Boosting the Performance of Myrinet Networks

José Flich, *Member*, *IEEE*, Pedro López,
M.P. Malumbres, *Member*, *IEEE*, and Jose Duato, *Member*, *IEEE*

Abstract—Networks of workstations (NOWs) are becoming increasingly popular as a cost-effective alternative to parallel computers. These networks allow the customer to connect processors using irregular topologies, providing the wiring flexibility, scalability, and incremental expansion capability required in this environment. Some of these networks use source routing and wormhole switching. In particular, we are interested in Myrinet networks because it is a well-known commercial product and its behavior can be controlled by the software running in network interfaces (Myrinet Control Program, MCP). Usually, the Myrinet network uses up*/down* routing for computing the paths for every source-destination pair. In this paper, we propose the In-Transit Buffer (ITB) mechanism to improve network performance. We apply the ITB mechanism to NOWs with up*/down* source routing, like Myrinet, analyzing its behavior on both networks with regular and irregular topologies. The proposed scheme can be implemented on Myrinet networks by only modifying the MCP, without changing the network hardware. We evaluate by simulation several networks with different traffic patterns using timing parameters taken from the Myrinet network. Results show that the current routing schemes used in Myrinet networks can be strongly improved by applying the ITB mechanism. In general, our proposed scheme is able to double the network throughput on medium and large NOWs. Finally, we present a first implementation of the ITB mechanism on a Myrinet network.

Index Terms—Networks of workstations, wormhole switching, minimal routing, source routing, performance evaluation.

1 Introduction

Due to the increasing computing power of microprocessors and the high cost of parallel computers, networks of workstations (NOWs) are currently considered as a cost-effective alternative for small scale parallel computing. Although NOWs do not provide the computing power available in large multicomputers and multiprocessors, they meet the needs of a great variety of parallel computing problems at a much lower cost.

Currently, the evolution of NOWs is closely related to that of local area networks (LANs). LANs are migrating from shared medium to dedicated medium networks. Although Ethernet is very popular, other commercial LANs have arisen in the high-speed networking arena. Among the current gigabit LAN technologies, Myrinet [1] has one of the highest performance/cost ratio [3].

In Myrinet, packets are delivered using wormhole switching and source routing (as opposed to distributed routing) [7]. With source routing, the path to destination is built at the source host and it is written into the packet header before it is transmitted. Switches route packets through the path found at the packet header. Although the path followed by a packet is fixed (i.e., it can not be dynamically modified at each switch) switches are simpler and, thus, faster than those used with distributed routing.

Myrinet design is simple and very flexible. In particular, it allows the user to change the network behavior through the Myrinet Control Program (MCP). This software is loaded in the memory of the network interface card (NIC) at boot time. It initializes the NIC, performs the network configuration automatically, does the memory management, defines and applies the routing algorithm, formats packets, transfers packets from local processors to the network and vice versa, etc.

One of the tasks managed by the MCP is the selection of the route to reach the destination of each packet. As the Myrinet routing scheme uses source routing, the NIC has to build network routes to each destination during the initialization phase. NICs have mechanisms to discover the current network configuration, being able to build routes between itself and the rest of network hosts. Myrinet uses up*/down* routing [17] to build these paths. Up*/down* routing is based on an assignment of direction labels ("up" or "down") to links. To eliminate deadlocks, a route must traverse zero or more "up" links followed by zero or more "down" links. While up*/down* routing is simple, many of the provided paths are not minimal on certain networks. Even more, the probability of finding minimal paths in accordance with the up*/down* restriction decreases as network size increases. Also, another drawback of up*/down* routing is that it forces most of the traffic to cross the vicinity of the root switch, leading to saturation at relatively low traffic.

2 Motivation

In [20], [19], we proposed the use of adaptive routing in irregular topologies with distributed routing, showing that up*/down* routing can be strongly improved by providing more routing flexibility and allowing minimal paths. Thus, it would be very interesting to develop similar techniques to improve the performance of networks with source routing. Moreover, the proposed techniques should be easily applied to source-based commercial networks. In the case of Myrinet, this should be done without modifying network

• *The authors are with Universidad Politécnica de Valencia, E-46010 Valencia, Spain. E-mail: {jflich,plopez,mperez,jduato}@gap.upv.es.*

Manuscript received 16 Nov. 2000; revised 27 July 2001; accepted 17 Dec. 2001.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 113167

Fig. 1. Variables v_1 and v_2 mapped onto the chromosome.

hardware, by changing only the MCP software and without incurring in an excessive overhead.

Some adaptivity can be easily provided by computing more than one path between every source-destination pair of hosts, if available, and later performing a selection. However, this solution increases the size of the routing table and makes routing operation more complex. Hence, we will focus only on supplying minimal paths. In this paper, we propose a method that always guarantees minimal routing between every pair of hosts. In particular, our approach divides forbidden up*/down* minimal routes into valid subroutes and forces a special kind of virtual cut-through through the NICs at some intermediate hosts. At these hosts, packets are completely ejected and later reinjected again into the network, thus breaking dependences between the input and output channels of the affected switch. As a consequence of providing minimal routing, the ITB mechanism also achieves a better network traffic balance, removing another important drawback of up*/down* routing as mentioned above. However, notice that those packets that are ejected and reinjected may suffer a latency increase that has to be taken into account in order to make a fair comparison.

Myrinet networks can be built following any kind of network topology. Indeed, users can decide the network connectivity attaching hosts and switches to their own. Depending on the location of network hosts and the final application of the system, the topology can be regular or irregular. For example, if we plan to install a Myrinet network that replaces an existing low-performance Ethernet, the physical locations of hosts would force us to an irregular topology [14]. However, if our goal is to build a cluster of workstations to run computation intensive programs, the network would fit in a single room, the switches would be in a cabinet, and the topology should be regular in order to get the best system performance [15], [9]. Also, the up*/down* routing is usually less restrictive in regular topologies, because it supplies minimal paths for almost all destinations. Hence, the influence of topology has to be considered when evaluating our proposal, so we analyze its behavior on both irregular and regular topologies.

The rest of the paper is organized as follows. In Section 3, the current Myrinet source routing scheme is introduced. In Section 4, the ITB mechanism is described. In Section 5, the performance of the proposed mechanism is evaluated by simulation. In Section 6, a first implementation of the ITB mechanism on a Myrinet network is presented. Finally, in Section 7 some conclusions are drawn.

3 Myrinet Source Routing

Myrinet uses source routing to transmit packets between hosts. In this technique, the source host computes the path that the packet has to follow to reach its destination and stores it into the packet header (see Fig. 1). Each packet

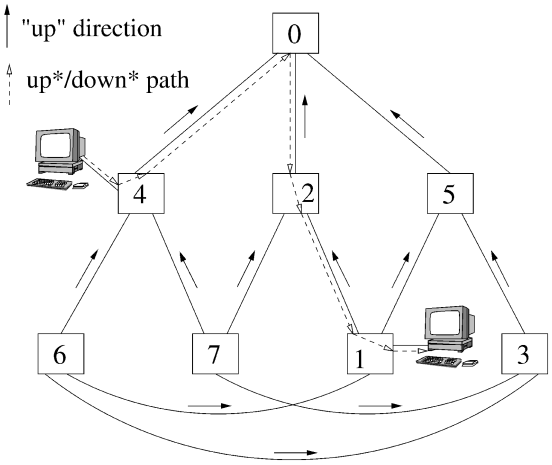

Fig. 2. A randomly generated task graph and height methods.

header consists of an ordered list of output link identifiers that are used by each intermediate switch to properly route the packet (the header also stores the packet type). The first link identifier corresponds to the one that the first switch will use, the second link identifier will be used by the second switch, and so on. Each link identifier is removed after being used. In order to build and maintain routes between the source host and each potential destination host, each network host must have a representation of the current network topology. Routes are built before sending any packet during the initialization phase. In addition, each network adapter checks for changes in the network topology (shutdown of hosts, link/switch failures, start-up of new hosts, etc.), in order to maintain the routing tables updated.

Although several routing algorithms can be used [4], [16], Myrinet uses up*/down*routing [17] to build network routes. Up*/down* routing is based on an assignment of direction to the operational links. To do so, a breadth-first spanning tree is computed and the "up" end of each link is defined as: 1) the end whose switch is closer to the root in the spanning tree and 2) the end whose switch has the lower ID, if both ends are at switches at the same tree level (see Fig. 2). 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 eliminate 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 dependences between channels are avoided because a packet cannot traverse a link along the "up" direction after having traversed one in the "down" direction.

Up*/down* routing is not always able to provide a minimal path between some pair of nodes, as shown in the following example. In Fig. 2, a packet transmitted from a host connected to switch 4 to a host connected to switch 1 cannot go through any minimal path. The shortest path (through switch 6) is not allowed since the packet should traverse a link in the "up" direction after one in the "down" direction. All the allowed paths (through switches 0, 2, and

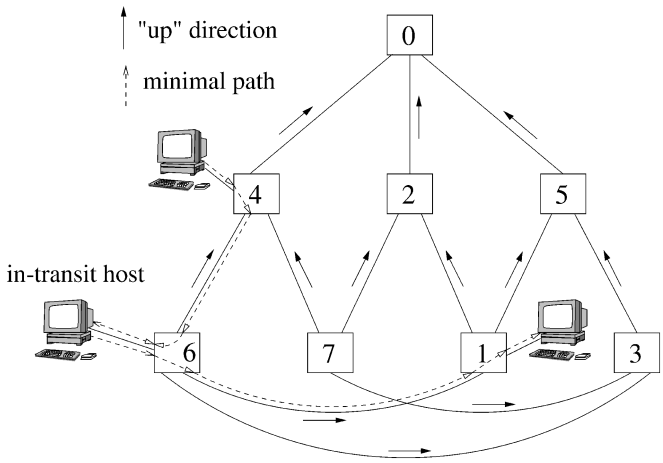

Fig. 3. A task graph and priority methods.

through switches 0, 5) are nonminimal and only one of them will be included in the routing table. The number of forbidden minimal paths increases as the network becomes larger.

Moreover, up*/down* routing un-balances network traffic. As network becomes larger, more restrictions (down → up transitions) appear in the network and, therefore, more traffic is forced to cross the vicinity of the root switch.

4 In-Transit Buffers: A Mechanism to Implement Minimal Source Routing

The up*/down* routing algorithm is deadlock-free. It avoids cyclic dependences between network links by not allowing packets to reserve "up" links after having reserved "down" links. Due to this restriction many minimal routes are forbidden. The basic idea to eliminate this restriction consists of dividing such forbidden paths into several valid up*/down* subpaths. On each subpath, an intermediate host is selected as the destination and, at this host, packets are completely ejected from the network and later reinjected in order to take the next subpath. In other words, the dependences between "down" and "up" links are removed by using some buffers at the intermediate host (in-transit buffer or ITB). In Fig. 3, we can see that, with the ITB mechanism, a minimal route can be used to route packets from the host connected to switch 4 to the host connected to switch 1. To break channel dependences, packets are sent to a host connected to the intermediate switch 6. This host will reinject packets as soon as possible towards its real destination.

Hence, when the up*/down* routing algorithm for a given packet does not provide any minimal path, the proposed routing strategy is able to find a minimal path. In this path, one or more in-transit hosts are chosen, verifying that each subroute is a valid up*/down* path. The packet will be addressed to the first in-transit host. The in-transit host will reinject the packet into the network as soon as possible, forwarding it to the destination host or to the next in-transit host.

