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

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

Abstract

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

In this paper, we analyze in detail the latency overhead of ITBs proposing several mechanisms in order to reduce, hide, and remove it. Firstly, we show by simulation the effect of an ITB implementation much slower than the one implemented. Then, we propose three mechanisms that will try to overcome the latency penalty. All the mechanisms are simple and can be easily implemented. Also they are out of the critical path of the ITB packet processing procedure. Results show a very good behavior of the proposed mechanisms, reducing considerably, and even removing the latency overhead.

1 Introduction

Clusters Of Workstations (COWs) are currently being considered as a cost-effective alternative for small and large-scale parallel computing. Usually, the interconnection network used is the local area network (LAN) all computers are attached to. When performance is the primary concern, a high performance network fabric is used. Myrinet network [1] is one of the most popular interconnect.

Topology is usually fixed by the physical location constraints of the computers, being the resulting topology typically irregular. On the other hand, either source or distributed routing may be used. In source routing, the path to destination is built at the source host and it is written into the packet header before delivery. Switches route packets through the fixed path found at the packet header. Myrinet uses wormhole switching and source routing.

Different source routing algorithms have been proposed, being the up*/down* the most well-known one. Other source routing algorithms, like DFS [8] and smart-routing [2], have been proposed to improve the performance of up*/down*. In [4] we evaluated these algorithms, identifying that the use of non-minimal paths, traffic unbalance and network contention are the three major factors that limit network performance. In [3] we proposed the in-transit buffers (ITB) mechanism for networks with source routing in order to reduce the impact of this factors on network performance. Basically, this mechanism avoids routing restrictions by ejecting packets at intermediate hosts, storing them in (in-transit) buffers and later re-injecting them into the network. Although it was originally proposed for up*/down* routing, it can be efficiently applied to any source routing algorithm [4]. An implementation of the ITB mechanism on Myrinet can be found in [5].

By avoiding routing restrictions, the ITB mechanism allows the use of minimal paths for every source-destination pair. Moreover, the mechanism allows a good traffic balance not achieved by routings based on spanning trees (up*/down* and DFS). This behavior is comparable to the smart-routing proposal in terms of load balancing but using a faster path-computation algorithm. Also, network contention can be reduced by using more ITBs than the strictly needed [6], showing their benefits when a high network throughput is required.

*This work was supported by the Spanish CICYT under Grant TIC2000–1151–C07.

2 Motivation

However, the use of ITBs adds some latency to those messages that use them. This overhead is proportional to the number of ITBs that a message uses to reach its destination. In [5] we measured this overhead in an unloaded network, showing that it may be too high, especially for short messages and at low traffic loads. This behavior forces us to design mechanisms to reduce, hide or even remove this latency overhead.

In this paper, we propose and evaluate three different mechanisms that will try to minimize and, if possible, remove the latency penalty of the ITB mechanism in different ways. So, with these techniques the will exploit the ability of the ITB mechanism to achieve a high network performance without the limiting factor of the latency penalty.

The rest of the paper is organized as follows. In Section 3 we present the proposed mechanisms to hide and remove the ITB latency overhead. In Section 4 evaluation results are presented, analyzing in detail the benefits of the proposals. Finally, in section 5 some conclusions are drawn and future work is anticipated.

3 Description of the Proposed Mechanisms

Basically, all the mechanisms will rely on the use of two paths for every source-destination pair. The way each of the two paths is computed will determine if ITBs are used and also how many ITBs will be used (if any). All three mechanisms will differ in the way the host selects the path to be used every time it needs to send a message. The remaining of this section describes the proposed mechanisms.

3.1 Using Fewer ITBs for Short Messages

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

We must ensure that using UD_MITB and UD_ITB together does not introduce cycles in the CDG. As the computation of both path sets is based on the up*/down* routing and the same root switch is used, using both sets together does not introduce cycles.

