Improving Routing Performance in Myrinet Networks [*]

J. Flich, M. P. Malumbres, P. López, 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 [1], packets are delivered using source routing. Due to the irregular topology, the routing scheme is often non-minimal. In this paper we analyze the routing scheme used in Myrinet networks in order to improve its performance. We propose new routing algorithms that balance the utilization of the available routes and always use minimal paths.

We show through simulation that the current routing schemes used in Myrinet networks can be improved by modifying only the routing software without increasing the software overhead significantly. The overall throughput can be doubled without modifying the network hardware.

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

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.

Among the current gigabit LAN technologies, Myrinet [1] has one of the highest performance/cost ratio. One of the reasons is that the research effort made on parallel computers has been applied to the design of high-speed local area networks. We will focus on this network because its 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 message. As the Myrinet routing scheme uses source routing, the network adapter has to build network routes to each destination during the initialization phase. The network adapter 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 [5] 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 the up*/down* routing are non-minimal on certain networks.

2 Motivation

In previous works [8, 9], we analyzed the behavior of distributed routing algorithms in irregular topologies, showing that adaptive routing schemes outperform up*/down* routing schemes by improving the routing flexibility and providing minimal paths. Therefore, it would be interesting to analyze the feasibility of implementing minimal routing in commercial networks and evaluate its behavior. In the case of Myrinet, this would be possible thanks to the flexibility of the MCP program. If we could change the MCP program in order to improve the network behavior without

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

significantly increasing the software overhead, performance may considerably improve.

In this paper, we take on such a challenge. We propose a method to implement minimal source routing. Also, we analyze different routing policies and compare them with the current scheme, in order to increase network performance.

In particular, we propose a new approach to increase overall network throughput using minimal routing. The idea is based on dividing forbidden minimal routes into valid sub-routes and forcing a special kind of virtual cut-through in the network interface cards at the in-transit hosts. However, splitting routes requires the collaboration of network hosts to forward packets from one sub-route to the next. This overhead must be taken into account in order to make a fair comparison.

Our main goal is to improve network performance without modifying the network design, by reusing existing hardware and changing only the MCP program in network adapters.

The rest of the paper is organized as follows. In Section 3, the current Myrinet routing scheme is introduced, and several proposals are described in order to improve routing flexibility. In Section 4, the performance of the proposed techniques are evaluated by simulation. Finally, in Section 5, some conclusions are drawn.

3 Improving the Routing Flexibility in Myrinet Networks

Myrinet uses up*/down* routing [5] 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 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. 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 in the “down” direction.

There may exist different valid up*/down* paths between the same source-destination pair. However, Myrinet routing software only uses one of the shortest paths. This path will be used to send packets to the corresponding destination until a network topology change is detected. In this case, the routing tables will be updated according to the new topology. On the other hand, to guarantee deadlock freedom, up*/down* routing is not always able to provide a minimal path between every pair of hosts. As network size increases, this effect becomes more important.

Two ideas can be exploited to improve routing flexibility. First, more than a single choice can be stored in the routing table for each source-destination pair, and several alternatives can be used to select the route that will be used. Second, other routing schemes together with a deadlock handling mechanism can be considered. Among all the feasible choices, we will analyze two possible ways to increase the performance of Myrinet networks:

(a) Using up*/down* routing with a more efficient route selection algorithm. Several selection policies that avoid using the same route between each source-destination pair have been considered, trying to balance the use of the available routes. Up*/down* routing can supply several valid routes between two network hosts and, in some cases, there exist more than one shortest up*/down* route. Therefore, we will store in the table a set of routes for each source-destination pair, including the best ones, and selecting one of them according to some criteria. We have considered the following path selection algorithms:

- *OSUD (One Shortest Up*/Down* path)*: Always the same shortest up*/down* path. This is the current routing policy used in Myrinet.
- *RSUD (Random Shortest Up*/Down* path)*: Randomly selection among all the shortest up*/down* paths.
- *RRSUD (Round-Robin Shortest Up*/Down* path)*: Round-robin selection among all the shortest up*/down* paths.
- *PUD (Probabilistic Up*/Down* path)*: 80% of selected paths using RSUD, and 20% randomly chosen among up*/down* paths that are one hop longer than the shortest up*/down* path.

