Performance Evaluation of a New Routing Strategy for Irregular Networks with Source Routing

J. Flich, M. P. Malumbres, P. López and J. Duato
Dpto. Informática de Sistemas y Computadores*
Universidad Politécnica de Valencia
Camino de Vera, 14, 46071–Valencia, Spain
E-mail: {jflich,mperez,plopez,jduato}@gap.upv.es

ABSTRACT

Networks of workstations (NOWs) are becoming increasingly popular as a cost-effective alternative to parallel computers. Typically, these networks connect processors using irregular topologies, providing the wiring flexibility, scalability, and incremental expansion capability required in this environment. In some of these networks, messages are delivered using the up*/down* routing algorithm [9]. However, the up*/down* routing scheme is often non-minimal. Also, some of these networks use source routing [1]. With this technique, the entire path to destination is generated at the source host before the message is sent.

In this paper we develop a new mechanism in order to improve the performance of irregular networks with source routing, increasing overall throughput. With this mechanism, messages always use minimal paths. To avoid possible deadlocks, when necessary, routes between a pair of hosts are divided into sub-routes, and a special kind of virtual cut-through is performed at some intermediate hosts. We evaluate the new mechanism by simulation using parameters taken from the Myrinet network. We show that the current routing schemes used in Myrinet can be improved by modifying only the routing software without increasing its overhead significantly and, most importantly, without modifying the network hardware. The benefits of using the new routing scheme are noticeable for networks with 16 or more switches, and increase with network size. For 32 and 64-switch networks, throughput is increased on average by a factor ranging from 1.3 to 3.3.

Keywords Networks of workstations, irregular topologies, wormhole switching, minimal routing, source routing.

*This work was supported by the Spanish CICYT under Grant TIC97–0897–C04–01 and by Generalitat Valenciana under Grant GV98-15-50

1. INTRODUCTION

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

Currently, the evolution of NOWs is closely related to that of local area networks (LANs). LANs are migrating from shared medium to point-to-point links. As an example, consider the evolution of the Ethernet family up to recent Gigabit Ethernet networks [11]. Although Ethernet is very popular, other commercial LANs have arisen in the high-speed networking arena, trying to provide solutions for some of the Ethernet weaknesses such as quality of service, priority-based traffic, gigabit channels, and flow control mechanisms (ATM, VG100AnyLan, Autonet, Myrinet). Some of them were considered in the recent Gigabit Ethernet Standard IEEE 802.3z.

In some of these networks, packets are delivered using source routing. In these kinds of networks, the path to destination is built at the source host and it is written into the packet header before it is sent. Switches route packets through the fixed path found at the packet header. One example of a network with source routing is Myrinet [1]. Myrinet design is simple and very flexible. In particular, it allows us to change the network behavior through the Myrinet Control Program (MCP) software. This software is loaded on the network adapter program memory at boot time. It initializes the network adapter, 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. The network adapter has to build network routes to each destination during the initialization phase. Network adapters 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 [9] to build these paths. Although the original distributed up*/down* routing scheme provides partial adaptivity, in Myrinet only one of the routes is selected to be included into the routing table, thus resulting in a deterministic routing algorithm. On the other hand, many paths provided by up*/down* routing are non-minimal on certain networks. The probability of finding minimal paths in accordance with the up*/down* restriction decreases as network size increases.

In previous work [12; 13], we analyzed the behavior of distributed routing algorithms on irregular topologies, showing that adaptive routing schemes outperform up*/down* routing schemes by improving routing flexibility and providing minimal paths. Therefore, it would be interesting to analyze the feasibility of using minimal routing in networks with source routing and evaluate its behavior.

In this paper, we take on such a challenge. We propose a new mechanism to implement minimal routing. The mechanism is valid for any network with source routing. In the case of Myrinet, it is easy to implement thanks to the flexibility provided by the MCP program.

The new mechanism always provides minimal routes. In order to be deadlock-free, this technique splits, when necessary, the route between a pair of hosts into sub-routes and forces a special kind of virtual cut-through in the network interface card of the intermediate host. However, splitting routes requires the collaboration of network hosts to forward packets from one sub-route to the next one. This overhead must be taken into account. We will focus on Myrinet networks to evaluate the new mechanism. Actual Myrinet parameters will be taken into account in order to make a fair comparison.

The rest of the paper is organized as follows. In Section 2, the current Myrinet source routing scheme is described. In Section 3 the new mechanism is introduced. In Section 4, the performance of the proposed mechanism is evaluated by simulation. Finally, in Section 5 some conclusions are drawn.

2. MYRINET SOURCE ROUTING

Myrinet uses wormhole switching with source routing to transmit packets between hosts. With this routing technique, the packet header stores the route that the packet has to follow to reach its destination (see Figure 1). To simplify switch operation, each packet 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 header type of the payload). 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, an so on. Each link identifier is discarded after using it. Therefore, each network host must have a representation of the current network topology, in order to build and maintain routes between itself and each potential destination host. 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.

