Improving Network Performance by Reducing Network Contention in Source-Based COWs with a Low Path-Computation Overhead*

J. Flich, P. López, M. P. Malumbres, and J. Duato
Dept. of Computer Engineering (DISCA)
Universidad Politécnica de Valencia
Camino de Vera, 14, 46071–Valencia, Spain
{jflich,plopez,mperez,jduato}@gap.upv.es

T. Rokicki
Instantis, Incorporated
Menlo Park, California, USA
rokicki@instantis.com

Abstract

In previous papers, we have proposed the in-transit buffer mechanism (ITB) to improve network performance in COWs with irregular topology and source routing. This mechanism allows the use of minimal paths among all hosts, breaking cyclic dependences between channels by storing and later re-injecting packets at some intermediate hosts. However, it also has two additional features that can improve even more network performance. First, the ITB mechanism reduces network contention because some messages are ejected from the network freeing network links. Second, the ITB mechanism allows the use of any path between each source-destination pair, improving traffic balance.

In this paper, we present a new routing algorithm that takes advantage of ITB by exploiting both issues: traffic balance and network contention reduction. The evaluation results show that network throughput can be considerably improved. On average, network throughput increases with respect to up/down* by factors of 2.51 and 3.77 in 32 and 64-switch networks, respectively.*

1. Introduction

Clusters Of Workstations (COWs) are currently being considered as a cost-effective alternative for small and large-scale parallel computing. In COWs, topology is usually fixed by the physical location constraints of the computers, so the resulting topology is typically irregular. Some networks, like Myrinet [1] use source routing (as opposed to distributed routing, like Servernet [7]) because switches are simpler and faster. In this case, 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.

Up*/down* [9] is the most well-known routing algorithm for irregular networks. It is based on an assignment of direction labels to links. To do so, a breadth-first spanning tree is computed and then, the "up" end of each link is defined as: (1) the end whose switch is closer to the root in the spanning tree; (2) the end whose switch has the lower ID, if both ends are at switches at the same tree level. As a result, each cycle in the network has at least one link in the "up" direction and one link in the "down" direction. Thus, cycles in the channel dependence graph (CDG) [3] are avoided by prohibiting packets to traverse links in the "up" direction after having traversed one in the "down" direction.

A similar routing algorithm is the DFS [10]. It 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. Cycles are broken by restricting routing. However, channels are labeled using a heuristic to reduce routing restrictions.

Smart-routing [2] considers all possible paths for every source-destination pair. Then, it searches through the CDG for cycles, removing dependences taking into account a heuristic. 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.

In [6] we evaluated these routing algorithms for several topologies, identifying three major factors that limit performance of networks with source routing. The first one is the 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.

*This work was supported by the Spanish CICYT, Generalitat Valenciana and Universidad Politénica de Valencia under Grants TIC97–0897–C04–01, TIC2000–1151–C07 and GV98-15-50

The second limiting factor is traffic unbalance. Routings based on spanning-trees tend to saturate the zone near the root switch. This effect is more noticeable as network size increases. On the other hand, the smart-routing algorithm highly balances network traffic.

The last limiting factor of performance is 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.

Therefore, the best routing algorithm should be the one that uses short paths, highly balances network traffic, and helps in reducing network contention, guaranteeing also deadlock freedom. Finally, it should be also desirable for this algorithm to require a reasonable low computation time and to be scalable. Existing routing algorithms have always focused in the first two issues. In this paper, we also consider network contention and computation time.

2. Motivation

The ITB mechanism [4, 5, 6] avoids routing restrictions by ejecting packets at intermediate hosts and later reinjecting them, thus breaking those dependences that cause cycles in the CDG. By avoiding routing restrictions, the ITB mechanism allows the use of minimal paths for every source-destination pair. Moreover, the mechanism allows a better traffic balance than routings based on spanning trees (up*/down* and DFS). On the other hand, when we applied this mechanism to smart-routing, we noticed that although smart-routing highly balances traffic, the resulting combined routing algorithm yields even higher network performance. This is due to the reduction in network contention achieved by ejecting packets at some hosts.