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

3.2 Reducing Latency by Using Fewer ITBs

The second mechanism will restrict the use of ITBs according to the desired overhead reduction. For instance, if we want to reduce the average latency penalty of the ITB mechanism by a 50%, we need to reduce the use of ITBs by a 50%. Therefore, each host will randomly select between two paths: one with ITBs and the other one without ITBs. By limiting the use of ITBs in this way, the latency overhead will be reduced to the half.

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

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

However, this mechanism will introduce some penalty to the overall performance in terms of network throughput. As stated earlier, the more ITBs the higher network throughput will be achieved. Therefore, if we limit the use of ITBs, the network throughput achieved will be also decreased.

3.3 Using ITBs only at Medium and High Network Loads

Finally, the third mechanism will remove the ITB latency penalty at low traffic loads. To do that, ITBs will be used only when network load is medium or high. Therefore, we need a mechanism at each host to measure network traffic. To keep this mechanism simple, it can only rely on local information to the host.

In particular, traffic load is locally estimated by means of the contention coefficient. Once a host is able to estimate network traffic, the proposed mechanism is simple. If the

contention coefficient exceeds a threshold then the host will begin to use paths with ITBs (as it considers network traffic is medium or high). On the other hand, if the contention coefficient decreases and goes down another threshold then the host will use paths without ITBs.

The contention coefficient is based on measuring the injection delay of each packet, which is computed as follows. When the host is going to send a packet, it stores the current time (T_{start}). Once the packet completely leaves the host, the current time is also stored (T_{end}). The injection delay is computed as $T_{end} - T_{start}$.

If there is no contention along the path, packet does not block and therefore flits can be smoothly injected into the network with a rate equal to the link bandwidth. Therefore, the minimum injection delay is equal to $\frac{P_{size}}{I_r}$, where I_r is the link bandwidth (in flits/cycle) and P_{size} is the packet size (in flits). On the other hand, if the packet finds contention along the path, the injection delay will be higher. Notice that if there is some contention but the packet leaves the host before stopping (i.e. a very short packet), the injection delay will be low, leading to a wrong estimation. However, if this situation remains, packets will be queued along the path and contention will finally reach the source host. So, although late, network contention is also detected.

---

procedure Compute_Contention_Coefficient
begin
  if $C_P > C_C$ *then*
    $C_C = \alpha * C_C + (1 - \alpha) * C_P$
  else
    $C_C = \beta * C_C + (1 - \beta) * C_P$
  endif
endp

Figure 1. Updating the contention coefficient.

The contention coefficient is computed after injecting a packet by comparing the actual time required to inject the packet (i.e. the injection delay) with the time needed without contention (i.e., the minimum injection delay):

$$C_P = \frac{T_{end} - T_{start}}{P_l * \frac{1}{I_r}} = I_r * \frac{(T_{end} - T_{start})}{P_l}$$

where I_r is the link bandwidth (in flits/cycle), T_{start} and T_{end} are the times that correspond to the start and the end of packet injection (in cycles), respectively and P_l is the packet length (in flits). Note that if $I_r = 1$ flits/cycle, in absence of contention, C_P will be equal to one. As there is some contention in the network, C_P becomes higher.

As more packets are injected into the network, the contention coefficient is dynamically updated, according to the procedure shown in Figure 1 where C_C is the current contention coefficient and C_P is the contention coefficient computed for the last packet sent. The α constant will determine the speed the contention coefficient adapts to higher traffic levels. On the other hand, the β constant will determine the speed the contention coefficient adapts to lower traffic levels. Both constants have values between 0 and 1. We use two different constants because we are interested in a fast contention detection mechanism (low α) in order to use ITBs as soon as possible. On the other hand, we are also interested in a slow transition (high β) of the contention coefficient from high to low values in order to avoid using up*/down* paths with medium or high traffic loads that would quickly congest the network.

