Combining In-Transit Buffers with Optimized Routing Schemes to Boost the Performance of Networks with Source Routing*

J. Flich[1], P. López[1], M. P. Malumbres[1], J. Duato[1], and T. Rokicki[2]

[1] Dpto. Informática de Sistemas y Computadores
Universidad Politécnica de Valencia, Spain
jflich@gap.upv.es
[2] Instantis, Inc. Menlo Park, California, USA
rokicki@instantis.com

Abstract. In previous papers we proposed the ITB mechanism to improve the performance of up*/down* routing in irregular networks with source routing. With this mechanism, both minimal routing and a better use of network links are guaranteed, resulting on an overall network performance improvement. In this paper, we show that the ITB mechanism can be used with any source routing scheme in the COW environment. In particular, we apply ITBs to DFS and routing algorithms, which provide better routes than up*/down* routing. Results show that ITB strongly improves DFS and Smart throughput (by 63% and 23%, respectively, for 64-switch networks).

1 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 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.

Up*/down* [7] is one of the best known routing algorithms for irregular networks. It is based on an assignment of direction labels ("up" or "down") to links. To eliminate deadlocks a route must traverses zero or more "up" links followed by zero or more "down" links. While up*/down* routing is simple, it concentrates traffic near the "root" switch and uses a large number of non-minimal paths.

Other routing algorithms like Smart [2] and DFS [6] achieve better performance than up*/down*. Smart first computes all possible paths for every source-destination pair, building the channel dependence graph (CDG). Then it uses an iterative process to remove dependencies in the CDG taking into account a heuristic cost function. Although Smart routing distributes traffic better than other approaches, it has the drawback of its high computation overhead. DFS computes a depth-first spanning tree with

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

no cycles. Then, it adds the remaining channels to provide minimal paths, breaking cycles by restricting routing. A heuristic is also used to reduce routing restrictions.

These routing strategies remove cycles by restricting routing. As a consequence, many of the allowed paths are not minimal, increasing both latency and contention in the network. Also, forbidding some paths may result in an unbalanced network traffic distribution, which leads to a rapid saturation. In this paper, we propose the use of a mechanism that removes channel dependences without restricting routing. This mechanism has been first proposed in [3] to improve up*/down* routing, but it can be applied to any routing algorithm. In this paper we will apply it to improved routing schemes (Smart and DFS).

The rest of the paper is organized as follows. Section 2 summarizes the mechanism works and its application to Smart some optimized routing strategies. In Section 3, evaluation results for different networks and traffic load conditions are presented, analyzing the benefits of using our mechanism combined with previous routing proposals. Finally, in Section 4 some conclusions are drawn.

2 Applying the ITB Mechanism to Remove Channel Dependences

We will firstly summarize the basic idea of the mechanism. 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, a intermediate host inside the path is selected (in-transit buffer, ITB), 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 may be needed. As packets are completely ejected at intermediate hosts and enough buffer space is provided, the dependences between the input and output switch channels are completely removed. CDG can become acyclic by repeating this process until no cycles are found.

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 increases the latency of these packets and requires 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 ones 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. Therefore, when we apply the ITB mechanism to up*/down*, we will use these two approaches. 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 approach, 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 even if there exist valid minimal up*/down* paths between the same source-destination pair.

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. However, for Smart routing, we will use a different approach. We first compute the paths between source-destination pairs that better balance network traffic. Notice that the obtained routes are not the same that Smart computes, because it 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 one of Smart routing.

3 Performance Evaluation

3.1 Network Model and Network Load

Network topology is irregular and has been generated randomly, imposing three restrictions: (i) all the switches have the same size (8 ports), (ii) there are 4 hosts connected to each switch 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 [1]. Concerning links, we assume Myrinet short LAN cables [5] (10 meters long, 160 MB/s, 4.92 ns/m). Flits are one byte wide. Physical links are one flit wide. Transmission of data across channels is pipelined [8] with a rate of one flit every 6.25 ns and a maximum of 8 flits on the link at a given time. A hardware "stop and go" flow control protocol [1] is used to prevent packet loss. The slack buffer size in Myrinet is fixed at 80 bytes. Stop and go marks are fixed at 56 bytes and 40 bytes, respectively.

