Using Minimal Routing 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

R. Felderman
Myricom, Inc.
and Computer Science Dept.
University of Southern California
E-mail: feldy@myri.com

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. Research in NOWs is advancing relatively fast because the research effort made on parallel computers is now being transferred to this environment. As a result, recent network products like Myrinet [1] and ServerNet [2] use this technology to compete with other high-speed local area network products.**

In some of these networks, messages 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 a new routing mechanism to significantly increase the overall throughput using 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. Instead, memory resources are used in the network interface cards, allowing network nodes to switch some packets by using a special kind of multi-hop routing.

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

I. Introduction

Due to the increasing computation 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 dedicated medium networks. As an example, consider the evolution of the Ethernet family up to recent Gigabit Ethernet networks [7]. 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.

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 very simple and 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 messages, transfers messages 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. Network adapters have mechanisms to discover the current network configuration, being able to build routes between itself and the rest of network nodes. 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.

II. Motivation

In previous works [8], [9], we analyzed the behavior of distributed routing algorithms in irregular topologies, showing that adaptive routing schemes

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

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 significantly increasing the software overhead, performance would considerably improve.

In this paper, we take on such a challenge. We propose a new approach to increase overall network throughput using minimal routing[1]. To achieve minimal routing we will split a non-minimal route into sub-routes making a special kind of virtual cut-through in the network interface cards of the in-transit nodes. However, splitting routes requires the collaboration of network nodes to forward messages from one sub-route to the next one. 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 III, the current Myrinet routing scheme is introduced, and several proposals are described in order to improve routing flexibility. In Section IV, the performance of the proposed techniques are evaluated by simulation. Finally, in Section V some conclusions are drawn.

III. Improving the Routing Flexibility in Myrinet Networks

A. Myrinet Routing

Myrinet uses source routing to transmit messages between nodes. In this routing technique, the message header stores the route that the message has to follow to reach its destination. To simplify switch operation, each message header consist of an ordered list of output link identifiers that are used by each intermediate switch to properly route the message. 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 node must have a representation of the current network topology, in order to build and maintain the routes between itself and each potential destination node. Myrinet uses up*/down* routing [5] to build network routes. Routes are built before sending any message in the network adapter of each node during its initialization phase. In addition, each network adapter checks for changes in the network topology (shutdown of network nodes, link/switch failures, start-up of new network nodes, etc.), in order to maintain its own routing table.

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 messages 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, up*/down* routing is not always able to provide a minimal path between every pair of nodes due to the restriction imposed by the up*/down* rule. As network size increases, this effect becomes more important.

Finally, timeout mechanisms are implemented in the switches in order to recover from possible deadlock situations due to transmission errors that result in header modifications or changes in topology that have not been detected yet.

B. Improving Routing Flexibility

The basic idea consist of storing more than one choice in the routing table for each destination. This will allow to use simple routing selection algorithms that improves the current scheme used in Myrinet.

Also we will try to find minimal routes for all destinations. If up*/down* routing does not provide a minimal route, we will break it in two or more sub-routes, using a special kind of virtual cut-through switching at intermediate nodes. We call this mechanism *Minimal Routing with In-Transit Buffers*. When a message is generated at a source node, the routing selection procedure will try to find a minimal route to the destination. If it is an up*/down* minimal route, then it is selected. Otherwise, a deadlock may appear. To avoid deadlock, the routing algorithm will split this route into several valid up*/down* minimal routes[2]. The dependencies between the routes will be broken by absorbing the message at the intermediate nodes and later re-injecting it into the network. Thus, in this case, the routing algorithm will try to find an intermediate network node that meets three conditions: It is in the path from source to destination, the path is minimal, and the path from source to the intermediate node is a valid up*/down* route.

[1] Note that many paths provided by up*/down* routing are not minimal on certain networks.

[2] Note that this is possible since every single link is a valid up*/down* path.

Then, the source node sends the message to the intermediate node, and this one will forward it to its final destination[3]. Note that there may exist more than one intermediate node along the path from source to destination. 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 nodes. Some memory to buffer in-transit messages is needed and the MCP program has to be modified to detect in-transit messages and process them accordingly. In order to minimize the introduced overhead, a DMA transfer to re-inject the in-transit message can be programmed as soon as its header is processed and the output channel is free. So, the delay to forward this message 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 message has been completely received, because it will arrive at the same rate that it is transmitted[4], assuming that all the links in the network have the same bandwidth[5]. Note that Myrinet does not implement virtual channels. Therefore, once a message header reaches the network interface card, flits will continue arriving at a constant rate. The only additional requirement is that the message is completely stored in the network adapter memory at the source node 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 message that is being re-injected can be completely ejected from the network if the re-injected part of the message becomes blocked, thus removing potential channel dependencies that may result in a deadlock. So, when an in-transit message arrives at a given node, 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 message in the host memory, considerably increasing the overhead. A fixed amount of buffer space will be allocated in the network interface card for in-transit messages. The host memory will be used in case of buffer overflow.