We have chosen different thresholds to switch from UD to UD_ITB (1.5) and to switch from UD_ITB to UD (1.1)[1]. The threshold that determines the use of UD paths is lower than the one established to use UD_ITB paths. This is due to the fact that, when using UD_ITB paths network contention is reduced due to the natural behavior of the ITB mechanism (the mechanism ejects packets temporarily and later re-injects them). If we put a low threshold to switch to UD paths (1.1) then we ensure that we will switch to UD paths in the case traffic is low (and not due to the natural effect of the mechanism in reducing network contention).

Finally, the implementation of this mechanism only requires that each host store the contention coefficient, updating it every time it sends a packet. This computation can be performed in Myrinet out of the critical path of each packet, that is, at the same time packets are being injected.

4 Performance Evaluation

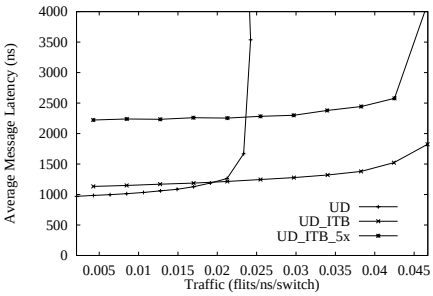
4.1 Network Model and Simulation Parameters

Network topologies are completely irregular and have been generated randomly, taking into account three restrictions: i) there are exactly 4 hosts connected to each switch; ii) all the switches have the same size (8 ports) and iii) two neighboring switches are connected by a single link. These assumptions have already been used in other evaluation studies [4].

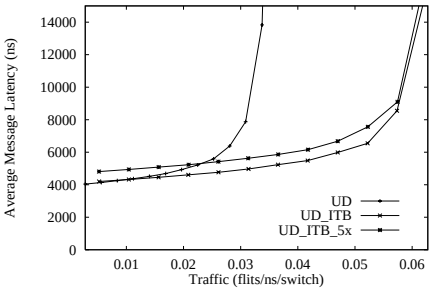
We will use different network sizes of 8, 16, 32, and 64 switches. To make results independent of the topology, we will evaluate 10 random topologies for each network size.

Links, switches, and interface cards are modeled based on the Myrinet network. Concerning links, we assume Myrinet short LAN cables (10 meters long, 160 MB/s, 4.92 ns/m (1.5 ns/ft)) [7]. Flits and physical links are one byte wide. Transmission of data across channels is pipelined [9]. Hence, a new flit can be injected into the physical channel every 6.25 ns and there will be a maximum of 8 flits on the link at a given time.

[1] Please note that we assume $I_r = 1$.

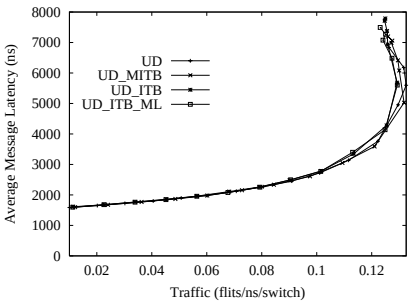


(a) 32-byte packets

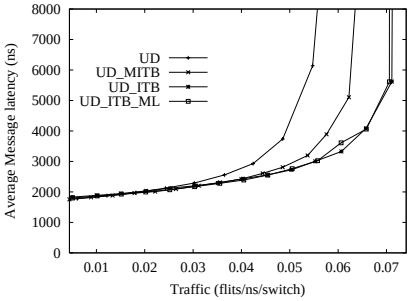


(b) 512-byte packets

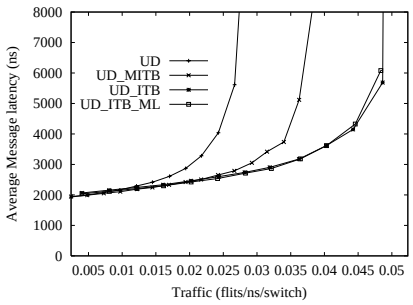
Figure 2. Impact of a Slow ITB Mechanism on Performance



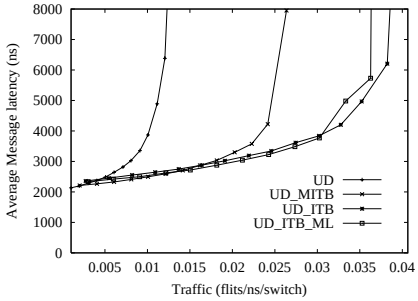
(a) 8-switch network



(b) 16-switch network



(c) 32-switch network



(d) 64-switch network

Figure 3. Using ITBs depending on message length. Bimodal traffic.

