Improving the Performance of Regular Networks with Source Routing *

J. Flich, P. Lpez, M. P. Malumbres, J. Duato
Dpto. Informática de Sistemas y Computadores
Universidad Politécnica de Valencia
Camino de Vera, 14, 46071–Valencia, Spain
E-mail: {jflich,plopez,mperez,jduato}@gap.upv.es

Abstract

Networks of workstations (NOWs) are becoming increasingly popular as a cost-effective alternative to parallel computers. Network products like Myrinet [1] and ServerNet [8] use the technology developped in parallel computers to compete with other high-speed local area network products. These networks allow the customer to connect processors using irregular topologies, providing the wiring flexibility, scalability, and incremental expansion capability required in this environment. Also, when performance is the primary concern, these network products are being used to build large commodity clusters with regular topologies [12].

In previous papers [5, 6], we have proposed the in-transit buffer mechanism to improve network performance, applying it to NOWs with irregular topology and source routing. This mechanism allows the use of minimal paths among all hosts, breaking cyclic dependencies between channels by storing and later reinjecting packets at some intermediate hosts. In this paper, we apply the in-transit buffer mechanism to regular networks with source routing in order to improve their performance. Also, two path selection policies are evaluated. The first one will always choose the same minimal path from source to destination, whereas the second one will choose from different alternative minimal paths in a round-robin fashion.

We evaluate by simulation several regular networks with different traffic patterns using timing parameters taken from the Myrinet network. Results show that the overall network throughput can be doubled for large networks.

1 Introduction

Due to the increasing computing power of microprocessors and the high cost of parallel computers, networks of workstations (NOWs) are currently considered as a cost-effective alternative for parallel computing.

In some of these networks, packets are delivered using source routing. In this kind of networks, the path to destination is built at the source host and it is written into the packet header before it is transmitted. Switches route packets through the fixed path found at the packet header. One example of network with source routing is Myrinet [1]. Myrinet design is simple and very flexible. In particular, it allows us to change the network behavior through the Myrinet Control Program (MCP). This software is loaded in the memory of the network interface card (NIC) at boot time. It initializes the network adapter, performs the network configuration automatically, does the memory management, defines and applies the routing algorithm, formats packets, transfers packets from local processors to the network and vice versa, etc.

One of the tasks managed by the MCP is the selection of the route to reach the destination of each packet. As the Myrinet routing scheme uses source routing, the network adapter has to build network routes to each destination during the initialization phase. Network adapters have mechanisms to discover the current network configuration, being able to build routes between itself and the rest of network hosts. Myrinet uses up*/down* routing [13] to build these paths. Although the original distributed up*/down* routing scheme provides partial adaptivity, in Myrinet only one of the routes is selected to be included in the routing table, thus resulting in a deterministic routing algorithm.

The up*/down* routing scheme does not always provide minimal paths. Also, another drawback of up*/down* routing is that it forces most of the traffic to cross the vicinity of the root switch, leading to saturation at relatively low traffic. In order to always use minimal paths and balance traffic, we have proposed the in-transit buffer mechanism [5] that consists of breaking cyclic dependencies between channels by storing and later reinjecting packets at some intermediate hosts. In [6] we have evaluated the in-transit buffer mechanism in networks with irregular topology, showing that this routing mechanism increases network throughput significantly.

When performance is the primary concern, Myrinet switches are also used to build commodity clusters with regular topologies [12]. In these topologies, up*/down* is less restrictive than in irregular ones because the up*/down* scheme supplies minimal paths for almost all destinations. Therefore, it is not clear whether the in-transit buffer mechanism will significantly improve performance with respect to up*/down* routing as in irregular topologies. In this pa-

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

per we evaluate the in-transit buffer mechanism in regular networks with source routing.

The rest of the paper is organized as follows. In Section 2, the current Myrinet source routing scheme is introduced. In Section 3 the in-transit buffer mechanism is described. In Section 4, the performance of the proposed mechanism is evaluated by simulation. Finally, in Section 5 some conclusions are drawn.

2 Myrinet Source Routing

Myrinet uses source routing to transmit packets between hosts. In this technique, the packet header stores the route that the packet has to follow to reach its destination. To simplify switch operation, each packet header consists of an ordered list of output link identifiers that are used by each intermediate switch to properly route the packet (the header also stores the header type of the payload). The first link identifier corresponds to the one that the first switch will use, the second link identifier will be used by the second switch, and so on. Each link identifier is discarded after being used. Therefore, each network host must have a representation of the current network topology, in order to build and maintain routes between itself and each potential destination host. Routes are built before sending any packet during the initialization phase. In addition, each network adapter checks for changes in the network topology (shutdown of hosts, link/switch failures, start-up of new hosts, etc.), in order to maintain the routing tables.

Myrinet uses up*/down* routing [13] to build network routes. Up*/down* routing is based on an assignment of direction to the operational 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). The result of this assignment is that each cycle in the network has at least one link in the “up” direction and one link in the “down” direction. To eliminate deadlocks while still allowing all links to be used, this routing scheme uses the following up*/down* rule: a legal route must traverse zero or more links in the “up” direction followed by zero or more links in the “down” direction. Thus, cyclic dependencies between channels are avoided because a message cannot traverse a link along the “up” direction after having traversed one in the “down” direction.