Depending on network topology, there may exist different minimal paths for each source-destination pair. If so, some of them might be valid up*/down* paths and others might not. We will use only one minimal path for each source-destination pair, randomly selected among all the feasible paths. In the case a nonvalid up*/down* path is chosen for a particular source-destination pair, ITBs will be placed at down → up transitions along the path. Thus, the places where ITBs will be allocated in the network depends on the final set of minimal paths. Obviously, we assume that every switch has at least one host attached to it.

Fig. 4. Average computation times for different population sizes.

To make this mechanism deadlock-free, it must be guaranteed that an in-transit packet that is being reinjected can be completely ejected from the network if the reinjected part of the packet becomes blocked, thus removing potential channel dependences (down → up transitions) that may result in a deadlock configuration. So, when an in-transit packet arrives at a given host, care must be taken to ensure that there is enough buffer space to store it at the interface card before starting its reinjection. If the buffer space at the network interface card has been exhausted, the MCP should store the packet in the host memory, considerably increasing the overhead in this case. Although this strategy requires an infinite number of buffers in theory, a very small number of buffers are required in practice. In fact, in all the simulations ran (see Section 5) the reserved memory space (512KB) was enough to handle all the in-transit packets without using the host memory. Taking into account that current Myrinet interface cards are shipped with 8MB and less than 128KB are set aside for the MCP, there is more than enough memory space in the NIC to allocate the ITBs. In Section 5, we will analyze the amount of memory needed at NICs to handle all the in-transit packets. Another feasible solution to NIC memory overflow could be discarding packets, relying on the GM acknowledgments to retransmit those missing packets.

In order to route packets requiring ITBs, the Myrinet packet header format must be changed. In Fig. 4, we can see the new header format that supports ITBs. The entire path to destination is built at the source host. In the case an ITB is needed along the path, a mark (ITB mark) is inserted in the packet header in order to notify the in-transit host that the packet must be reinjected into the network after removing that mark. After the mark, the path from the in-transit host to the final destination (or to another in-transit host) follows.

Obviously, the ITB mechanism may add some latency to those packets that use it to reach their destinations. Also, the mechanism requires some additional resources in both network (links) and network interface cards (memory pools and DMA engines).

Fig. 5 shows the implementation of the in-transit buffer mechanism. In the case of Myrinet networks, some memory is needed at the network interface card to store in-transit packets and the MCP program needs to be modified to detect in-transit packets and process them accordingly. In

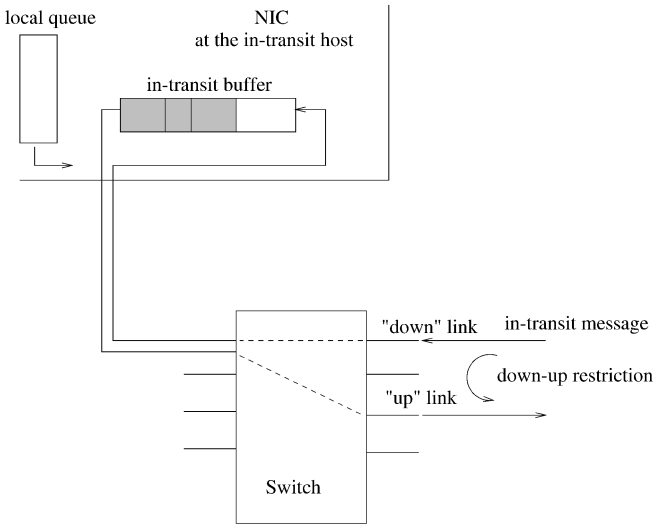

Fig. 5. Average speedup for different population sizes.

order to minimize the introduced overhead, as soon as the in-transit packet header is processed and the required output channel is free, a DMA transfer can be programmed to reinject the in-transit packet. So, the delay to forward this packet will be the time required for processing the header and starting the DMA (when the output channel is free). As the MCP allows this kind of DMA programming, it is possible to implement the ITB mechanism in Myrinet without modifying the network hardware. On the other hand, there is no problem if the DMA transfer begins before the packet has been completely received, because it will arrive at the same rate that it was transmitted,[1] assuming that all the links in the network have the same bandwidth.[2] Note that Myrinet does not implement virtual channels. Therefore, once a packet header reaches the network interface card, flits will continue arriving at a constant rate. The only additional requirement is that the packet is completely stored in the network adapter memory at the source host before starting transmission to avoid interference with its I/O bus.

The use of the mechanism on a particular in-transit host could affect the performance of its local traffic. The in-transit host output channel will be shared among local and in-transit packets. An excessive number of in-transit packets could block the injection of local packets. In order to minimize this effect, a dynamic priority scheme has been taken into account. With this mechanism, local traffic priority is increased as the number of queued local packets increases. In particular, when this number is large, both kinds of packets have the same priority. By using this strategy, in all the simulations ran, starvation was avoided in all the nodes.

1. Due to limited memory bandwidth in the network interfaces, a source host may inject *bubbles* into the network, thus lowering the effective reception rate at the in-transit host. This problem has been addressed and can be easily avoided when implementing the MCP code. Also, future implementations of Myrinet interfaces will remove this problem.

2. Myrinet supports mixing links with different bandwidth.

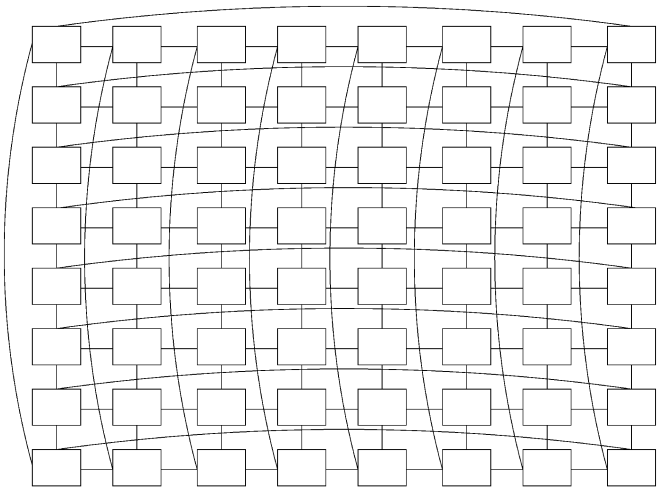

Fig. 6. Crossover type and the average speedup.

5 Performance Evaluation

In this section, we evaluate the ITB mechanism and compare it with the original up*/down* routing used in Myrinet. First, we describe the different topologies and traffic patterns used in the study. Then, we describe the assumed network parameters (links, switches, and network interfaces) which are based on current Myrinet networks. Finally, we present the simulation results.

5.1 Network Model

The network is composed of a set of switches and hosts, all of them interconnected by links. To perform a detailed study, we evaluate several regular and irregular topologies. First, we have selected three well-known regular topologies:[3]

- 2D Torus. The evaluated 2D Torus network is made up of 64 16-port switches. Each switch is connected to each of its four neighbors through a single link. There are eight hosts connected to each switch, so there are 512 hosts in the whole system. There are four ports left open in each switch. This topology is shown in Fig. 6.
- 2D Torus with express channels. This topology is similar to the 2D Torus except that all the switches are also connected to their second-order neighbors (neighbors located two hops away in each dimension) using express channels [5]. Each switch has 16 ports. There are 8 hosts connected to each switch, so there are 512 hosts in the whole system. All ports are used in all switches. The 2D Torus with express channels network is shown in Fig. 7.
- CPLANT. This topology is used in the Computational Plant (CPLANT) at the Sandia National Laboratories [15]. It is made up of 50 16-port switches connecting 400 nodes (each switch has eight hosts attached to it). Forty-eight switches are grouped into six groups of eight switches. Each switch uses four ports to connect to other switches in the same group and four ports to connect to its equivalent switches in the remaining groups. Each

3. Some of them have already been implemented using Myrinet switches.

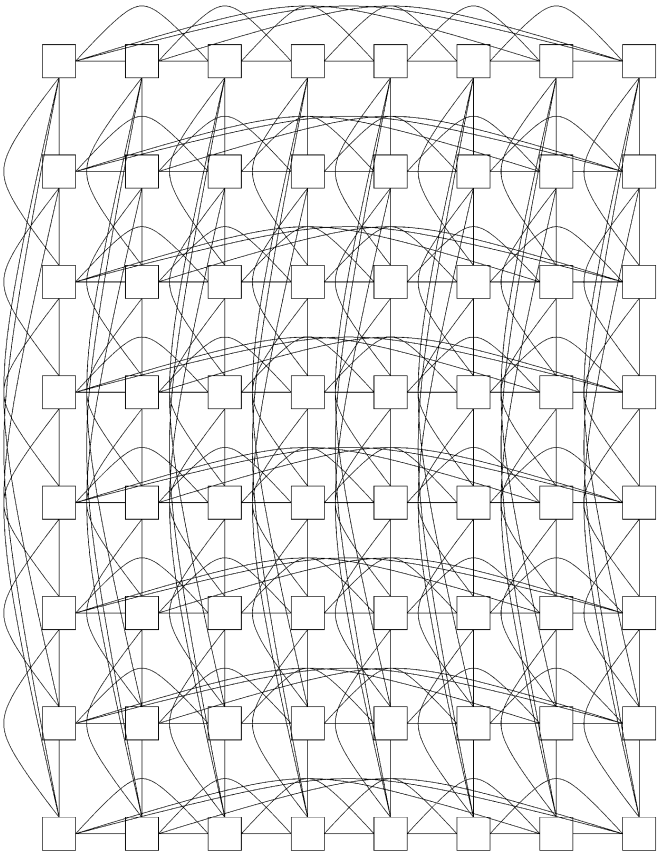

Fig. 7. Average time for each type of crossovers.

group forms an hypercube topology in which an additional link is used at each switch to connect to the farthest node in the group. The six groups form an incomplete hypercube, which also contains connections between farthest nodes. The remaining two switches form an additional group. Therefore, the resulting topology in not completely regular. The CPLANT topology is shown in Fig. 8.

In order to evaluate the ITB mechanism with irregular topologies, we have randomly generated several topologies taking into account only three restrictions. First, we assume that there are exactly four hosts connected to each switch. Second, all the switches in the network have the same size.

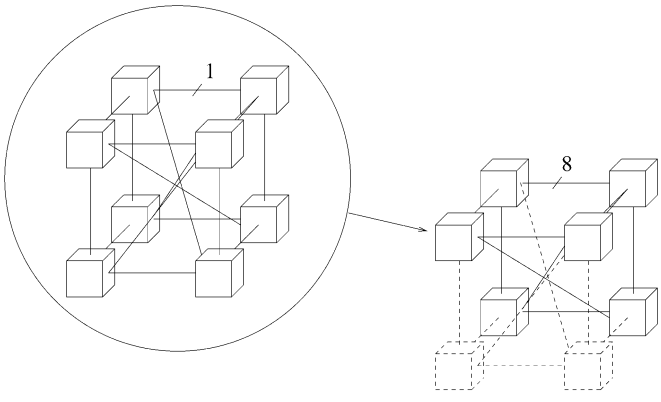

Fig. 8. Average speedup for each mutation type.