(b) Using a special kind of virtual cut-through switching at intermediate hosts. In this case, when a message is generated at a source host, the routing selection procedure will try to find a minimal route to the destination. If it is a valid up*/down* minimal route, then it is used without modification. Otherwise, it would try a forbidden but minimal route. To avoid deadlock, the routing algorithm will split this route into several valid up*/down* sub-routes. The dependencies between the sub-routes (down-to-up transitions) will be broken by absorbing the message at the intermediate host and later re-injecting it into the network. Thus, in this case, the routing algorithm will try to find an intermediate network host that meets two conditions: It is along a minimal path from source to destination and the sub-paths from source to the intermediate host and from the intermediate host to the destination are valid up*/down* routes. Then, the source host sends the message to the intermediate host, and this one will forward it to its final destination. Note that there may exist more than one intermediate host along the path from source to destination. In this case, all the subpaths between intermediate hosts must be valid up*/down* routes. Also, note that this routing strategy requires that at least one host is connected to every switch where there may exist down-to-up transitions.

The critical part of this proposal is the overhead introduced at the intermediate hosts. Some memory to buffer in-transit packets is needed at the network adapter of the in-transit host and the MCP program has to be modified to detect in-transit packets and process them accordingly. In order to minimize the introduced overhead, a DMA transfer to re-inject the in-transit packet can be programmed as soon as its header is processed and the output channel is free. So, the delay to forward this packet will be the time required for processing the header and initiating the DMA. As the MCP allows this kind of DMA programming, it is possible to implement it 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, assuming that all the links in the network have the same bandwidth. 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 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. 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. Otherwise, the MCP should store the packet in the host memory, considerably increasing the overhead. In this case, a fixed amount of buffer space will be allocated in the network interface card for in-transit packets. The host memory will be used in case of buffer overflow. 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.

On the other hand, several minimal routes (with or without in-transit buffers) can be considered for a given source-destination pair. The possible path selection choices considered lead to these new routing algorithms:

- *OMIT (One Minimal In-Transit path)*: Always uses the same minimal path from source to destination.

- *RMIT (Random Minimal In-Transit path)*: Random choice among all possible minimal paths.

- *RRMIT (Round-Robin Minimal In-Transit path)*: Round-robin selection among all minimal paths.

- *PIT (Probabilistic In-Transit path)*: 80% of selected paths using RMIT, and 20% randomly selected among paths that are one hop longer than minimal paths.

- *RRMIT-MIN (Round-Robin Minimal In-Transit path MINimizing in-transit hosts)*: Round robin selection among minimal paths that minimize the number of in-transit hosts. Hence, in-transit hosts will be used only when the up*/down* routing algorithm does not provide a minimal route.

4 Performance Evaluation

4.1 Network Model

The network is composed of a set of switches. Network topology is completely irregular and has been randomly generated taking into account three restrictions. First, we assumed that there are exactly 4 hosts attached to each switch. Second, all the switches in the network have the same size. We assumed 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 studies [8, 9].

In order to evaluate the influence of the network size on system performance, we varied the number of switches in the network keeping the number of hosts connected to each switch constant and equal to 4. We have used network sizes of 16, 32, and 64 switches.

For each simulation run, we assume that the message 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 have evaluated the full range of traffic, from low load to saturation. Message destinations are randomly chosen among all the hosts.

4.2 Myrinet Links, Switches, and Interfaces

We assume short LAN cables [3] to interconnect switches and hosts. 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 [6]. 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.

Each Myrinet switch has a simple routing control unit that removes the first flit of the header and uses it to select the output channel (when the output channel becomes 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 only process one message header at a time. An output port is assigned to waiting packets in a demand-slotted round-robin fashion. A crossbar inside the switch allows multiple packets to traverse it simultaneously without interference.

We do not use virtual channels since current 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.

Each host has a routing table with one or more entries for every possible destination of packets. The way tables are filled determines the routing scheme that will be used. We will analyze the two schemes and the path selection algorithms proposed in Section 3. Although tables can be filled with all the possible routes to every possible destination, to avoid using a huge table that may result in a high look-up delay, we imposed a limit of 10 alternative routes for each source-destination pair.

