In-Transit Buffers: A Mechanism to Improve Performance of Networks with Source Routing

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

Resumen— **Clusters Of Workstations (COWs) 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. To guarantee deadlock freedom, messages are usually delivered using a routing algorithm with an acyclic channel dependence graph. However, the routing scheme is often non-minimal and network traffic tends to saturate some network areas, resulting in an unbalanced use of network links. Also, some networks use source routing [1], so the paths to reach every potential destination have to be known in advance.**

In [3], [4] we have presented a new mechanism (ITB) that improves the performance of the up*/down* routing algorithm in irregular networks with source routing. In this paper, we show that the ITB mechanism may be used in any source routing scheme in the COWs environment. In particular, we apply ITBs to DFS [6] and smart-routing [2]. DFS and smart-routing use better routes than up*/down*. So, our challenge will be to determine if ITBs can improve the performance achieved by these routing algorithms. Results show that ITB improves DFS and smart-routing by a 63% and a 23%, respectively.

Palabras clave— **Networks of workstations, irregular topologies, wormhole switching, minimal routing, source routing.**

I. Introduction

CLUSTERS of workstations (COWs) are currently being considered as a cost-effective alternative for small-scale parallel computing. In these networks, topology is usually fixed by the physical location constraints of the computers, making it irregular. On the other hand, source routing is often used as an alternative to distributed routing, because switches are simpler and faster. In this routing technique, the path to destination is built at the source host and it is written into the packet header before delivering. Switches route packets through the fixed path found at the packet header. Myrinet network [1] uses source routing.

Up*/down* [7] is one of the best known routing algorithms for irregular networks. It is used in Myrinet networks. It is based on an assignment of direction labels to links. To do so, a breadth-first spanning tree is computed and the "up" end of each link is defined as: (1) the end whose switch is closer to the root in the spanning tree; (2) the end whose switch has the lower ID, if both ends are at switches at the same tree level (see Figure 1). To eliminate deadlocks a legal route must traverse zero or more links in the "up" direction followed by zero or more links in the "down" direction. While up*/down* routing is simple, it concentrates traffic near the root switch and uses a large number of non-minimal paths.

Smart-routing [2] first computes all possible paths for every source-destination pair, building also the channel dependence graph (CDG). Then it searches through the CDG for cycles. An iterative process breaks cycles by removing dependences taking into account a heuristic cost function. This process finishes when the CDG has no cycles. Although smart-routing distributes traffic better than other approaches, it has the drawback of its high computation overhead, since it uses a linear programming solver to balance the traffic while it tries to break cycles.

The DFS routing algorithm [6] computes a depth-first spanning tree with no cycles. Then, it adds the remaining channels to provide minimal paths, which leads to cycles in the CDG. As in previous approaches, cycles are broken by restricting routing. However, channels are labeled using a heuristic in such a way that the number of routing restrictions is low.

In this paper, we propose a mechanism that removes channel dependences without restricting routing. Although this mechanism has been first proposed in [3], [4] to improve up*/down* routing, it can be applied to any routing algorithm. In this paper we will apply it to other routing schemes.

The rest of the paper is organized as follows. Section II presents how the mechanism works and some details about its implementation. In Section III the mechanism is applied to some routing strategies. In Section IV evaluation results for different networks are presented, analyzing the benefits of using our mechanism combined with previous routing proposals. Finally, in Section V some conclusions are drawn.

II. ITBs: A Mechanism to Remove Channel Dependences

The basic idea of the mechanism is the following. The paths between source-destination pairs are computed following any given rule and the corresponding CDG is obtained. Then, the cycles in the CDG are broken by splitting some paths into sub-paths. To do so, one intermediate host inside the path is selected, and, at this host, packets are completely ejected from the network as if it was their destination. Later, packets are re-injected into the network to reach their final destination. Notice that more than one intermediate host can be needed. As packets are completely ejected at intermediate hosts, the dependences between the input and output switch channels

DISCA, Universidad Politécnica de Valencia, 46071 Valencia. Correo electrónico: jflich@gap.upv.es.
This work was supported by the Spanish CICYT under Grant TIC97–0897–C04–01 and by Generalitat Valenciana under Grant GV98-15-50

are completely removed. CDG can become acyclic by repeating this process until no cycles are found.