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.

Each Myrinet switch has a simple routing control unit. Assuming that the requested output link is free, the first flit latency is 150 ns through the switch. After that, the switch is able to transfer flits at the link rate (one flit every 6.25 ns). Each output port can process only one packet header at a time. An output port is assigned to waiting packets in a demand-slotted round-robin fashion. A crossbar inside the switch allows multiple packets to traverse it simultaneously without interference.

Each Myrinet network interface card has a routing table with one or more entries for every possible destination of messages. For each source-destination pair, different paths will be computed according to the mechanism being used.

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

For each simulation run, we assume that the packet generation rate is constant and the same for all the hosts. Once the network has reached a steady state, the generation rate is equal to the reception rate. We evaluate the full range of traffic, from low load to saturation. We use different message sizes (32 and 512 bytes), and in some evaluation experiments we will use bimodal traffic consisting of two different message lengths, short (32 bytes) and long (512 bytes) messages. Also, we will use a uniform distribution of message destinations in all cases.

4.2 Impact of a Slow ITB Mechanism on Performance

In order to see the impact of the ITB mechanism on latency, we evaluate in this section a slow ITB mechanism, five times slower the one used in previous studies [3, 4, 6]. A 32-switch topology with 32-byte and 512-byte packets are used. Figure 2 shows evaluation results for the UD_ITB routing with timing parameters of $T_d = 275$ ns and $T_r = 200$ ns and for the UD_ITB routing with timing parameters of $T_d = 1375$ ns and $T_r = 1000$ ns (referred to as UD_ITB_5x). The up*/down* routing (UD) is also plotted in order to see the improvement of the ITB mechanism.

As we can see, the latency overhead is significant, spe-

	UD_MITB		UD_ITB		UD_ITB_ML	
Sw	Min/Max	Avg	Min/Max	Avg	Min/Max	Avg
8	0.96/1.08	1.01	0.93/1.21	1.01	0.95/1.21	1.02
16	1.05/1.29	1.17	1.20/1.47	1.33	1.11/1.46	1.31
32	1.36/2.03	1.59	1.66/2.70	2.10	1.65/2.72	2.07
64	1.99/2.40	2.11	2.70/3.50	2.94	2.59/3.31	2.87

Table 1. Factor of throughput increase (UD_MITB, UD_ITB, UD_ITB_ML vs UD). Bimodal traffic.

	UD_MITB		UD_ITB		UD_ITB_ML	
Sw	Min/Max	Avg	Min/Max	Avg	Min/Max	Avg
8	0.7/1.0	0.9	1.2/4.1	1.9	0.8/1.7	1.3
16	-1.6/0.6	-0.6	1.6/5.8	3.5	0.3/3.0	1.6
32	0.8/2.1	1.3	6.7/8.2	7.4	4.6/6.3	5.3
64	2.6/4.6	3.3	9.4/12.2	10.8	7.6/9.5	8.2

Table 2. Percentage of latency increase (UD_MITB, UD_ITB, UD_ITB_ML vs UD). Bimodal traffic.

cially for short messages. In particular, average packet latency is doubled when using a slow ITB mechanism. However, network throughput is not affected. This is due to the fact that a slow mechanism still has the advantages of supplying minimal paths, balancing traffic and, even more, reducing network contention. Therefore, the quickness of the mechanism only affects the latency of messages. Finally, as we can see, the impact of a slow ITB mechanism on larger packets (512 bytes) is less important.

