Removing the Latency Overhead of the ITB Mechanism in COWs with Source Routing

J. Flich, M. P. Malumbres, P. López and J. Duato

Resumen—
CLUSTERS of workstations (COWs) are becoming increasingly popular as a cost-effective alternative to parallel computers. In previous papers we presented the in-transit buffer mechanism (ITB) to improve network performance, applying it to COWs with irregular topology and source routing. This mechanism considerably improves the performance of this kind of networks when compared to current source routing algorithms, however it introduces a latency penalty. Moreover, an implementation of this mechanism was performed, showing that the latency overhead of the mechanism may be noticeable, especially for short messages and at low network loads.

In this paper, we analyze in detail the latency overhead of ITBs proposing several mechanisms in order to reduce, hide, and remove it. Firstly, we show by simulation the effect of an ITB implementation much slower than the one implemented. Then, we propose three mechanisms that will overcome the latency penalty. Results show a very good behavior of the proposed mechanisms, reducing considerably, and even removing the latency overhead.

Keywords Clusters of workstations, wormhole switching, source routing, in-transit buffers, Myrinet networks.

I. Introduction

Clusters Of Workstations (COWs) are currently being considered as a cost-effective alternative for small and large-scale parallel computing. Although COWs 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. The interconnection network used is usually a local area network (LAN) all computers are attached to. Research in interconnection networks for COWs is advancing relatively fast due to the research effort made on parallel computers.

In COWs, topology is usually fixed by the physical location constraints of the computers, being the resulting topology typically irregular. On the other hand, either source or distributed routing may be used. In source routing, the path to destination is built at the source host and it is written into the packet header before delivery. Switches route packets through the fixed path found at the packet header. Myrinet network [1] uses source routing. However, with source routing, the path followed by a message can not be dynamically changed in order to avoid congested areas, thus loosing the routing adapativity behavior.

Departamento de Informática de Sistemas y Computadores, Universidad Politécnica de Valencia, E-46022 Valencia (España). Correo electrónico: jflich,mperez,plopez,jduato@gap.upv.es

Different source routing algorithms have been proposed, being the up*/down* the most well-known routing algorithm. Other source routing algorithms, like DFS [8] and smart-routing [2], have been proposed to improve the performance of up*/down*.

In [4] we have evaluated these algorithms. From this study we have identified three major factors that limit the performance of source routing networks:

- Use of non-minimal paths. As network size increases, routing algorithms based on spanning trees (up*/down* and DFS) tend to use long paths. In the case of smart-routing, the good traffic balance achieved also leads to the use of long paths. The use of long paths increases network contention as messages use, on average, more links in the network.
- Traffic unbalance. Routings based on spanning-trees obtain a more unbalanced traffic. These routings tend to saturate the zone near the root switch.
- Network contention. Because wormhole switching is used and virtual channels are not allowed, contention on one link can instantly block other links, cascading throughout the network. This serious limiting factor increases latency and reduces overall performance.

In [3] we presented a new mechanism (in-transit buffers, ITB) for networks with source routing in order to improve network performance. Basically, this mechanism avoids routing restrictions by ejecting packets at intermediate hosts and later re-injecting them. Although it was originally proposed to be used with up*/down* routing, it can be efficiently applied to any source routing algorithm [4]. Also, an implementation of the ITB mechanism [5] was made on Myrinet by modifying the network control program without changing the network hardware.

II. Motivation

However, the use of ITBs adds a latency overhead to messages that use them. This overhead will be proportional to the number of ITBs that a message uses to reach its destination. In [5] we have measured the latency overhead of the ITB mechanism in an unloaded network, showing that the impact of latency penalty may be excessive, especially for short messages and at low traffic loads. This behavior forces us to design mechanisms to reduce, hide or even remove the latency overhead.

In this paper, we propose and evaluate three different mechanisms. All of them will try to minimize

and, if possible, remove the latency penalty of the ITB mechanism in different ways. So, with these tecniques the ITB source routing mechanism will have a lot of advantages in terms of performance, removing the limiting factor of the latency penalty overhead.

The rest of the paper is organized as follows. In Section III we present the proposed mechanisms to hide and remove the ITB latency overhead. In Section IV evaluation results are presented, analyzing in detail the benefits of the proposals. Finally, in section V some conclusions are drawn and future work is anticipated.