Moreover, network contention is highly affected by the number of ITBs we put in the network. The more ITBs we put, the less network contention will appear. On the other hand, as ITBs add some latency to in-transit packets [4, 5, 6], with more ITBs in the network, packets will suffer higher latency penalties. Therefore, when using ITBs, there is a trade-off between reduction in network contention and the increase in message latency.

In this paper, we analyze this trade-off. We study the effect on network contention by using different number of ITBs. In particular, we propose three methods to compute and allocate ITBs in the network.

On the other hand, routing algorithms that highly balance traffic, like smart-routing, have a high computation cost. These routing algorithms select paths considering simultaneously load balancing and deadlock freedom. Often, as new paths are computed, the decision made on other paths already computed needs to be modified, requiring the use of backtracking. As an example, the smart-routing algorithm is unable to obtain paths for networks with more that 32 switches in a reasonable amount of computing time.

By using ITBs to break cycles, path computation can be performed without taking into account deadlocks. Therefore, the path computation algorithm first selects the set of paths that offer a good traffic balance. As the deadlock-free restriction is not being considered, the number of available paths to choose from is higher and backtracking is not longer needed. Later, ITBs will be inserted to avoid deadlock. So, by using ITBs we can design a routing algorithm that offers good load balancing and can be computed quickly.

Therefore, the contribution of the paper is twofold. First, a new path computation algorithm is presented. This algorithm will balance traffic while requiring a low computation time. And second, three approaches will be used (by using different number of ITBs) in order to reduce network contention. These two contributions will we presented together as a new routing algorithm.

The rest of the paper is organized as follows. In Section 3 we present the new routing algorithm. In Section 4 evaluation results for different networks and traffic load conditions are presented, analyzing in detail the benefits of the new routing algorithm. Finally, in section 5 some conclusions are drawn and future work is anticipated.

3. A New Routing Algorithm with ITBs

Routing algorithms for networks with source routing can be divided into two stages, as shown in Figure 1. First, the routing algorithm computes a limited set of paths for every source-destination pair. After that, it selects the final set of paths that will be used. Usually, only one path for every source-destination pair is selected. These two stages (computing and selecting paths) are influenced by some restrictions. The first one is deadlock freedom. This restriction can be applied to the first stage when routes are computed (up*/down* and DFS). Therefore, only paths that do not introduce cycles in the CDG are computed. However, this restriction can be also applied to the second stage (smart-routing). The smart-routing algorithm considers all paths for every source-destination pair and then selects the path subset that does not introduce cycles in the CDG.

The second restriction is that short path lengths are preferred. However, often shortest paths are not deadlock-free. Hence, the routing algorithm must choose longer paths.

Finally, paths may be computed using a load balancing algorithm. However, again, the deadlock freedom condition restricts the number of alternative paths that can be used and therefore traffic balance may not be achieved (up*/down* and DFS). On the other hand, the smart-routing algorithm has some feedback between both stages to override this

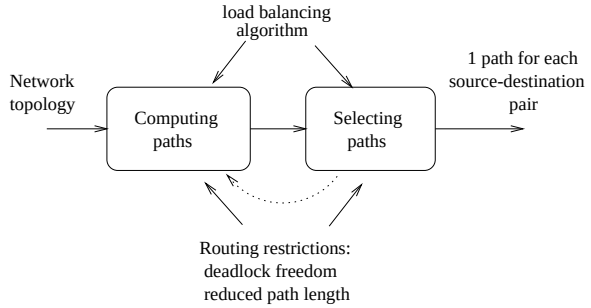

Figure 1. Stages in the design of typical routing algorithms for networks with source routing.