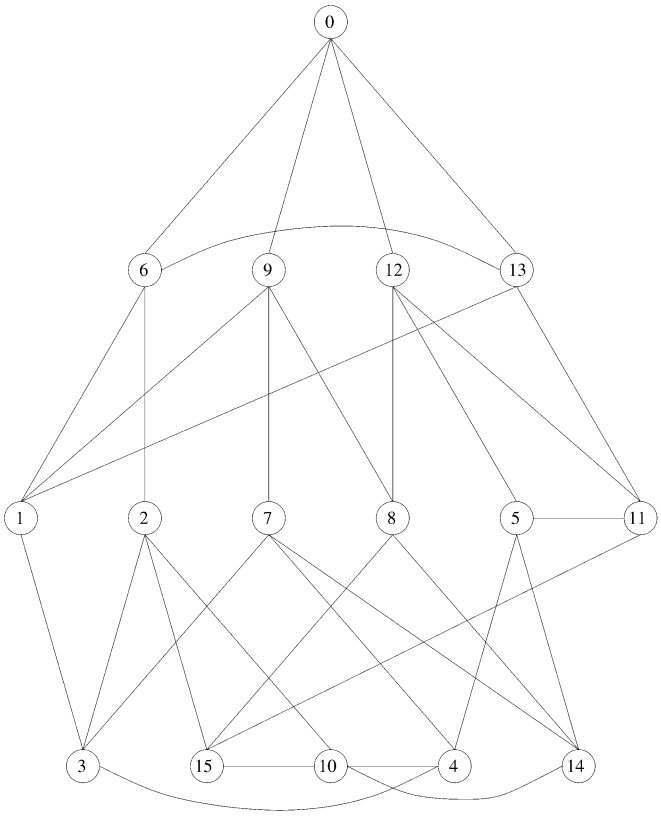

Fig. 9. Average time for both forms of mutation.

We assume that each switch has eight ports. So, there are four ports available to connect to other switches. Finally, two neighboring switches are connected by a single link. These assumptions are quite realistic and have already been considered in other evaluation studies [20], [19]. In Fig. 9, we show one of the irregular topologies that we have used in the evaluation.

To analyze the influence of the network size on system performance, we used different network sizes. In particular, irregular networks with 8, 16, 32, and 64 switches have been analyzed, so there are 32, 64, 128, and 256 hosts in the system, respectively. Finally, to make results independent of a particular topology, we evaluate up to 40 random irregular topologies, 10 for each network size.

5.2 Traffic Patterns

In order to evaluate the ITB mechanism with different traffic patterns, we used different message destination distributions to generate network traffic. The distributions are the following:

- Uniform distribution. The destination of a message is randomly chosen with the same probability for all the hosts. This pattern has been widely used in other evaluation studies [2], [6].
- Bit-reversal distribution. The destination of a message is computed by reversing the bits of the source host identification number. This pattern has been selected taking into account the permutations that are usually performed in parallel numerical algorithms [10], [11]. Notice that this distribution can only be used when the number of nodes is a power

of 2. This distribution is applied to all the networks presented above but to the CPLANT network.

- Hotspot distribution. A percentage of traffic is sent to one host (the hotspot). The selected hotspot location is chosen randomly. The rest of the traffic is generated randomly using a uniform distribution.
- Local distribution. Message destinations are, at most, l switches away from the source host, and are randomly computed. We will study the effects of local distribution with $l = 3$, $l = 4$, and $l = 5$, depending on the topology used.

For each simulation run, we assume that the message generation rate is constant and the same for all the hosts. We evaluate the full range of traffic, from low load to saturation. And finally, we have considered different message lengths. In particular, 32, 512, and 1,024-byte messages have been considered.

5.3 Myrinet Links, Switches, and Interfaces

We assume short LAN cables [13] to interconnect switches and workstations. These cables are 10 meters long, offer a bandwidth of 160 MB/s, and have a delay of 4.92 ns/m (1.5 ns/ft). Flits are one byte wide. Physical links are also one flit wide. Transmission of data across channels is pipelined [18]. Hence, a new flit can be injected into the physical channel every 6.25 ns and there can be a maximum of 8 flits on the link at a given time.

A hardware "stop and go" flow control protocol [1] is used to prevent packet loss. In this protocol, the receiving switch transmits a stop(go) control flit when its input buffer fills over (empties below) 56 bytes (40 bytes) of its capacity. The slack buffer size is fixed at 80 bytes.

Each Myrinet switch has a simple routing control unit that removes the first flit of the header and uses it to select the output link. That link is reserved when it becomes free. Assuming that the requested output link is free, the first flit latency is 150 ns through the switch. After that, the switch is able to transfer flits at the link rate, that is, one flit every 6.25 ns. Each output port can process only one packet header at a time. An output port is assigned to waiting packets in a demand-slotted round-robin fashion. When a packet gets the routing control unit, but it cannot be routed because the requested output link is busy, it must wait in the input buffer until its next turn. A crossbar inside the switch allows multiple packets to traverse it simultaneously without interference.

Each host is connected to the Myrinet network through a network interface card (NIC). This card contains the LANai processor, some buffer memory (ranging from 2MB in the earlier versions to 8MB in the latest ones) [12], and three DMA devices. Each NIC contains a routing table with only one entry for every possible packet destination. The LANai processor fills the routing table and uses it to send packets to other hosts. The way tables are filled determines the routing scheme that will be used. We are interested in comparing the performance achieved by the original Myrinet routes using the up*/down* routing algorithm with the ones supplied by the ITB mechanism.

The original Myrinet routes have been obtained by using the `simple_routes` program from the Myricom GM [8] protocol distribution. This program computes the entire set of up*/down* paths and then selects the final set of up*/down* paths (one path for every source-destination pair) trying to balance traffic among all the links. This is done by using weighted links. So, it may happen that the `simple_routes` program selects a nonminimal up*/down* path, instead of an available minimal up*/down* path. In fact, we have compared the performance of the `simple_routes` routing scheme versus the original up*/down* one that uses always a minimal up*/down* path, if available. We concluded that the routes given by the `simple_routes` program always achieve higher network throughput than the original up*/down* routes. Also, by using the routes generated by the `simple_routes` program, we simulate the behavior of Myrinet with its original routing algorithm.

The latest versions of the `simple_routes` program also offer the choice of computing minimal paths by using the `-shortest-path` option. However, with this option deadlock-freedom is not guaranteed. Also, there is another option (`-remove-deadlocks`) that searches for cycles and tries to heuristically remove them by computing alternative paths. However, when both options were used, the routes computation time grew exponentially and the program was unable to find deadlock-free routes for medium size (32-switch) networks.

In the case of minimal routing with ITBs, the incoming packet must be recognized as in-transit and the transmission DMA must be reprogrammed. We have used a delay of 275 ns (44 bytes received) to detect an in-transit packet, and 200 ns (32 additional bytes received) to program the DMA to reinject the packet.[4] Also, 512KB of memory has been reserved for ITB packets at each Myrinet interface card.

5.4 Simulation Results for Irregular Networks

In this section, we show the results obtained from the simulation of Myrinet networks using both the original Myrinet up*/down* routing scheme and the new routing strategy on irregular networks. We refer to the original routing as UD, and to the new routing mechanism as ITB. We group results by traffic patterns. For each destination distribution, we show the increase in network throughput when using ITBs with respect to the original routing algorithm. In particular, we show the minimum, maximum, and average increase for the randomly generated irregular topologies we have used for different network sizes. Also, for the uniform distribution of message destinations we will show results using different message sizes and we will also plot the average message latency (measured in nanoseconds) versus the average accepted traffic (measured in flits/ns/switch) for some selected topologies.

5.4.1 Uniform Distribution of Message Destinations

In Table 1, we can see the increase in network throughput when using the ITB mechanism for different network and message sizes for the uniform distribution of message destinations. For small networks (eight switches), the use of ITBs does not always increase throughput. The reason is that in small networks many up*/down* routes are

4. These timings have been measured on a real Myrinet network. Average timings have been computed from the transmission of more than 1000 messages using the Real Time Clock register (RTC) of the Myrinet interface card.

TABLE 1
Factor of Throughput Increase When Using the ITB Mechanism

Sw	Msg.Size	Min	Max	Avg
8	32	0.90	1.10	0.97
16	32	1.09	1.63	1.33
32	32	1.66	2.40	2.00
64	32	2.60	3.89	3.21
8	512	0.81	1.05	0.92
16	512	1.00	1.50	1.25
32	512	1.44	2.17	1.76
64	512	2.38	3.25	2.72
8	1024	0.83	1.13	0.92
16	1024	1.00	1.50	1.27
32	1024	1.50	2.01	1.77
64	1024	2.25	3.20	2.65

minimal. Hence, the ITB mechanism does not help very much. Moreover, it introduces some overhead that decreases performance.

As network size increases, the up*/down* routing algorithm does not scale well [20]. However, this is not the case of ITB. For 16-switch networks, ITB always increases network throughput, allowing from 25 percent to 33 percent more traffic on average. In larger networks (32 switches), benefits are even more noticeable with an average improvement ranging from a factor increase of 1.76 when using 512-byte messages to more than doubling network throughput when using 32-byte messages. Moreover, ITB always increases throughput for 32-switch networks (the minimum factor of throughput increase obtained is 1.44 for 512-byte messages).

For large network sizes (64 switches), ITB clearly outperforms UD routing. The minimum factor of throughput increase is 2.25 in a particular network with 1,024-byte messages, whereas the maximum factor of improvement is 3.89 for 32-byte messages. The average results show that ITB drastically increases performance over UD.

As mentioned before, a drawback of the ITB mechanism is the additional latency suffered by messages that use ITB, especially when network traffic is low. Table 2 shows the percentage of message latency increase for low traffic when using the ITB mechanism with respect to the original up*/down* routing. The table shows minimum, maximum, and average increases for different network and message sizes. For small networks (eight switches), the average increase in message latency is small. On average, it ranges from 0.22 percent for 1024-byte messages to 2.24 percent for 32-byte messages. In these small networks, most up*/down* routes are minimal and a low number of ITBs are used on average. So, average message latency does not increase too much.

Latency is increased significantly only for short messages (32 bytes) in medium and large networks (16 switches or more). The average latency increase is about 12 percent. However, for 512 and 1,024-byte messages the maximum latency increase is only 2.73 percent. As ITBs add a constant latency increase, the larger the message, the lower the relative introduced overhead.

TABLE 2
Percentage of Message Latency Increase for Low Traffic When Using the ITB Mechanism

Sw	Msg.Size	Min	Max	Avg
8	32	-0.28	7.86	2.24
16	32	7.31	14.87	10.32
32	32	10.59	14.16	12.93
64	32	10.12	15.15	12.69
8	512	-0.02	1.39	0.48
16	512	0.99	2.73	1.65
32	512	1.19	2.52	1.82
64	512	-1.12	1.52	0.42
8	1024	0.00	0.98	0.22
16	1024	-0.08	1.15	0.52
32	1024	-2.22	0.34	-0.85
64	1024	-3.90	-1.34	-2.27

Indeed, for large networks (32 and 64 switches) and long messages (1,024 bytes), ITB even reduces message latency on average. This is due to the fact that the reduction in message latency due to allow crossing minimal paths is more important than the latency overhead due to the use of ITBs.

In Figs. 10a, 10b, and 10c we show the behavior of the UD and ITB routing algorithms for different network sizes (16, 32, and 64 switches, respectively) on selected topologies. They are the ones in which the improvement achieved by ITB is closer to the average improvement for the corresponding network sizes. Message size is 512 bytes. As it can be seen, in this case, the ITB mechanism does not increase latency at low loads with respect to UD. Moreover, as ITB saturates at a much higher load, the improvement over UD rapidly increases with network size, as indicated in Table 1.