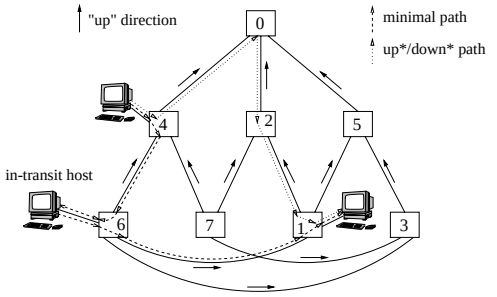

Fig. 1. Link direction assignment and use of the ITB mechanism for an irregular network.

This mechanism was first proposed to remove the dependences between 'down' and 'up' channels in the up*/down* routing algorithm [3], [4]. It was referred to as *in-transit buffers* (ITB). By removing dependences between 'down' and 'up' channels, minimal paths between some hosts, forbidden by the original up*/down* routing algorithm, are now provided. For example, although in Figure 1 there is a minimal path between switch 4 and switch 1 ($4 \rightarrow 6 \rightarrow 1$), it is forbidden because it uses an 'up' link after a 'down' link at switch 6. However, with the ITB mechanism this path is allowed by using one host at switch 6 as an in-transit host to break the dependence. By using ITBs, minimal routing can be guaranteed while keeping the deadlock-free condition.

On the other hand, ejecting and re-injecting packets at some hosts also improves performance by reducing network contention. Packets that are ejected free the channels they have reserved, thus allowing other packets requiring these channels to advance through the network (otherwise, they would become blocked). Therefore, adding some extra ITBs at some hosts may help in improving performance. Hence, the goal of the ITB mechanism is not only to provide minimal paths by breaking some dependences but also to improve performance by reducing network contention. However, ejecting and re-injecting packets at some intermediate hosts also introduces some drawbacks. First, latency of these packets is increased by some amount of time. As network load is low, this effect will be more prominent. An efficient implementation of the mechanism can help to keep this overhead low. And second, ITBs use some additional resources in both network (links), and network interface cards (memory pools and DMA engines).

If the rules used to build the paths between source-destination pairs lead to an unbalanced traffic distribution, then adding more ITBs than the strictly needed will help. This is the case for up*/down*, because this routing tends to saturate the area near the root switch. Thus, there is a trade-off between using the minimum number of ITBs that guarantees deadlock-free minimal routing and using more than these to improve network throughput.

Figure 2 shows the implementation of the in-

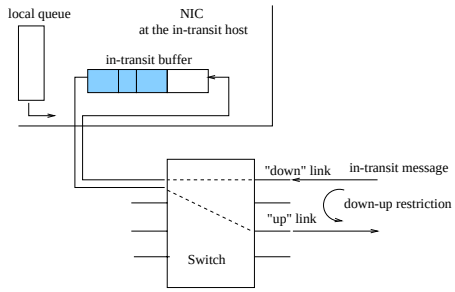

Fig. 2. In-Transit buffer mechanism.

transit buffer mechanism. Some changes in the packet header format and in the Myrinet Control Program (MCP) [5] are required. A mark is inserted in the packet header in order to notify the in-transit host that the packet must be re-injected into the network after removing that mark. 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 output channel is free, a DMA transfer can be programmed to re-inject the in-transit packet. To make the 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 dependences that may result in a deadlock (down-up transitions). Thus, 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 has been exhausted, the MCP should store the packet in the host memory, considerably increasing the overhead in this case.