III. Description of the Proposed Mechanisms

Basically, all the mechanisms will rely on the use of two paths for every source-destination pair. The way each of the two paths is computed will determine the use of ITBs (or not) and also how many ITBs will be used. All three mechanisms will differ in the way the host selects the path to be used every time it needs to send a message. The remaining of this section describes all the mechanisms.

A. Using Fewer ITBs for Short Messages

The first mechanism will limit the use of ITBs only for short messages. In order to implement this mechanism we need two sets of paths. The first one will contain a minimal path for every source-destination pair by using the minimum number of ITBs that guarantee minimal routing. This routing algorithm is called UD_MITB (Up*/Down* with Minimum ITBs) and will be used to send short messages. The second set of paths will be computed by using the so called UD_ITB algorithm that puts ITBs without restrictions in order to provide minimal paths for every source-destination pair. This second set of paths will be used to send long messages. Both routing algorithms (UD_MITB and UD_ITB) have been evaluated [4] and results showed that with the UD_MITB routing the latency penalty was drastically reduced because the use of ITBs was restricted. However, network throughput was also reduced (although it was higher than the throughput achieved by up*/down*).

Once the paths are computed, the mechanism will work as follows. Upon sending any message, its length is checked. If it is a short message (i.e. less than 64 bytes) then, the UD_MITB path is selected and appended to its header. Otherwise, the path supplied by the UD_ITB routing algorithm will be used.

B. Reducing Latency by Using Fewer ITBs

The second mechanism will restrict the use of ITBs in order to reduce the average latency overhead. Therefore, each host will randomly select between two paths: one with ITBs and the other one without ITBs. By limiting the use of ITBs in this way, the latency overhead will be reduced to the half.

This mechanism will use two path sets. In the first one, ITBs are prohibited and therefore paths are computed by using the up*/down* routing (referred to as UD). The second path set will be computed by using the UD_ITB routing algorithm (ITBs without restriction).

To limit the utilization of ITBs, we define the maximum allowed latency overhead (MAXLO) as a percentage that ranges from 0% (UD behavior) to 100% (UD_ITB behavior). So, we limit the utilization of ITBs by selecting one UD_ITB path with a probability of $\frac{MAXLO}{100}$, and the corresponding UD path with probability of $1 - \frac{MAXLO}{100}$. In other words, MAXLO can be viewed as the percentage of reduction of the latency overhead introduced by UD_ITB.

However, this mechanism will introduce some penalty to the overall performance in terms of network throughput. As stated earlier, the more ITBs the higher network throughput will be achieved. Therefore, if we limit the use of ITBs, the network throughput achieved will be also decreased. However, at first sight we do not know by how much the network throughput will be reduced. We will evaluate this issue later.

C. Using ITBs only at Medium and High Network Loads

Finally, the third mechanism will remove latency penalty at low traffic loads. To do that, ITBs will be used only when network load is medium or high. Therefore, in order to implement this mechanism, we need to know how the network traffic is at each moment. Because source routing is used and the entire path is decided at the time of sending any packet, we can only rely on local information to know the current traffic load.

In order to know the traffic load we define a contention coefficient. This contention coefficient will be implemented at each host and will figure out how loaded the network is. It is based on the injection delay of each sent packet.

So, when the host is going to send a packet, it annotates the current time (T_{start}). Switches along the path have buffers on each input port. These buffers are depth enough to decouple inputs from outputs. In the case there is no network contention, packet does not block and therefore injection continues without delays. Once the packet completely leaves the host, the current time is also annotated (T_{end}). The injection time will be computed as:

$$T_{end} - T_{start} = \frac{P_{size}}{I_r} \qquad (1)$$

where I_r is the link bandwidth (measured in flits/cycle) and P_{size} is the packet size.

If the packet finds contention along the path, the injection time will be higher showing a estimation of the network contention along the path. However, if there is network contention but the packet leaves the host before stopping (i.i a very short packet), this estimation would be wrong. However, if this situation remains, packets will be queued along the

path and finally contention will reach the source host. So, although late, network contention is also detected and measured.

procedure Compute_Contention_Coefficient
begin
  if $C_p > C_c$ *then*
    $C_{c_{new}} = \alpha * C_{c_{old}} + (1 - \alpha) * C_p$
  else
    $C_{c_{new}} = \beta * C_{c_{old}} + (1 - \beta) * C_p$
  endif
endp

Fig. 1. Computation of the contention coefficient.