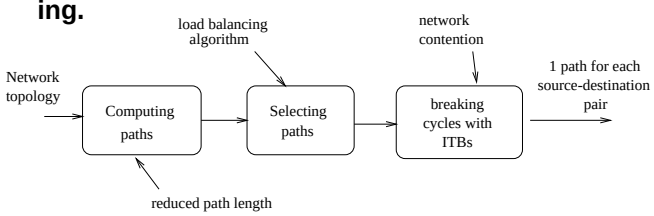

Figure 2. Stages in the design of the new routing algorithm for networks with source routing.

problem. In this case, selecting some paths can involve changing some already computed paths. This feedback can introduce a high computation time penalty.

By using ITBs, we can modify the design shown in Figure 1 into the one shown in Figure 2. First, the routing algorithm computes the whole set of minimal paths. Since there may be a high number of minimal paths between a pair of nodes, we randomly select a predefined maximum number of such paths. Notice that deadlock freedom is not longer considered in this stage. In the next step, traffic is balanced. As deadlock freedom related restrictions have not been considered, we can play with more paths and therefore, we can try to balance traffic without requiring feedback between the stages, considerably reducing computation time.

After selecting the final set of paths, the last stage breaks cycles in the CDG. Here, in-transit buffers are used in order to guarantee deadlock freedom. So, the minimum number of ITBs are placed to break the existing cycles in the CDG. On the other hand, network contention can be reduced by adding more ITBs than those needed to guarantee deadlock freedom. At the end, we have just one path for every source-destination pair and the resulting path set is deadlock free.

3.1. Balancing and Selecting Paths

Usually, routing algorithms balance traffic having in mind a uniform distribution of message destinations. Although actual traffic patterns change depending on running

```
procedure balance-and-select-paths()
var     p:path
begin
   repeat
        p=nil
        for each path
             eliminate path from weight-map
             compute standard deviation of weight-map
             add path to weight-map
             annotate the path that shows the lowest standard
                 deviation and has alternative paths (p)
        end
        if p<>nil
             eliminate path from weight-map
             delete path
        end
   until p==nil
end procedure
```

Figure 3. Balancing and selecting paths.

applications, this message destination distribution is often considered as the worst case. In fact, the smart-routing algorithm balances traffic for a uniform distribution and performs well for different traffic patterns [6]. We will take the same approach in this paper, balancing paths assuming a uniform distribution of message destinations.

The way we balance traffic is by counting the number of paths that cross each link and trying to keep that number uniform on all network links. Therefore, once paths are computed, we select those paths that best balance the number of paths per link. Figure 3 shows the algorithm used for balancing and selecting paths. This algorithm removes paths by minimizing a cost function. The cost function is the standard deviation of the number of paths that cross each link. To do that, at each loop iteration it removes one path temporarily and computes the standard deviation of the number of paths per link using the rest of paths. This is repeated for all the paths. After this process, the algorithm knows the standard deviations resulting for the elimination of each path. Then, it removes the path that shows the minimum increase (the maximum decrease) in the standard deviation. Notice that a given path will be removed only if there are still alternative paths for the corresponding source-destination pair. The algorithm finishes when none path can be removed. At the end, there will be only one path for each source-destination pair.

3.2. Breaking Cycles and Reducing Contention

The second part of our routing algorithm deals with the removal of cycles in the CDG and also with reducing network contention. As explained before, there is a trade-off between network contention and latency penalty when

```
procedure breaking-cycles()
begin
   for each link
      link-number[link] = random-number
   end
   for each path
      previous-number = link-number[path.link[0]]
      for each hop
         if previous-number < link-number[path.link[hop]]
            put ITB on path
         end
         previous-number = link-number[path.link[hop]]
      end
   end
end procedure
```

Figure 4. Breaking cycles and reducing network contention.

adding ITBs. We will evaluate three different approaches.

In the first approach, we will place the minimum number of ITBs to guarantee deadlock freedom. This is the best solution for reducing the latency increase. The second approach will put ITBs on all the switches. Therefore, packets will cross the network visiting one in-transit host at each switch. This approach obtains an acyclic CDG because there are only dependences between channels connecting two switches and channels connecting switches to hosts and vice-versa. On the other hand, this is the best approach to reduce network contention.