4.3 Using ITBs Depending on Message Length

We evaluate the first mechanism to reduce the latency overhead with a bimodal traffic composed of a 30% of long packets (512 bytes) and 70% of short packets (32 bytes). The new routing algorithm will be referred to as UD_ITB_ML (*Up*/Down* with ITBs considering Message Length*). This routing algorithm is compared against the UD, UD_MITB, and UD_ITB routings. Figure 3 shows the evaluation results for 8, 16, 32, and 64-switch networks.

As we can see, the UD_ITB_ML obtains the same network throughput of UD_ITB, except for the 64-switch network where it is slightly decreased. The UD_ITB_ML routing uses paths from UD_MITB and from UD_ITB, therefore its network throughput will lie between the two ones.

Table 1 shows minimum, maximum, and average factors of throughput increase when using UD_MITB, UD_ITB, and UD_ITB_ML with respect to the UD routing. On the other hand, Table 2 shows the minimum, maximum, and average percentages of latency increase of the UD_MITB, UD_ITB, and UD_ITB_ML routings with respect to UD for a low traffic condition. As we can see, the UD_ITB_ML latency overhead is lower than the one obtained by the UD_ITB routing but it is still noticeable.

Therefore, with the first proposed mechanism, the latency overhead is slightly decreased with respect to UD_ITB. However, it depends mainly on the percentage of long messages used in the system. If all the messages were short, network throughput would be drastically reduced and would be the same the UD_MITB routing obtains.

	UD_ITB			UD_ITB_50		
Sw	Min	Max	Avg	Min	Max	Avg
8	0.96	1.36	1.03	0.96	1.22	1.02
16	1.20	1.65	1.39	1.08	1.31	1.19
32	1.69	2.88	2.30	1.33	1.60	1.49
64	2.84	3.76	3.22	1.53	1.75	1.66

Table 3. Factor of throughput increase (UD_ITB and UD_ITB_50 vs. UD). 32-byte messages.

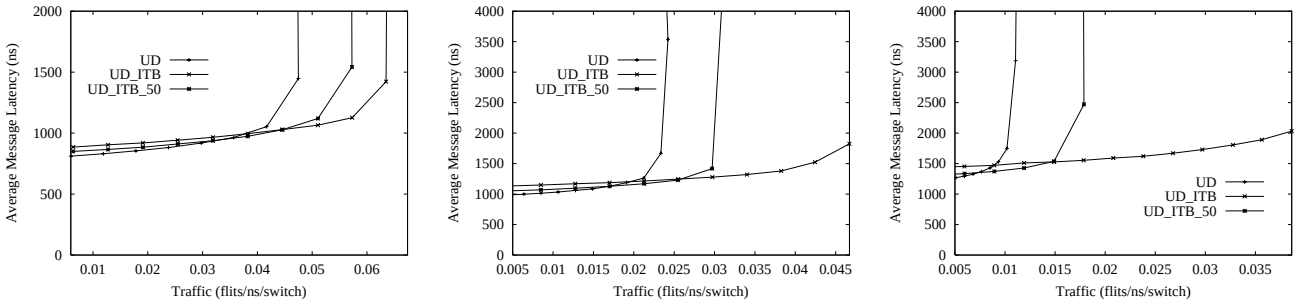
4.4 Reducing Latency by Using Fewer ITBs

In this section we evaluate the second proposed mechanism. We fix the percentage of paths that will use ITBs in 50%. Therefore, every time a host decides to send a packet it will randomly choose one from two paths (UD and UD_ITB). The resulting routing will be referred to as UD_ITB_50. Figure 4 shows the evaluation results for UD, UD_ITB, and UD_ITB_50 routings in 16, 32, and 64-switch networks. Message size is 32 bytes.

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

Tables 3 and 4 show the factor of throughput increase and the percentage of latency increase, respectively. As expected, on average, the UD_ITB_50 routing decreases UD_ITB latency penalty by a half. However, the network throughput increases are much lower than the ones obtained by UD_ITB.

Therefore, the UD_ITB_50 effectively reduces latency penalty but paying a high price in terms of throughput decrease.