This contention coefficient will be computed based on the procedure shown in Figure 1 where C_c is the contention coefficient and C_p is the contention coefficient computed for the last packet sent. C_p is computed using the following expression:

$$C_p = I_r * \frac{(T_{end} - T_{start})}{P_l} \qquad (2)$$

where I_r is the link bandwidth (measured in flits/cycle), T_{start} and T_{end} are the timings corresponding to the start and the end of packet injection (measured in cycles), and P_l is the packet length (measured in flits). Note that if $I_r = 1$ flit/cycle and in absence of traffic, C_p will be one. So, under these circumstances, C_p will indicate how many times the packet injection time is delayed only by the detected path contention. Note also that this way of computing C_p makes the coefficient independent from the packet length.

In Figure 1 we can observe that depending on the current value of C_p it will affect differently to the computation of C_c. The differences in the C_c computation are focused in the α and β constants with values between 0 and 1.

We use two different constants because we are interested in a fast detection mechanism (low α) in order to use ITBs as soon as possible. On the other hand, we are also interested in a slow transition (high β) of the contention coefficient from high to low values in order to avoid using up*/down* paths in medium or high traffic loads that would congest quickly the network.

In order to decide when using ITBs, each host will keep track of the contention coefficient evolution. Therefore, if the contention coefficient increases and exceeds a threshold then, the host will begin to use paths with ITBs (as it considers network traffic is medium or high). In the same way, when using paths with ITBs, if the contention coefficient decreases and goes down a threshold then, the host will use paths without ITBs.

We have chosen two thresholds to switch from UD to UD_ITB (1.5) and to switch from UD_ITB and UD (1.1). The threshold that determines the use of UD paths is lower than the one established to use UD_ITB paths. This is due to the fact that, when using UD_ITB paths, network contention is reduced due to the natural behavior of the ITB mechanism (the mechanism ejects packets temporarily and later re-injects them). If we put a low threshold to switch to UD paths (1.1) then we ensure that we will switch to UD paths in the case traffic is really low (and not due to the natural effect of the mechanism of reducing network contention).

IV. Performance Evaluation

In this section we evaluate the mechanisms presented above. First, we describe the network model and traffic patterns we will use. Then, we evaluate the latency overhead of the ITB mechanism and the mechanisms that try to hide and remove the latency overhead.

A. Network Model and Simulation Parameters

The network is composed of a set of switches and hosts, all of them interconnected by links. Network topologies are completely irregular and have been generated randomly, taking into account only 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 (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 [4][10]. We will use different network sizes of 8, 16, 32, and 64 switches. To make results independent of the topology, we will evaluate 10 random topologies for each network size.

Links, switches, and interface cards are modeled based on the Myrinet network. Concerning links, we assume Myrinet short LAN cables [7] 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 [9]. 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 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 (one flit every 6.25 ns). Each output port can process only one packet header at a time. An output port is assigned to waiting packets in a demand-slotted round-robin fashion. When a packet gets the routing control unit, but it cannot be routed because the requested output link is busy, it must wait in the input buffer until its next turn. A crossbar inside the switch allows multiple packets to traverse it simultaneously without interference.

Each Myrinet network interface card has a routing table with one or more entries for every possible des-

tination of messages. When using in-transit buffers, the incoming packet must be recognized as in-transit and the transmission DMA must be re-programmed. We will use different timings to specify the delay of both operations, detection of an incoming in-transit packet (T_d) and programming the DMA to re-inject the packet (T_r). In particular we will use two timing sets: Best case (T_d = 275 ns, and T_r = 200 ns) and worst case (5 times slower than the best case). Please note that the delays obtained in the real implementation [5] fall between both timing sets. Also, the total capacity of the in-transit buffers has been set to 512 KB at each interface card.

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. We use different message sizes (32 and 512 bytes), and in some evaluation experiments we will use bimodal traffic consisting of two different message lengths, short (32 bytes) and long (512 bytes) messages. Also, we will use a uniform distribution of message destinations for all the evaluations.

B. Impact on performance by using a slow ITB mechanism

In order to see the importance of the latency overhead of the ITB mechanism, we evaluate in this section a slow ITB mechanism that will be five times slower the one used in previous studies [3][4][6]. We evaluate both mechanisms in a 32-switch topology and use message sizes of 32 bytes and 512 bytes. Figure 2 shows evaluation results when using the UD_ITB routing with timing parameters of T_d = 275 ns and T_r = 200 ns (referred to as UD_ITB) and when using the UD_ITB routing with timing parameters of T_d = 1375 ns and T_r = 1000 ns (referred to as UD_ITB_5x). The up*/down* routing (UD) is also plotted in order to see the improvements of the mechanism in network throughput.