Each switch has a simple routing control unit that removes the first flit of the header and uses it to select the output link. The first flit latency is 150 ns through the switch. After that, the switch is able to transfer flits at the link rate. Each output port can process only one packet header at a time. A crossbar inside the switch allows multiple packets to traverse it simultaneously.

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.

In order to evaluate different workloads, we use different message destination distributions to generate network traffic: *Uniform* (the destination is chosen randomly with the same probability for all the hosts), *Bit-reversal* (the destination is computed by reversing the bits of the source host id.), *Local* (destinations are, at most, 5 switches away from the source host, and are randomly computed), *Hot-spot* (a percentage of traffic (20%, 15%, and 5% for 16, 32, and 64-switch networks, respectively) is sent to one chosen randomly host and the rest of the traffic randomly among all hosts) and a *Combined* distribution, which mixes the previous ones. In the later case, each host will generate messages using each distribution with the same probability.

Packet generation rate is constant and the same for all the hosts. Although we use different message sizes (32, 512, and 1K bytes), for the sake of brevity results will be shown only for 512-byte messages.

3.2 Simulation Results

First, we analyze the behavior of the routing algorithms without using in-transit buffers. Results for up*/down*, DFS and the Smart routing algorithms will be refered to as UD,DFS and SMART, respectively. Then, we evaluate the use of in-transit buffers over up*/down* and DFS routing. For up*/down* routing, we analyze the two approaches mentioned above: using the minimum number of ITBs needed to guarantee deadlock-free minimal routing (UD_MITB), and using more ITBs (UD_ITB). For DFS routing, we only use the second approach, which 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 B_ITB (B from "balanced").

For each network size analyzed, we show the the increase in throughput when using the in-transit buffer mechanism with respect to the original routing algorithms. Minimum, maximum and average results for 10 random topologies are shown. In addition, we will plot the average message latency versus the accepted traffic for selected topologies.

Routing Algorithms without ITBs Figure 1 shows the results for the uniform distribution of message destinations for selected topologies of 16, 32, and 64 switches, respectively. SMART routing is not shown for the 64-switch network 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 we could evaluate. In particular, it increases throughput over UD and DFS routing by factors up to 1.77 and 1.28, respectively.

The performance improvement achieved by SMART is due to its better traffic balancing. Figure 3.a shows the utilization of links connecting switches for the 32-switch network. Links are sorted by utilization. Traffic is 0.03 flits/ns/switch. For this traffic value, UD routing is reaching saturation. When using UD routing, half the links are

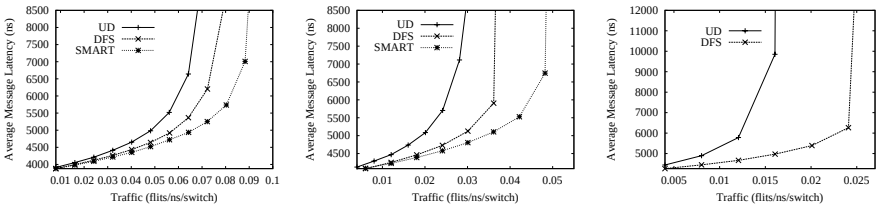


Fig. 1. Average message latency vs. accepted traffic. Message length is 512 bytes. Uniform distribution. Network size is (a) 16 switches, (b) 32 switches, and (c) 64 switches.

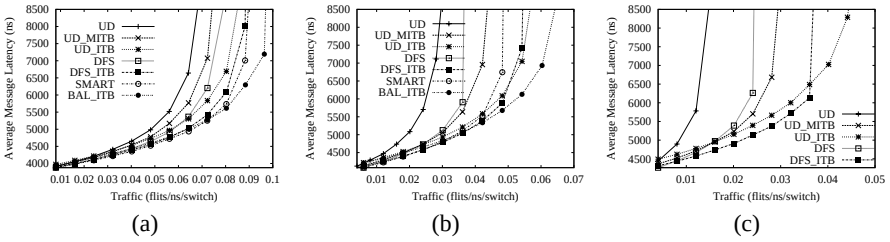


Fig. 2. Average message latency vs. accepted traffic. UD, DFS, SMART, UD_MITB, UD_ITB, DFS_ITB and B_ITB routing. Message length is 512 bytes. Uniform distribution. Network size is (a) 16 switches, (b) 32 switches, and (c) 64 switches.