With this scheme we will test different routing selection algorithms:

- *OMIT (One Minimal In-Transit path)*: Always considers only one 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 Nodes)*: In the previous algorithms the route lengths are obtained by counting the number of switches crossed to reach the destination. In-transit buffering adds latency to messages and forces them to cross a switch twice. Thus, in this algorithm minimal paths are selected taking into account the total number of times switches are crossed. In other words, when an in-transit node is visited, the adjacent new switch is counted twice. This algorithm restricts the use of in-transit nodes.

IV. Performance Evaluation

A. Network Model

The network is composed of a set of switches. Network topology is completely irregular and has been generated randomly[6], taking into account three restrictions. First, we assumed that there are exactly 4 workstations connected 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 workstations connected to each switch constant

[3] The entire route is fixed at the source node. The in-transit node only re-injects the message.

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

[5] Myrinet supports mixing links with different bandwidth.

[6] Most Myrinet networks are based on regular topologies, specially when they are used to build low-cost supercomputers by connecting many processors together. However, other implementations may need an irregular topology. For example, a department may have several computers spreading over several offices and/or floors. In this case it seems that several distributed switches is the best solution. In other words, the network topology is adapted to the building constraints.

and equal to 4. We have used network sizes of 16, 32, and 64 switches, so there are 64, 128, and 256 workstations in the system, respectively.

For each simulation run, we assume that the message generation rate is constant and the same for all the nodes. 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. The message destination is randomly chosen among all the nodes.

As Myrinet network allows any message size, we will also analyze the influence of different message sizes. We will show results using message sizes of 32, 128, 512, and 1K bytes.

B. Myrinet Links

We assume short LAN cables [3] to interconnect switches and workstations. These cables are 10 meters long, offer a bandwidth of 160 MB/s, and have a delay of 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 in a given time.

C. Myrinet 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 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 messages in a demand-slotted round-robin fashion. When a message gets the routing control unit, but it cannot be routed because the output channel is busy, it must wait in the input buffer until its next turn. A crossbar inside the switch allows multiple messages to traverse it simultaneously without interference.

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.

D. Myrinet Interfaces

Each workstation has a routing table with one or more entries for every possible destination. The way tables are filled determines the routing scheme that will be used. We will analyze the scheme proposed in previous Section. Tables are filled with routes that follow minimal paths to destinations (if the path follows the up*/down* rules), and/or with minimal paths to in-transit nodes that are in a minimal path to the destination (following the up*/down* rules). This strategy is used to analyze the performance of OMIT, RMIT, RRMIT, PIT, and RRMIT-MIN routing selection algorithms.

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.

When minimal routing with in-transit buffers is used, the network DMA must be re-programmed for each in-transit message. To do so, some special registers must be loaded up. We have assumed a delay of 275 ns [7](44 bytes received) to detect an in-transit message, and 200 ns (32 bytes received more) to program the DMA to re-inject the message. Also, the total capacity of the in-transit buffers has been set to 90KB at each Myrinet interface card.

E. Simulation Results

In this section we show the results obtained from the simulation of the Myrinet network. We will compare the new selection algorithms (OMIT, RMIT, RRMIT, PIT, and RRMIT-MIN) with the current Myrinet routing algorithm that uses only one up*/down* route from each source-destination pair. We call this algorithm OSUD (One Shortest Up*/Down* path). We vary network size from 16 switches to 64 switches and use message sizes of 32, 128, 512, and 1024 bytes.

Figures 1, 2, 3, and 4 show the results for a 16-switch network using message sizes of 32, 128, 512, and 1024 bytes, respectively. 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., 0.032 flits/ns/switch versus 0.017 flits/ns/switch with 32-byte messages).