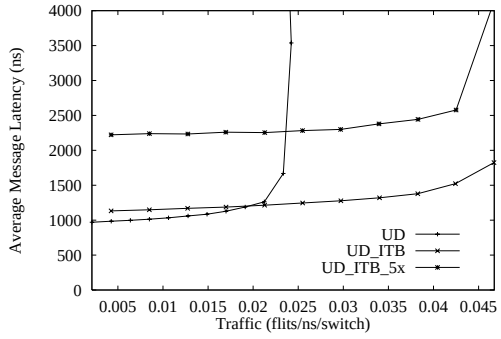

Fig. 2. Traffic vs latency. 32-switch network. Message size is 32 bytes. Uniform distribution.

As shown, the latency overhead of the mechanism is significant when using short messages. In particular, average packet latency is doubled when using a slow ITB mechanism. However, the network throughput is not reduced. The UD_ITB_5x routing (with an ITB mechanism five times slower) gets almost the same network throughput than a fast ITB mechanism. This is due to the fact that even a slow mechanism will still have the properties of using minimal paths, balancing traffic and, even more, reducing network contention. Therefore, the quickness of the mechanism only affects the latency of messages. Finally, the impact of a slow ITB mechanism on larger packets (512 bytes) is less important.

C. Using ITBs Depending on Message Length

We evaluate the first mechanism to reduce the latency overhead by using a bimodal traffic. So, each host will send an 30% of long packets (512 bytes length) and 70% of short packets (32 bytes length). The resulting routing algorithm will be referred to as UD_ITB_ML (*Up*/Down* with ITBs considering Message Length*). We compare this routing algorithm with the UD, UD_MITB, and UD_ITB routings. Figure 3 shows the evaluation results for 32-switch networks.

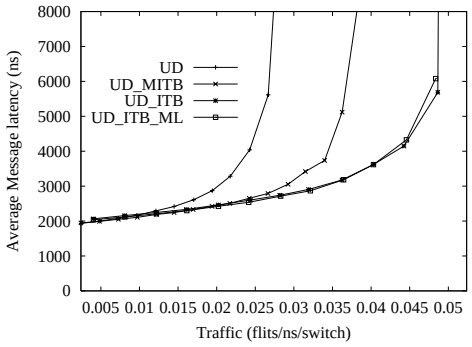

Fig. 3. Traffic vs latency. Network size is 32 switches. Bimodal traffic.

As we can see, the UD_ITB_ML obtains the same network throughput of UD_ITB, except in the 64-switch network where network throughput is slightly decreased. The UD_ITB_ML routing uses paths from UD_MITB and from UD_ITB, therefore its network throughput will be between the ones achieved by both routings.

Table I shows the average factors of throughput increase when using UD_MITB, UD_ITB, and UD_ITB_ML with respect to the UD routing and their average percentages of latency increases, also with respect to UD routing. The latency percentages were taken at low traffic conditions.

The UD_ITB_ML latency overhead is lower than the one obtained by the UD_ITB routing but it is still noticeable. Therefore, by using the first mechanism, the latency overhead is slightly decreased with respect to UD_ITB. However, it depends mainly on the percentage of long messages used in the system.

D. Reducing Latency by Using Fewer ITBs

In this section we evaluate the second mechanism that will limit the use of ITBs. We fix the percentage of paths that will use ITBs in 50%. Therefore every

TABLA I
THROUGHPUT AND LATENCY INCREASE OF UD_MITB, UD_ITB, AND UD_ITB_ML WITH RESPECT TO UD. BIMODAL TRAFFIC.

	UD_MITB		UD_ITB		UD_ITB_ML	
Sw	Thr	Lat%	Thr	Lat%	Thr	Lat%
8	1.01	0.90	1.01	1.93	1.02	1.30
16	1.17	-0.62	1.33	3.48	1.31	1.61
32	1.59	1.34	2.10	7.37	2.07	5.25
64	2.11	3.29	2.94	10.66	2.87	8.22

time a host decides to send a packet it will choose randomly from two paths, being the first one a UD path and the second one a UD_ITB path. The resulting routing will be referred to as UD_ITB_50. Figure 4 shows the evaluation results for UD, UD_ITB, and UD_ITB_50 routings in 32-switch networks using short messages.