In the case of minimal routing with in-transit buffers the network DMA must be re-programmed. To do so, some special registers must be loaded up. We have used a delay of 275 ns (44 bytes received) to detect an in-transit message, and 200 ns (32 bytes more) to program the DMA to re-inject the message[1]. Also, the total capacity of the in-transit buffers has been limited to 90KB at each Myrinet interface card.

4.3 Simulation Results

In this section, we show the results obtained from the simulation of the Myrinet network for the different routing algorithms analyzed. First we will analyze the impact of route selection using up*/down* routes (OSUD, RSUD, RRSUD, and PUD). Later, we will analyze the benefits of using minimal routing by means of the in-transit buffer mechanism (OMIT, RMIT, RRMIT, PIT, and RRMIT-MIN). We have used different message sizes, but for the shake of brevity, we only show results for 512-byte packets.

[1]These timings have been obtained from a real Myrinet network.

4.3.1 Up*/Down* Routing with Alternative Routes

We first evaluate the behavior of the up*/down* routing algorithm using the different path selection choices presented in Section 3 (OSUD, RSUD, RRSUD, and PUD). We will show results for networks sizes of 16, 32, and 64 switches.

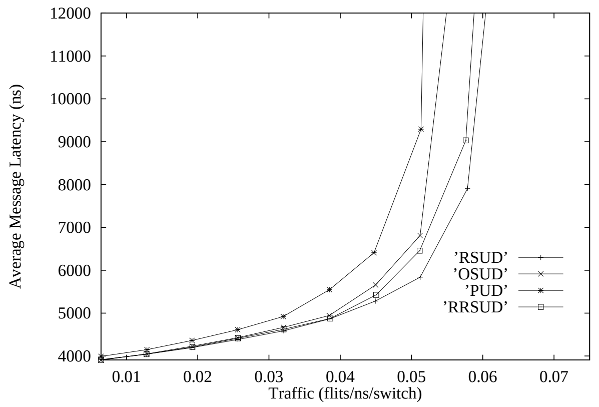

Figure 1. Average message latency vs. traffic. Network size is 16 switches.

Figure 1 shows the results for 16-switch networks for a message size of 512 bytes. With low traffic rate, the different path selection algorithms have almost the same performance, with the exception of the PUD algorithm, which achieves higher latency due to the use of longer paths. As traffic grows, we observe that RRSUD and RSUD algorithms provide better performance than OSUD because they balance traffic among the different available routes. However, the OSUD algorithm forces all the traffic from a source to a given destination to follow always the same path.

Now, we focus on the impact of network size on the different routing algorithms. Figures 2 and 3 show results for networks with 32 and 64 switches, respectively.

The conclusions are roughly the same as for 16-switch networks. PUD path selection algorithm performs poorly, and the others (OSUD, RSUD, RRSUD) perform similarly except for high traffic loads. However, RSUD and RRSUD algorithms do not improve performance over OSUD. This is because these algorithms are highly dependent on the network topology and the number and length of available shortest up*/down* routes. As network size increases, shortest up*/down* routes are longer on average, and different valid routes will share more links. Thus, these routes tend to perform as a single route, leading the RSUD and RRSUD to behave in the same way as OSUD.

We conclude that RRSUD and RSUD generally reduce network contention by using different routes to reach a given destination. However, these algorithms merely improve the current Myrinet routing algorithm (OSUD) by a small

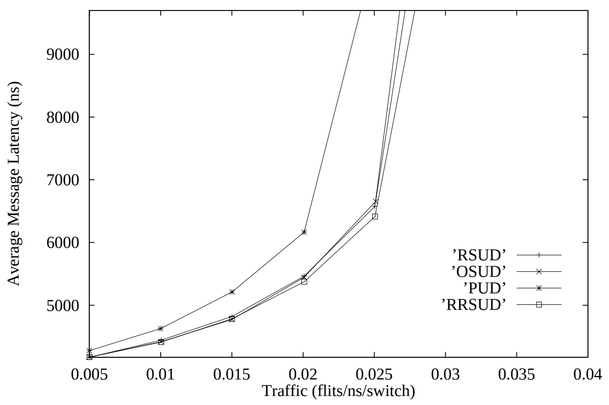

Figure 2. Average message latency vs. traffic. Network size is 32 switches.