5.4.2 Other Message Destination Distributions

Table 3 shows the increase in network throughput provided by ITB for the bit-reversal, local, and hotspot traffic patterns. In the case of the bit-reversal, the obtained results are qualitatively similar to the ones obtained for the uniform distribution. For small networks (eight switches), ITB performs similarly to UD. For large networks, the larger the network the higher the benefits obtained.

When using a local traffic pattern, with $l = 3$ and $l = 5$, the increase in network throughput depends on the locality factor (l). As it can be seen, when locality is high ($l = 3$ switches), ITB does not improve over UD because the latter offers minimal paths almost for all source-destination pairs. As a consequence, ITB adds few minimal paths to UD routing. In fact, the average number of ITBs used per message is very low (0.0008 for 64-switch networks). Therefore, even for large networks, ITB does not help to increase throughput.

However, as locality decreases ($l = 5$ switches),[5] the behavior of ITB improves significantly. In particular, for 16-switch networks, throughput is increased for all the topologies. Also, for 32 and 64-switch networks, ITB always increases throughput. In this case, messages use, on average,

5. Notice that results are not shown for eight-switch networks because all destinations are, at most, four switches away from sources.

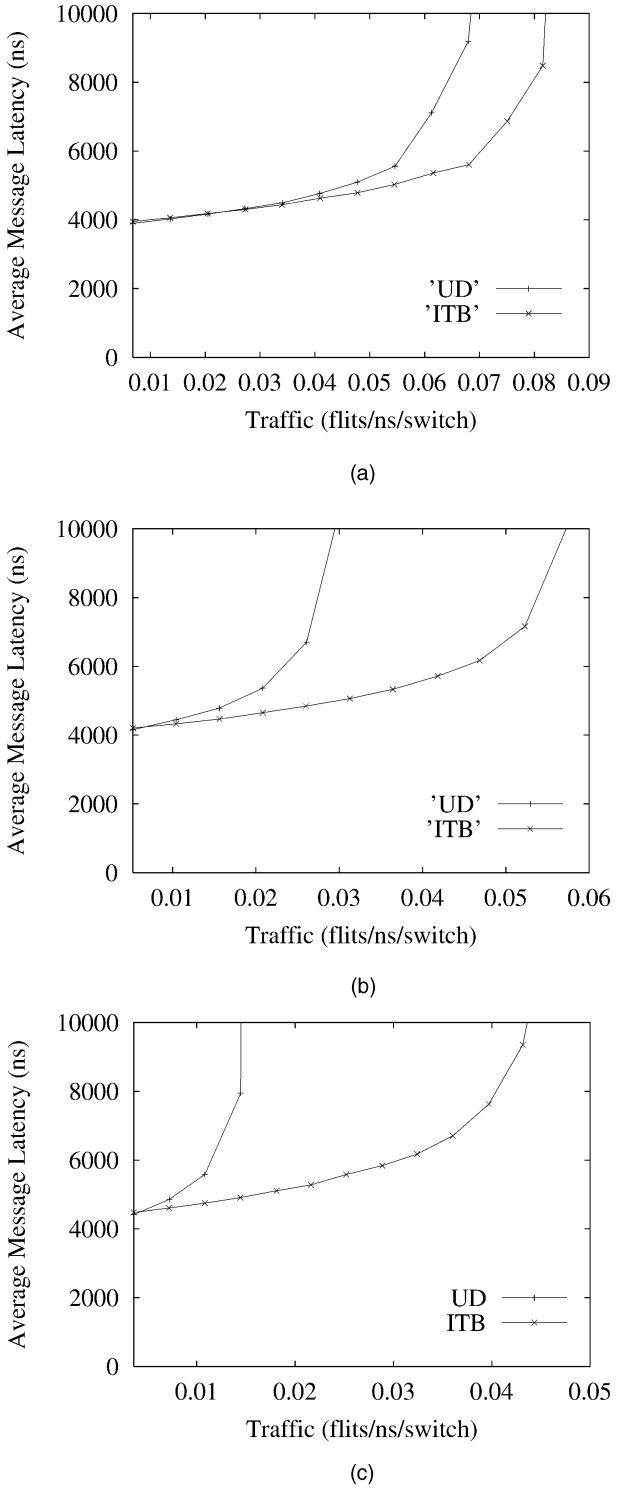

Fig. 10. Algorithm type and speedup.

0.381 in-transit buffers (for the same 64-switch network mentioned above).

Finally, when using the hotspot traffic pattern, most messages are sent to one host. The selected host is chosen randomly. The same host number is used for all the topologies. In order to use a representative hotspot distribution, we have used different percentages depending on network size. In particular, we have used 30 percent, 20 percent, 15 percent, and 5 percent for 8, 16, 32, and 64-switch networks, respectively. The rest of the traffic is randomly generated using a uniform distribution. Although the improvements achieved by ITB are slightly lower than the ones obtained when using the uniform distribution (see Table 1), they are still noticeable. As with the other traffic patterns analyzed, ITB does not help for 8-switch networks. But as network size increases, the benefits of using ITB also increase.

TABLE 3
Factor of Throughput Increase When Using the ITB Mechanism for the Bit-Reversal, Local, and Hotspot Traffic Patterns

Sw	Traffic	Min	Max	Avg
8	Bit-Reversal	0.86	1.15	0.99
16	Bit-Reversal	0.89	2.00	1.29
32	Bit-Reversal	1.64	2.52	2.00
64	Bit-Reversal	2.31	3.56	2.79
8	Local ($l = 3$)	0.84	1.00	0.93
16	Local ($l = 3$)	0.91	1.09	0.99
32	Local ($l = 3$)	0.91	1.08	1.01
64	Local ($l = 3$)	0.89	1.01	0.99
8	Local ($l = 5$)	-	-	-
16	Local ($l = 5$)	1.05	1.55	1.32
32	Local ($l = 5$)	1.40	1.63	1.51
64	Local ($l = 5$)	1.32	2.03	1.60
8	Hotspot	0.91	1.10	1.01
16	Hotspot	1.00	1.20	1.09
32	Hotspot	1.10	1.62	1.30
64	Hotspot	1.58	3.00	2.21

5.4.3 Memory Requirements of the ITB Mechanism

As stated earlier, to ensure deadlock-freedom, it must be guaranteed that the amount of memory for in-transit buffers is not exhausted at any host. This is guaranteed with the memory available at the NIC and, if required, with the host main memory. Another solution is to discard packets when the NIC buffer overflows.

In order to quantify the memory requirements of the ITB mechanism, we have obtained the number of in-transit packets that are simultaneously stored at each NIC for different traffic rates. In particular, we have analyzed an irregular 32-switch network using a uniform traffic pattern with 32-byte messages. Simulation was run for a period of time enough to deliver two million of packets. Results are shown in Fig. 11. Fig. 11a plots the average message latency versus traffic. Figs. 11b and 11c show the average and maximum number of 32-byte ITB packets that are simultaneously at each NIC of the network for different injection rates, respectively. In particular, in Fig. 11b traffic rate is equal to 0.034 flits/ns/switch (network is reaching saturation) and in Fig. 11c traffic rate is 0.039 flits/ns/switch (network is beyond saturation).

Even when the network is approaching saturation (Fig. 11b), we can observe that the amount of memory needed at the NICs is negligible. No more than 15 packets (480 bytes without considering headers) are hold at the same time at a particular NIC. When network is saturated (Fig. 11c), we can observe that more space is required to store in-transit packets. However, the maximum number of

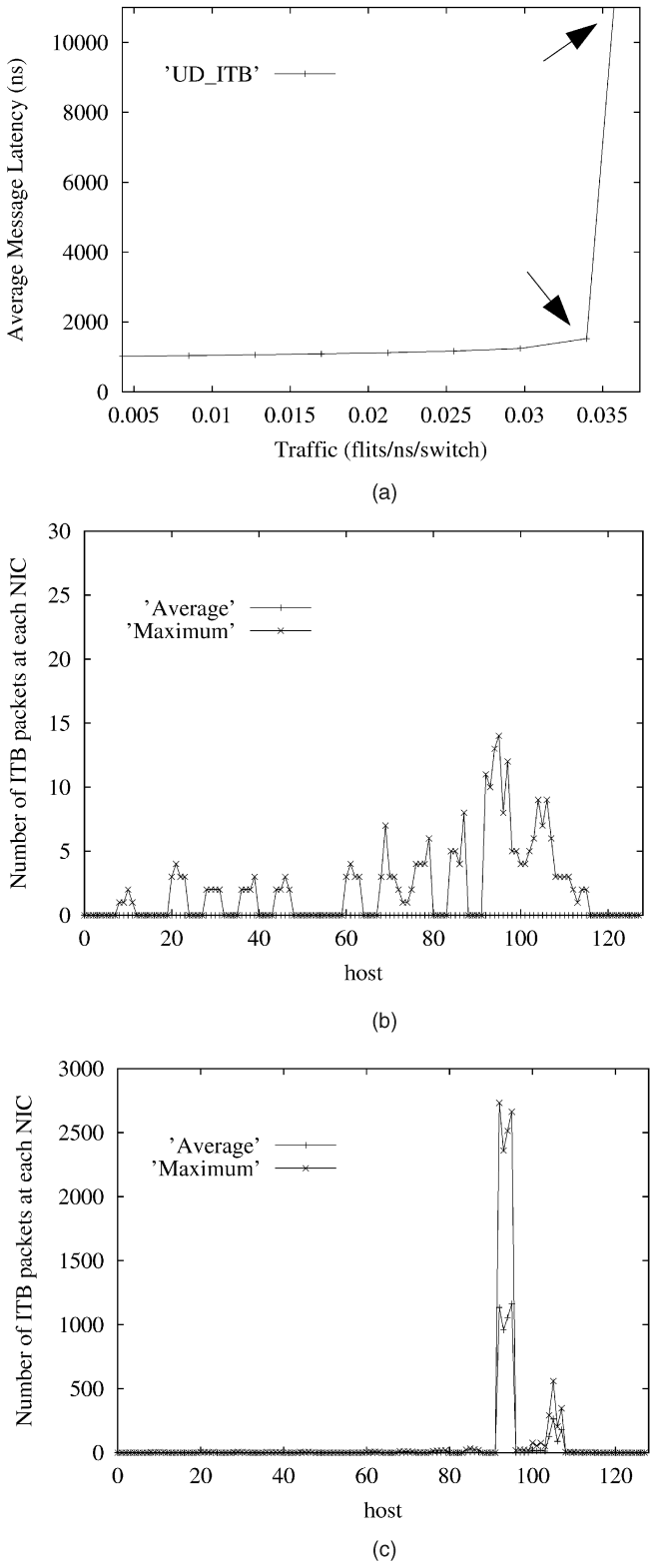

Fig. 11. Computation time for different algorithms.

ITB packets at a particular NIC for a two-million packets simulation is lower than 3,000 (96KB without considering headers). The same behavior has been experienced for other network topologies and packet sizes.