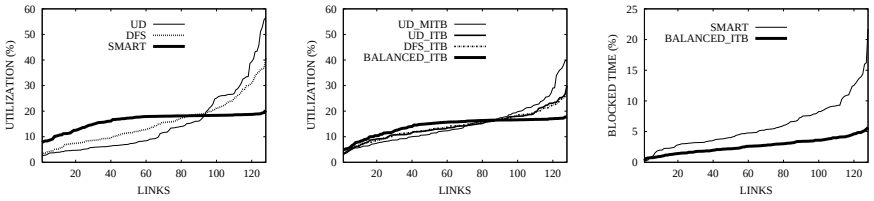
poorly used (52% of links with a link utilization lower than 10%) and a few links highly used (only 11% of links with a link utilization higher than 30%), some of them being overused (3 links with a link utilization higher than 50%). Traffic is clearly unbalanced among all the links. DFS routing reduces this unbalancing and has 31% of links with link utilization lower than 10% and 9% of links with link utilization higher than 30%. The best traffic balancing is achieved by SMART routing. For the same traffic value, links are highly balanced, link utilization ranging from 7.76% to 20.26% (76% of links with a link utilization between 15% and 20%). As traffic is better balanced, more traffic can be handled by the SMART routing and therefore, higher throughput is achieved.

Routing algorithms with ITBs Figure 2 shows the performance results obtained by the UD_MITB, UD_ITB, DFS_ITB and B_ITB routing algorithms for the uniform distribution of message destinations for selected 16, 32, and 64-switch networks, respectively. Table 1 shows the average results for 30 different topologies. For 64-switch networks, Smart routes were not available due to its high computation time.

Let us first comment the influence of in-transit buffers on up*/down* and DFS. As can be seen, the ITB mechanism always improves network throughput over both original routing algorithms. Moreover, as network size increases, more benefits are obtained. In particular, UD_MITB improves over UD by factors of 1.12, 1.50, and 2.00 for 16, 32, and 64-switch networks, respectively. However, when more ITBs are used,

	UD_MITB vs UD			UD_ITB vs UD			DFS_ITB vs DFS			B_ITB vs SMART		
Sw	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
16	1.00	1.29	1.13	1.00	1.57	1.29	1.01	1.20	1.12	1.00	1.16	1.07
32	1.16	1.72	1.46	1.50	2.14	1.88	1.25	1.56	1.41	1.11	1.33	1.23
64	1.60	2.25	1.91	2.20	3.00	2.57	1.50	1.85	1.63	N/A	N/A	N/A

Table 1. Factor of throughput increase when using in-transit buffers on the UD, DFS, and SMART routing. Uniform distribution. Message size is 512 bytes.



(a) (a) (a)

Fig. 3. Link utilization and blocked time. Network size is 32 switches. Message size is 512 bytes. Uniform distribution. (a) Link utilization for original routings. (b) Link utilization for routings with ITBs. (c) Blocked time for SMART and B_ITB routings. Traffic is (a, b) 0.03 and (c) 0.05 flits/ns/switch.

more benefits are obtained. In particular, UD_ITB improves over UD 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 over DFS by factors of 1.10, 1.39, and 1.54. for the same network sizes.

Notice that UD_ITB and DFS_ITB achieves roughly the same network throughput. These routing algorithms use the same minimal paths and the main difference between them is where the in-transit buffers are allocated and how many in-transit buffers are needed. Also, DFS_ITB routing exhibits lower average latency than UD_ITB. This is because DFS routing is less restrictive than UD routing, and therefore, DFS_ITB needs less ITBs on average than UD_ITB. When using DFS_ITB routing in the 64-switch network, messages use 0.3 ITBs on average, while the average number of ITBs per message is 0.55 in UD_ITB. This also explains the higher throughput achieved by UD_ITB since messages using ITBs are removed from the network, thus reducing congestion.

On the other hand, as network size increases, network throughput increases with respect to routing algorithms that do not use ITBs. UD and DFS routing are computed from a spanning tree and one of the main drawbacks of such approach is that, as network size increases, a smaller percentage of minimal paths can be used. For the 16-switch network, 89% of the routes computed by UD are minimal. However, for 32 and 64-switch networks, the percentage of minimal routes goes down to 71% and 61%, respectively. When DFS routing is used, something similar occurs. There are 94%, 81%, and 70% of minimal routes for the 16, 32, and 64-switch networks, respectively. When using in-transit buffers, all the computed routes are minimal.