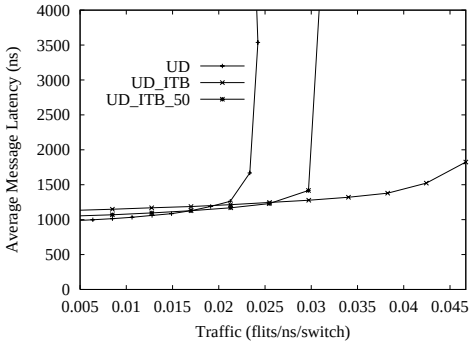

Fig. 4. Traffic vs latency. Network size is 32 switches. Message size is 32 bytes.

As we can see, the latency overhead of the UD_ITB_50 routing with respect to UD is half the latency overhead of the UD_ITB routing (with respect to UD). However, network throughput is drastically reduced by UD_ITB_50. Although network throughput of UD_ITB_50 is higher than throughput of UD, it is much lower than the one obtained by UD_ITB. By using up*/down* paths the UD_ITB_50 routing unbalances highly traffic and increases network contention.

Table II show the average factor of throughput increase and the percentage of latency increase of both algorithms, UD_ITB and YB_ITB_50. As expected, on average, the UD_ITB_50 routing decreases UD_ITB latency penalty by a half.

TABLA II
THROUGHPUT AND LATENCY INCREASES OF UD_ITB AND UD_ITB_50 WITH RESPECT TO UD. MESSAGE SIZE IS 32.

	UD_ITB		UD_ITB_50	
Sw	Thr	Lat%	Thr	Lat%
8	1.03	2.85	1.02	1.46
16	1.39	10.84	1.19	5.24
32	2.30	16.78	1.49	7.97
64	3.22	19.90	1.66	9.63

E. Using ITBs only on Medium and High Network Loads

Finally, in this section we evaluate the last mechanism. This mechanism will use ITBs only for medium and high network traffic loads, whereas for low traffic loads the use of ITBs will be avoided in order to obtain low message latencies.

We fix the parameters for computing the contention coefficient to $\alpha = 0.7$ and $\beta = 0.95$. Therefore, the mechanism will adapt faster to a high traffic load condition than to a low one ($\alpha < \beta$). We also fix the thresholds to the ones fixed in previous section (assuming $I_r = 1$). The resulting routing algorithm will be referred to as UD_ITB_DET. This routing will be compared to UD and UD_ITB.

Figure 5 shows the performance results obtained for the UD, UD_ITB, and UD_ITB_DET routings for 32-switch networks using 32-byte messages. For all the topologies, the UD_ITB_DET routing algorithm obtains the same latency as the UD does at low traffic loads. As traffic increases, the behavior of UD_ITB_DET in latency moves from the latencies of UD to the latencies of UD_ITB. However, the UD_ITB_DET routing obtains lower latencies than the UD_ITB routing does. Moreover, the UD_ITB_DET routing obtains the same network throughput as UD_ITB does.

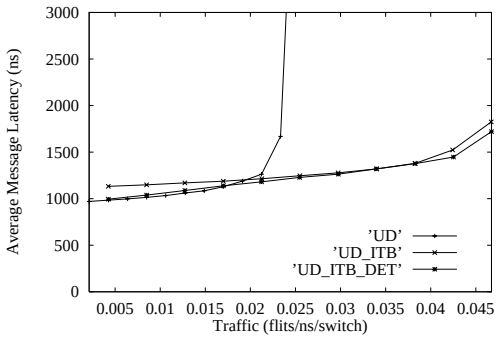

Fig. 5. Traffic vs latency. Network size is 32 switches. Message size is 32 bytes.

On the other hand, for longer messages, the behavior of the UD_ITB_DET routing is practically the same of the UD_ITB routing. Therefore, with this mechanism we have obtained a low latency overhead at low traffic loads without losing network throughput performance.

For more topologies, Table III shows the factor of throughput increase when using the UD_ITB_DET and UD_ITB with respect to UD in different network sizes and different message sizes. We can see that the network throughput is not decreased when using the UD_ITB_DET. Therefore, the good behavior achieved by UD_ITB is not compromised with the new mechanism (UD_ITB_DET). Also, the percentage of latency increase at low traffic loads when using UD_ITB_DET and UD_ITB with respect to UD are shown. We can observe that the high latency overhead exhibited by UD_ITB with short messages is re-

duced by UD_ITB_DET in all networks and allways is less, on average, than 3.6%.