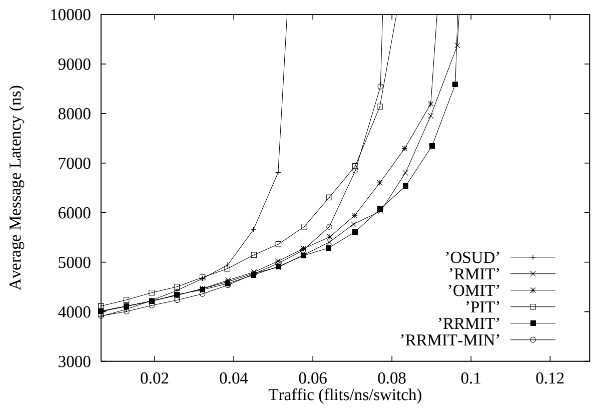

Figure 4. Average message latency vs. traffic. Network size is 16 switches.

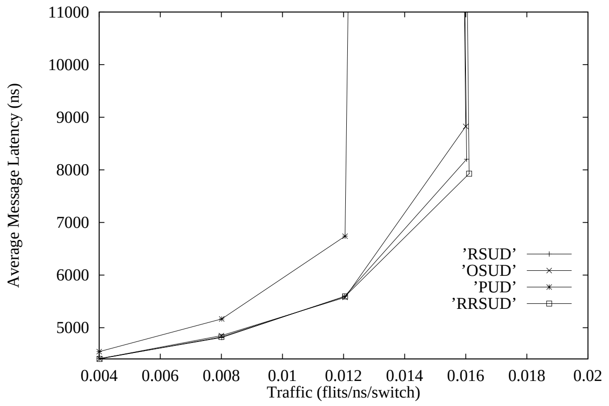

Figure 3. Average message latency vs. traffic. Network size is 64 switches.

amount.

4.3.2 Minimal Routing with In-Transit Buffers

We focus now on the performance of minimal routing using in-transit buffers. We compare the OMIT, RMIT, RRMIT, PIT, and RRMIT-MIN path selection algorithms with the basic up*/down* routing algorithm (OSUD). We vary the network size from 16 switches to 64 switches.

Figure 4 shows the results for a 16-switch network using 512-byte packets. As we can see, all the path selection algorithms that use minimal paths (OMIT, RMIT, RRMIT, PIT, and RRMIT-MIN) significantly outperform the OSUD routing algorithm that uses only up*/down* paths. In particular, RMIT and RRMIT algorithms almost double the throughput achieved by OSUD algorithm (e.g., 0.051 flits/ns/switch versus 0.096 flits/ns/switch).

However, all these minimal algorithms (except RRMIT-MIN), exhibit a slightly higher average latency for low traffic than the original OSUD algorithm (4011 ns with RMIT versus 3906 ns with OSUD for 0.006 flits/ns/switch). This is due to the use of in-transit buffers. Crossing one in-transit host adds 625 ns (475 ns to detect the message and program the re-injection and 150 ns to cross the switch again) to the latency of the message. The RRMIT-MIN algorithm avoids this problem by minimizing the use of in-transit hosts. RRMIT-MIN algorithm achieves near the same average latency for low traffic (3915 ns) as OSUD and offers better throughput (reaching to 0.077 flits/ns/switch, that is, 1.51 times better than OSUD). However, this algorithm offers less throughput than the others algorithms do. As message and network size increase, the latency added by in-transit hosts becomes less significant. Also, new Myrinet interface implementations will lead to a reduction in the detection time of in-transit packets and DMA reprogramming times.

Let us consider the influence of path selection strategies. Random (RMIT) and round-robin (RRMIT) are the best choices due to their ability of choosing any of the available minimal paths versus the use of only one path in OMIT. As was expected, the use of non-minimal paths (PIT) leads to the worst performance. PIT path selection algorithm saturates at 0.08 flits/ns/cycle and shows higher latency.

As network size increases, the up*/down* routing algorithm does not scale well [8]. Figures 5 and 6 show the results for networks of 32 and 64 switches, respectively. In a 32-switch network, the new routing scheme doubles the performance achieved by the basic up*/down* routing (OSUD) except for RRMIT-MIN that is 1.8 times better. In particular, RMIT achieves more than twice the throughput achieved by OSUD. In a 64-switch network, the enhancement achieved by the new approach is even higher. In this case, the best minimal routing scheme achieves three times better throughput than the basic up*/down* scheme.