Up*/down* routing is not always able to provide a minimal path between some pairs of nodes, as shown in the following example. In Figure 1, a message transmitted from switch 4 to switch 1 cannot go through any minimal path. The shortest path (through switch 6) is not allowed since the message should traverse a link in the “up” direction after one in the “down” direction. All the allowed paths (through switches 0, 2, and through switches 0, 5) are non-minimal and only one of them will be included in the routing table. The number of forbidden minimal paths increases as the network becomes larger.

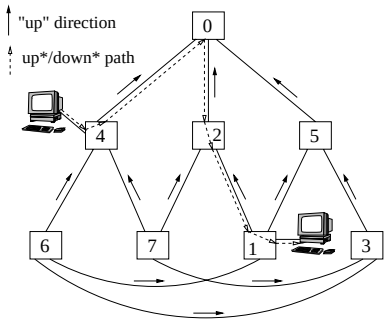


Figure 1. Link direction assignment for an irregular network.

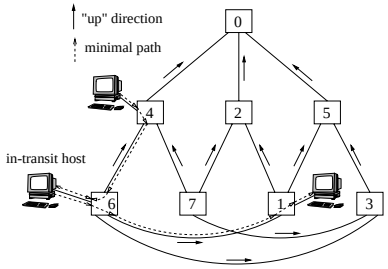


Figure 2. Use of the in-transit buffer mechanism in an irregular network.

3 In-Transit Buffers: a Mechanism to Implement Minimal Source Routing

The up*/down* routing algorithm is deadlock-free. It avoids cyclic dependencies between network links by not allowing messages to reserve “up” links after having reserved “down” links. Due to this restriction many minimal routes are forbidden. The basic idea to eliminate this restriction consists of splitting such forbidden paths into several valid up*/down* paths. On each path, an intermediate host is selected as the destination and, at this host, packets are completely ejected from the network and later re-injected into it. In other words, the dependencies between “down” and “up” links are removed by using some buffers at the intermediate hosts (in-transit buffers). In Figure 2 we can see that, with the in-transit buffer mechanism, a minimal route can be used to route packets from switch 4 to switch 1. To break channel dependencies, packets are sent to a host connected to the intermediate switch 6. This host will re-inject packets as soon as possible.

When the up*/down* routing algorithm for a given packet does not provide any minimal path, the proposed routing strategy selects a minimal path. In this path, one or more in-transit hosts are chosen, verifying that each subroute is a valid up*/down* path, and therefore, the routing algorithm is deadlock-free. The packet will be addressed to

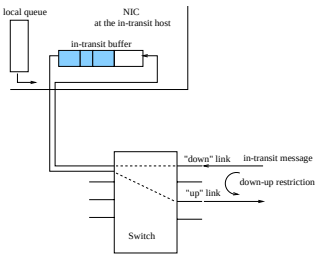


Figure 3. In-Transit buffer mechanism.

the first in-transit host. The in-transit host will re-inject the packet into the network as soon as possible, forwarding it to the destination host or to the next in-transit host.

In order to route packets requiring in-transit buffers, the packet header format must be changed. In particular, a mark (ITB mark) is inserted in order to notify the in-transit host that the packet must be re-injected into the network after removing that mark.

The in-transit buffer mechanism adds latency to the message and also uses some additional resources in both network (links) and network interface cards (memory pools and DMA engines). On the other hand, with this mechanism, "down" to "up" transitions are allowed. As a consequence, the resulting routing algorithm is less restrictive than the original up*/down* routing algorithm, as it always uses minimal paths among all hosts.

The critical part of this mechanism is the introduced overhead at the intermediate hosts. Figure 3 shows the implementation of the in-transit buffer mechanism. To implement the in-transit buffer mechanism in Myrinet networks, 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 required output channel is free, a DMA transfer can be programmed to re-inject the in-transit packet. So, the delay to forward this packet will be the time required for processing the header and starting the DMA (when the output channel is free). As the MCP allows this kind of DMA programming, it is possible to implement the in-transit buffer mechanism in Myrinet without modifying the network hardware. On the other hand, there is no problem if the DMA transfer begins before the packet has been completely received, because it will arrive at the same rate that it is transmitted[1], assuming that all the links in the network have the same bandwidth[2]. Note that Myrinet does not implement virtual channels. Therefore, once a packet header reaches the network interface card, flits will continue arriving at a constant rate. The only additional requirement is that the packet is completely stored in the network adapter memory at the source host before starting transmission to avoid interference with the host I/O bus.

To make this mechanism deadlock-free, it must be guaranteed that an in-transit 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 dependencies that may result in a deadlock (down-up transitions). So, 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. Although this strategy requires an infinite number of buffers in theory, a very small number of buffers are required in practice. We rely on dynamic allocation of buffers to simulate infinite buffer capacity.

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

[2]Myrinet supports mixing links with different bandwidth.

4 Performance Evaluation

In this section, we evaluate the new mechanism and compare it with the original up*/down* routing used in Myrinet. First, we describe the different topologies and traffic patterns used in the study. Then, we describe the simulation parameters concerning links, switches, and network interfaces. These parameters are based on the Myrinet network. Finally, we present the simulation results.