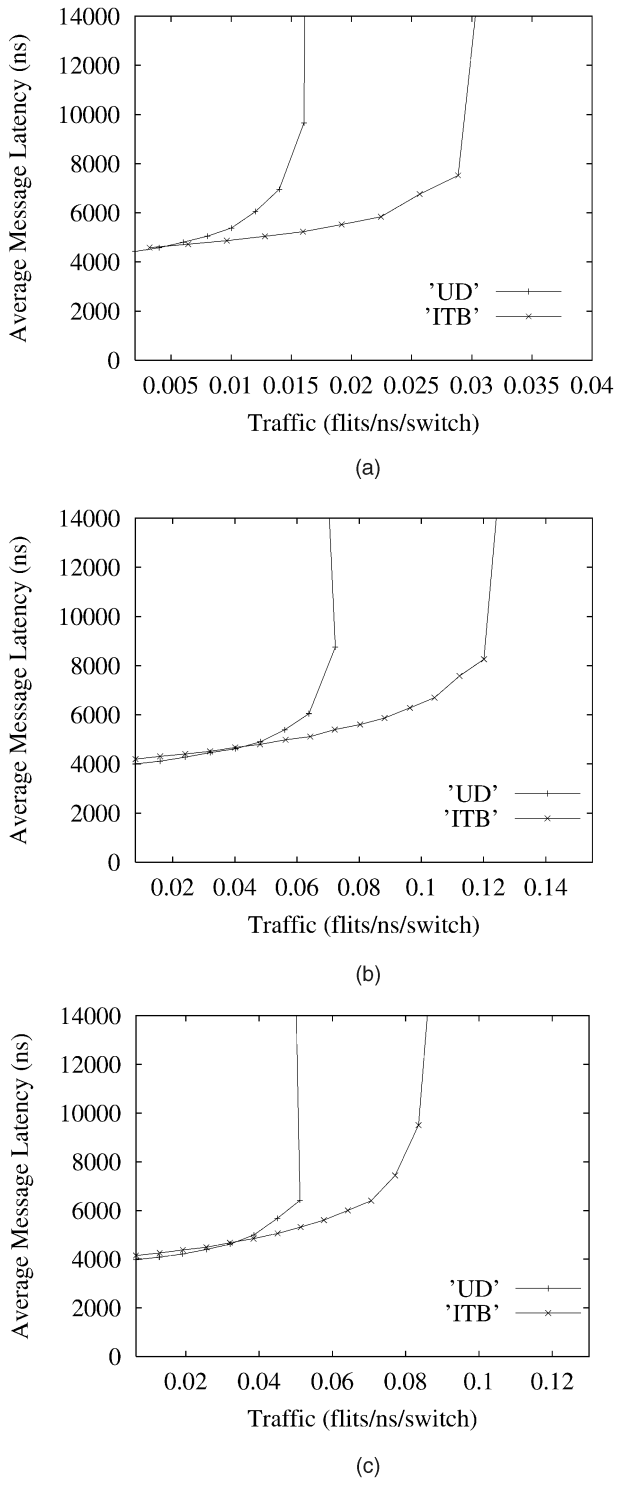

Fig. 12. Computation time for different mutation values.

Taking into account that current NICs offer up to 8MB of available memory and that network load will not be usually beyond saturation, we can conclude that the probability of NIC memory overflow is very low. So, in the case of NIC memory overflow, the best solution will be to discard the packet.

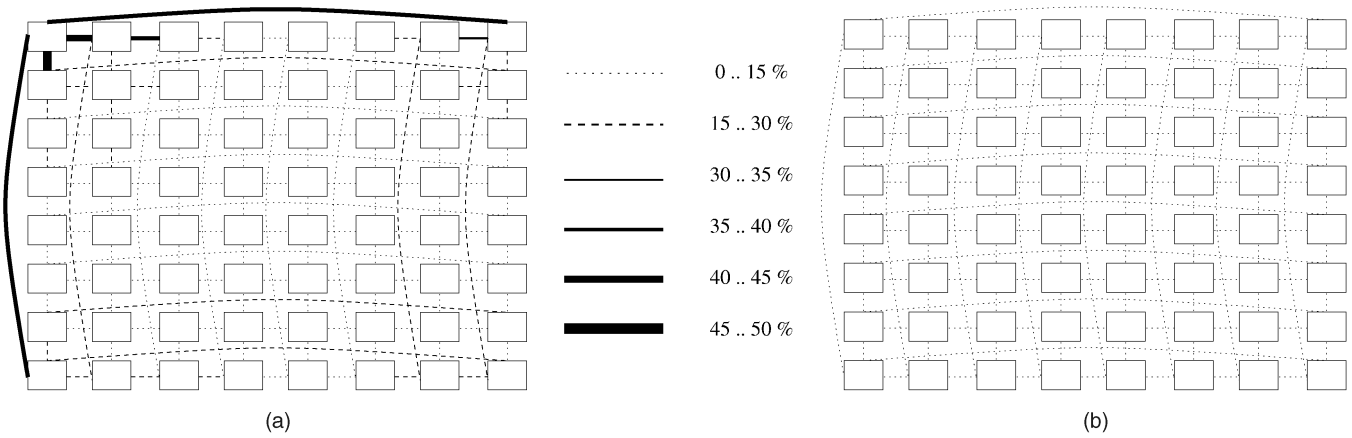

Fig. 13. Computation time for each crossover probability.

5.5 Simulation Results for Regular Topologies

In this section, we show the results obtained from the simulation of Myrinet networks using both the original up*/down* routing scheme and the new routing strategy on regular networks. As with irregular topologies, we group results by traffic patterns. For the sake of brevity, we will only show plots of average message latency versus traffic for the uniform distribution of message destinations.

As stated in the introduction section, when using regular topologies, Myrinet up*/down* routing is less restrictive than in irregular ones because the up*/down* scheme supplies minimal paths for almost all destinations. However, network un-balancing will be still noticeable in such regular networks. So, we will also show the link utilization under different traffic conditions to better analyze the effect of ITBs on traffic balancing.

5.5.1 Uniform Distribution of Message Destinations

Fig. 12 shows the performance of the ITB mechanism and the original Myrinet up*/down* routing algorithm for the uniform distribution of message destinations. In the 2D Torus network (Fig. 12a) ITB doubles the throughput achieved by the original Myrinet routing algorithm. In particular, ITB reaches 0.03 flits/ns/switch while UD saturates at 0.015 flits/ns/switch.

Although up*/down* routing does not always provide minimal paths, 80 percent of the paths computed by the original Myrinet routing algorithm are minimal in this topology. However, the ITB mechanism uses always minimal paths. So, the ITB mechanism adds 20 percent of minimal paths to up*/down* routing. As a consequence, the average distance to destination (measured as the number of traversed links) for up*/down*routing is 4.57 whereas with the ITB mechanism is 4.06. However, this does not fully justify the performance improvement achieved when using in-transit buffers.

As we stated before, another drawback of up*/down* routing is that it forces most of the traffic to cross the root switch. The ITB mechanism distributes network traffic better by allowing the use of alternative paths. Fig. 13 shows the link utilization for UD and ITB when traffic is 0.015 flits/ns/switch (UD saturation point). When using UD routing (Fig. 13a), links near the root switch (the top leftmost switch) are congested (utilization in those links reaches 50 percent), whereas the utilization of the rest of links in the network is low (65 percent of links have a utilization lower than 10 percent). On the other hand, the ITB routing algorithm (Fig. 13b) balances traffic among all the links in the network. The utilization of all the links is less than 12 percent. For higher traffic, when ITB is reaching its saturation point, it still distributes traffic evenly among all the links. Fig. 14 shows the link utilization for ITB routing at injection rate equal to 0.03 flits/ns/switch (ITB saturation point). Traffic is quite balanced among all

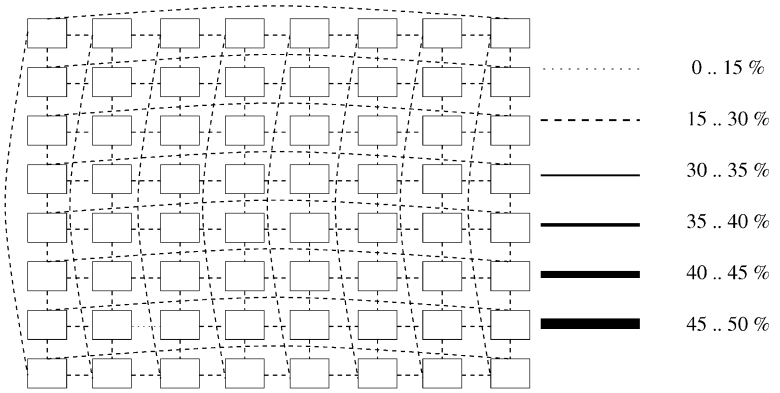

Fig. 14. Time taken for different numbers of tasks.

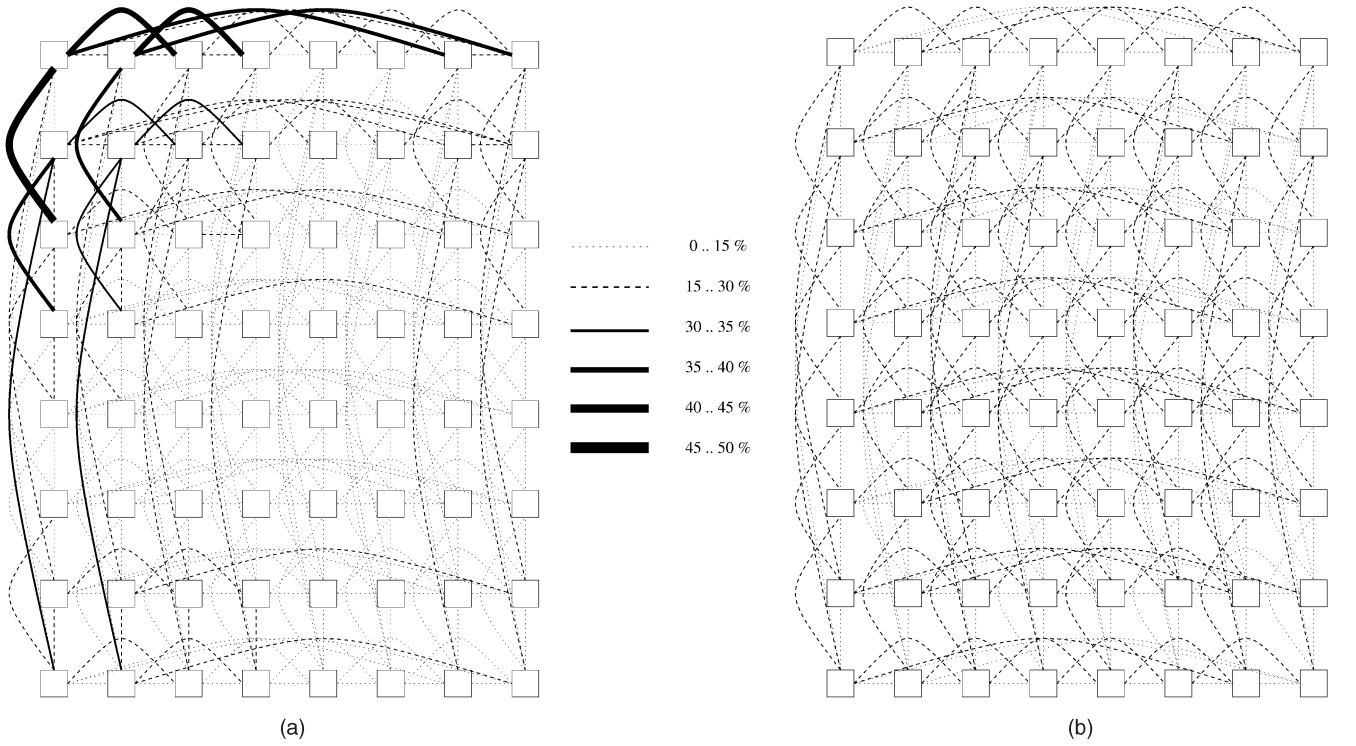

Fig. 15. Percentage error of the heuristic optimal solution.

the links (link utilization ranges from 14 percent to 29 percent). So, when using ITB, minimal paths are always used and, most important, traffic is evenly balanced among all the links in the network. This is the reason of doubling the network throughput achieved by up*/down* routing.