However, all these minimal algorithms (except RRMIT-MIN), exhibit higher average latency for low traffic than the original OSUD algorithm (978 ns with RMIT versus 895 ns with OSUD for 0.01 flits/ns/switch). This is due to the use of in-transit buffers. Crossing one in-transit node adds 625 ns

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

(475 ns to detect the message and program the reinjection and 150 ns to cross the switch again) to the latency of the message. The RRMIT-MIN algorithm avoids this problem by taking into account that using in-transit nodes is equivalent to crossing one more switch, so that less in-transit nodes will be used. We can see that the RRMIT-MIN algorithm achieves the same average latency for low traffic as OSUD and offers better throughput (reaching to 0.025 flits/ns/switch, that is 1.47 times better than OSUD). On the other hand, this algorithm offers less throughput than the others algorithms do. As message size and network size increase the latency added by in-transit nodes becomes less significant. Also, new Myrinet interface implementations will lead to a reduction in the detection and reprogramming times.

The MCP software design at the interface cards gives us the posibility of switching among different routing algorithms depending on the workload. So, if low latency is needed (e.g., in a DSM system), the MCP will switch to the RRMIT-MIN algorithm, offering more throughput than up*/down* routing while keeping latency. On the other hand, if high throughput is needed for an intensive traffic workload, then the MCP will switch to RMIT or RRMIT routing algorithms.

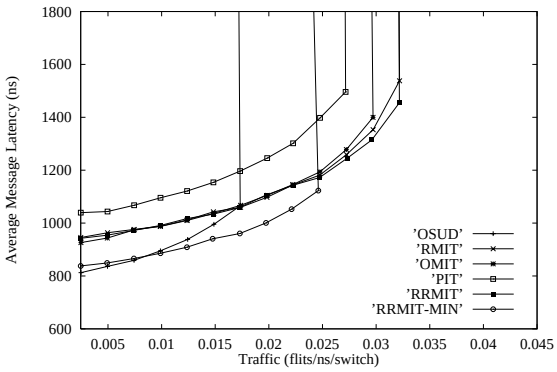

Fig. 1. Average message latency vs. traffic. Network size is 16 switches. Message length is 32 bytes

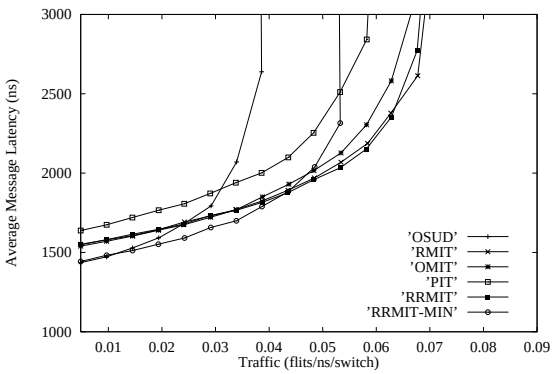

Fig. 2. Average message latency vs. traffic. Network size is 16 switches. Message length is 128 bytes

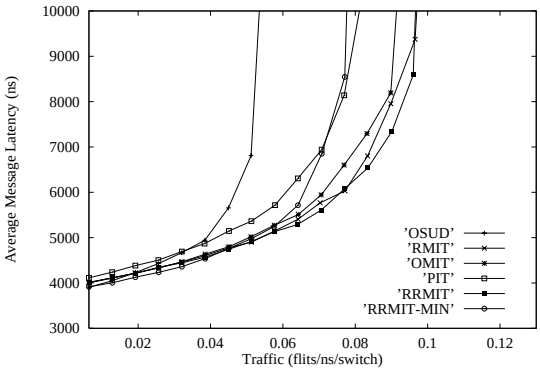

Fig. 3. Average message latency vs. traffic. Network size is 16 switches. Message length is 512 bytes

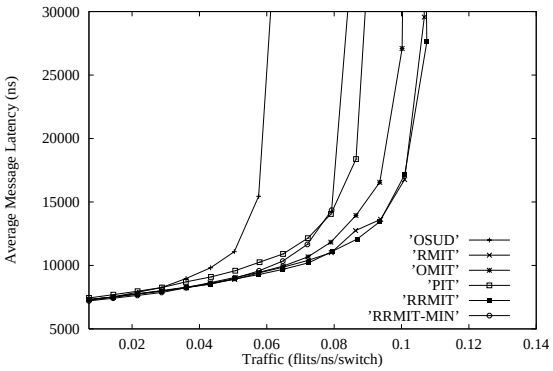

Fig. 4. Average message latency vs. traffic. Network size is 16 switches. Message length is 1024 bytes

Let us consider only the path selection strategies for minimal routing. RMIT and RRMIT are the best choices due to their possibility of choosing any of the available minimal paths versus the use of only one path in OMIT. The OMIT routing algorithm enters saturation at 0.03 flits/ns/switch for 32-byte messages while the RMIT and RRMIT routing algorithms do it at 0.032 flits/ns/switch. As was expected, the use of non-minimal paths (PIT) leads to the worst performance. PIT path selection algorithm saturates at 0.026 flits/ns/cycle and achieves a higher latency.