4.1 Network Model

The network is composed of a set of switches and hosts, all of them interconnected by links. In order to perform a detailed study, we evaluated several regular topologies. Some of them are well-known topologies and others have already been implemented using Myrinet switches. These topologies are the following:

- 2-D Torus (Figure 4). It is made up of 64 16-port switches. Each switch is connected to each of its four neighbors through a single link. There are 8 hosts connected to each switch, so there are 512 hosts in the whole system. There are 4 ports left open in each switch.
- 2-D Torus with express channels (Figure 5). This topology is similar to the 2-D Torus except that all switches are also connected to their second-order neighbors using express channels [3] (neighbors located two hops away in each dimension). Each switch has 16 ports. There are 8 hosts connected to each switch, so there are 512 hosts in the whole system. All ports are used in all switches.
- CPLANT (Figure 6). This topology is used in the Computational Plant (CPLANT) at the Sandia National Laboratories [12]. It is made up of 50 16-port switches connecting 400 nodes (each switch has 8 hosts attached to it). Out of them, 48 switches are grouped into 6 groups of 8 switches. Each switch uses 4 ports to connect to other switches in the same group and 4 ports to connect to its equivalent switches in the remaining groups. Each group forms a hypercube topology in which an additional link is used at each switch to connect to the farthest node in

the group. The six groups form an incomplete hypercube, which also contains connections between farthest nodes. The remaining 2 switches form an additional group. Therefore, the resulting topology i not completely regular.

4.2 Traffic Patterns

Message generation rate is constant and the same for all the hosts. Several message destination distributions have been used to generate network traffic:

- Uniform distribution. The destination of a message is randomly chosen with the same probability for all the hosts. This is the most widely used pattern.
- Bit-reversal distribution. The destination of a message is computed by reversing the bits of the source host identification number. This pattern has been selected taking into account the permutations that are usually performed in parallel numerical algorithms [9, 10]. Note that this distribution can only be used when the number of nodes is a power of 2. This distribution is applied to all the networks presented above except to the CPLANT network.
- Hotspot distribution. A percentage of traffic is sent to one host (different percentages are used). The selected hotspot location is chosen randomly (10 different simulations are performed using 10 different hotspot locations). The rest of the traffic is generated randomly using a uniform distribution.
- Local distribution. Message destinations are, at most, 3 switches away from the source host, and are randomly computed. Also, we studied the effects of local distribution with 4 switches distance.

For message length, 32, 512, and 1024-byte messages have been considered. However, taking into account that the obtained results are qualitatively similar, for the sake of brevity, onle results for 512-byte messages are presented.

4.3 Myrinet Links

We assume short LAN cables [11] 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 [14]. 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.

We do not use virtual channels since current Myrinet switches do not support them. A hardware "stop and go" flow control protocol [1] is used to prevent packet loss. In this protocol, the receiving switch transmits a stop(go) control flit when its input buffer fills over (empties below) 56 bytes (40 bytes) of its capacity. The slack buffer size in Myrinet is fixed at 80 bytes.

4.4 Myrinet Switches

Each Myrinet switch has a simple routing control unit that removes the first flit of the header and uses it to select the output link. That link is reserved when it becomes free. If 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.

4.5 Myrinet Interfaces

Each host is connected to the Myrinet network through a network interface card (NIC). This card contains the LANai processor, some buffer memory (4MB in the current version), and three DMA devices. Each NIC contains a routing table with one or more entries for every possible destination of messages. The LANai processor fills the routing table and uses it to send packets to other hosts. The way tables are filled determines the routing scheme that will be used. We are interested in comparing the performance achieved by the original Myrinet routes using the up*/down* routing algorithm with the new routes that use in-transit buffers.

Although tables can be filled with all the possible routes to every possible destination, to avoid using a huge table that may result in a long look-up delay, we imposed a limit of 10 alternative routes for each source-destination pair. When using the original Myrinet routes, only one route is inserted in the table for every source-destination pair. These routes have been obtained from the `simple_routes` program that comes with the GM [7] protocol from Myricom. This program computes the entire set of up*/down* paths and then selects the final set of up*/down* paths (one path for every source-destination pair) trying to balance traffic among all the links. This is done by using weighted links. So, it may happen that the `simple_routes` program selects a non-minimal up*/down* path, instead of an available minimal up*/down* path. In fact, we have compared the performance of the `simple_routes` routing scheme versus using all the minimal up*/down* paths available. We concluded that the routes given by the `simple_routes` program always achieve higher network throughput. Also, by using the routes generated by the `simple_routes` program, we simulate the behavior of Myrinet using its original routing algorithm.

In the case of minimal routing with in-transit buffers, 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 [3]. Also, the total capacity of the in-transit buffers has been set to 90KB at each Myrinet interface card.

[3] These timings have been measured on a real Myrinet network. Average timings have been computed from the transmission of more than 1000 messages using the Real Time Clock register (RTC) of the Myrinet interface card.

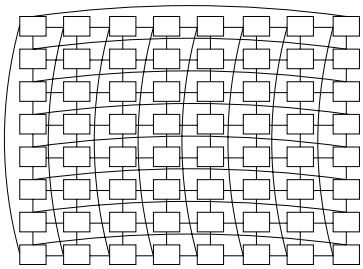


Figure 4. 2-D Torus network.