On the other hand, it can be observed that the network saturates when link utilization is still low. The long routing time (150 ns) and the small capacity of slack buffers (80 bytes) lead to a high correlated message blocking.

Fig. 12b shows results for the 2D Torus with express channels. The benefits of using ITBs are slightly smaller than in the 2D Torus because when using express channels there are twice as many links and there are more alternative paths towards the root switch. UD multiplies the throughput achieved in the 2D Torus network by 4.6, reaching 0.07 flits/ns/switch, whereas ITB multiplies network throughput by 4, reaching 0.12 flits/ns/switch. The increase in network throughput (with respect to the 2D Torus network) is due to the added express channels.

The use of express channels also increases the number of minimal paths provided by UD. In this case, the percentage of minimal paths is 94 percent. So, providing more minimal paths with ITBs is less important than in the 2D Torus network. On the other hand, traffic balance also plays a key role in this network. Fig. 15 shows link utilization for UD and ITB routing at the saturation point for UD (0.07 flits/ns/switch). We can see that when using UD (Fig. 15a) links near the root switch have a utilization near 50 percent, whereas the rest of links in the network have a low utilization (as in the 2D Torus). However, when using ITB routing (Fig. 15b) link utilization is more balanced, as in the 2D Torus. All links have a utilization lower than 30 percent. If we take a closer look at Fig. 15b, we can observe that there are links more frequently used than others. In particular, the added express channels have a utilization of 25 percent, whereas the rest of links have a utilization of 10 percent. Express channels are more frequently used because they provide shorter paths to destinations, while the other links are only used to deliver packets to their final switch (when the packet is one hop away from destination).

Hence, the topology itself promotes an unbalanced use of its channels and, as a consequence, traffic is not evenly distributed among all the links. So, the throughput increase when using ITB is slightly smaller than the one achieved in the 2D Torus. Even though, ITB achieves a throughput increase of 1.71 with respect to UD routing.

Regarding the CPLANT network (Fig. 12c), ITB almost doubles the network throughput achieved by UD. In particular, UD saturates at 0.05 flits/ns/switch whereas ITB saturates at 0.082 flits/ns/switch. CPLANT has a complex topology formed by groups of switches. In this case, this improvement is again mainly due to its ability to better balance network traffic. When UD is used as the routing algorithm, most of the traffic must cross the root switch (that is located in a certain group of switches), unbalancing traffic among all the groups. On the other hand, when using the ITB mechanism, traveling across the root switch is not needed, therefore distributing better the traffic among the groups.

5.5.2 Other Message Destination Distributions

Let us start with the bit-reversal traffic pattern. The obtained performance results (not shown) are similar to the uniform distribution one. In particular, for the 2D Torus, UD throughput is almost doubled when using ITBs. For the

TABLE 4
Factor of Throughput Increase for Different Hotspot Locations and Different Hotspot Traffic

	2-D Torus		2-D Torus w/exp.		CPLANT
	5 %	10 %	3 %	5 %	5 %
Min	2.00	1.31	1.12	1.08	1.11
Max	2.16	1.40	1.17	1.08	1.32
Avg	2.13	1.40	1.13	1.08	1.24

2D Torus with express channels, the benefits of using ITBs are slightly smaller (as for the uniform distribution).

With respect to hotspot traffic pattern, ten different hotspot locations have been considered for each topology. Also, different hotspot traffic loads have been used (3 percent, 5 percent, and 10 percent hotspot traffic). Table 4 shows the throughput achieved by each routing algorithm in the 2D Torus, 2D Torus with express channels and CPLANT networks.

Fig. 16 shows the link utilization for UD and ITB routing schemes with a 10 percent hotspot traffic at the UD saturation point (0.0123 flits/ns/switch) for a 2D Torus. In UD (Fig. 16a), links near the root switch are much more heavily used than links near the hotspot switch (marked as H). On the other hand, when using the ITB routing (Fig. 16b), only links near the hotspot switch start to saturate. So, for the UD routing, the root switch behaves as a big hotspot and saturates the network, whereas ITB routing saturates later due to the hotspot.

Taking into account the results obtained for the uniform distribution in an 2D Torus with express channels, the throughput achieved by UD routing with a uniform distribution is reduced by 26 percent and 49 percent for 3 percent and 5 percent hotspot traffic, respectively. When using ITBs, the reduction of throughput achieved with respect to the uniform distribution is 50 percent and 67 percent, respectively. So, ITB is more heavily affected by hotspots. This confirms our explanation for the 2D Torus network. The root switch is the true hotspot when using UD. Fig. 17 shows the link utilization for UD and ITB at their saturation points (0.0483 flits/ns/switch and 0.0542 flits/ns/switch, respectively) using 3 percent hotspot traffic.

Finally, when a local distribution is used with $l = 3$, the benefits of using ITBs are considerably reduced. In the three topologies analyzed, ITB obtains better results than UD, although the differences are smaller than the ones obtained with other message destination distributions. Factor of throughput increases are 1.19, 1.21, and 1.13 for the 2D Torus, 2D Torus with express channels, and the CPLANT network, respectively. UD achieves better results because, due to the local traffic pattern, traffic is evenly distributed among all the links in the network. Moreover, up*/down* routing is always able to use a minimal path when the destination is one hop away (two switches away) or is connected to the same switch. So, there are no benefits of using ITBs in these circumstances. Anyway, the ITB mechanism does not perform worse than UD.

If the locality of messages is reduced (destinations now are, at most, $l = 4$ switches away from sources), the improvement achieved by the ITB mechanism is middle way between the results obtained for the uniform (Fig. 12) and the uniform with high locality ($l = 3$). In particular, UD throughput is increased by ITB with factors of 1.33, 1.5, and 2.18 for the 2D Torus, 2D Torus with express channels and CPLANT networks, respectively.

6 A First Implementation of ITBs on Myrinet Networks

In order to demonstrate the feasibility of using the ITB mechanism in real networks, in this section, we present a first implementation of the ITB mechanism on Myrinet GM software [8].

First, a new packet type (ITB packet) has to be created to distinguish between normal Myrinet packets and in-transit packets. New packet types are assigned by Myricom upon request. In particular, in our implementation, the following types of packets are allowed: a normal GM packet, a mapping packet, a packet with an IP packet in its payload, and an ITB packet.

In order to support in-transit packets, the MCP has been modified. The modifications have been made taking care of introducing the minimum overhead. Therefore, we need a fast detection mechanism of an incoming in-transit packet in order to reprogram a DMA transfer and reinject it as soon as possible (even if the packet is still being received), thus providing virtual cut-through switching for ITB packets.

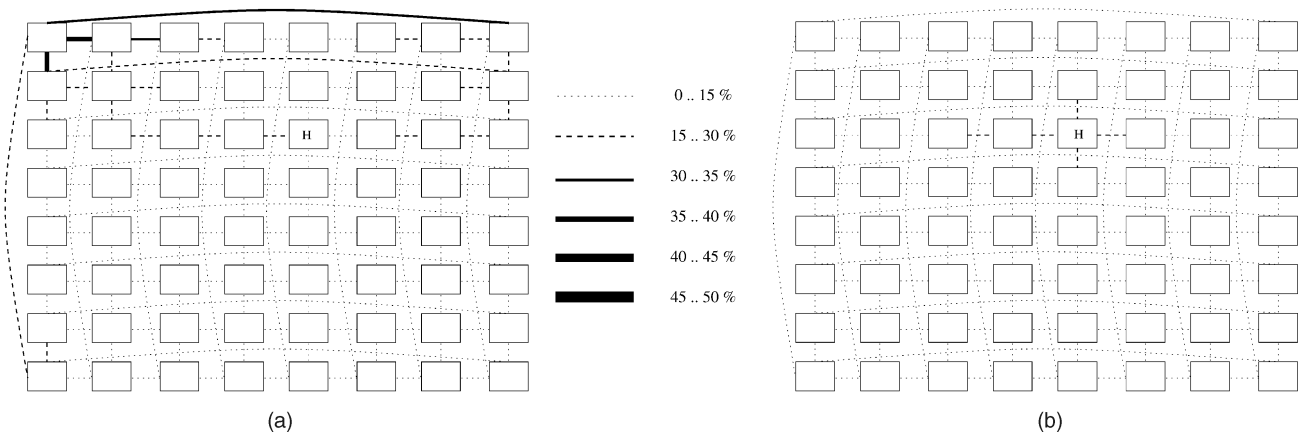

Fig. 16. The Height Algorithm

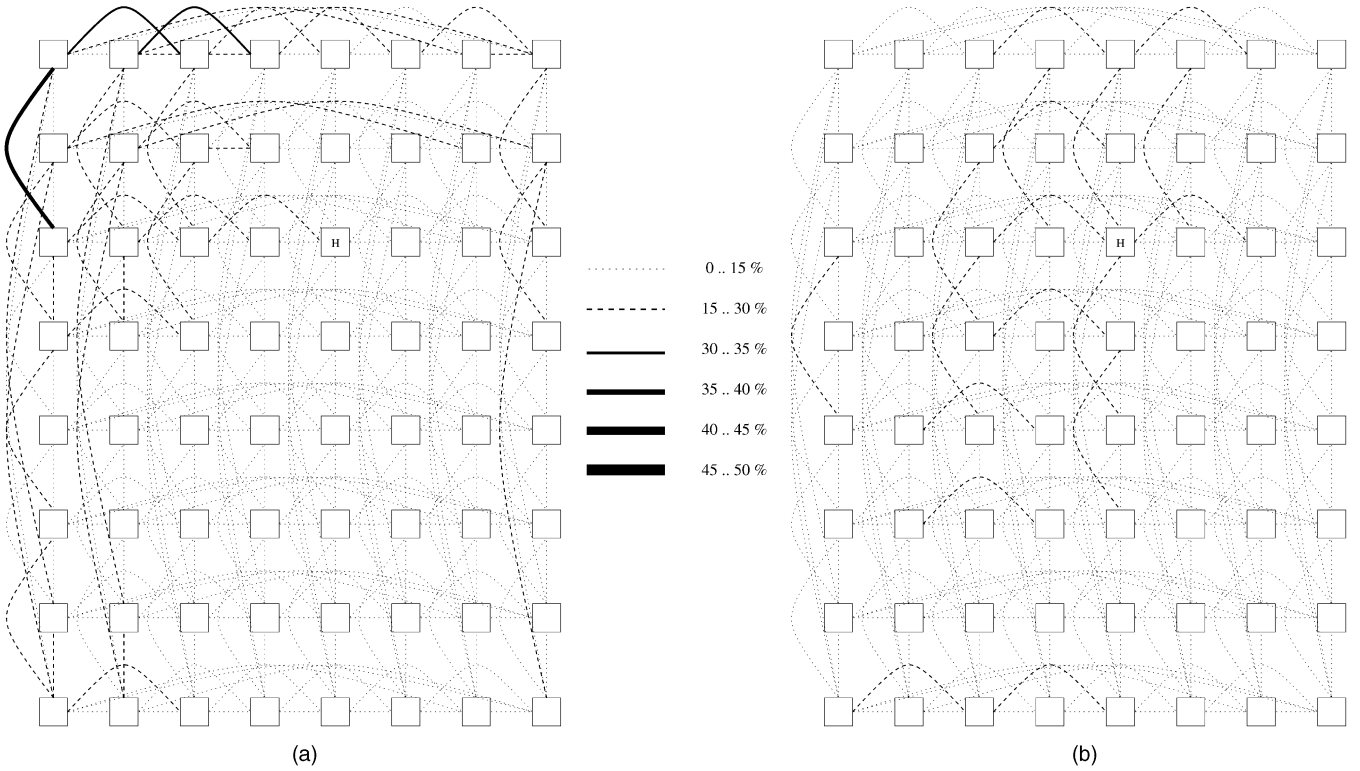

Fig. 17. The Priority Algorithm.