The last approach is half-way between the previous ones. It will try to place the appropriate number of ITBs to achieve low network contention while keeping a low latency overhead. Figure 4 shows the algorithm. First, we assign a random identifier to each link. We take care to assign unique identifiers to each link. Then, in order to guarantee that the routing algorithm is deadlock free, we will arrange the use of links. So, we introduce the following routing restriction. A message can use a link if the previously used has a lower identifier. Routing restrictions will be overcame with ITBs. The corresponding CDG will be acyclic, since in any potential cycle there always will be at least two consecutive channels that do not follow the established link order. Notice that, as link identifiers have been randomly assigned, this approach will place ITBs in about half the places the previous approach does.

4 Performance Evaluation

In this section, we evaluate the new routing algorithm. First, we present the network model and traffic patterns we will use. Then, we present the simulation results comparing different source routing algorithms with the new one proposed in this paper.

4.1. 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 neighbor switches are connected by a single link. These assumptions are quite realistic and have already been considered in other evaluation studies [5, 12]. We have analyzed networks with 16, 32, and 64 switches (64, 128, and 256 hosts, respectively). To make results independent of the topology, we evaluate up to 10 random topologies for each network size.

Links, switches, and interface cards are modeled based on the Myrinet network [1]. Concerning links, we assume Myrinet short LAN cables [8] (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 [11] 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 host chosen randomly and the rest of traffic is sent using a uniform distribution) and a *Combined* distribution, which mixes the

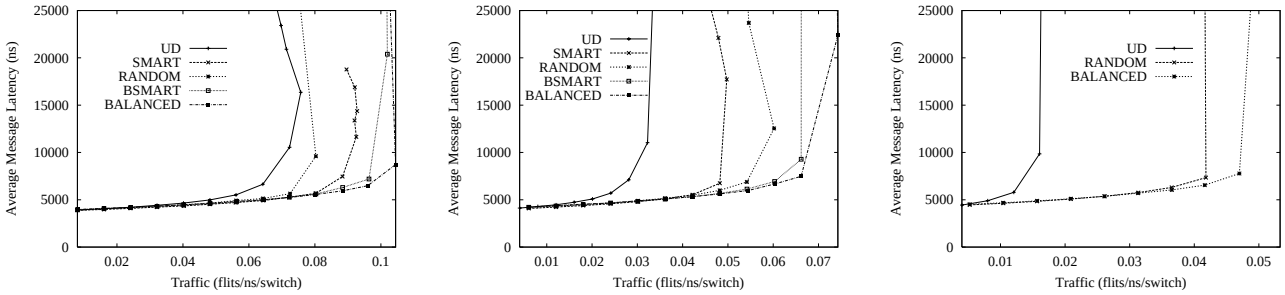

Figure 5. Evaluation results for the uniform distribution of message destinations. 512-byte messages. Network size is: (a) 16, (b) 32, and (c) 64 switches.

previous ones. In the later case, each host will generate messages using each distribution with the same probability.

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. Although we use different message sizes (32, 512, and 1K bytes), for the sake of brevity most results will be shown only for 512-byte messages.

4.2. Simulation Results

4.2.1 Load Balancing Behavior

In order to isolate the benefits achieved only by the traffic balancing method, in this section, we only use the minimum number of ITBs that guarantee deadlock freedom. The resulting routing algorithm will be referred to as BALANCED. We will compare it with classical routing algorithms such as the up*/down* (UD) and smart-routing (SMART). We will also use the routing algorithm proposed in [6] for comparison purposes. This routing algorithm combines the load balancing algorithm of the smart-routing with the use of in-transit buffers. It also uses the minimum number of ITBs and will be referred to as BSMART.