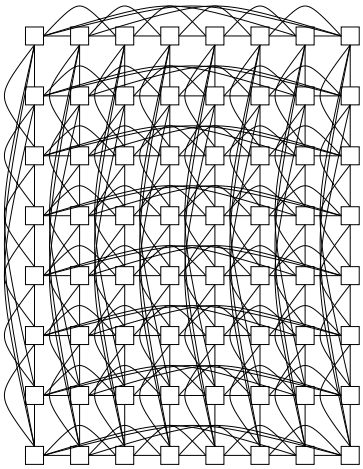


Figure 5. 2-D Torus with express channels.

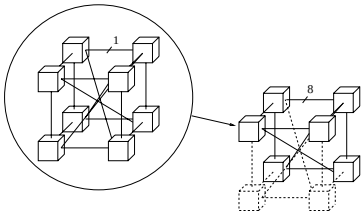


Figure 6. CPLANT network at Sandia National Labs.

4.6 Path Selection Policies

Although Myrinet uses only one of the shortest paths for each source-destination pair to send packets, different paths may be available for each source-destination pair. We will use two path selection policies when using in-transit buffers: The first one will always select the same (minimal) path for each source-destination pair and the second one will select the path from all the alternative minimal paths in a round-robin fashion.

4.7 Simulation Results

In this section we show the results obtained from the simulation of Myrinet networks using both the original up*/down* routing scheme and the new routing strategy with in-transit buffers. We will refer to the original routing as UP/DOWN. We combined the new routing scheme based on in-transit buffers with the different path selection policies described above. We will refer to them as ITB-SP for the routing using in-transit buffers and single path selection policy and ITB-RR for the routing using in-transit buffers and round-robin path selection policy. We group results by traffic pattern. For all the cases, we show the average message latency[4] measured in nanoseconds versus the average accepted traffic[5] measured in flits/ns/switch.

4.7.1 Uniform Distribution

Figure 7 shows the performance of the in-transit buffer mechanism and the original Myrinet up*/down* routing algorithm for a uniform distribution of message destinations and different topologies. In the 2-D Torus network (Figure 7.a) both routings schemes using in-transit buffers (ITB-SP and ITB-RR) double the throughput achieved by the original Myrinet routing algorithm (UP/DOWN). In particular, ITB-SP and ITB-RR reach 0.029 and 0.032 flits/ns/switch, respectively. UP/DOWN saturates at 0.015 flits/ns/switch.

One of the drawbacks of up*/down* routing is that it does not always provide minimal paths. In this topology, 80 % of the paths computed by the original Myrinet routing algorithm are minimal paths. On the other hand, the in-transit buffer mechanism uses always minimal paths. So, the in-transit buffer mechanism adds 20% of minimal paths to up*/down* routing. The average distance to destination (measured as the number of traversed links) for up*/down* routing is 4.57 whereas with the in-transit buffer mechanism is 4.06. However, this does not justify the performance improvement achieved when using in-transit buffers.

The other drawback of up*/down* routing is that it forces most of the traffic to cross the root switch. The in-transit buffer mechanism distributes network traffic better by allowing the use of alternative paths. Figure **??** shows the link utilization for UP/DOWN and ITB-RR when traffic is 0.015 flits/ns/switch (UP/DOWN reaches its saturation point). When using UP/DOWN routing (Figure **??**.a), links near the root switch (the top leftmost switch) are congested (utilization in those links reaches 50 %), whereas the utilization of the rest of links in the network is low (65 % of links have a utilization less than 10 %). On the other hand, the ITB-RR routing algorithm (Figure **??**.b) balances traffic among all the links in the network. The utilization of all the links is less than 12 %. For higher traffic, when ITB-RR is reaching its saturation point, ITB-RR still distributes traffic evenly among all the links. Figure **??** shows the link utilization for ITB-RR routing at injection rate equal to 0.03 flits/ns/switch. Traffic is quite balanced among all the links (link utilization ranges from 14 % to 29 %). So, when using in-transit buffers, minimal paths are always used and, most important, traffic is balanced among all the links in

[4]Latency is the elapsed time between the injection of a message into the network at the source host until it is delivered at the destination host.

[5]Accepted traffic is the amount of information delivered by the network per time unit.

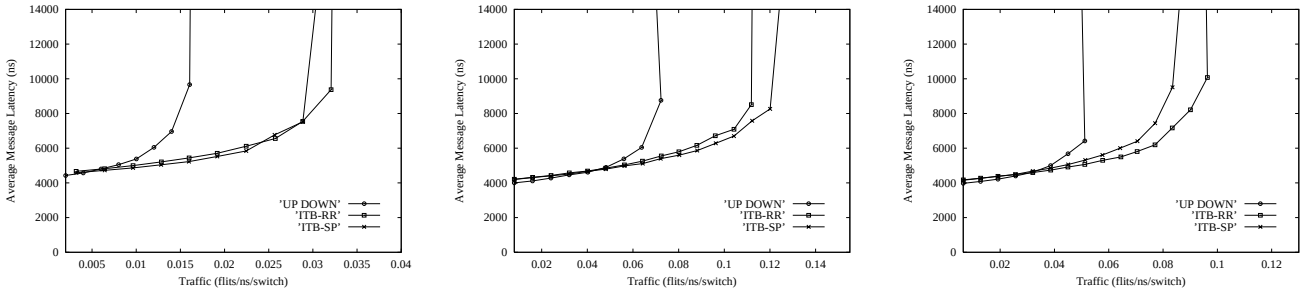


Figure 7. Performance results for the uniform distribution. a) 2-D Torus, b) 2-D Torus with express channels, c) CPLANT