The MCP is composed of four state machines: SDMA, RDMA, Send and Recv (see Fig. 18). SDMA and RDMA control memory transactions between the host and the sending/receiving buffers located in the NIC memory. Send and Recv are responsible for controlling transactions to and from the network. On the other hand, the MCP is also responsible of setting up and detecting the completion of transactions on each interface. It can do this by means of start and finish events that modify the state of the system. The overall state of the system is managed through a number of status bits, being some of them maintained by the LANai hardware while the remaining ones being controlled by software. An event handler is in charge of monitoring the state of the MCP, giving control to the state machine that handles the highest priority pending event.

The MCP Recv state machine has to detect in-transit packets and check whether the Send state machine is free. Figs. 18 and 19 show the changes required in the MCP code. Once an incoming ITB packet is detected, if the output channel is free, the send DMA has to be programmed to reinject the packet. Notice that in this case the Recv state machine is responsible for the in-transit packet reinjection in order to minimize the overhead (avoiding one dispatching cycle delay). This is shown in Fig. 18 as dashed lines. On the other hand, if the output channel is busy, the packet will be sent as soon as it becomes free, as indicated in Fig. 18 in dotted lines. Finally, note that when the Send state machine is reinjecting an ITB packet and this packet is blocked in the network, the Myrinet Stop&Go flow control will stop the transmission and, as a consequence, will temporarily stop the DMA send operation. In this situation, the rest of the ITB packet will remain in its buffer until the transmission is resumed again. If the ITB packet was not completely received in the above situation, the receiving process will not be stopped until the last byte of the ITB packet is received and stored in its corresponding buffer.

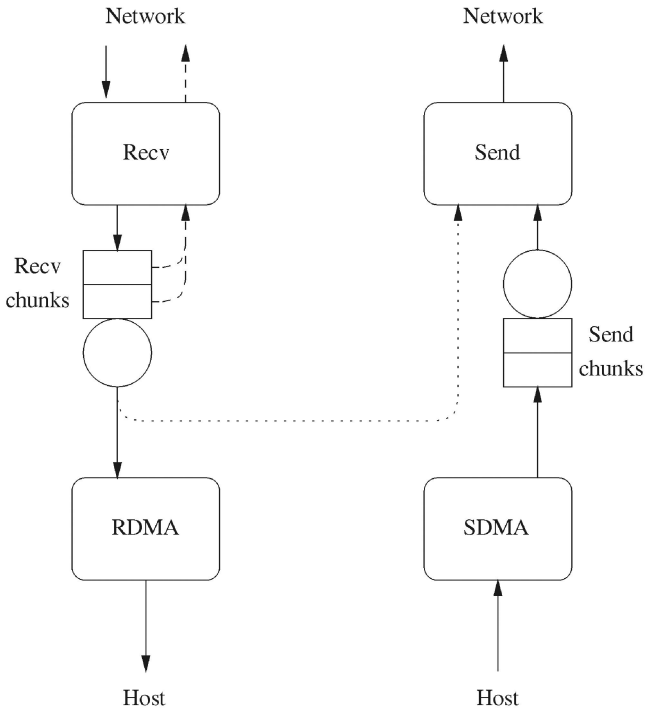

Fig.18. The PMX Crossover Algorithm.

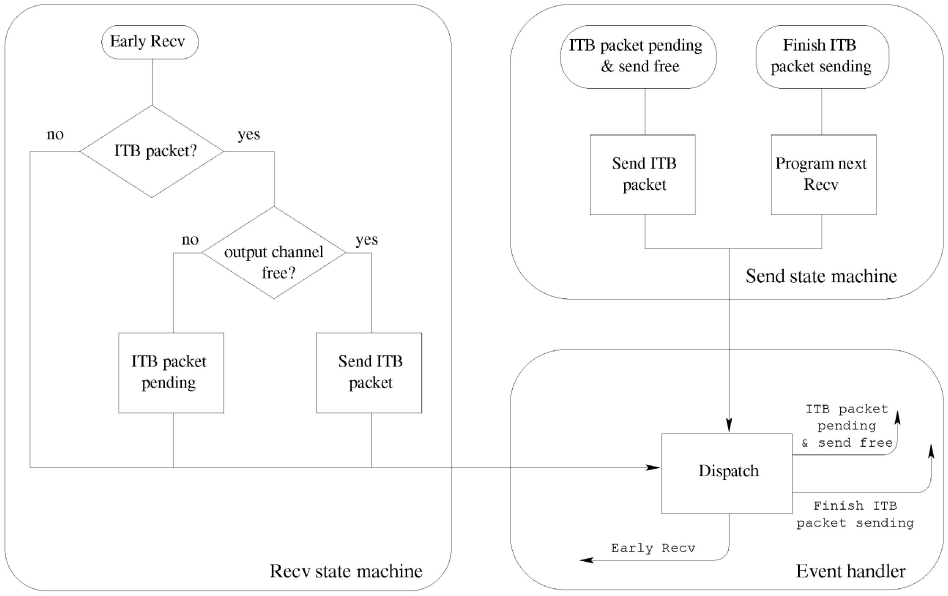

Fig. 19. The Cycle Crossover Algorithm.

In order to detect as soon as possible an in-transit packet, a new high priority event has been included (Early_Recv_Packet event). It is triggered by the LANai hardware when the first four bytes of a packet are received. Fig. 19 illustrates these modifications.

6.1 Preliminary Performance Results

The implementation described above has been made on the GM-1.2pre16 software distribution for Linux [8]. The testbed is comprised of three 450 MHz Pentium III-based computers running SuSE Linux 6.1 with kernel 2.0.36. Computers are attached to a Myrinet network with two M2FM-SW8 switches (eight-Port Myrinet Switch with four LAN ports and 4 SAN ports [12]) according to Fig. 20. Host 1 and in-transit host use M2L-PCI64A-2 NICs (universal, 64/32-bit, 66/33MHz, Myrinet-LAN/PCI interface, PCI short card-2 MB [12]) and host 2 uses a M2M-PCI64A-2 (universal, 64/32-bit, 66/33MHz, Myrinet-SAN/PCI interface, PCI short card-2 MB [12]).

Once the modified GM/MCP has been verified to deliver packets correctly, several tests have been made using the `gm_allsize` test distributed by Myricom in GM-1.2pre16. Firstly, the overhead introduced by the new code in the normal MCP operation has been measured. This test evaluates the impact of adding ITB support to the network. Notice that both normal and ITB packets will suffer this overhead, but only once per packet, as it only affects to the receiving part of the MCP. The test has been done by comparing the point-to-point half-round-trip latency of the original MCP with the modified one when sending packets between hosts 1 and 2, using up*/down* routes. Latencies have been obtained by averaging 100 iterations for each

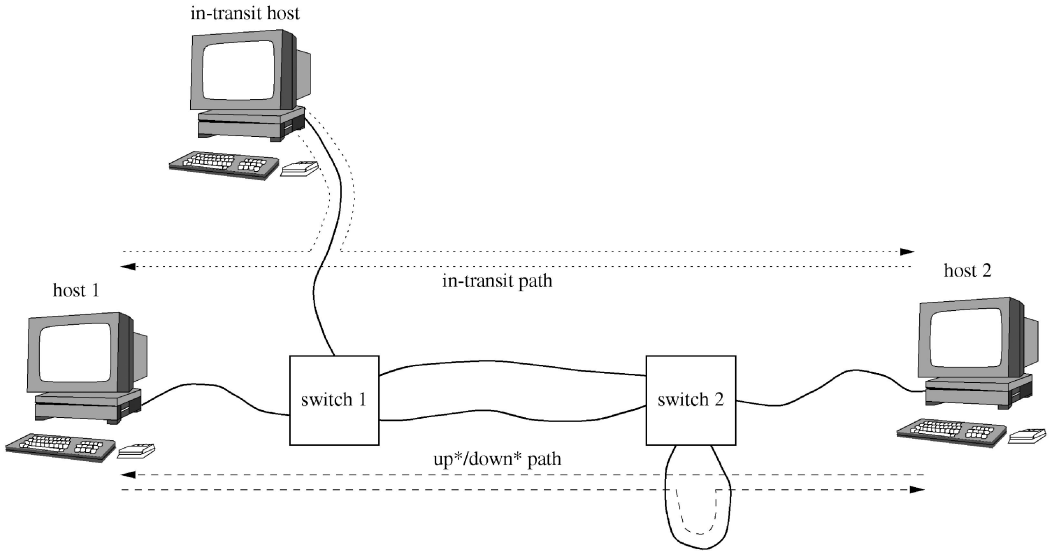

Fig. 20. The Mutation Algorithms.

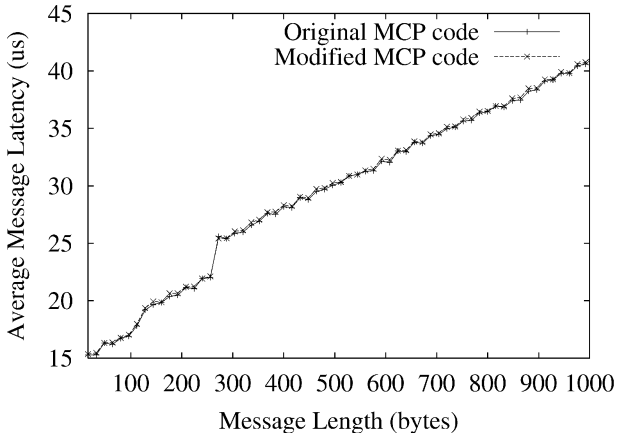

Fig. 21. Message latency overhead of the new GM/MCP code.

packet size. Fig. 21 shows the average latencies measured versus packet length for both MCPs.

As can be seen, overhead is very low. Difference in measured latencies (not appreciated in the figure) does not exceed 300 ns and, on average, is equal to 125 ns. Notice that, as message latency increases (more switches to cross and/or longer packets) the relative impact of this overhead decreases. In our testbed, with packets traversing 2.5 switches, the relative overhead (not shown) ranges from 1 percent for very short packets to 0.4 percent for long packets.