(a) 16-switch network (b) 32-switch network (c) 64-switch network

Figure 4. Reducing latency by using fewer ITBs. 32-byte messages.

	UD_ITB			UD_ITB_50		
Sw	Min	Max	Avg	Min	Max	Avg
8	0.07	8.35	2.85	0.07	4.14	1.46
16	7.71	17.46	10.84	3.84	8.19	5.24
32	15.29	19.13	16.78	7.39	8.66	7.97
64	16.72	21.64	19.90	8.23	10.91	9.63

Table 4. Percentage of latency increase (UD_ITB and UD_ITB_50 vs. UD). 32-byte messages.

4.5 Using ITBs only on Medium and High Network Loads

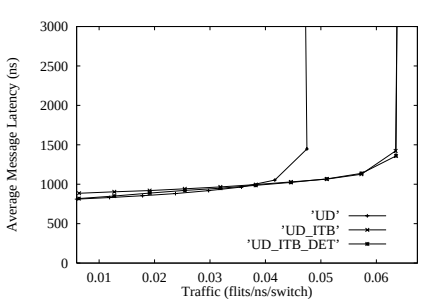
Finally, in this section we evaluate the last mechanism. We fix the parameters for computing the contention coefficient to $\alpha = 0.7$ and $\beta = 0.95$. Therefore, the mechanism will adapt faster to high traffic loads than to low ones ($\alpha < \beta$). We also fix the thresholds to 1.5 and 1.1 to start using and not using ITBs (assuming $I_r = 1$). The resulting routing algorithm will be referred to as UD_ITB_DET. This routing will be compared to UD and UD_ITB.

Figure 5 shows the performance results obtained for the UD, UD_ITB, and UD_ITB_DET routings for different network and message sizes. In all cases, the UD_ITB_DET routing algorithm obtains the same latency as the UD does at low traffic loads. As traffic increases, the behavior of UD_ITB_DET in latency moves from the latencies of UD to the latencies of UD_ITB. However, the UD_ITB_DET routing obtains lower latencies than the UD_ITB routing does. Moreover, the UD_ITB_DET routing obtains the same network throughput as UD_ITB does. On the other hand, for longer messages (Figure 5.c) we can see that the behavior of the UD_ITB_DET routing is roughly the same of the UD_ITB routing. Therefore, with this mechanism we have obtained a low latency overhead at low traffic loads without losing network throughput performance.

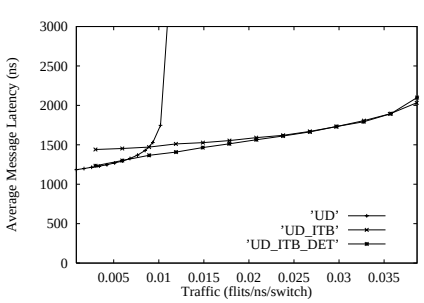
Figure 6 shows the number of ITBs used at each traffic point for different network sizes and for 32 and 512-byte messages. For UD, the number of ITBs is 0 as it does not use any ITB. The number of ITBs used by UD_ITB is always the same. However, as it can be seen, the number of ITBs used by the UD_ITB_DET algorithm increases as traffic injection increases. For instance, for the 64-switch network (Figure 6.c) at low traffic points, the number of ITBs used is low and less than 5000 (less than 10% of ITBs used by UD_ITB). Therefore, for this traffic load, the average latency increase with respect to UD is very low and near to zero. The higher the traffic the more contention is found and therefore, more ITBs are used. We can see that the number of ITBs grows quickly in all networks and finally reaches the number of ITBs used by UD_ITB at high traffic loads.

For other topologies, Table 5 shows the factor of throughput increase when using the UD_ITB_DET and UD_ITB with respect to UD for different network and message sizes. We can see that the network throughput is not decreased when using the UD_ITB_DET. Therefore, the good behavior achieved by UD_ITB is not compromised with the new mechanism (UD_ITB_DET). Table 6 shows the percentage of latency increase at low traffic loads when using UD_ITB_DET and UD_ITB with respect to UD. We can observe that the high latency overhead exhibited by UD_ITB with short messages is reduced by UD_ITB_DET in all networks and always is less, on average, than 3.6%, which seems negligible.