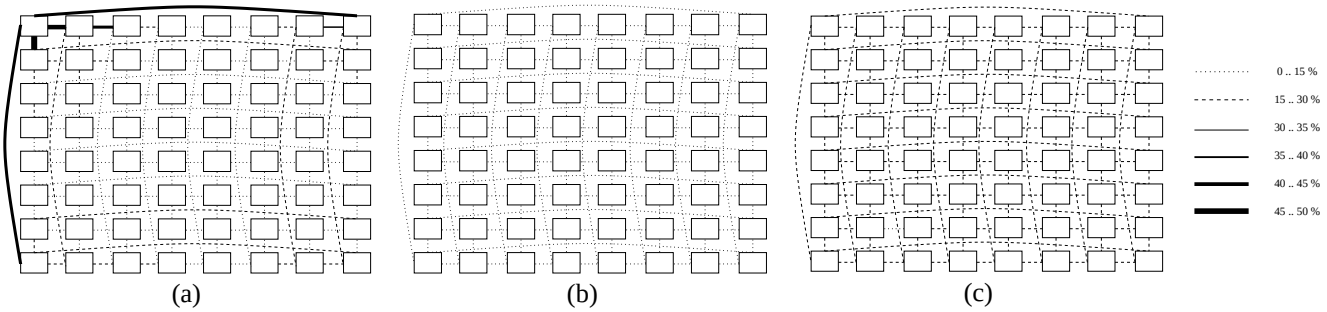


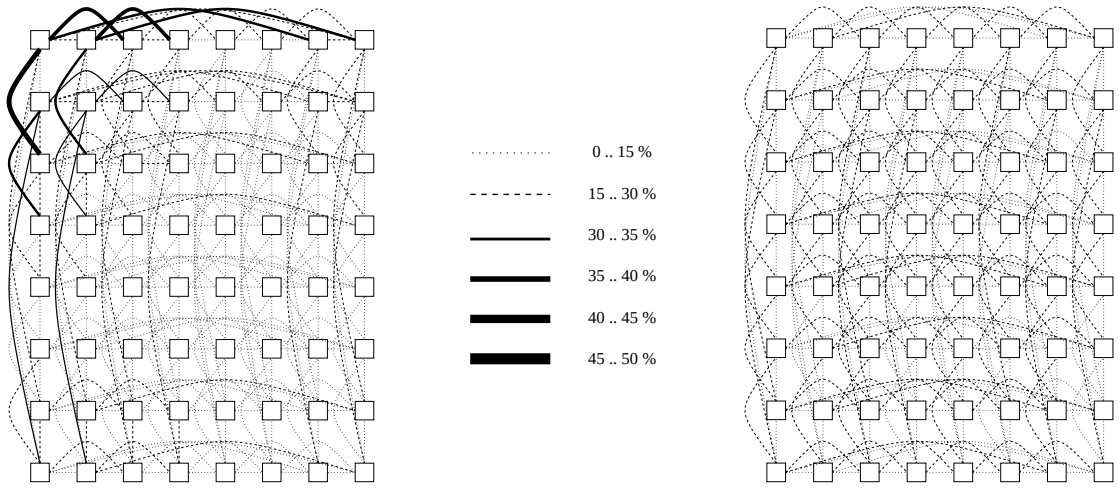
Figure 8. Link utilization in a 2-D Torus. a) UP/DOWN (0.015 flits/ns/switch) b) ITB-RR (0.015 flits/ns/switch) c) ITB-RR (0.03 flits/ns/switch).

the network, doubling the network throughput achieved by up*/down* routing. It can be observed that the network saturates when link utilization is still low. The long routing time (150 ns) and the small capacity of slack buffers (80 bytes) lead to a high correlated message blocking. Thus, at the saturation point for the ITB-RR routing algorithm, 20 % of links are iddle more than 10 % of the total time due to the flow control mechanism, reaching 30 % of the total time for some links.

Comparing ITB-SP and ITB-RR performance (Figure 7.a), ITB-SP achieves slightly lower latency. This is due to the fact that, on average, more in-transit buffers are used by messages when using ITB-RR (0.43 in-transit buffers per message when using ITB-SP and 0.54 when using ITB-RR, on average). However, ITB-RR has a higher saturation point (0.032 flits/ns/switch) than ITB-SP (0.029 flits/ns/switch). This is due to the higher number of choices that ITB-RR offers to forward messages toward their destinations, distributing traffic even better.

Figure 7.b shows results for the 2-D Torus with express channels. ITB-SP and ITB-RR almost double the throughput achieved by UP/DOWN routing. The benefits of using in-transit buffers are slightly smaller than in the 2-D Torus because when using express channels, there are twice as many links and there are more alternative paths towards the root switch (where UP/DOWN routing saturates). UP/DOWN multiplies the throughput achieved in the 2-D Torus network by 4.6, reaching 0.07 flits/ns/switch, whereas ITB-SP and ITB-RR multiply network throughput by 4, reaching 0.12 and 0.11 flits/ns/switch, respectively. The increase in network throughput (with respect to the 2-D Torus network) is due to the added express channels. The number of links in the network is doubled, so more messages can be crossing the network at the same time. Also, average distance to message destinations is almost reduced to the half. Thus, each packet uses half the resources (slack buffers). Also, as a consequence, routing time for the overall path is also reduced.