III. Applying the ITB Mechanism to Some Routing Algorithms

In this section, we will apply the ITB mechanism to several source-based routing algorithms: up*/down*, DFS and smart-routing. In the case of up*/down*, we will use the two mentioned approaches: Allocating the minimum and non-minimum number of ITBs. In the first case, we will place the minimum number of ITBs that guarantees deadlock-free minimal routing. Thus, given a source-destination pair, we will compute all minimal paths. If there is a valid minimal up*/down* path it will be chosen. Otherwise, a minimal path with ITBs will be used. In the second one, we will use more ITBs than strictly needed to guarantee deadlock-free minimal routing. In particular, we will randomly choose one minimal path. If the selected path complies with the up*/down* rule, it is used without modification. Otherwise, ITBs are inserted. Notice that there may exist valid minimal up*/down* paths between the

Hence, more ITBs than the strictly necessary will be used. By using these two approaches, we can evaluate the trade-off mentioned above.

In the case of DFS, we will use ITBs in the same way as in the second approach used for up*/down* but verifying if the paths comply with the DFS rule before inserting ITBs. However, for smart-routing, we will use the following approach. We first compute the paths between source-destination pairs that better balance network traffic. It is important to notice that the obtained routes are not the same that smart-routing computes. Smart-routing computes both balanced and deadlock-free routes whereas we compute only balanced routes. For this reason, we will refer to these routes as "balanced" rather than "smart". Then, we compute the CDG and place ITBs to convert it into an acyclic one. On the other hand, since computing balanced routes alone is easier than computing both balanced and deadlock-free routes, the computational cost of the resulting routing algorithm is lower than the computational cost of the smart-routing. Hence, the resulting routing algorithm relaxes one of the most claimed drawbacks of smart-routing.

IV. Performance Evaluation

A. Network Model

Network topologies are completely irregular and have been generated randomly, taking into account three restrictions: (i) all the switches in the network have the same size (8 ports), (ii) there are exactly 4 hosts connected to each switch (so, there are 4 ports available to connect to other switches) and (iii) two neighboring switches are connected by a single link. We have analyzed networks with 16, 32, and 64 switches (64, 128, and 256 hosts, respectively).

Links, switches, and interface cards are modeled based on the Myrinet network. Concerning links, we assume Myrinet short LAN cables. 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 [8]. 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, that is, one flit every 6.25 ns. Each output port can process only one packet header at a time. An output port is assigned to waiting packets in a demand-slotted round-robin fashion. When a packet gets the routing control unit, but it cannot be routed because the requested output link is busy, it must wait in the input buffer until its next turn. A crossbar inside the switch allows multiple packets to traverse it simultaneously without interference. Each Myrinet network interface card has a routing table with one entry for every possible destination of messages. The tables are filled according to the routing scheme used.

In the case of using ITBs, 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. These timings have been taken on a real Myrinet network. Also, the total capacity of the in-transit buffers has been set to 512KB at each interface card.

B. Network Load

Several message destination distributions and sizes have been analyzed. However, for the sake of brevity, results will be shown only for 512-byte messages and uniform distribution. For each simulation run, we assume that the packet generation rate is constant and the same for all the hosts. We evaluate the full range of traffic, from low load to saturation.

C. Simulation Results

In this section we evaluate the performance of the ITB mechanism when it is applied to the above mentioned source-based routing algorithms. First, we analyze the behavior of the routing algorithms without using in-transit buffers. We evaluate the up*/down* routing (UD), the depth-first search spanning tree based routing (DFS), and the smart-routing algorithm (SMART).

Then, we evaluate the use of in-transit buffers over the up*/down* and DFS routings. For the up*/down* routing, we analyze the two above-mentioned approaches: using the minimum number of ITBs needed to guarantee deadlock-free minimal routing (UD_MITB), and using more ITBs (UD_ITB). For the DFS routing, we use the second approach. This new routing will be referred to as DFS_ITB. Finally, we evaluate the use of in-transit buffers over balanced but deadlocking routes supplied by the smart-routing algorithm. This routing will be referred to as BALANCED_ITB.