Myrinet uses up*/down* routing [9] to build network routes. Up*/ down* routing is based on an assignment of direction

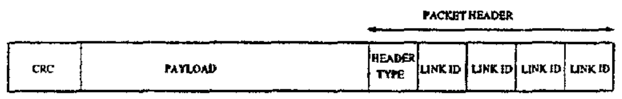

Figure 1: Myrinet packet header.

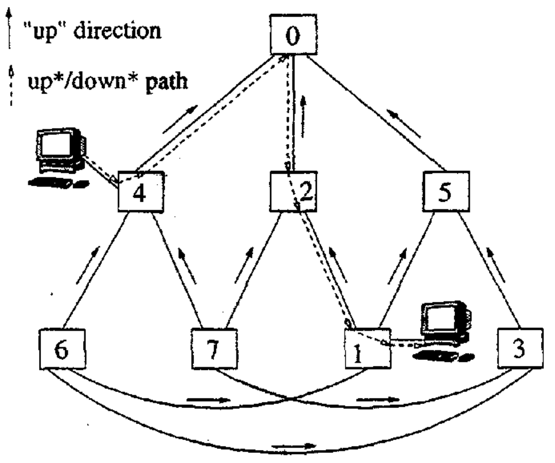

Figure 2: Link direction assignment for an irregular network.

to the operational links. To do so, a breadth-first spanning tree is computed and then, 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 lower ID, if both ends are at switches at the same tree level (see Figure 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 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 dependencies between channels are avoided because a message cannot traverse a link along the "up" direction after having traversed one along the "down" direction.

Up*/down* routing is not always able to provide a minimal path between some pairs of hosts, as shown in the following example. In Figure 2, a message transmitted from switch 4 to switch 1 cannot go through any minimal path. The shortest path (through switch 6) is not allowed since the message should traverse a link in the "up" direction after one in the "down" direction. All the allowed paths (through switches 0, 2, and through switches 0, 5) are non-minimal and only one of them will be included in the routing table. The number of forbidden minimal paths increases as the network becomes larger.

3. IN-TRANSIT BUFFERS: A MECHANISM TO IMPLEMENT MINIMAL SOURCE ROUTING

The up*/down* routing algorithm is deadlock-free. It avoids cyclic dependencies between network links by not allow-

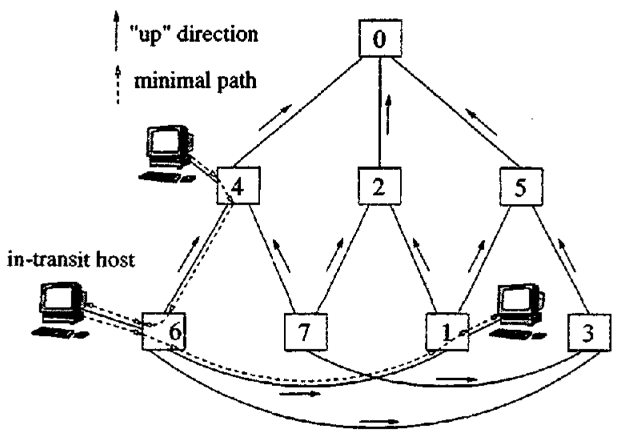

Figure 3: Use of the in-transit buffer mechanism in an irregular network.

ing messages to reserve "up" links after having reserved "down" links. Due to this restriction minimal routes are usually forbidden. The basic idea to eliminate this restriction consists of splitting such forbidden paths into several valid up*/down* paths. On each path, an intermediate host is selected as the destination and, at this host, packets are completely ejected from the network and later re-injected into it. In other words, the dependencies between "down" and "up" links are removed by using some buffers at the intermediate hosts (in-transit buffers). In Figure 3 we can see that, with the in-transit buffer mechanism, a minimal route can be used to route packets from switch 4 to switch 1. To break channel dependencies, packets are sent to a host connected to the intermediate switch 6. This host will re-inject packets as soon as possible.

When the up*/down* routing algorithm for a given packet does not provide a minimal path, the proposed routing strategy selects a minimal path. In this path, one or more in-transit hosts are chosen, verifying that each subroute is minimal and a valid up*/down* path. Therefore, the routing algorithm is deadlock-free. The packet will be addressed to the first in-transit host. The in-transit host will re-inject the packet into the network as soon as possible, forwarding it to the destination host or to the next in-transit host. Although several minimal paths may exist between each pair of nodes, only one of them will be used because performance does not increase significantly when using several minimal paths [4].

In order to route packets requiring in-transit buffers, the packet header format must be changed. In Figure 4 we can see the new header format that supports in-transit buffers. The entire path to destination is built at the source host. A mark (ITB mark) is inserted in order to notify the in-transit host that the packet must be re-injected 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.