Finally, as a reference, we will also use a random traffic balancing algorithm. This algorithm computes all the possible minimal paths and selects the final set of paths randomly. This routing algorithm will also use the minimum number of ITBs and will be referred to as RANDOM.

Figure 5 shows the performance results for the different routing algorithms for 16, 32, and 64-switch networks. Message size is 512 bytes and uniform distribution of message destinations is used. Notice that SMART and BSMART routing algorithms are not shown for 64-switch networks because for this network size it was not possible to compute the paths.

As we can see, the routing algorithms that do not use in-transit buffers (UD and SMART) obtain the worst results, except for the SMART routing in the 16-switch network that outperforms the RANDOM routing. As this behavior has already been evaluated in previous papers [4, 5, 6] we only focus on the load balancing behavior.

Regarding the routing algorithms that use ITBs (BALANCED, BSMART, and RANDOM) the BALANCED and the BSMART routing algorithms achieve a higher network throughput than the one achieved by the RANDOM routing algorithm for all networks. To better analyze traffic balancing behavior, Figure 6 shows the link utilization of these algorithms when they are reaching saturation in the 32-switch network. Links are sorted by utilization.

The best traffic balance is achieved by BSMART and BALANCED, with some advantage of BSMART. However, this better balancing does not have an important impact on network performance (as shown in Figure 5). Moreover, BSMART shows a slightly decrease in performance with respect to BALANCED, due to the higher number of ITBs used by BALANCED. In particular, BSMART uses on average 0.32 ITBs per message, whereas BALANCED uses a bit more, 0.38. However, the great difference between both routing algorithms is that BSMART uses a linear programming solver when computing paths, whereas the new proposed routing algorithm (BALANCED) uses an iterative process that is faster. This is an important issue in systems that are prone to changes in network topology (i.e. COWs).

On the other hand, RANDOM routing achieves a worse traffic balance than BALANCED and BSMART. Indeed, as shown in Figure 5.b, BALANCED and BSMART increase RANDOM throughput by a factor of 1.3. This result indicates that the effort made in balancing network traffic is worth while. With a simple and easy traffic balancing algorithm, good results are achieved, and additional efforts (BSMART) yields no further improvements.

Finally, notice that SMART routing achieves a load balancing similar to the one obtained by the BSMART, as both routing algorithms use the same traffic balancing algorithm.

Table 1 shows average results for other randomly gen-

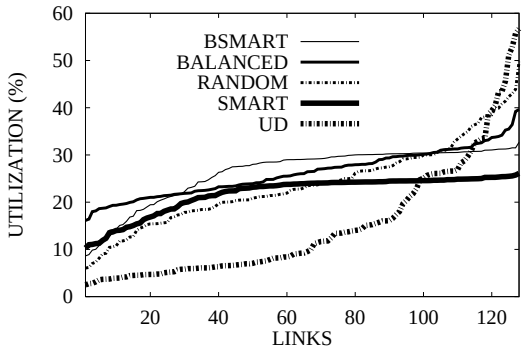

Figure 6. Link utilization at saturation points for each routing. Network size is 32 switches.

		UD	SMART	BSMART	RAND
Sw	Pattern	Avg	Avg	Avg	Avg
16	Uniform	1.44	1.12	1.03	1.21
32	Uniform	1.99	1.33	1.03	1.18
64	Uniform	2.80	N/A	N/A	1.16
16	Hot-spot	1.06	1.01	1.00	1.02
32	Hot-spot	1.25	1.09	1.05	1.07
64	Hot-spot	2.00	N/A	N/A	1.02
16	Reversal	1.45	0.93	1.09	1.18
32	Reversal	2.13	1.29	1.05	1.12
64	Reversal	2.94	N/A	N/A	1.07
16	Local	1.04	1.07	1.04	1.06
32	Local	1.04	1.10	1.07	1.04
64	Local	1.06	N/A	N/A	1.03
16	Combined	1.40	1.10	1.02	1.16
32	Combined	1.84	1.25	1.06	1.17
64	Combined	2.45	N/A	N/A	1.11