As network size increases, the up*/down* routing algorithm does not scale well [8]. Figure 5 shows the results for a 32-switch network. For the sake of brevity, results are only shown for message sizes of 1024 bytes. We can see that the new routing schemes double the performance achieved by the basic up*/down* routing (OSUD) except for RRMIT-MIN that is 0.66 times better. In particular, RMIT achieves more than twice the throughput achieved by OSUD for 1024 byte messages. Figure 6 shows the results for a 64-switch network. The enhancement achieved by the new approaches is even higher. In this case, the best minimal routing scheme achieves three times better throughput than the basic up*/down* scheme.

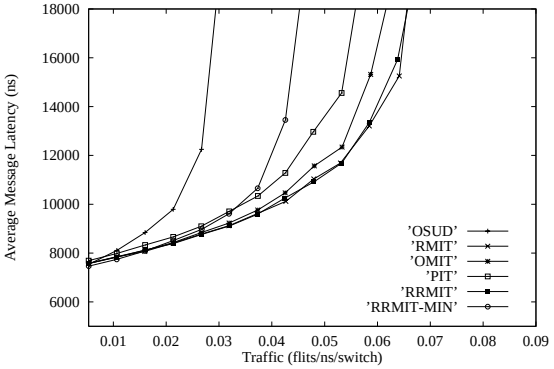

Fig. 5. Average message latency vs. traffic. Network size is 32 switches. Message length is 1024 bytes

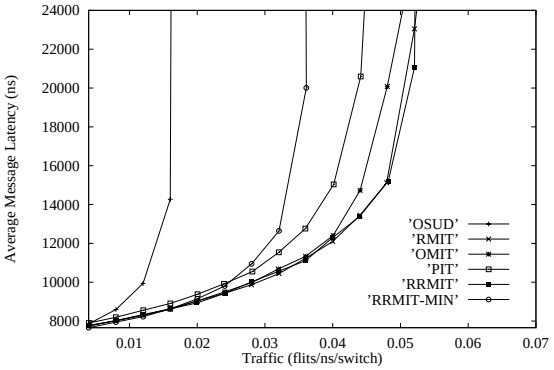

Fig. 6. Average message latency vs. traffic. Network size is 64 switches. Message length is 1024 bytes

Let us take a look at the in-transit buffers usage. Figure 7 shows the average number of in-transit buffers used per message with the different routing algorithms in a 16-switch network for 512-byte messages. From the Figure we can observe that on average less that 0.4 in-transit buffers are used. So, there is not an excessive use of in-transit buffers that could lead to the appearing of hot-spots in the network.

To conclude, the proposed software implementation of in-transit buffers in Myrinet interface card 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).

V. Conclusions

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

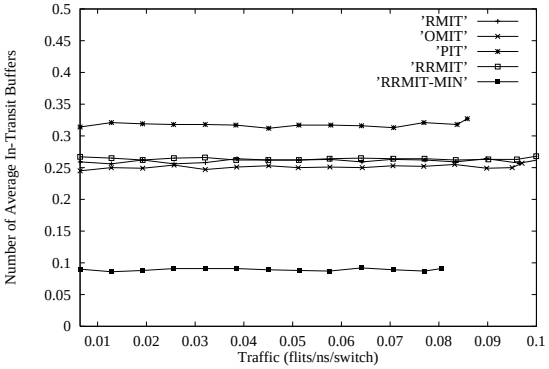

Fig. 7. Average number of in-transit buffers per message. Network size is 16 switches. Message length is 512 bytes

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

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. Horst, "ServerNet deadlock avoidance and fractahedral topologies," in *Proc. of the Int. Parallel Processing Symp.*, Apr. 1996.

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

[4] T.M. Pinkston and S. Warnakulasuriya, "On deadlocks in interconnection networks," in *The 24th International Symposium on Computer Architecture*, June 1997.

[5] 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.

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

[7] R. Sheifert, "Gigabit Ethernet," in *Addison-Wesley*, ISBN: 0-201-18553-9, April 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*, December 1997.

[9] 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.