However, the in-transit buffer mechanism adds some latency to the message and also uses some additional resources in both network (links) and network interface cards (memory

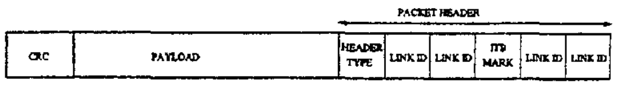

Figure 4: Myrinet packet header for routing schemes based on in-transit buffers.

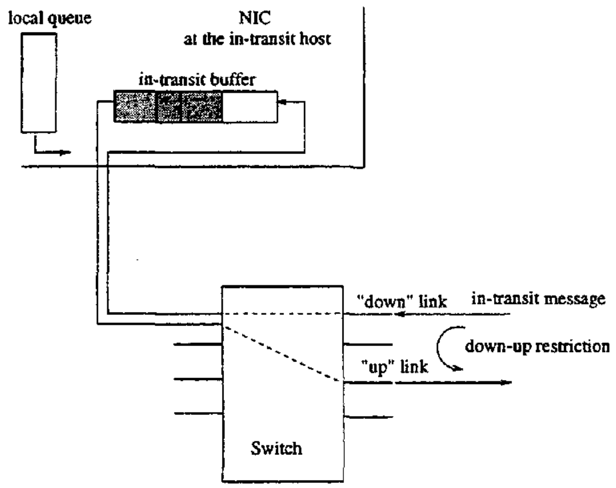

Figure 5: In-Transit buffer mechanism.

pools and DMA engines). On the other hand, with this mechanism, "down" to "up" transitions are allowed. As a consequence, the resulting routing algorithm is less restrictive than the original up*/down* routing algorithm, as it always uses minimal paths among all hosts.

The critical part of this mechanism is the introduced overhead at the intermediate hosts. Figure 5 shows the implementation of the in-transit buffer mechanism. To implement the in-transit buffer mechanism in Myrinet networks, some memory is needed at the network interface card to store in-transit packets and the MCP program has to be modified to detect in-transit packets and process them accordingly. In 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 re-inject 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 in-transit buffer 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 is 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

[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 eliminate this problem.
[2]Myrinet supports mixing links with different bandwidth.

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 the host I/O bus.

To make this mechanism deadlock-free, it must be guaranteed that an in-transit packet that is being re-injected can be completely ejected from the network if the re-injected part of the packet becomes blocked, thus removing potential channel dependencies that may result in a deadlock (down-up transitions). 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 the DMA transfer. If the buffer space at the network interface card exceeds, 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. We rely on dynamic allocation of buffers to simulate infinite buffer capacity.

4. PERFORMANCE EVALUATION

In this section, we evaluate the new mechanism and compare it with the original up*/down* routing algorithm used in Myrinet. First, we describe the different topologies used in the study, and enumerate the different traffic patterns we will use. Also, we describe in this section the parameters used in the simulation concerning links, switches, and network interfaces. These parameters are based on the Myrinet network. Finally, we present the simulation results.

4.1 Network Model

The network is composed of a set of switches and hosts, all of them interconnected by links. Many Myrinet networks are based on regular topologies, especially when they are used to build low-cost supercomputers by connecting many processors together. However, other implementations may need an irregular topology.

Network topologies are completely irregular and have been generated randomly, taking into account three restrictions. First, we assume that there are exactly 4 hosts connected to each switch. Second, all the switches in the network have the same size. We assume that each switch has 8 ports. So, there are 4 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 [12; 13].

In order to evaluate the influence of the network size on system performance, we vary the number of switches in the network. We use network sizes of 8, 16, 32, and 64 switches, so there are 32, 64, 128, and 256 hosts in the system, respectively. To make results independent of the topology, we evaluate up to 40 random topologies, ten for each network size.

4.2 Traffic Patterns

In order to evaluate different workloads, we use different message destination distributions to generate network traffic. The distributions are the following:

- Uniform distribution. The destination of a message is chosen randomly with the same probability for all the hosts. This pattern has been widely used in other evaluation studies [2; 3].
- 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 [6; 7].
- Local distribution. Message destinations are, at most, l switches away from the source host, and are randomly computed. Two values of l have been considered: $l = 3$ and $l = 5$.
- Hot-spot distribution. A percentage of traffic is sent to one host. The selected host is chosen randomly. The same host number will be used for all the topologies. In order to use a representative hot-spot distribution, we have used different percentages depending on the network size. In particular, we have used 30%, 20%, 15%, and 5% for 8, 16, 32, and 64-switch networks, respectively. The rest of the traffic is randomly generated using a uniform distribution.

For each simulation run, we assume that the packet generation rate is constant and the same for all the hosts. Once the network has reached a steady state, the flit generation rate is equal to the flit reception rate. We evaluate the full range of traffic, from low load to saturation.

As Myrinet networks allow any packet size, we also analyze the influence of different packet sizes. We show results using packet sizes of 32, 512, and 1K bytes.