Table 1. Factor of throughput increase when using BALANCED with respect to UD, SMART, BSMART, and RANDOM routings for different traffic patterns. 512-byte messages.

erated topologies. It shows the average factor of throughput increase when using the BALANCED routing algorithm with respect to the UD, SMART, BSMART, and RANDOM routing algorithms, respectively. As we can see, on average, for the uniform distribution, UD throughput is strongly increased (almost tripled for 64 switches) whereas the SMART throughput is moderately increased (up to 33% for 32 switches). On the other hand, BALANCED also outperforms RANDOM and nearly achieves the same throughput than BSMART. This confirms that by using a simple traffic balancing algorithm (BALANCED) we can achieve the performance obtained by a sophisticated and time-consuming load balancing algorithm (BSMART).

The main drawback of using ITBs is the added latency to messages [6]. This penalty is only noticeable for low traffic loads and short messages. To analyze this effect, Table 2 shows the percentage of latency increase when using the BALANCED routing algorithm with respect to the rest of routing algorithms for low traffic loads. Messages of 32 and 512 bytes are used. Also, a combination of both message sizes (bimodal) is considered (70% of 32-byte messages plus 30% of 512-byte messages). We can see that the latency penalty is higher than 10% only for short messages on medium and large networks (32 and 64 switches) and when we compare BALANCED with respect to the UD and SMART algorithms, which do not use ITBs. Comparing BALANCED with BSMART and RANDOM, which also use ITBs (although not exactly the same number), the latency increase is small (about 4% in the worst case).

		UD	SMART	BSMART	RAND
Sw	Msgs	Avg	Avg	Avg	Avg
16	32	6.83	8.04	0.25	3.13
32	32	**15.22**	**15.57**	4.12	4.23
64	32	**17.18**	N/A	N/A	4.31
16	512	0.81	1.63	0.00	0.48
32	512	3.22	3.38	1.02	0.94
64	512	1.76	N/A	N/A	1.12
16	bimodal	2.00	3.69	-0.08	1.11
32	bimodal	**5.91**	**8.23**	2.27	1.93
64	bimodal	**2.95**	N/A	N/A	2.79

Table 2. Percentage of latency increase when using BALANCED with respect to UD, SMART, BSMART, and RANDOM. Uniform traffic pattern.

As stated earlier, the load balancing algorithms usually rely on an expected traffic pattern (usually the uniform traffic pattern). However, a different traffic pattern may appear in the network. For this, we have also evaluated the routing algorithms under different traffic patterns. Table 1 shows the factor of throughput increase when using the BALANCED routing algorithm with respect to the other routing algorithms under different traffic patterns. Although, for some traffic patterns, BALANCED achieves lower throughput increases than for the uniform traffic pattern, they are still noticeable. As traffic pattern changes, path lengths also change. With ITBs, the longer the path the more the improvement over alternative routing algorithms. For the local traffic pattern, roughly the same throughput is achieved with all the routing algorithms because path lengths are very short and fewer ITBs are needed. Finally, note that BALANCED also achieves nearly the same throughput as BSMART for the different traffic patterns considered.

4.2.2 Network Contention Behavior

Now, we focus on the second part of the routing algorithm (network contention reduction). We have proposed three approaches to assign ITBs in the network while breaking