As evaluation results, we will plot the average accepted traffic measured in flits/ns/switch versus the average message latency measured in ns.

C.1 Original Routing Algorithms

Figures 3.a, 3.b, and 3.c show the behavior of the analyzed routing algorithms (UD, DFS, and SMART) for the uniform distribution of message destinations for topologies of 16, 32, and 64 switches, respectively. SMART routing is not shown for the 64-switch network because it was not available due to its high computation time. As it was expected, the best routing algorithm is SMART. It achieves the highest network throughput for all the topologies where it is used. In particular, for the 16-switch network, SMART increases throughput over UD and DFS routings by factors of 1.22 and 1.10

respectively. Also, for larger networks (32 switches) SMART increases network throughput by factors of 1.77 and 1.28 with respect to the UD and DFS routings, respectively. For the 64-switch network, the best routing algorithm is the DFS routing (SMART routing was not available). Concerning only DFS and UD routing algorithms, we observe that DFS improves UD for all the network sizes.

The higher network throughput achieved by SMART routing is due to its better traffic balancing. Figures 4.a, 4.b, and 4.c show the utilization of links connecting switches for the 32-switch network when using UD, DFS, and SMART routings, respectively. Links are sorted by utilization. Traffic is near 0.03 flits/ns/switch (where the UD routing is reaching saturation). We observe that, when using the UD routing, half the links are poorly used (52% of links with a link utilization lower than 10%) and a few links highly used (only 11% of links with a link utilization higher than 30%), being some of these overused (3 links with a link utilization higher than 50%). Traffic is clearly unbalanced among all links. On the other hand, the DFS routing softens this unbalancing. But the best traffic balancing is achieved by the SMART routing. We observe that links are highly balanced, ranging link utilization between 7.76% and 20.26%. As traffic is better balanced, more traffic can be handled by the SMART routing and therefore, higher throughput is achieved.

C.2 In-transit Buffers with UD and DFS

We focus now on the performance of the in-transit buffer mechanism when it is applied to the UD and DFS routings. Figures 5.a, 5.b, and 5.c show the performance results obtained by the resulting routing algorithms (UD_MITB, UD_ITB, and DFS_ITB) for the uniform distribution of message destinations for 16, 32, and 64-switch networks, respectively.

The in-transit buffer mechanism always improves network throughput over both original routings. Moreover, as network size increases, more benefits are obtained by the in-transit buffer mechanism. Concerning UD_MITB routing, we observe that UD is improved by factors of 1.12, 1.50, and 2.00 for 16, 32, and 64-switch networks, respectively. However, when more ITBs are used, more benefits are obtained. In particular, UD is improved by UD_ITB by factors of 1.22, 2.14, and 2.75 for the 16, 32, and 64-switch networks, respectively. Concerning DFS, DFS_ITB routing improves network throughput by factors of 1.10, 1.39, and 1.54.

The most important drawback of routings computed from spanning trees is traffic unbalancing. As network size increases, routing algorithms tend to overuse determined links (links near the root switch) and this leads to unbalanced traffic. As in-transit buffers allow the use of alternative routes, network traffic is not forced to pass through the root switch (in the spanning tree) achieving better network performance (see Figures 6.a 6.b, and 6.c).

As stated before, the ITB mechanism introduces some latency penalty. The worst case occurs when network traffic is very low. For 512-byte messages, we can observe that this increase is never higher than 5%. In general, it is noticeable only for short messages and also depends on the number of ITBs allocated (not shown).