4.3 Myrinet Links

We assume short LAN cables [8] 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 [10]. Hence, a new flit can be injected into the physical channel every 6.25 ns and there will be a maximum of 8 flits on the link at a given time.

We do not use virtual channels since the actual Myrinet switches do not support them. 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 in Myrinet is fixed at 80 bytes.

4.4 Switches

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.

4.5 Interface Cards

Each Myrinet network interface card has a routing table with one or more entries for every possible destination of messages. The way tables are filled determines the routing scheme. We simulate the in-transit buffer mechanism, comparing it with the original Myrinet up*/down* routing. For each source-destination pair, only one route will be computed for both routing algorithms. In particular, we fill tables in two different ways:

- *Up*/down* routing.* Tables are filled with routes that follow the up*/down* rules. These routes have been obtained from the `simple_routes` program that comes with the GM [5] protocol from Myricom. 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 non-minimal 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 using all the minimal up*/down* paths available. We concluded that the routes given by the `simple_routes` program always achieve higher network throughput. By using the routes generated by this program, we simulate the behavior of Myrinet using its original routing algorithm.
- *Minimal routing with in-transit buffers.* Tables are filled with routes that follow minimal paths to destinations (if the path follows the up*/down* rules), or with minimal sub-paths to in-transit hosts that are in a minimal path to the destination (sub-paths follow the up*/down* rules). When several minimal paths exist between a pair of hosts only one is selected (randomly). This strategy is used to simulate the performance of the in-transit buffer mechanism.

In the case of minimal routing with in-transit buffers, the incoming packet must be recognized as in-transit and the transmission DMA must be re-programmed. 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 re-inject the packet[3]. Also, the total capacity of the in-transit buffers has been set to 512KB at each Myrinet interface card.

[3]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.

4.6 Simulation Results

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 with in-transit buffers proposed in this paper. We refer to the original routing as UD, and to the new mechanism as ITB. We group results by traffic pattern and message size. For all the cases, we show the increase in throughput[4] when using in-transit buffers with respect to the original routing algorithm. In particular, we show the minimum, maximum, and average increase for the topologies we have analyzed.

We also show more detailed performance results for the topologies in which the in-transit buffer mechanism improvement is closer to the average improvement obtained. In particular, we will plot the average accepted traffic[5] measured in flits/ns/switch versus the average message latency[6] measured in nanoseconds. For the shake of brevity, these results are shown only for 512-byte messages.

4.6.1 Uniform Distribution

In Table 1 we can see the increase in network throughput when using the in-transit buffer mechanism (ITB) for different network and message sizes when message destinations are uniformly distributed. For small networks (8 switches), the use of ITB sometimes increases throughput (up to 13% increase for 1024-byte messages) but sometimes decreases it (up to 19% reduction for 512-byte messages). On average, ITB behavior is slightly worse than UD behavior for 8-switch networks because, in most of the topologies we have evaluated, many up*/down* routes are minimal and the in-transit buffer 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 [12]. However, for 16-switch networks, ITB always increases network throughput, allowing from 25% to 33% more traffic on average (increase factor of 1.25 and 1.33, respectively).

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 doubling network throughput when using 32-byte messages. In some topologies, ITB more than doubles network throughput (throughput is increased by a factor of up to 2.4 for 32-byte messages). Moreover, ITB always increases throughput for 32-switch networks (the minimum factor of throughput increase obtained is 1.44 when using 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 1024-byte mes-

[4]Throughput is the maximum amount of information delivered by the network per time unit. It is equal to the maximum accepted traffic (see next footnote).

[5]Accepted traffic is the amount of information delivered by the network per time unit. In order to make it independent from the number of switches in the network, it is measured in flits/ns/switch.

[6]Latency is the elapsed time between the injection of a message into the network at the source host until it is delivered at the destination host.

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

Table 1: Factor of throughput increase when using ITB for the uniform distribution.

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

Table 2: Percentage of message latency increase for low traffic when using ITB. Uniform distribution.

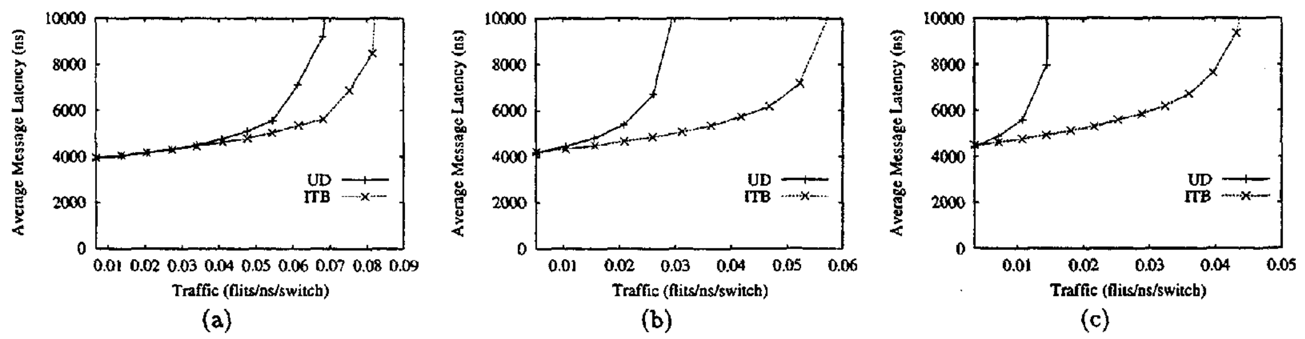

Figure 6: Average message latency vs. traffic. Network size is (a) 16 switches, (b) 32 switches, and (c) 64 switches. Message length is 512 bytes. Uniform distribution.

