In particular, 8-switch networks increase,

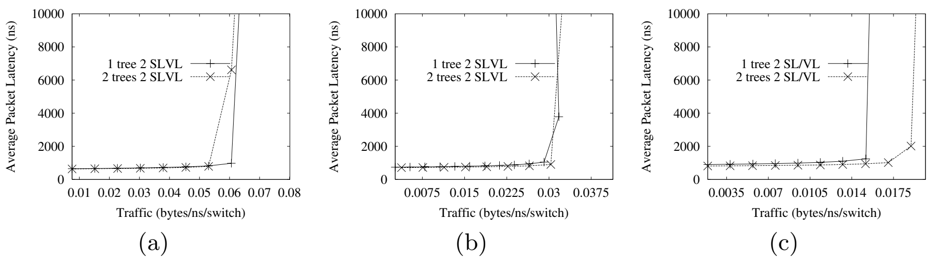


Fig. 3. Average packet latency vs. traffic. Network size is (a) 16, (b) 32, and (c) 64 switches. Packet size is 32 bytes. Hotspot distribution. One hotspot. 20% of hotspot traffic.

Table 1. Factor of throughput increase when using two up*/down* trees. Packet size is 32 bytes. Uniform distribution of packet destinations.

Sw	Min	Max	Avg
8	0.96	1.47	1.18
16	1.01	2.01	1.55
32	1.59	3.00	2.18
64	1.81	2.83	2.14

on average, their throughput by a factor of 1.18. For larger networks, average factor of network throughput increases significantly.

We can see also that for large networks of 64 switches (Figure 1.c) the increase in network throughput when using two up*/down* trees is 1.95 and, on average, is 2.14 (Table 1). This improvement is significantly lower than the improvement obtained with a 32-switch network (Figure 1.b). The reason for this lower improvement will be explained in the next section and basically will be related to the need of using an additional up*/down* tree.

Previous results have been obtained with a uniform distribution of packet destinations. Although this traffic pattern is commonly used, other traffic patterns could be more realistic in these kind of networks. Indeed, in these networks is typical to find some disk devices (TCA) attached to switches that are accessed by hosts (HCA). We model this behavior with a hotspot traffic distribution. With this distribution, all the hosts in the network will send a percentage of the traffic to a particular host (the hotspot). The rest of the traffic will be sent by using a uniform distribution of packet destinations. The hotspot will be randomly selected for each network. Different percentages of traffic sent to the hotspot will be evaluated. We will refer to this traffic as hotspot traffic. Also, different number of hotspots will be used in order to model networks with several attached disks. When dealing with more than one hotspot, the hotspot traffic will be uniformly distributed among all the hotspots.

In Figures 3.a, 3.b, and 3.c, we can observe the performance results when there is one hotspot in 16, 32, and 64-switch networks, respectively. Hotspot traffic is 20%. We can observe that for the 16-switch network, the use of two

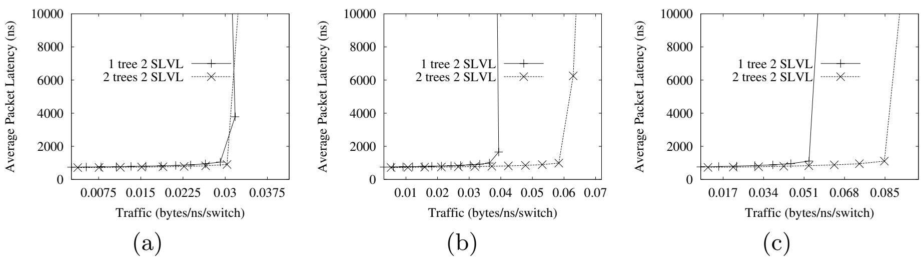


Fig. 4. Average packet latency vs. traffic. Network size is 32 switches. Packet size is 32 bytes. Hotspot distribution. One hotspot. Hotspot traffic is (a) 20%, (b) 10%, and (c) 5%.

Table 2. Factor of throughput increase when using two up*/down* trees. Packet size is 32 bytes. Hotspot traffic. One hotspot. Different percentages of hotspot traffic.