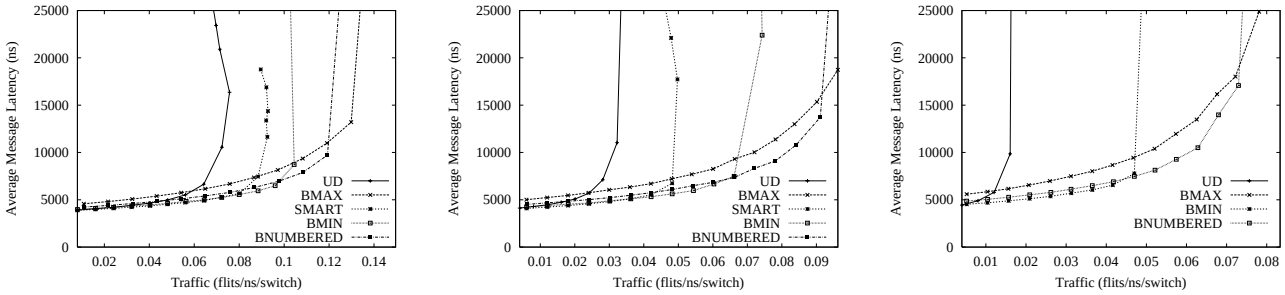

Figure 7. Evaluation results for the uniform distribution of message destinations. 512-byte messages. Network size is: (a) 16, (b) 32, and (c) 64 switches.

cycles. The first one that uses the minimum number ITBs to make the CDG acyclic will be referred to as BMIN. The one that uses ITBs in all the switches will be referred to as BMAX and the one that uses ITBs in about half the possible places will be referred to as BNUMBERED (as it numbers the links). Notice that BMIN, BNUMBERED, and BMAX use the new load balancing algorithm, using different number of ITBs. The comparison of BNUMBERED and BMAX with BMIN will evaluate the impact of ITBs on network contention. We compare also these three routing algorithms with the up*/down* (UD) and the smart-routing (SMART) algorithms.

Figure 7 shows the performance results for the different routing algorithms using 512-byte messages and a uniform traffic pattern. We observe that both BNUMBERED and BMAX are able to increase the network throughput achieved by BMIN. Notice, however, that average message latency is also increased for the full range of traffic. We will address this issue later.

By using more ITBs, BNUMBERED and BMAX reduce network contention. To analyze this fact, Figure 8 shows the percentage of stopped time of links for different routing algorithms. Network size is 32 switches and traffic is 0.066 flits/ns/switch (BMIN is reaching saturation). We observe that, with the BMIN routing algorithm, some links are stopped more than 10% of the total time. However, links with BNUMBERED are stopped less than 10%, and the 95% of links with BMAX are stopped less than 7%. Therefore, we conclude that the ITB mechanism also helps in increasing network throughput by reducing network contention. But, as average message latency is also increased, it seems that the BMAX approach is not appropiated. Indeed, the BNUMBERED routing algorithm reaches roughly the same network throughput as BMAX and uses half the ITBs.

For more network topologies, similar results have been obtained. Table 3 shows average factors of throughput increase for different network sizes and different routing algorithms. On average, BNUMBERED increases network

	UD	SMART	BMIN	BMAX
Sw	Avg	Avg	Avg	Avg
16	1.70	1.32	1.18	0.98
32	2.51	1.69	1.27	0.99
64	3.77	N/A	1.35	1.04

Table 3. Factor of throughput increase when using BNUMBERED with respect to UD, SMART, BMIN, and BMAX routings. Uniform traffic pattern. 512-byte messages.

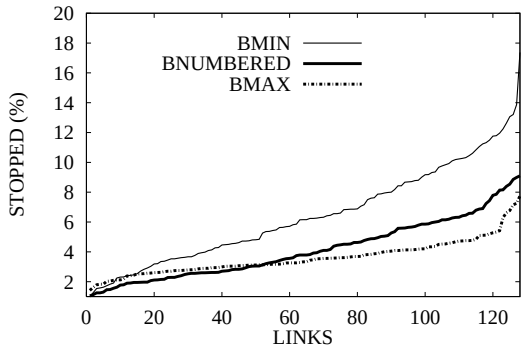

Figure 8. Stopped time (network contention). Traffic is 0.066 flits/ns/switch. Network size is 32 switches.

throughput of BMIN by factors of 1.18, 1.27, and 1.35 for 16, 32, and 64-switch networks, respectively. With respect to the traditional routings the throughput is improved by almost four times (UD). We also confirm that the BNUMBERED routing algorithm achieves near the same network throughput as BMAX does.