The use of express channels increases the number of minimal paths provided by UP/DOWN. In this case, the percentage of minimal paths is 94 %. So, providing more minimal paths is not as important as in the 2-D Torus. On the other hand, traffic distribution plays a key role in this network. Figure 9 shows link utilization for UP/DOWN and ITB-RR routing at the saturation point for UP/DOWN (0.066 flits/ns/switch). We can see that when



(a) UP/DOWN (b) ITB-RR

Figure 9. Link utilization in a 2-D Torus with express channels at saturation point for UP/DOWN.

using UP/DOWN (Figure 9.a) links near the root switch have a utilization near 50 %, whereas the rest of links in the network have a low utilization (as in the 2-D Torus). On the other hand, when using ITB-RR routing (Figure 9.b), link utilization is more balanced, as in the 2-D Torus. All links have a utilization lower than 30 %. If we take a closer look at Figure 9.b we can observe that there are links more frequently used than others. The added express channels have a utilization of 25 %, whereas the rest of links have a utilization of 10 %. Express channels are more frequently used because they provide shorter paths to destinations, while the other links are only used to deliver packets to their final switch (when the packet is one hop away from destination). So, traffic is not evenly distributed among all the links due to the different use of links, and therefore, ITB-RR does not evenly distribute traffic among all the links. So, the throughput increase when using in-transit buffers is slightly smaller than the one achieved in the 2-D Torus. Even though, the in-transit buffer mechanism (with the single path selection policy) increases the throughput achieved by up*/down* routing by a factor of 1.71.

Regarding the CPLANT network (Figure 7.c), ITB-SP almost doubles the network throughput achieved by UP/DOWN whereas ITB-RR doubles it. UP/DOWN saturates at 0.05 flits/ns/switch whereas ITB-RR saturates at 0.095 flits/ns/switch. CPLANT has a complex topology formed by groups of switches. When UP/DOWN is used as the routing algorithm, most of the traffic must cross the root switch (that is located in a certain group of switches), unbalancing traffic among all the groups. On the other hand, when using the in-transit buffer mechanism, traveling across the root switch is not needed, therefore distributing better the traffic among the groups. ITB-RR achieves better performance than ITB-SP because it uses different alternative paths to forward messages (unbalancing traffic even more). Regarding minimal paths, UP/DOWN always uses minimal paths in this topology, so the improvement achieved by in-transit buffers is only due to a more balanced traffic distribution.

4.7.2 Bit-Reversal Traffic Pattern

For the bit-reversal traffic pattern, similar results have been obtained. Figure 10 shows the performance results obtained for UP/DOWN, ITB-SP, and ITB-RR routing when using the bit-reversal traffic pattern. The figure shows results for the 2-D Torus, and for the 2-D Torus with express channels, respectively.

For the 2-D Torus (Figure 10.a), UP/DOWN throughput is almost doubled when using in-transit buffers (0.017 flits/ns/switch for UP/DOWN and 0.032 flits/ns/switch for ITB-RR). Again, when comparing ITB-SP and ITB-RR, ITB-RR incurs a slightly higher latency due to the use of more in-transit buffers on average but, on the other hand, it also increases network throughput.

For the 2-D Torus with express channels (Figure 10.b), the benefits of using in-transit buffers are slightly smaller (as for the uniform distribution). UP/DOWN saturates at 0.07 flits/ns/switch whereas ITB-RR saturates at 0.11 flits/ns/switch.

As for the uniform distribution, traffic distribution among all the links plays a key role in improving network performance. As soon as the root switch becomes congested when using UP/DOWN routing, the entire network saturates. The in-transit buffer mechanism allows the use of alternative paths without the constraint of crossing the root switch.

4.7.3 Hotspot

Ten different hotspot locations have been considered for each topology. Also different hotspot traffic loads have been

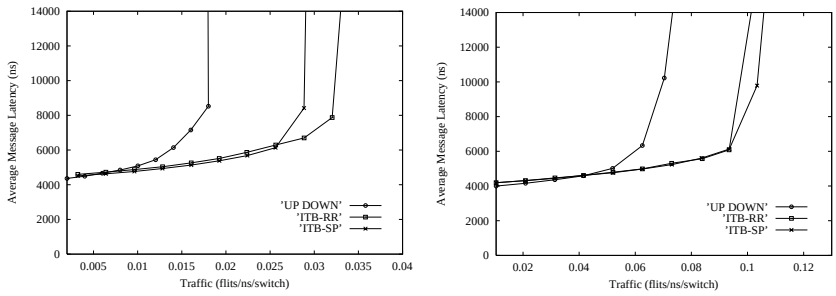


Figure 10. Performance results for the bit-Reversal traffic pattern. a) 2-D Torus, b) 2-D Torus with express channels.

used to model hotspot traffic. Table 1 shows the throughput achieved by each routing algorithm in the 2-D Torus network. In particular, we have used two hotspot traffic loads for the 2-D Torus: 5 % hotspot traffic and 10 % hotspot traffic. On average, ITB-SP and ITB-RR increase UP/DOWN throughput by a factor of 2.13 and 2.19, respectively, when 5 % hotspot traffic is used. On the other hand, for 10 % hotspot traffic, ITB-SP and ITB-RR decrease their performance. However, ITB-SP and ITB-RR still increase UP/DOWN throughput by a factor of 1.4 and 1.48, respectively.