	5%			10%			20%		
Sw	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
8	0.84	1.63	1.16	0.91	1.59	1.16	0.77	1.31	1.00
16	1.13	1.94	1.48	0.93	1.53	1.18	0.70	1.50	0.99
32	1.52	2.82	1.95	1.30	1.93	1.67	0.78	1.67	1.25
64	1.36	2.84	1.86	1.01	2.10	1.36	0.81	1.86	1.28

up*/down* trees does not help in increasing network throughput. Even, the use of two up*/down* trees may behave slightly worse than when using one up*/down* tree. This is due to the fact that the hotspot may affect differently both up*/down* trees (in the routing with two up*/down* trees), dealing to a sudden saturation of one tree. In particular, one of the up*/down* trees may behave worse than the another one simply because it handles much more traffic due to the hotspot location. This up*/down* tree enters saturation, prematurely leading to the overall network saturation. However, for 64-switch networks the use of two up*/down* trees increases network throughput by a factor of 1.27. In this case, the hotspot does not affect in a severe way any of both up*/down* trees.

On the other hand, if the hotspot traffic is reduced, the benefits of using two up*/down* trees become significant again. Figures 4.a, 4.b, and 4.c show the performance results for the 32-switch network when the hotspot traffic is 20%, 10%, and 5%, respectively. We can observe that for a 10% hotspot traffic the use of two up*/down* trees obtains an increase of network throughput by a factor of 1.5. If the hotspot traffic is much lower (5%), the factor improvement is 1.8.

Table 2 shows the average, minimum, and maximum factors of throughput increase when there is one hotspot in the network with different hotspot traffics and different network sizes. We can observe that, for severe hotspots (20% of hotspot traffic), only large networks of 32 and 64 switches obtain better results with the use of two up*/down* trees. When the hotspot is less severe (10% and 5% of hotspot traffic), the routing that uses two up*/down* trees behaves, on

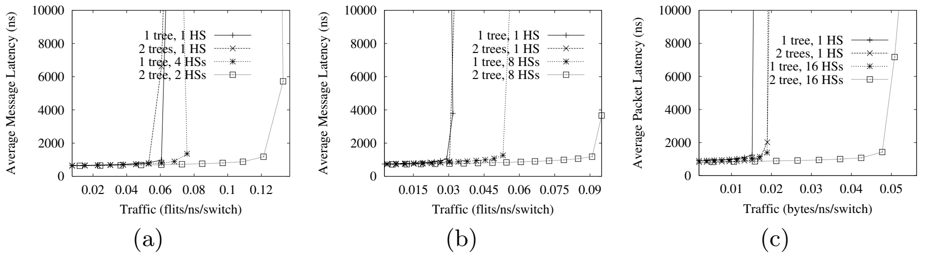


Fig. 5. Average packet latency vs. traffic. Network size is (a) 16, (b) 32, and (c) 64 switches. Packet size is 32 bytes. Hotspot traffic pattern. 20% of hotspot traffic.

Table 3. Factor of throughput increase when using two up*/down* trees. Packet size is 32 bytes. Hotspot traffic. Several hotspots. 20% of hotspot traffic.

Sw	HS	Min	Max	Avg
8	2	0.90	1.19	1.03
16	4	1.20	1.75	1.49
32	8	1.41	2.57	1.93
64	16	1.60	2.61	2.02

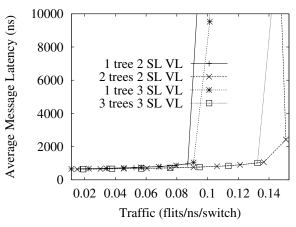
average, much better than the routing with only one up*/down* tree for every network size.

However, as more hotspots appear in the network (modeling more disks), the negative effect is diminished. We can see in Figures 5.a, 5.b, and 5.c the network performance when there is more than one hotspot in the network, for 16, 32, and 64-switch networks, respectively. We can observe that, as more hotspots appear in the network, higher throughput values are obtained when using the routing with two up*/down* trees. In particular, for 16-switch networks with 4 hotspots (Figure 4.a) network throughput is increased by a factor of 1.65. For 32 and 64-switch networks with 8 and 16 hotspots, respectively, the factor of throughput improvement ranges from 1.64 to 2.5.

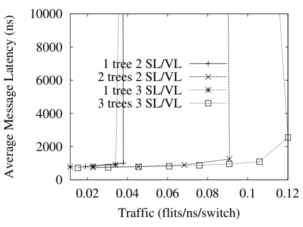
Table 3 shows average, minimum, and maximum factors of throughput increase for different number of hotspots. We can observe that when there is more than one hotspot in the network, the routing with two up*/down* trees significantly increases network throughput.