sages, whereas the maximum factor of improvement is 3.89 for 32-byte messages. The average results show that ITB drastically increases performance over UD routing, increasing throughput by a factor of 3.21 for 32-byte messages. For longer messages (512 and 1024 bytes), the factor of throughput increase ranges from 2.25 to 3.25.

In Figures 6.a, 6.b, and 6.c we show the behavior of the UD and ITB routing algorithms for different network sizes (16, 32, and 64 switches, respectively). The selected topologies are the ones in which the improvement achieved by ITB is closer to the average improvement for the corresponding network sizes. For all simulations, message size is 512 bytes. As can be seen, the ITB mechanism does not increase latency at low loads with respect to UD. Most important, ITB saturates at a much higher load, increasing the improvement over UD as network size increases, as already indicated in Table 1.

Let us analyze latency in more detail. As mentioned above, a drawback of the in-transit buffer mechanism is the additional latency suffered by messages that use in-transit buffers, especially when network traffic is low. Table 2 shows the percentage of message latency increase for low traffic when using the in-transit buffer mechanism with respect to the original up*/down* routing. The table shows minimum, maximum, and average increases for different network sizes and message sizes. For small networks (8 switches), the average increase in message latency is small. On average, it ranges from 0.22% for 1024-byte messages to 2.24% for 32-byte messages. In these small networks, most up*/down* routes are minimal and few in-transit buffers are used on average. So, average message latency does not increase too much (7.86% of increase in the worst case).

Latency is increased significantly only for short messages (32 bytes) in medium and large networks (16 switches or more). Average latency increase ranges from 10.32% to 12.93%. For 512 and 1024-byte messages the maximum latency increase is 2.73%. For long messages, the overhead added by in-transit buffers is a small fraction of the total transmission latency of the message. On the other hand, for short messages, this additional latency is much more important and represents a larger fraction of the total transmission latency of the message. Therefore, short messages suffer some penalty in latency. This effect is more noticeable for medium and large networks.

Finally, for large networks (32 and 64 switches) and long messages (1024 bytes), ITB even reduces message latency on average. This is due to the fact that, for large networks, the additional latency due to using non-minimal paths in UD is higher than the additional latency due to using in-transit buffers through minimal paths.

	32 bytes			512 bytes			1024 bytes		
Sw	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
8	0.90	1.27	1.05	0.86	1.15	0.99	0.89	1.16	1.02
16	1.00	1.96	1.31	0.89	2.00	1.29	0.86	1.76	1.34
32	1.44	2.26	1.89	1.64	2.52	2.00	1.67	2.70	2.00
64	2.31	4.79	3.33	2.31	3.56	2.79	2.18	3.76	2.91

Table 3: Factor of throughput increase when using ITB for the bit-reversal traffic pattern.

	32 bytes			512 bytes			1024 bytes		
Sw	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
8	-0.01	9.87	3.26	-0.10	2.12	0.58	-0.09	1.13	0.28
16	5.76	19.01	10.23	-1.55	3.89	1.32	-4.27	2.78	0.14
32	8.30	17.31	13.28	0.74	3.79	2.64	-1.07	1.54	0.41
64	7.30	14.78	10.18	-2.26	1.07	-0.16	-3.54	-0.73	-1.80

Table 4: Percentage of message latency increase for low traffic when using ITB. Bit-reversal traffic pattern.