C.3 In-transit Buffers with SMART

Smart-routing is not based on spanning trees. Moreover, its main goal is to balance network traffic. In fact, we have already seen the good traffic balancing achieved by this routing algorithm (Figure 4.c). Therefore, it seems that in-transit buffers will have little to offer to the smart-routing. Figures 7.a and 7.b show the performance results for the SMART and BALANCED_ITB routings for 16 and 32-switch networks. In 64-switch networks, smart routes were not available due to its high computation time. We observe that, also for smart-routing, the in-transit buffer mechanism increases network throughput. In particular, for 32-switch networks, the BALANCED_ITB routing increases network throughput by a factor of 1.33.

We have shown before that traffic is highly distributed among all links when SMART routing is used. Figures 8.a and 8.b show traffic balancing among all links for the SMART and BALANCED_ITB routings at 0.03 flits/ns/switch traffic point. Results are very similar in both cases. The SMART routing is quite good in balancing traffic among all links, and therefore, the in-transit buffer mechanism does not improve network throughput by balancing traffic even more.

To fully understand the better performance achieved by the BALANCED_ITB routing we focus now on network contention. For this reason, Figures 9.a and 9.b show the blocked time of links for both routing algorithms. Blocked time is the percentage of time that the link is stopped due to the flow control protocol. This is a direct measure of network contention. We observe that, smart-routing have some links blocked more than 10% of time, being some particular links blocked more than 20% of time. On the other hand, by using in-transit buffers, for the same traffic point, blocked time is kept lower than 5% for all links.

V. Conclusions

In previous papers, we proposed the in-transit buffer mechanism (ITB) to improve network performance in networks with source routing and up*/down* routing. Although the mechanism was primarily intended for breaking cyclic dependences between channels that may result in a deadlock, we have found that it also serves as a mechanism to reduce network contention and better balance network traffic.

In this paper we apply the ITB mechanism over up*/down*, DFS, and smart-routing schemes, analyzing its behavior. Results show that the in-transit

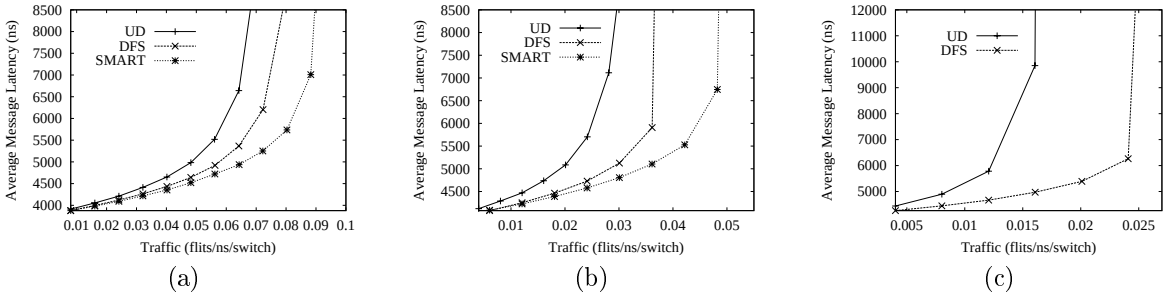

Fig. 3. Average message latency vs. traffic. Network size is (a) 16 switches, (b) 32 switches, and (c) 64 switches.

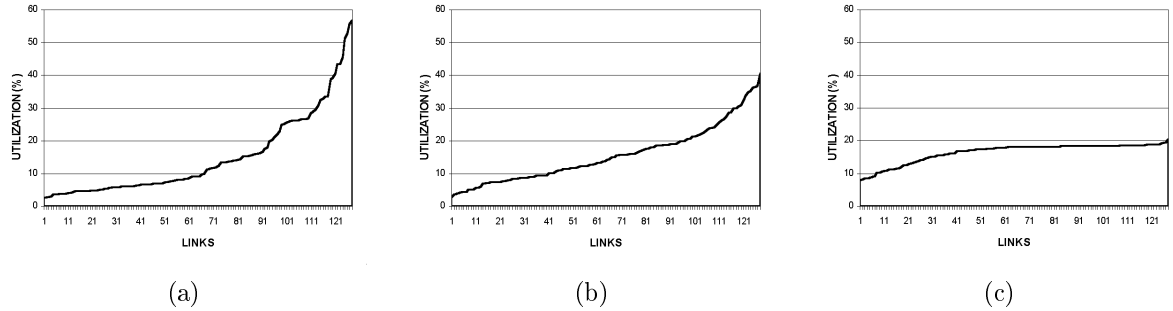

Fig. 4. Link utilization. Traffic is 0.03 flits/ns/switch. Network size is 32 switches. (a) UD, (b) DFS, and (c) SMART routings.