From Table 1 we can observe that up*/down* routing is only slightly affected by the hotspot. For a uniform distribution, UP/DOWN routing reaches a network throughput of 0.015 flits/ns/switch. With a 5 % hotspot traffic, network throughput is decreased only by 16 % (0.0125 flits/ns/switch on average) and for 10 % hotspot traffic, network throughput is almost the same (on average, network throughput is 0.0123 flits/ns/switch). This is because the root switch behaves as a big hotspot and UP/DOWN saturates when the root switch is overloaded. Figure 11 shows the link utilization for UP/DOWN and ITB-RR routing for 10 % hotspot traffic when the saturation point is being reached by UP/DOWN (0.0123 flits/ns/switch). For UP/DOWN (Figure 11.a) links near the root switch are much more heavily used than links near the hotspot switch (marked as H). On the other hand, when using ITB-RR routing (Figure 11.b), only links near the hotspot switch start to saturate. So, for UP/DOWN routing, the root switch behaves as a big hotspot and saturates the network, whereas ITB-RR routing saturates due to the hotspot.

Table 2 shows the throughput achieved when using different hotspot locations and different hotspot traffic loads for the 2-D Torus with express channels. In particular, we have used 3 % hotspot traffic and 5 % hotspot traffic. ITB-SP increases UP/DOWN throughput by a factor of 1.13 and 1.08 for 3 % and 5 % hotspot traffic, respectively. On the other hand, ITB-RR increases UP/DOWN throughput by a factor of 1.12 and 1.07 for the hotspot traffic loads.

Taking into account the results obtained for the uniform distribution, the throughput achieved by UP/DOWN routing with a uniform distribution is reduced by 26 % for 3 % hotspot traffic, and by 49 % for 5 % hotspot traffic. For ITB-SP, the reduction of throughput achieved with respect to the uniform distribution is 50 % and 67 %, respectively. For ITB-RR, the reduction is 54 % and 70 %, respectively. So, ITB-SP and ITB-RR are more heavily affected by hotspots. Figure **??** shows the link utilization for UP/DOWN and ITB-RR routing at their saturation points (0.0483 flits/ns/switch for UP/DOWN and 0.0542 flits/ns/switch for ITB-RR) using 3 % hotspot traffic. It can be observed that due to the hotspot in UP/DOWN (Figure **??**.a), the root switch is less congested, but links between the root switch and the hotspot are much more heavily used than the other links in the network. For ITB-RR (Figure **??**.b) only links near the hotspot switch are heavily used. It can be observed that all the saturated links are express channels.

Finally, for the CPLANT network, Table 3 shows the throughput achieved by each routing algorithm for different hotspot locations and for 5 % hotspot traffic. On average, ITB-SP and ITB-RR routing increase network throughput by a factor of 1.24 and 1.32, respectively.

4.7.4 Local Distribution

Figure 12 shows the performance achieved by the routing algorithms when using a local distribution of messages. Message destinations are at most 3 switches away from their source. For the 2-D Torus (Figure 12.a), ITB-SP and ITB-RR obtain higher throughput than UP/DOWN. UP/DOWN saturates near 0.1 flits/ns/switch, whereas ITB-SP and ITB-RR saturate near 0.13 flits/ns/switch. For the 2-D Torus with express channels (Figure 12.b) UP/DOWN performs as ITB-RR. ITB-SP achieves slightly higher throughput than UP/DOWN. Also, for the CPLANT network (Figure 12.c)

	5 %			10 %		
Hotspot	U/D	ITB-SP	ITB-RR	U/D	ITB-SP	ITB-RR
1	0.0120	0.0264	0.0288	0.0120	0.0168	0.0168
2	0.0144	0.0240	0.0288	0.0121	0.0191	0.0168
3	0.0120	0.0287	0.0289	0.0145	0.0168	0.0192
4	0.0120	0.0289	0.0289	0.0120	0.0168	0.0192
5	0.0120	0.0241	0.0264	0.0120	0.0168	0.0193
6	0.0120	0.0312	0.0265	0.0120	0.0168	0.0193
7	0.0144	0.0241	0.0264	0.0120	0.0191	0.0193
8	0.0120	0.0241	0.0264	0.0120	0.0168	0.0191
9	0.0120	0.0290	0.0264	0.0120	0.0169	0.0168
10	0.0120	0.0265	0.0264	0.0120	0.0168	0.0168
Avg	0.0125	0.0267	0.0274	0.0123	0.0173	0.0183

Table 1. Throughput for different hotspot locations and different hotspot traffic. 2-D Torus.

	3 %			5 %		
Hotspot	U/D	ITB-SP	ITB-RR	U/D	ITB-SP	ITB-RR
Avg	0.0483	0.0546	0.0542	0.0334	0.0363	0.0359

Table 2. Throughput for different hotspot locations and different hotspot traffic. 2-D Torus with express channels.

	5 %		
Hotspot	U/D	ITB-SP	ITB-RR
Avg	0.0340	0.0423	0.0451

Table 3. Throughput for different hotspot locations. CPLANT network.

small benefits are obtained. In general, UP/DOWN achieves good results because traffic is evenly distributed among all the links in the network (due to the traffic pattern). Note that up*/down* routing is always able to use a minimal path when the destination is one hop away (two switches away) or is connected to the same switch. So, the benefits of using in-transit buffers are small. However, the in-transit buffer mechanism does not decrease UP/DOWN performance.