5.4 Using an Additional Virtual Network

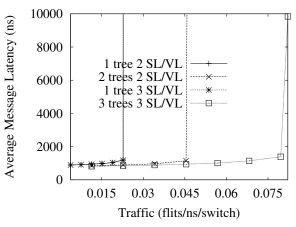
In this section we focus on the evaluation of the routing algorithm that uses an additional virtual network. This virtual network will be used by a new up*/down* tree. The selection of the root node will be done in the same way as the previous case. That is, the three chosen root nodes will be at the same distance each other and at the farthest positions. We are interested in obtaining the trade-off between throughput improvement and the use of an additional virtual network (SL/VL).



(a)



(b)



(c)

Fig. 6. Average packet latency vs. traffic. Network size is (a) 16, (b) 32, and (c) 64 switches. Packet size is 32 bytes. Uniform distribution of packet destinations.

Table 4. Factor of throughput increase when using three up*/down* trees. Packet size is 32 bytes. Uniform distribution of packet destinations.

	3 trees vs 2 trees			3 trees vs 1 tree		
Sw	Min	Max	Avg	Min	Max	Avg
8	1.00	1.20	1.12	1.03	1.47	1.22
16	0.95	1.23	1.08	1.09	2.17	1.61
32	0.96	1.60	1.24	1.70	3.42	2.58
64	1.21	1.78	1.52	2.63	3.61	3.22

Figures 6.a, 6.b, and 6.c show the performance results when one, two, and three up*/down* trees are used for 16, 32, and 64-switch networks, respectively. Packet size is 32 bytes and a uniform distribution of packet destinations is used. When using one up*/down* tree, and for comparison reasons, two and three virtual lanes are equally used.

We can observe that, when using an additional up*/down* tree, improvements are lower except for the 32 and 64-switch networks. In particular, network throughput is only increased by a factor of 1.15 for the 16-switch network. The use of an additional up*/down* tree does not help in small networks because the use of two up*/down* trees is enough to balance the network in an efficient way.

However, in medium-sized and large networks (32 and 64 switches), the use of an additional up*/down* tree helps to increase significantly network throughput. In particular, network throughput is increased by factors of 1.33 and 1.78 for 32 and 64-switch networks, respectively.

Table 4 shows minimum, maximum, and average factors of throughput increase when using three up*/down* trees. We can observe how, on average, the benefits are noticeable only for 32 and 64-switch networks. Throughput is increased, on average, by a factor of 2.58 and 3.22 for 32 and 64-switch networks, respectively. We can observe that it is not worth for small and medium networks the use of an additional virtual network (SL/VL) to increase network throughput. However, for larger networks is highly recommended as network throughput is even tripled, on average.

To sum up, the use of an additional up*/down* tree is highly related to the network size and it is worth using it only with large networks at the expense of consuming an additional SL/VL resource.

5.5 Selection of the Root Nodes

In this last study we evaluate how the way we choose the root nodes for each up*/down* tree may affect the achieved network performance. In particular, we are interested in foreseeing the throughput that could be achieved by using an alternative pair of root nodes and knowing if it is worth making a deeper study about the selection of root nodes.

For this, we have computed all the possible routings for every possible pair of root nodes for a particular 16-switch network. For each routing (240 routings) we have simulated its behavior and obtained its throughput for a uniform distribution of packet destinations and with 32-byte packets. Figure 7 shows all the achieved throughput values. Every pair of possible roots are shown in the X-bar and they are sorted by throughput. We have also plotted the throughput achieved when considering the farthest root nodes (0.0145 bytes/ns/switch).

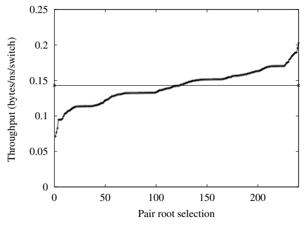


Fig. 7. Average packet latency vs. traffic. Network size is 32 switches. Packet size is 32 bytes. Uniform distribution of packet destinations.

As we can see, there is a high variation of network throughput depending on the selection of the root nodes. We can obtain a lower throughput (0.07 bytes/ns/switch) with some particular root nodes as well as a higher throughput (up to 0.2 bytes/ns/switch).

Notice that we are in the middle way when using the farthest root nodes in the topology. Therefore, with a clever selection of the root nodes we could even increase the network throughput by an additional factor of 1.40. As a conclusion, it is worth undertaking a deeper study of the selection of the root nodes.

6 Conclusions

In this paper, we have proposed a new routing methodology to improve IBA network performance by using virtual lanes in an efficient way. This routing methodology does not use virtual lanes just to reduce the head-of-line blocking effect on the input buffers. Rather, virtual lanes are viewed as separate virtual networks and different routing algorithms are used at each virtual network. In particular, different up*/down* trees are used at each virtual network. With this we try to achieve a better traffic balance and to allow most packets to follow minimal paths.