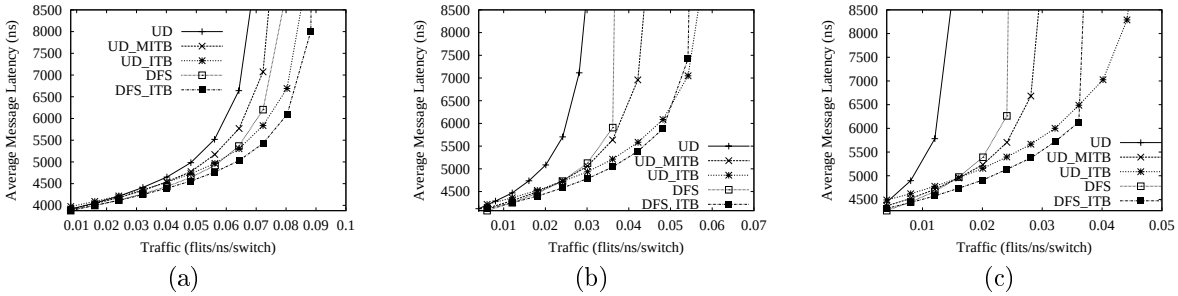

Fig. 5. Average message latency vs. traffic. UD, DFS, UD_MITB, UD_ITB, and DFS_ITB routings. Network size is (a) 16 switches, (b) 32 switches, and (c) 64 switches.

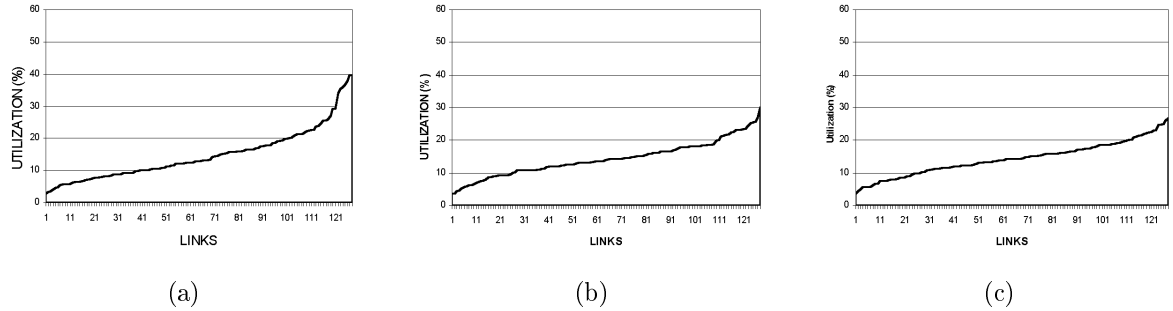

Fig. 6. Link utilization. (a) UD_MITB, (b) UD_ITB, and (c) DFS_ITB. Network size is 32 switches. Traffic is 0.03 flits/ns/switch.

all cases. Up*/down* is significantly improved due to the high routing restrictions imposed to the network and the unbalanced traffic nature of the spanning trees. Better source routing algorithms, like DFS and smart-routing, are also improved by the ITB mechanism, due to its ability to reduce network contention. Finally, we have observed that as more ITBs are added to the network, network throughput increases but the latency also increases due to the small penalty of using in-transit buffers. So, when using ITB, there is a trade-off between network throughput and message latency that network designers have to esta-