5 Conclusions

When performance is the primary concern, Myrinet products are being used to build large commodity clusters with regular topologies. In this paper we have evaluated the in-transit buffers mechanism for different networks with regular topologies and with different traffic patterns. Results show that network performance is always improved when using the in-transit buffer mechanism. For uniform and bit-reversal distributions, the in-transit buffer mechanism doubles the network throughput achieved by the original routing used in Myrinet (up*/down*). For hotspot distribution benefits are lower but still significant. For local distribution the in-transit buffer mechanism offers a low improvement. This mechanism is valid for any network with source routing. It can be implemented in Myrinet thanks to the flexibility offered by the MCP.

As for future work, we plan to implement the proposed mechanism on an actual Myrinet network in order to confirm the simulation results obtained. Also, we are working on reducing the latency overhead and on new route selection algorithms that implement some adaptivity at the source host.

References

[1] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. Seizovic and W. Su, "Myrinet - A gigabit per second local area network," *IEEE Micro*, pp. 29–36, February 1995.

[2] R. V. Bopana and S. Chalasani, "A comparison of adaptive wormhole routing algorithms," *Proc. 20th Annu. Int. Symp. Comp. Architecture*, May 1993.

[3] W.J. Dally, "Express Cubes: Improving the Performance of k-ary n-cube Interconnection Networks," in *IEEE Transactions on Computers*, Vol. 40, No. 9, September 1991.

[4] W. J. Dally, "Virtual-channel flow control," *IEEE Trans. Parallel Distributed Syst.*, vol. 3, no. 2, pp. 194-205, Mar. 1992.

[5] J. Flich, M.P. Malumbres, P.Lopez and J. Duato, "Improving Routing Performance in Myrinet Networks," accepted for the *International Parallel and Distributed Processing Symposium*, May 2000.

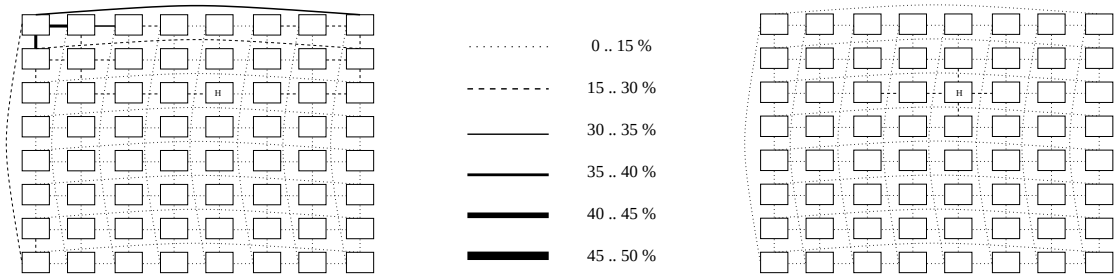
[6] J. Flich, M.P. Malumbres, P.Lopez and J. Duato, "Performance Evaluation of a New Routing Strategy for Irregular Networks with Source Routing," submitted to the *International Conference on Supercomputing*, May 2000.

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

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

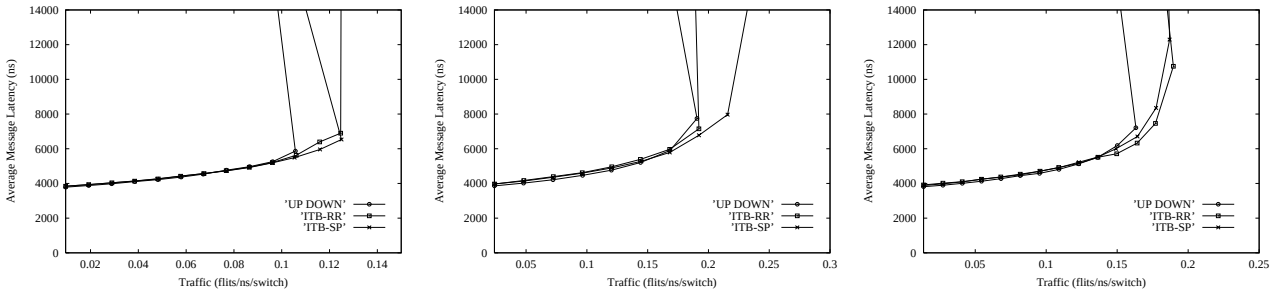
[9] J. Kim and A. Chien, "An evaluation of the planar/adaptive routing," in *Proc. 4th IEEE Int. Symp. Parallel Distributed Processing*, December 1992

[10] P. R. Miller, "Efficient comunications for fine-grain distributed computers," Ph.D. Thesis, Southampton University, 1991.



(a) UP/DOWN (b) ITB-RR

Figure 11. Link utilization in a 2-D Torus with hot-spot traffic at saturation point for UP/DOWN.



(a) (b) (c)

Figure 12. Performance results for the local distribution (destinations are at most 3 switches away). a) 2-D Torus, b) 2-D Torus with express channels, c) CPLANT

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

[12] R. Riesen et al., "CPLANT, " in *Proceedings of the Second Extreme Linux Workshop,* June 1999.

[13] 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.

[14] 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.

[15] R. Sheifert, "*Gigabit Ethernet,*" in Addison-Wesley, April 1998.