Another drawback of routing algorithms computed from spanning trees is unbalanced traffic. As network size increases, routing algorithms tend to overuse some 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), thus improving network performance. Figure 3.b shows the link utilization for UD_MITB, UD_ITB, DFS_ITB and B_ITB routing, respectively, in the 32-switch network. Network traffic is 0.03 flits/ns/switch (where UD routing saturates). We observe that UD_MITB routing achieves a better traffic balancing than UD (see Figure 3.a). Only 33% of links have a link utilization lower than 10% and only 10% of links are used more than 30% of time. However, as this algorithm uses ITBs only when neccesary to ensure deadlock-free minimal routing, a high percentage of routes are still valid minimal up*/down* paths, and therefore, part of the traffic is still forced to cross the root switch. UD_MITB traffic balancing is improved by UD_ITB and DFS_ITB. UD_ITB routing has all the links with a utilization lower than 30% and only 20% of links are used less than 10% of time. DFS_ITB routing shows roughly the same traffic balancing.

Let us analyze now in-transit buffers with Smart routing. 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 (see Figure 3.a). Therefore, it seems that in-transit buffers will have little to offer to Smart routing. However, we observe that for Smart routing, the in-transit buffer mechanism also increases network throughput (except for one 16-switch network where it obtains the same network throughput). For a 32-switch network, B_ITB routing increases network throughput by a factor of 1.33.

In order to analyze the reasons for this improvement, Figure 3.b shows traffic balancing among all the links for B_ITB routing at 0.03 flits/ns/switch. As can be seen, it is very similar to the ones obtained by Smart (see Figure 3.a). The reason is that SMART routing is quite good in balancing traffic among all the 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 B_ITB routing we focus now on network contention. For this reason, we plot the link blocked time for both routing algorithms. Blocked time is the percentage of time that the link stops transmission due to flow control. This is a direct meassure of network contention. Figure 3.c shows the link blocked time for a 32-switch network when using SMART and B_ITB routing. Traffic is near 0.05 flits/ns/switch. We observe that Smart routing has some links blocked more than 10% of time, some particular links being blocked more than 20% of time. On the other hand, when using in-transit buffers, blocked time is kept lower than 5% for all the links for the same traffic point.

In order to analyze the overhead introduced by ITBs, Table 2 shows the latency penalty introduced by in-transit buffers for very low traffic (the worst case). We show results for 512 and 32-byte messages. For 512-byte messages we observe that, on average, the in-transit buffer mechansim slightly increases average message latency. This increase in never higher than 5%. Latency increase is only noticeable for short messages (32 bytes). In this case, maximum latency increase ranges from 16.66% to

		UD_MITB vs UD			UD_ITB vs UD			DFS_ITB vs DFS			B_ITB vs SMART		
Msg. size	Sw	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
512	16	-0.24	0.76	0.20	1.01	2.63	1.69	0.22	0.74	0.57	-0.20	2.22	1.64
512	32	0.26	2.42	1.24	2.31	4.20	3.33	0.90	1.08	0.93	1.78	2.95	2.34
512	64	-3.85	-1.03	-2.22	-1.23	1.46	0.31	0.67	1.27	1.02	N/A	N/A	N/A
32	16	0.80	3.41	2.29	8.52	16.66	10.52	1.56	5.50	3.49	-0.35	10.18	7.77
32	32	2.33	7.57	5.32	13.16	18.07	16.44	6.09	7.28	6.59	9.26	13.28	11.00
32	64	1.40	5.97	3.64	11.69	22.09	16.87	6.44	8.56	7.64	N/A	N/A	N/A

Table 2. Percentage of message latency increase for very low traffic when using in-transit buffers on UD, DFS, and SMART routing. Uniform distribution.