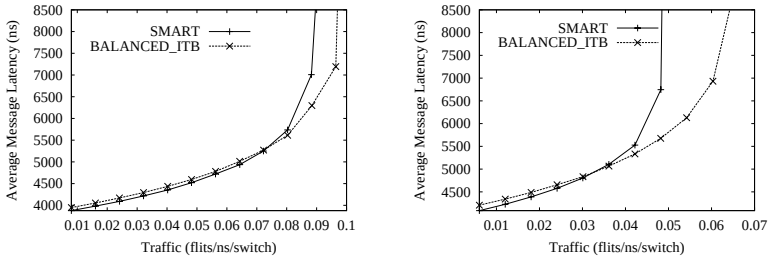

Fig. 7. Average message latency vs traffic. SMART and BALANCED_ITB routings. Network size is (a) 16 switches and (b) 32 switches.

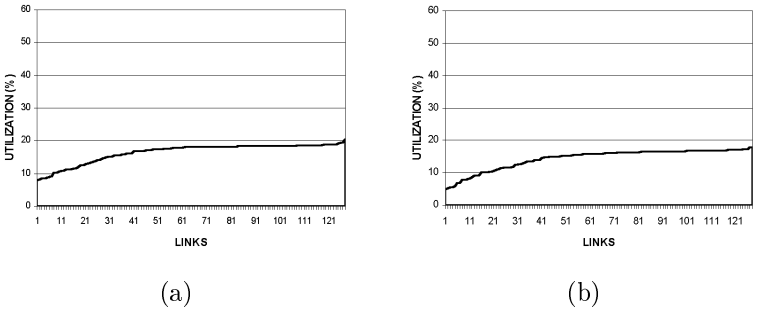

Fig. 8. Link utilization of (a) SMART and (b) BALANCED_ITB routings. Network size is 32 switches. Traffic is 0.03 flits/ns/switch.

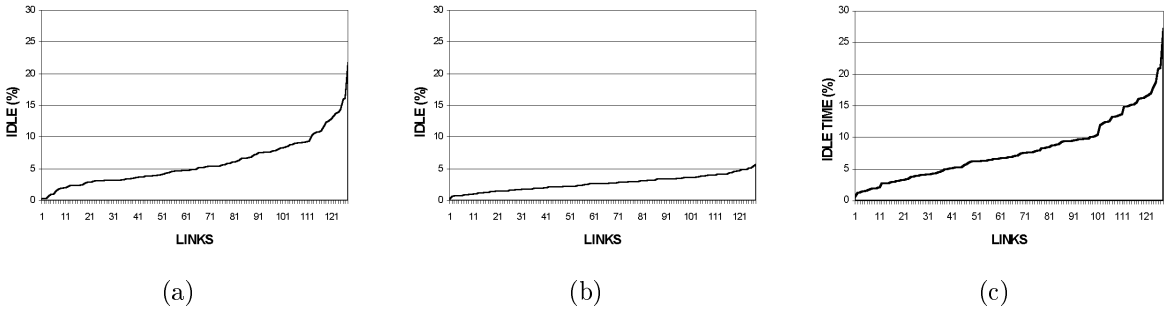

Fig. 9. Blocked Time. Traffic is (a, b) 0.05 flits/ns/switch and (c) 0.06 flits/ns/switch. Network size is 32 switches. (a) SMART and (b, c) BALANCED_ITB.

blish.

As for future work, we plan to implement the proposed mechanism on an actual Myrinet network in order to confirm the obtained simulation results. Also, we are working on techniques that reduce ITB overhead.

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 International Conference on System Sciences*, Feb. 1995.

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

[4] J. Flich, M.P. Malumbres, P.Lopez, and J. Duato, "Performance Evaluation of a New Routing Startegy for Irregular Networks with Source Routing," in *International Conference on Supercomputing (ICS 2000)*, May 2000.

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

[6] 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.

[7] 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.

[8] 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.