In order to measure the overhead experienced by messages that use ITBs, a second test has been made. This test evaluates the delay associated with the detection of an incoming in-transit packet, its ejection and the reinjection into the network. It has been calculated by subtracting the point-to-point half-round-trip latency of messages being sent between hosts 1 and 2 using the up*/down* path (shown in Fig. 20 as dashed lines) from the equivalent latency of messages that use an ITB at the in-transit host (shown in Fig. 20 as dotted lines). Care has been taken to assure that both the in-transit and up*/down* paths cross the same number of switches (five switches). Notice that the up*/down* path requires a loop in switch 2 for this purpose. On the other hand, given that the test program measures the half-round-trip latency and only one ITB is used, the overhead due to this ITB has been obtained multiplying the result of the above difference by two. In the same way it has been done for the first test, 100 iterations have been averaged for each message size.

To guarantee that this comparison shows only the overhead due to the ITB, both paths have been generated taking into account that messages have to traverse the same number of switches and also they have to cross the same kind of ports (LAN or SAN). It has to be stated that the latency through a switch depends on the type of traversed ports.

Fig. 22a shows the point-to-point half-round-trip latency for messages sent between host 1 and 2 without in-transit buffers (UD) and with one ITB (ITB) versus message length. The resulting absolute overhead (Fig. 22b) is also plotted. As it can be seen, the cost of detecting an ITB packet and handling its reinjection is around 1.3 μs. This value is higher than the

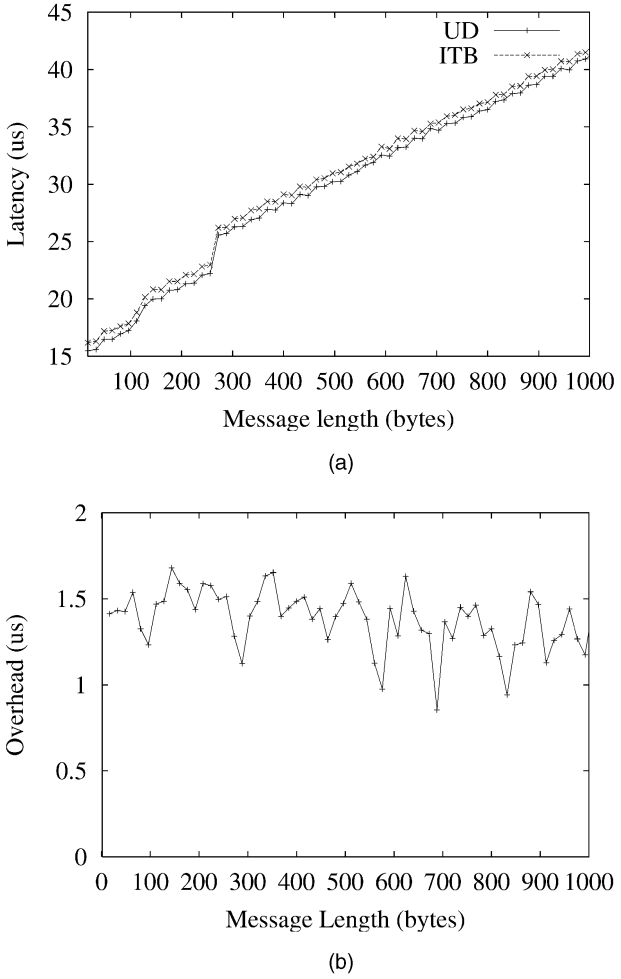

Fig. 22. Message latency overhead of the ITB mechanism: (a) Compared with up*/down* and (b) absolute latency overhead.

one used in the evaluation results section (around 0.5 μs, see Section 5.3). Notice that the value of 0.5 μs was taken from a raw MCP. Hence, a final optimized ITB implementation should be able to reduce the cost of detecting and reinjecting an ITB packet.

7 Conclusions

In this paper, we have proposed the In-Transit Buffer (ITB) mechanism to improve network performance in networks with source routing and wormhole switching. With this mechanism, messages always are forwarded using minimal paths and also network traffic is better balanced.

This mechanism is valid for any network with source routing and it has been evaluated by simulation for the Myrinet network. It can be easily implemented in Myrinet thanks to the flexibility offered by the MCP. We have compared the original Myrinet up*/down* routing algorithm with the new one that uses ITBs. We have used different random irregular topologies (up to 40 topologies), three well-known regular topologies, and different traffic patterns (uniform, bit-reversal, local, and hotspot) with different message sizes (32, 512, and 1,024-byte messages) and different network sizes (8, 16, 32, and 64 switches).

The proposed mechanism achieves its best results with irregular topologies. For small networks (eight switches) and local traffic patterns, the ITB mechanism does not significantly improve network performance, because few minimal paths are added. But, as network size and average distance increase, the benefits obtained by the ITB mechanism also increase. In particular, we have obtained that throughput increases, on average, by a factor ranging from 1.3 to 3.33 with respect to the original Myrinet routing algorithm for 32 and 64-switch irregular networks. In some particular networks, throughput is increased by a factor of 4.7.

For regular networks, the network performance is always improved when using the ITB mechanism but the benefits are slightly lower than in irregular networks. In the best case (uniform and bit-reversal distributions of message destinations) network throughput can be doubled. We have found that this improvement is mainly due to the ability of the ITB mechanism to balance network traffic.

On the other hand, although the proposed technique may increase message latency, the evaluation results show that the overhead is only noticeable when network traffic is not intense and is composed of short messages.

Finally, in order to fully analyze the feasibility of adding the ITB mechanism on a real Myrinet network, we have developed a first implementation on MCP/GM software.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their helpful and valuable suggestions that have improved the presentation and technical contents of this paper. This work was supported by the Spanish Comisión Interministerial de Ciencia y Tecnologí;a (CICYT) under Grant TIC2000-1151.

REFERENCES

[1] N.J. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J. Seizovic, and W. Su, "Myrinet —A Gigabit Per Second Local Area Network," *IEEE Micro,* pp. 29-36, Feb. 1995.
[2] R.V. Bopana and S. Chalasani, "A Comparison of Adaptive Wormhole Routing Algorithms," *Proc. 20th Ann. Int'l Symp. Computer Architecture,* May 1993.
[3] H. Chen and P. Wyckoff, "Performance Characterization of a Terabit Switch and Myrinet as Cluster Interconnects," *Proc. HOT Interconnects,* Aug. 2000.
[4] L. Cherkasova, V. Kotov, and T. Rokicki, "Fibre Channel Fabrics: Evaluation and Design," *29th Int'l Conf. System Sciences,* Feb. 1995.
[5] W.J. Dally, "Express Cubes: Improving the Performance of k-ary n-cube Interconnection Networks," *IEEE Trans. Computers,* vol. 40, no. 9, Sept. 1991.
[6] W.J. Dally, "Virtual-Channel Flow Control," *IEEE Trans. Parallel and Distributed Systems,* vol. 3, no. 2, pp. 194-205, Mar. 1992.
[7] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks, An Eng. Approach,* IEEE CS Press, 1997.
[8] GM Protocol, http://www.myri.com/GM. 2001.
[9] R. Horst, "ServerNet Deadlock Avoidance and Fractahedral Topologies," *Proc. Int'l Parallel Processing Symp.,* Apr. 1996.
[10] J. Kim and A. Chien, "An Evaluation of the Planar/Adaptive Routing," *Proc. Fourth IEEE Int'l Parallel Distributed Processing Symp.,* Dec. 1992.
[11] P.R. Miller, "Efficient Communications for Fine-Grain Distributed Computers," PhD thesis, Southampton Univ., 1991.
[12] Myrinet web page, http://www.myri.com. 2001.
[13] Myrinet, M2-CB-35 LAN cables, http://www.myri.com/myrinet/product_list.html. 2001.
[14] S.S. Owicki and A.R. Karlin, "Factors in the Performance of the AN1 Computer Network," *Digital SRC Research Report 88,* June 1992.
[15] R. Riesen et al., "CPLANT," *Proc. Second Extreme Linux Workshop,* June 1999.
[16] J.C. Sancho, A. Robles, and J. Duato, "New Methodology to Compute Deadlock-Free Routing Tables for Irregular Networks," *Proc. Workshop Comm. and Architectural Support for Network-Based Parallel Computing,* Jan. 2000.
[17] M.D. Schroeder et al., "Autonet: A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links," Technical Report SRC research report 59, DEC, Apr. 1990.
[18] S.L. Scott and J.R. Goodman, "The Impact of Pipelined Channels on k-ary n-Cube Networks," *Proc. IEEE Trans. Parallel and Distributed Systems,* vol. 5, no. 1, pp. 2-16, Jan. 1994.
[19] F. Silla, M.P. Malumbres, A. Robles, P. López, and J. Duato, "Efficient Adaptive Routing in Networks of Workstations with Irregular Topology," *Proc. Workshop Comm. and Architectural Support for Network-Based Parallel Computing,* Feb. 1997.
[20] F. Silla and J. Duato, "Improving the Efficiency of Adaptive Routing in Networks with Irregular Topology," *1997 Int'l Conf. High Performance Computing,* Dec. 1997.

José Flich received the MS and PhD degrees in computer science from the Technical University of Valencia (Universidad Politécnica de Valencia), Spain, in 1994 and 2001, respectively. He joined the Department of Computer Engineering (DISCA), Universidad Politécnica de Valencia in 1998 where he is currently an associate professor of computer architecture and technology. His research interests are related to high performance interconnection networks for multiprocessor systems and cluster of workstations. He is a member of the IEEE and the IEEE Computer Society.

Pedro López received the BEng degree in electrical engineering and the MS and PhD degrees in computer engineering from the Technical University of Valencia (Universidad Politécnica de Valencia), Spain, in 1984, 1990, and 1995, respectively. He joined the Department of Computer Engineering (DISCA), Universidad Politécnica de Valencia in 1986 where he is currently an associate professor of computer architecture and technology. He has taught several courses on computer organization and computer architecture. His research interests include high performance interconnection networks for multiprocessor systems and cluster of workstations. Dr. López is a member of the editorial board of *Parallel Computing*.

M.P. Malumbres received the MS and PhD degrees in computer science from Technical University of Valencia (UPV), Spain, at 1991 and 1996, respectively. He is currently an associate professor in the Computer Engineering Department (DISCA) at UPV. His research and teaching activities are related to multimedia and high-speed networking. He is member of the IEEE and the IEEE Computer Society.

Jose Duato received the MS and PhD degrees in electrical engineering from the Technical University of Valencia, Spain, in 1981 and 1985, respectively. He is currently a professor in the Department of Computer Engineering (DISCA), Technical University of Valencia (Universidad Politécnica de Valencia), Spain, and an adjunct professor in the Department of Computer and Information Science, at The Ohio State University. He is currently researching on multiprocessor systems, networks of workstations, interconnection networks, and multimedia systems. His theory on deadlock-free adaptive routing has been used in the design of the routing algorithms for the MIT Reliable Router, the Cray T3E router, and the router embedded in the new Alpha 21364 microprocessor. He coauthored the text *Interconnection Networks: An Engineering Approach* with S. Yalamanchili and L.M. Ni (published by IEEE CS Press). Dr. Duato served as an associate editor of *IEEE Transactions on Parallel and Distributed Systems* from 1995 to 1997. He is currently serving as an associate editor of *IEEE Transactions on Computers*. Also, he has been or is a member of the Program Committee for several major conferences (ICPADS, ICDCS, Europar, HPCA, ICPP, MPPOI, HiPC, PDCS, ISCA, IPPS/SPDP, ISPAN, CCGrid). He has been the general cochair for ICPP 2001. He is a member of the IEEE.