TABLA III
THROUGHPUT AND LATENCY INCREASES OF UD_ITB_DET AND UD_ITB WITH RESPECT TO UD.

		UD_ITB_DET		UD_ITB	
Sw.	Msg. size	Thr	Lat%	Thr	Lat%
8	32	1.0	0.32	1.0	2.85
16	32	1.4	1.04	1.4	10.84
32	32	2.3	2.56	2.2	16.78
64	32	3.2	3.51	3.1	19.90
8	512	1.0	0.72	1.0	1.18
16	512	1.4	1.24	1.4	2.61
32	512	2.0	2.17	2.1	3.70
64	512	2.9	2.33	2.9	4.39

V. CONCLUSIONS

In previous papers [3], [4] we presented the in-transit buffer mechanism (ITB) to improve network performance of COWs with irregular topology and source routing. This mechanism considerably improves the performance of this kind of networks when compared to current source routing algorithms. However it introduces a latency penalty. In [5] an implementation of this mechanism was performed, showing that the latency overhead of the mechanism may be noticeable, especially for short messages and at low network loads.

In this paper we have evaluated the impact of the latency overhead of using ITBs with different in-transit packet processing delays, showing that in the worst case, a slow ITB packet management only affects to the average message latency considerably. However, the network throughput remains near the same as the one achieved by the fastest ITB version.

Therefore, in this paper we have proposed several mechanisms to reduce considerably the latency penalty introduced by the ITB mechanism.

In a first mechanism, the use of ITBs is restricted when sending short messages. Although it reduces the latency overhead of the ITB, its effectiveness highly depends on the message lengths.

A second mechanism has been proposed. It restricts the utilization of ITBs in order to reduce the latency. However, the network throughput achieved by this mechanism is highly restricted compared with the network throughput achieved by ITB without restrictions.

Finally, a third mechanism has been proposed. With this mechanism ITBs are used only for medium and high traffic loads. The mechanism relies on a contention coeficient computed at each host from local information. From this coeficient each host figures out the network load in order to use ITBs only at medium and high network loads. The evaluation of this mechanism have shown that latency penalty is practically eliminated (4% of latency overhead at maximum) with this mechanism at low loads, without compromising the good network throughput achieved by the ITB mechanism.

As future work we plan to incorporate the third mechanism in our final implementation of the ITB mechanism on Myrinet.

REFERENCIAS

[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] L. Cherkasova, V. Kotov, and T. Rokicki, "Fibre channel fabrics: Evaluation and design," in *29th Int. Conf. on System Sciences*, Feb. 1995.

[3] J. Flich, M.P. Malumbres, P.Lopez, and J. Duato, "Performance Evaluation of a New Routing Strategy for Irregular Networks with Source Routing," in *International Conference on Supercomputing (ICS 2000)*, ACM Press, ISBN: 1-58113-270-0, vol.1, pp. 34-43, May 2000.

[4] J. Flich, P.Lopez, M.P. Malumbres, J. Duato, and T. Rokicki, "Combining In-Transit Buffers with Optimized Routing Schemes to Boost the Performance of Networks with Source Routing," *Int. Symp. on High Performance Computing (ISHPC2K)*,Lecture Notes in Computer Science, vol.1, pp.300-309, October 2000.

[5] S.Coll, J. Flich, M.P. Malumbres, P.Lopez, J. Duato, and F.J.Mora, "A first implementation of In-Transit Buffers on Myrinet GM software," in *Workshop on Communication Architecture for Clusters (CAC '01)(IPDPS 2001)*, April 2001.

[6] J. Flich, P.Lopez, M.P. Malumbres, J. Duato, and T. Rokicki, "Improving network performance by reducing network contention in source-based COWs with a low path-computation overhead," in *Int. Parallel and Distributed Processing Symp. (IPDPS 2001)*. April 2001.

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

[8] J.C. Sancho, A. Robles, and J. Duato, "New Methodology to Compute Deadlock-Free Routing Tables for Irregular Networks," in *Workshop on Communications and Architectural Support for Network-based Parallel Computing*, January, 2000.

[9] 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.

[10] 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.