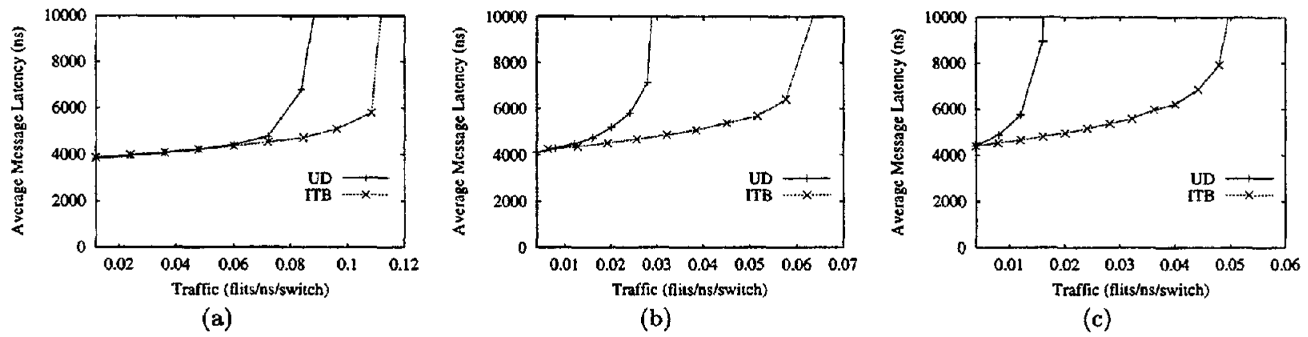

Figure 7: Average message latency vs. traffic. Network size is (a) 16 switches, (b) 32 switches, and (c) 64 switches. Message length is 512 bytes. Bit-reversal traffic pattern.

4.6.2 Bit-reversal Traffic Pattern

For the bit-reversal traffic pattern, Table 3 shows the increase in network throughput provided by ITB. Results for the bit-reversal distribution are qualitatively similar to the ones obtained for the uniform distribution. For small networks (8 switches), ITB performs similarly to UD. For larger networks, the larger the network the higher the benefits obtained. For 32 and 64-switch networks, ITB always improves performance, doubling the throughput achieved by UD in some 32-switch networks, and increasing it even more for 64-switch networks. In particular, for some 64-switch networks, ITB can outperform UD by a factor of up to 4.79.

Figures 7.a, 7.b, and 7.c show the behavior of the routing algorithms for several networks with different sizes (16, 32, and 64 switches, respectively). As network size increases (from left to right), the benefits obtained when using the in-transit buffer mechanism are more noticeable.

Table 4 shows the percentage of message latency increase for low traffic when using in-transit buffers. As for the uniform distribution, latency increase is noticeable only for 32-byte messages, reaching an increase of 19% in the worst case. As network size increases and longer messages are transmitted, this penalty decreases.

4.6.3 Local Distribution of Message Destinations

Table 5 shows the increase in network throughput provided by ITB when messages are sent, at most, 3 switches away from their source. As can be seen, ITB does not improve over UD. UD offers more minimal paths on average for this distribution. As a consequence, when using ITB, the average number of in-transit buffers used per message is very low (0.008 for a 64-switch network), resulting in few minimal paths added to UD routing. Therefore, even for large networks (64 switches), ITB does not help to increase throughput.

Table 6 shows the increase in network throughput when messages are sent farther. In this case, all messages sent are, at most, 5 switches away from their destinations. Notice that results are not shown for 8-switch networks because all destinations are, at most, 4 switches away from sources. As can be seen, ITB obtains higher throughput than UD. In particular, for 16-switch networks, throughput is increased for all the topologies (except for one out of ten topologies, where ITB performs the same as UD). For 32 and 64-switch networks, ITB always increases throughput. The increase ranges, on average, from 47% to 65% (increase factors of 1.47 and 1.65, respectively) in 32-switch networks and from 43% to 67% (increase factors of 1.43 and 1.67, respectively) in 64-switch networks. In this message destination distribution, messages use on average 0.381 in-transit buffers (for the same 64-switch network mentioned above).

	32 bytes			512 bytes			1024 bytes		
Sw	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
8	0.90	1.00	0.97	0.84	1.00	0.93	0.80	1.00	0.90
16	0.92	1.08	0.98	0.91	1.09	0.99	0.89	1.01	0.96
32	0.93	1.08	1.00	0.91	1.08	1.01	0.93	1.07	1.00
64	0.92	1.09	1.03	0.89	1.01	0.99	0.90	1.12	0.99

Table 5: Factor of throughput increase when using ITB for the local distribution ($l = 3$).

	32 bytes			512 bytes			1024 bytes		
Sw	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
16	1.00	1.67	1.31	1.05	1.55	1.32	1.00	1.55	1.27
32	1.20	2.00	1.65	1.40	1.63	1.51	1.27	1.81	1.47
64	1.50	2.00	1.67	1.32	2.03	1.60	1.32	1.60	1.43

Table 6: Factor of throughput increase when using ITB for the local distribution ($l = 5$).

	32 bytes			512 bytes			1024 bytes		
Sw	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
16	7.05	15.16	10.07	0.11	2.08	1.25	-0.91	0.97	0.17
32	15.71	20.45	17.56	2.06	3.65	2.85	0.53	1.91	1.18
64	18.83	22.05	20.25	1.31	3.39	2.22	-0.80	1.22	0.20

Table 7: Percentage of message latency increase for low traffic when using ITB. Local distribution ($l = 5$).