By using just two virtual networks (2 SL/VL) it is possible to increase network throughput by average factors of improvement of 1.55 and 2.18 for 16 and 32-switch networks, respectively. Only for very large networks (64 switches) it is worth using an additional virtual network (3 SL/VL). In particular, 64-switch network throughput is tripled. All of these results have been obtained assuming a straightforward strategy to select the root nodes and the final set of paths.

We have also evaluated the selection of alternative pairs of root nodes. We have seen that a better selection of the root nodes could lead to a better network performance. Therefore, it is worth deepening more in the selection of the root nodes.

As future work, we plan to study better criteria to select the appropriate number of root nodes and their locations in the network. Also, we plan to evaluate the use of better routing algorithms and the combination of different routing algorithms.

References

1. N. J. Boden et al., Myrinet - A gigabit per second local area network, *IEEE Micro*, vol. 15, Feb. 1995.
2. L. Cherkasova, V. Kotov, and T. Rokicki, “Fibre channel fabrics: Evaluation and design,” in *Proc. of 29th Int. Conf. on System Sciences*, Feb. 1995.
3. W. J. Dally, Virtual-channel flow control, *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194-205, March 1992.
4. W. J. Dally and C. L. Seitz, Deadlock-free message routing in multiprocessors interconnection networks, *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547-553, May. 1987.
5. J. Duato, A. Robles, F. Silla, and R. Beivide, A comparison of router architectures for virtual cut-through and wormhole switching in a NOW environment in Proc. of *13th International Parallel Processing Symposium*, April 1998.
6. J. Flich, P. Lopez, M.P. Malumbres, J. Duato, and T. Rokicki, “Combining In-Transit Buffers with Optimized Routing Schemes to Boost the Performance of Networks with Source Routing,” *Proc. of Int. Symp. on High Performance Computing*, Oct. 2000.
7. D. García and W. Watson, Servernet II, in *Proceedings of the 1997 Parallel Computer, Routing, and Communication Workshop*, Jun 1997.
8. InfiniBand^{TM} Trade Association, http://www.infinibandta.com.
9. InfiniBand^{TM} Trade Association, *InfiniBand^{TM} architecture. Specification Volumen 1. Release 1.0.a.* Available at http://www.infinibandta.com.
10. P. Kermani and L. Kleinrock, Virtual cut-through: A new computer communication switching technique, *Computer Networks*, vol. 3, pp. 267-286,1979.
11. C. Minkenberg and T. Engbersen, A Combined Input and Output Queued Packet-Switched System Based on PRIZMA Switch-on-a-Chip Technology, in *IEEE Communication Magazine*, Dec. 2000.

12. P. López, J. Flich, and J. Duato, Deadlock-free Routing in InfiniBand^{TM} through Destination Renaming, in *Proc. of 2001 International Conference on Parallel Processing (ICPP'01)*, Sept. 2001.
13. G. Pfister, *In search of clusters*, Prentice Hall, 1995.
14. W. Qiao and L. M. Ni, "Adaptive routing in irregular networks using cut-through switches," in *Proc. of the 1996 International Conference on Parallel Processing*, Aug. 1996.
15. J.C. Sancho, A. Robles, and J. Duato, Effective Strategy to Compute Forwarding Tables for InfiniBand Networks, in *Proc. of 2001 International Conference on Parallel Processing (ICPP'01)*, Sept. 2001.
16. J.C. Sancho and A. Robles, Improving the Up*/down* routing scheme for networks of workstations in *Proc. of Euro-Par 2000*, Aug. 2000.
17. J.C. Sancho, A. Robles, and J. Duato, New methodology to compute deadlock-free routing tables for irregular networks, in *Proc. of CANPC'2000*, Jan. 2000.
18. J.C. Sancho, J. Flich, A. Robles, P. López and J. Duato, "Analyzing the Influence of Virtual Lanes on InfiniBand Networks," submitted for publication.
19. F. Silla and J. Duato, Tuning the Number of Virtual Channels in Networks of Workstations, in *Proc. of the 10th International Conference on Parallel and Distributed Computing Systems (PDCS'97)*, Oct. 1997.
20. M. D. Schroeder et al., Autonet: A high-speed, self-configuring local area network using point-to-point links, *SRC research report 59*, DEC, Apr. 1990.
21. R. Sheifert, *Gigabit Ethernet*, Addison-Wesley, April 1998.