		UD_MITB vs UD			UD_ITB vs UD			DFS_ITB vs DFS			B_ITB vs SMART		
Distrib.	Sw	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
Hot-spot	16	0.99	1.17	1.04	0.99	1.21	1.10	1.00	1.17	1.05	0.85	1.17	0.96
Hot-spot	32	1.00	1.40	1.18	1.00	1.39	1.18	0.98	1.17	1.03	1.00	1.00	1.00
Hot-spot	64	1.60	2.08	1.71	1.66	2.57	2.03	1.21	1.49	1.35	N/A	N/A	N/A
Bit-rev.	16	0.94	1.44	1.16	0.87	1.81	1.17	0.79	1.27	1.03	0.73	1.13	0.93
Bit-rev.	32	1.12	2.00	1.59	1.56	2.57	1.87	1.20	1.99	1.51	0.99	1.45	1.21
Bit-rev.	64	1.74	2.99	2.05	2.21	3.50	2.76	1.46	2.20	1.78	N/A	N/A	N/A
Local	16	0.97	1.26	1.08	1.02	1.56	1.24	1.00	1.30	1.17	1.00	1.17	1.10
Local	32	1.00	1.40	1.16	1.12	1.60	1.44	1.15	1.45	1.29	1.10	1.29	1.17
Local	64	1.00	1.20	1.07	1.40	1.57	1.49	1.13	1.33	1.24	N/A	N/A	N/A
Combined	16	1.00	1.45	1.15	1.00	1.56	1.26	0.98	1.28	1.14	1.00	1.17	1.06
Combined	32	1.12	1.57	1.31	1.31	1.86	1.65	1.20	1.50	1.35	1.04	1.27	1.14
Combined	64	1.48	2.00	1.74	1.82	2.65	2.31	1.43	1.80	1.56	N/A	N/A	N/A

Table 3. Factor of throughput increase when using in-transit buffers on the UD, DFS, and SMART routing for different traffic patterns. Message size is 512 bytes.

22.09% for UD_ITB. The explanation is simple. The ITBs only increase the latency components that depend on the number of hops. Therefore, short messages suffer a higher penalty in latency. Additionally, the latency penalty depends on the number of ITBs needed to guarantee deadlock freedom. This is also shown in Table 2 where average latency penalty is lower when using ITBs with Smart, DFS or the minimum number of ITBs with UD (UD_MITB).

Table 3 shows the factor of throughput increase for the hot-spot, bit-reversal, local, and combined traffic patterns. We observe that the in-transit buffer mechanism always increases, on average, network throughput of UD and DFS routing. In particular, when the combined traffic pattern is used, UD_ITB improves over UD by factors of 1.26, 1.65, and 2.31 for 16, 32, and 64-switch networks, respectively. Also, DFS_ITB improves over DFS by factors of 1.14, 1.35, and 1.56 for 16, 32, and 64-switch networks, respectively. Finally, B_ITB increases, on average, network throughput by a factor of 1.14 for 32-switch networks.

We conclude that, by using in-transit buffers on all the routing schemes analyzed, network throughput is increased. As network size increases, higher improvements are obtained. In-transit buffers avoid congestion near the root switch (in the tree-based schemes), always provide deadlock-free minimal paths and balance network traffic. On

the other hand, average message latency is slightly increased, but this increase is only noticeable for short messages and small networks.

4 Conclusions

In previous papers, we proposed the ITB mechanism to improve network performance in networks with source routing and up*/down* routing. Although the mechanism was primarly 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. Moreover, it can be applied to any source routing algorithm.

In this paper we apply the ITB mechanism to up*/down*, DFS, and Smart routing schemes, analyzing its behavior in detail using up to 30 randomly generated topologies, different traffic patterns (uniform, bit-reversal, local, hot-spot, and combined), and network sizes (16, 32, and 64 switches). Network design parameters were obtained from a real Myrinet network.

Results show that, the in-transit buffer mechanism improves network performance for all the studied source routing algorithms. Up*/down* routing is significantly improved due to the many routing restrictions that it imposes and the unbalanced traffic nature of the spanning trees. Better source routing algorithms, like DFS and Smart, are also improved by the ITB mechanism. Finally, we have observed that as more ITBs are added to the network, throughput increases but the latency also increases due to the small penalty of using in-transit buffers. Therefore, there is a trade-off between network throughput and message latency. Thus, network designers have to decide on the appropiate number of ITBs depending on the application requirements.

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.

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. 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.López, and J. Duato, "Improving Routing Performance in Myrinet Networks," in *Int. Parallel and Distributed Processing Symp.*, May 2000.
4. GM protocol, 'http://www.myri.com/GM'
5. Myrinet, 'M2-CB-35 LAN cables, http://www.myri.com/myrinet/product_list.html'
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," Tech. Rep. 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 Trans. on Parallel and Distributed Systems*, vol. 5, no. 1, pp. 2–16, January 1994.