In order to analyze in detail latency overhead, Table 4 shows the percentage of latency increase when using the BNUMBERED routing algorithm with respect to UD, SMART, BMIN, and BMAX, respectively. We observe that,

		UD	SMART	BMIN	BMAX
Sw	Msgs	Avg	Avg	Avg	Avg
16	32	35.24	36.78	26.60	-25.00
32	32	**46.29**	**46.74**	26.96	-29.62
64	32	**48.33**	N/A	26.58	-35.84
16	512	7.92	8.80	7.05	-7.45
32	512	9.91	10.09	6.48	-10.25
64	512	9.55	N/A	7.66	-13.02
16	bimodal	15.86	17.77	13.49	-15.08
32	bimodal	**20.21**	**22.84**	16.07	-20.65
64	bimodal	**18.53**	N/A	N/A	-26.26

Table 4. Percentage of latency increase when using BNUMBERED with respect to UD, SMART, BMIN, and BMAX routings. Uniform traffic pattern.

by using many more ITBs, as BNUMBERED does, latency penalty can be increased up to 50% with respect to UD and SMART for short messages and in low traffic conditions. For bimodal traffic, latency penalty is smaller, up to 20% on average. Although this penalty should be taken into account, the great improvements in network throughput should also be considered.

For other traffic patterns (not shown) throughput improvements are lower (but still noticeable). For local traffic pattern, throughput is not decreased.

In summary, load balance is not the only key contributor to network performance. Reducing network contention with ITBs can yield significant improvements in network throughput. The BNUMBERED routing algorithm exploits both issues and outperforms previous proposals.

5. Conclusions

We have presented a new source routing algorithm based on the use of the in-transit buffer mechanism. Our routing algorithm tries to optimize three important factors that are strongly related with the overall network performance: minimal routing, traffic balance, and network contention.

Our routing algorithm computes the set of minimal paths among network nodes. Then, it chooses a subset that offers good traffic balance through a very simple algorithm. Finally, it guarantees deadlock freedom and also reduces network contention by a clever allocation of ITBs. By keeping simple the traffic balance algorithm, i) path computation can be afforded in a reasonable time and ii) paths for large networks can be computed.

The evaluation results show that traffic balance alone is able to strongly improve network throughput over any previous proposal. For instance, in a network with 64 switches, up*/down* is almost tripled. When combining traffic balance and network contention reduction, the obtained throughput is increased even more (up*/down* is increased almost 4 times) at the price of some latency overhead, specially for short messages and low traffic loads.

We are currently developing the ITB mechanism in a Myrinet network and implementing all the proposed routing algorithms. Also, we are working on techniques that try to hide and/or reduce the latency penalty.

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, Feb. 1995.

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

[3] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessors interconnection networks," in *IEEE Trans. on Computers*, vol C-36, no. 5, pp. 547-553, May 1987.

[4] J. Flich, M.P. Malumbres, P. Lopez, and J. Duato, "Improving Routing Performance in Myrinet Networks," in *Proc. of Int. Parallel and Distributed Processing Symp.*, May 2000.

[5] J. Flich, M.P. Malumbres, P. Lopez, and J. Duato, "Performance Evaluation of a New Routing Startegy for Irregular Networks with Source Routing," in *Proc. of Int. Conf. on Supercomputing*, May 2000.

[6] 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," *Proc. of Int. Symp. on High Performance Computing*, Oct. 2000.

[7] R. W. Horst, "ServerNet deadlock avoidance and fractahedral topologies," in *Proc. of the Int. Parallel Processing Symp.*, April 1996.

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

[9] S. S. Owicki and A. R. Karlin, "Factors in the performance of the AN1 computer network," *Performance Evaluation Review*, vol. 20, pp. 167-180, June 1992.

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

[11] 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, Jan. 1994.

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