Finally, it is important to point out that the 90KB re-

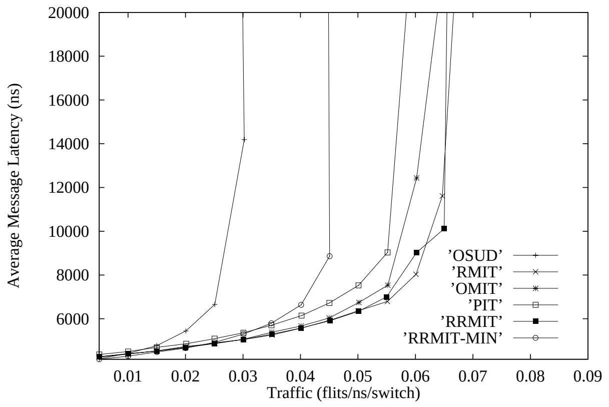

Figure 5. Average message latency vs. traffic. Network size is 32 switches.

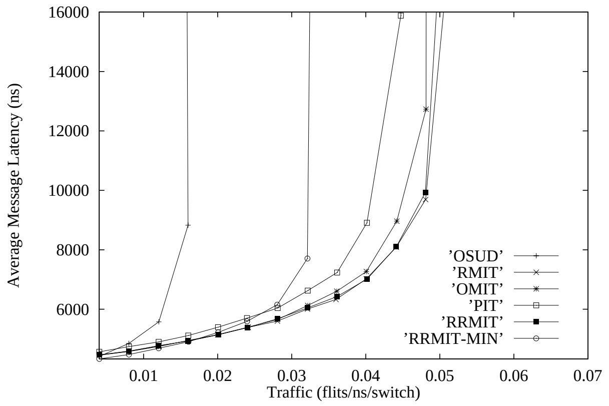

Figure 6. Average message latency vs. traffic. Network size is 64 switches.

served for buffers at the Myrinet interface card has been enough in all simulations to store all the in-transit packets without using the host memory. Current Myrinet interface cards are shipped with 4MB and less than 128KB are set aside for the MCP.

To conclude, the proposed software implementation of in-transit buffers in Myrinet interface cards allows the use of deadlock-free minimal routing, drastically increasing the overall network throughput, doubling it for small network sizes (16 switches), and more than tripling it for large network sizes (64 switches).

5 Conclusions

In this paper, we proposed several routing strategies to improve Myrinet performance. First, keeping the original up*/down* routing scheme, we proposed different path selection algorithms in order to improve network performance. These mechanisms are simple enough to avoid adding overhead in the MCP (Myrinet Control Program). The results show that performance is improved by a small amount for small networks. Unfortunately, for large networks, there are even diminishing returns.

Second, a deadlock avoidance-based minimal routing scheme that forwards packets by splitting routes into several deadlock-free sub-routes as well as several selection policies are proposed. This mechanism can be easily implemented in current Myrinet networks. Results show that throughput can be doubled for small networks and tripled for large networks (with respect to the original Myrinet routing algorithm).

As for future work, we plan to implement the proposed mechanism on an actual Myrinet network in order to confirm these results. Also, we are working on new path selection algorithms that increase adaptivity by reducing the resource sharing among alternative routes.

6 Acknowledgements

The authors thank Dr. R. Felderman for his helpful and valuable suggestions to improve the presentation and technical contents of this paper.

References

[1] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic and W. Su, "Myrinet - A gigabit per second local area network," *IEEE Micro*, pp. 29–36, Feb. 1995.

[2] R. Horst, "ServerNet deadlock avoidance and fractahedral topologies," in *Proc. of the Int. Par. Proc. Symp.*, Apr. 1996.

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

[4] T. Pinkston and S. Warnakulasuriya, "On deadlocks in interconnection networks," in *The 24th Int. Symp. on Computer Architecture*, Jun. 1997.

[5] M. 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.

[6] S. Scott and J. 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, Jan. 1994.

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

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

[9] F. Silla, M. Malumbres, A. Robles, P. López and J. Duato, "Efficient Adaptive Routing in Networks of Workstations with Irregular Topology," in *Workshop on Comms. and Arch. Support for Net-based Parallel Computing*, Feb. 1997.