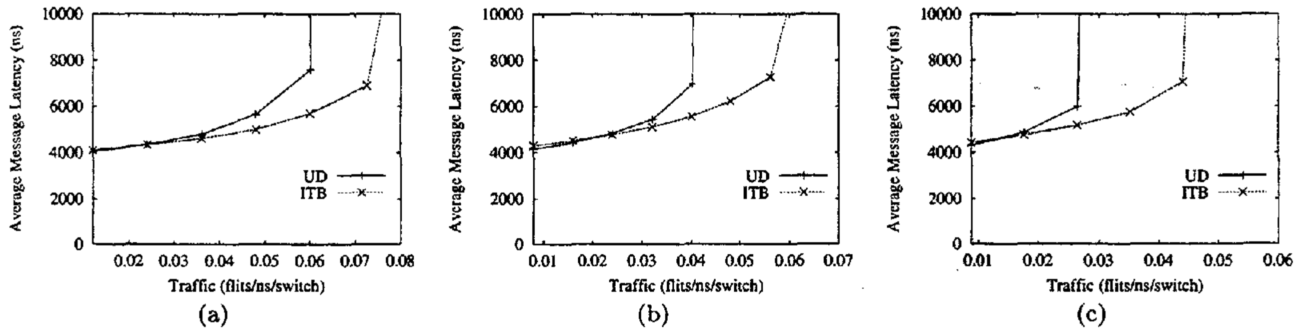

Figure 8: Average message latency vs. traffic. Network size is (a) 16 switches, (b) 32 switches, and (c) 64 switches. Message length is 512 bytes. Local distribution ($l = 5$).

In Figures 8.a, 8.b, and 8.c we can see some performance results for selected topologies. Table 7 shows the percentage of message latency increase. Latency increase is kept low, except for 32-byte messages (at most 22% of increase).

For a local distribution of message destinations, ITB performance depends on the length of paths to destinations. The longer the paths, the higher the probability of requiring in-transit buffers to supply minimal paths and therefore the larger the performance benefits achieved by ITB.

4.6.4 Hot-spot Distribution

When there is a hot-spot in the network, the ITB routing algorithm obtains the throughput increases shown in Table 8. 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 in-transit buffers also increase. In particular, 16-switch networks improve throughput by up to 38% (increase factor of 1.38) when using in-transit buffers (for 32-byte messages). For larger networks, the factor of improvement achieved by in-transit buffers ranges on average from 1.30 for 32-switch networks to 3.21 for 64-switch networks. In some particular networks, the improvement factor reaches up to 4.70.

In Figures 9.a, 9.b, and 9.c we can see the performance of both routing algorithms for different selected topologies. Table 9 shows the percentage of message latency increase for low traffic when using in-transit buffers. The relative behavior of ITB is similar to the one shown for other distributions.

As can be seen, for the hot-spot distribution, the in-transit buffer mechanism increases network throughput. Benefits are smaller than the ones obtained for the uniform and bit-reversal distribution of message destinations, but are still significant.

To conclude, the proposed in-transit buffer mechanism allows the use of deadlock-free minimal routing. This new

	32 bytes			512 bytes			1024 bytes		
Sw	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
8	0.93	1.04	0.97	0.91	1.10	1.01	0.91	1.10	1.02
16	1.00	1.38	1.17	1.00	1.20	1.09	1.00	1.17	1.10
32	1.12	1.81	1.55	1.10	1.62	1.30	1.04	1.63	1.31
64	1.98	4.70	3.21	1.58	3.00	2.21	1.67	2.66	2.11

Table 8: Factor of throughput increase when using ITB for the hot-spot distribution.

	32 bytes			512 bytes			1024 bytes		
Sw	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
8	-0.15	10.64	4.11	0.00	2.04	0.74	0.02	1.07	0.45
16	5.87	17.26	10.68	0.90	2.80	1.81	0.07	1.48	0.91
32	11.52	17.66	14.44	0.17	2.82	1.21	-1.27	0.89	-0.30
64	3.61	12.05	6.91	-6.00	-0.64	-2.89	-8.34	-2.45	-4.84

Table 9: Percentage of message latency increase for low traffic when using ITB. Hot-spot distribution.

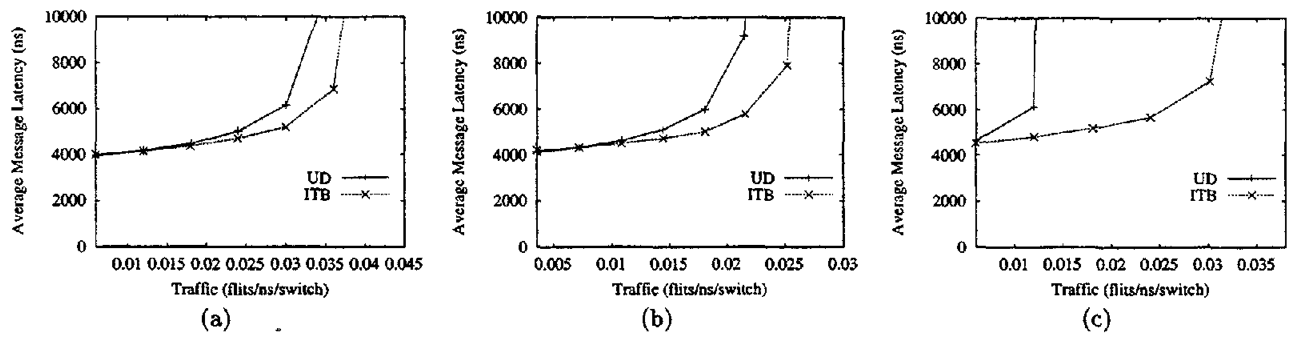

Figure 9: Average message latency vs. traffic. Network size is (a) 16 switches, (b) 32 switches, and (c) 64 switches. Message length is 512 bytes. Hot-spot distribution.