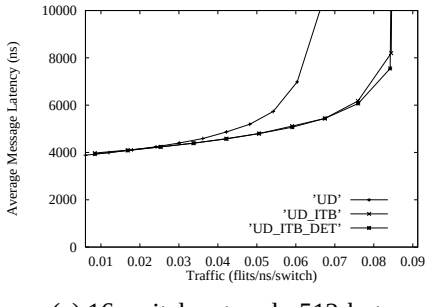
In order to analyze the dynamic behavior of the mechanism, we have also evaluated network performance after a change in its load. Network load is set to a low level (0.001 flits/ns/host) for 500000 clock cycles. Then, it is sharply changed to a load beyond saturation (0.033 flits/ns/host) for 700000 clock cycles. Afterwards, load is reduced again to the low level. Simulation finishes when 500000 messages have been delivered. Network size is 32 switches and message length is 32 bytes. With this experiment, we want to analyze if the detection mechanism is able to detect in an efficient way the changes in network traffic and therefore



(a) 16-switch network. 32-byte packets.

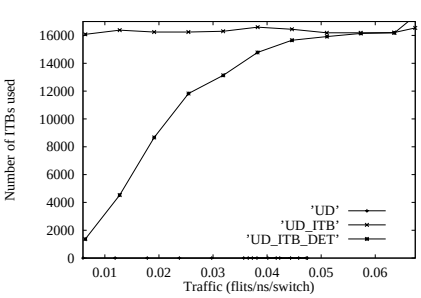


(b) 64-switch network. 32-byte packets.

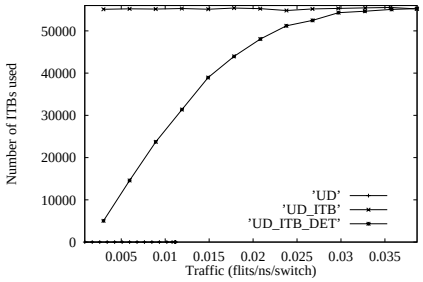


(c) 16-switch network. 512-byte packets.

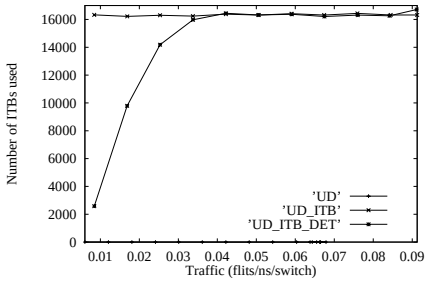
Figure 5. Using ITBs only on medium and high network loads.



(a) 16-switch network. 32-byte packets.



(b) 64-switch network. 32-byte packets.



(c) 16-switch network. 512-byte packets.

Figure 6. Using ITBs only on medium and high network loads. Number of ITBs used.

increase or decrease the number of ITBs used. We have evaluated both the UD and UD_ITB_DET routing algorithms.

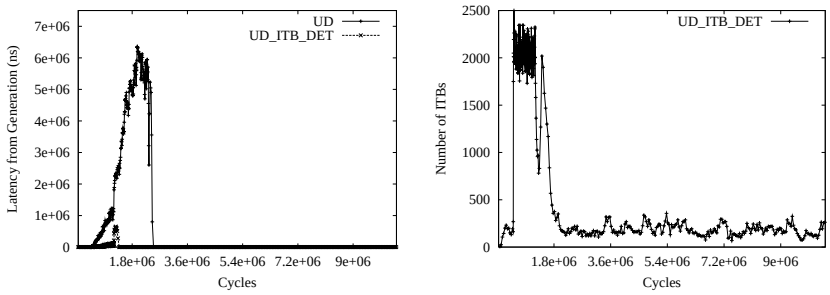
Figure 7.a shows the evolution of the average message latency (including the time spent in the source queue). As we can see, the UD routing is not able to manage all the messages generated inside the high traffic pulse. Indeed, the UD routing needs much more time to deliver all the generated messages. On the other hand, the UD_ITB_DET routing is able to efficiently handle the high traffic pulse. The maximum average message latency is almost 12 times lower than UD. This is due to the ability of the UD_ITB_DET routing to increase the number of ITBs used and thus reduce contention. In Figure 7.b we can observe the average number of ITBs used every 1000 messages. As we can see, this number increases sharply when congestion is detected. We can also observe that the number of ITBs decreases when congestion is not longer detected.