technique is useful for medium and large networks (more than 16 switches). For 32 and 64-switch networks we analyzed, throughput increases by a factor ranging from 1.3 to 3.33. On the other hand, the proposed technique increases message latency when network traffic is not intense, especially with short messages (32 bytes). However, the increase in latency does not exceed 22% in the worst case, which is a small price to pay when compared to the large improvement obtained in network throughput.

Finally, it is important to point out that the 512KB reserved for buffers in the network interface card has been enough in all cases to store all the in-transit messages without using the host memory. Current Myrinet interface cards are shipped with 4MB and less than 128KB are set aside for the MCP.

5. CONCLUSIONS

In this paper, we have proposed a new mechanism (in-transit buffers) to improve network performance in networks with source routing. With this mechanism, messages always are forwarded using minimal paths. To avoid deadlocks, when necessary, paths are divided into several deadlock-free subroutes, and a special kind of virtual cut-through is performed at the network interface card of some intermediate hosts.

This mechanism is valid for any network with source routing and it has been evaluated by simulation for the Myrinet network. It can be 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 in-transit buffers. We have used different random topologies (up to 40 topologies) and different traffic patterns (uniform, bit-reversal, local, and hot-spot) with different message sizes (32, 512, and 1024-byte messages) and different network sizes (8, 16, 32, and 64 switches).

Results show that for small networks (8 switches) and for local traffic patterns, the in-transit buffer mechanism does not improve network performance, because few minimal paths are added by the in-transit buffer mechanism. But, as network size and average distance increase, the benefits obtained by the in-transit buffer mechanism also increase. In particular, we have obtained 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 networks. In some particular networks, throughput is increased by a factor of 4.7.

Although the proposed technique may increase message latency, this effect is only noticeable for low network loads and short messages (32 bytes), and it does not exceed 22%. This increase in latency is a small price to pay when compared to the large improvement in network throughput.

As for future work, we plan to implement the proposed mechanism on an actual Myrinet network in order to confirm the simulation results obtained. Also, we are working on reducing the latency overhead and on new route selection algorithms to increase adaptivity.

Acknowledgements
The authors thank Drs. R. Felderman and T.M. Pinkston for their helpful and valuable suggestions to improve the presentation and technical contents of this paper.

6. 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, February 1995.

[2] R. V. Bopana and S. Chalasani, "A comparison of adaptive wormhole routing algorithms," *Proc. 20th Annu. Int. Symp. Computer Architecture*, May 1993.

[3] W. J. Dally, "Virtual-channel flow control," *IEEE Trans. Parallel Distributed Systems*, vol. 3, no. 2, pp. 194-205, Mar. 1992.

[4] J. Flich, M.P. Malumbres, P.Lopez and J. Duato, "Improving Routing Performance in Myrinet Networks," in *Int. Parallel and Distributed Processing Symp. (IPDPS 2000)*, May 2000.

[5] GM protocol, 'http://www.myri.com/GM'

[6] J. Kim and A. Chien, "An evaluation of the planar/adaptive routing," in *Proc. 4th IEEE Int. Symp. Parallel Distributed Processing*, 1992

[7] P. R. Miller, "Efficient communications for fine-grain distributed computers," Ph.D. Thesis, Southampton University, 1991.

[8] Myrinet, 'M2-CB-35 LAN cables, http://www.myri.com/myrinet/product_list.html'

[9] 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, April 1990.

[10] S. L. Scott and J. R. Goodman, "The Impact of Pipelined Channels on k-ary n-Cube Networks," in *IEEE Trans. on Parallel and Distributed Systems*, vol. 5, no. 1, pp. 2–16, January 1994.

[11] R. Sheifert, "Gigabit Ethernet," in *Addison-Wesley*, ISBN: 0-201-18553-9, April 1998.

[12] F. Silla and J. Duato, "Improving the Efficiency of Adaptive Routing in Networks with Irregular Topology," in *1997 Int. Conf. on High Performance Computing*, December 1997.

[13] F. Silla, M. P. Malumbres, A. Robles, P. López and J. Duato, "Efficient Adaptive Routing in Networks of Workstations with Irregular Topology," in *Workshop on Communications and Architectural Support for Network-based Parallel Computing*, February 1997.