5 Conclusions

In previous papers [3, 4] we presented the in-transit buffer mechanism (ITB) to improve network performance of COWs with source routing. The main drawback of the ITB mechanism is that it introduces some penalty in average message latency. This overhead is especially noticeable for short messages and at low network loads.

In this paper we have proposed three mechanisms to considerably reduce or even remove this latency penalty.

In the first mechanism, the use of ITBs is restricted when sending short messages. Although it reduces the latency overhead of the ITB, its effectiveness highly depends on the message lengths. The second mechanism restricts the utilization of ITBs. Although latency overhead is reduced, the network throughput achieved by this mechanism is also decreased. Finally, a third mechanism has been proposed. With this mechanism, ITBs are used only for medium and high traffic loads. The mechanism relies on a contention coefficient computed at each host using only local information. From this coefficient each host figures out the network



(a) (b)

Figure 7. Using ITBs only on medium and high network loads. Dynamic behavior.

		UD_ITB_DET			UD_ITB		
Sw.	Msg.	Min	Max	Avg	Min	Max	Avg
8	32	1.0	1.2	1.0	1.0	1.3	1.0
16	32	1.2	1.6	1.4	1.2	1.6	1.4
32	32	1.7	2.9	2.3	1.7	2.9	2.2
64	32	2.8	3.7	3.2	2.8	3.5	3.1
8	512	0.9	1.2	1.0	0.9	1.2	1.0
16	512	1.2	1.6	1.4	1.3	1.6	1.4
32	512	1.5	2.5	2.0	1.7	2.5	2.1
64	512	2.5	3.5	2.9	2.3	3.5	2.9

Table 5. Factor of throughput increase (UD_ITB_DET and UD_ITB vs. UD).

		UD_ITB_DET			UD_ITB		
Sw.	Msg.	Min	Max	Avg	Min	Max	Avg
8	32	0.07	0.62	0.32	0.07	8.35	2.85
16	32	0.67	1.89	1.04	7.71	17.46	10.84
32	32	2.26	3.18	2.56	15.29	19.13	16.78
64	32	2.88	4.22	3.51	16.72	21.64	19.90
8	512	0.62	0.88	0.72	0.63	2.34	1.18
16	512	0.97	1.64	1.24	2.06	3.85	2.61
32	512	2.02	2.38	2.17	3.23	4.10	3.70
64	512	2.10	2.60	2.33	3.80	4.72	4.39

Table 6. Percentage of latency increase for low traffic (UD_ITB_DET and UD_ITB vs. UD).

load. ITBs are only used when it surpasses some threshold value. The evaluation of this mechanism have shown that latency penalty is practically eliminated (4% of latency overhead at most at low loads), without compromising at all the good network throughput achieved by the ITB mechanism. As a consequence, this mechanism effectively enables the use of the ITB mechanism.

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

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 *29th Int. Conf. on System Sciences*, Feb. 1995.

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

[4] J. Flich, P.Lopez, M.P. Malumbres, J. Duato, and T. Rokicki, "Combining In-Transit Buffers with Optimized Routing Schemes to Boost the Performance of Networks with Source Routing," in *Int. Symp. on High Performance Computing 2000*, Oct. 2000.

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

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

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

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

[9] 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.