Congestion Control Based on Transmission Times*

E. Baydal, P. López, and J. Duato

Dept. of Computer Engineering, Universidad Politécnica de Valencia,
Camino de Vera s/n, 46071 – Valencia, Spain,
elvira,plopez,jduato@disca.upv.es

Abstract. Congestion leads to a severe performance degradation in multiprocessor interconnection networks. Therefore, the use of techniques that prevent network saturation are of crucial importance to avoid high execution times.
In this paper, we propose a new mechanism that uses only local information to avoid network saturation in wormhole networks. In order to detect congestion, each network node computes the quotient between the real transmission time of messages and its minimum theoretical value. If this ratio is greater than a threshold, the physical channel used by the message is considered congested. Depending on the number of congested channels, the available bandwidth to inject messages is reduced.
The main contributions of the new mechanism are three: i) it can detect congestion in a remote way, but without transmitting control information through the network; ii) it tries to dynamically adjust the effective injection bandwidth available at each node; and iii) it is starvation-free.
Evaluation results show that the proposed mechanism avoids network performance degradation for different network loads and topologies. Indeed, the mechanism does not introduce any penalty for low and medium network loads, where no congestion control mechanism is required.

1 Introduction

Massively parallel computers provide the performance that most scientific and commercial applications require. Their interconnection networks offer the low latencies and high bandwidth that is needed for different kinds of traffic. Usually, wormhole switching with virtual channels and adaptive routing is used [6]. However, multiprocessor interconnection networks may suffer from severe saturation problems with high traffic loads, which may prevent reaching the wished performance.

This problem can be stated as follows. With low and medium network loads, the accepted traffic rate is the same as the injection rate and latency slightly increases due to contention. When traffic injection rate exceeds certain level (the network saturation point), accepted traffic falls and message latency increases considerably. Notice that both latency and accepted traffic are dependent variables on injected traffic. When this situation is reached, we say that the interconnection network is congested. Performance degradation due to congestion is specially important when messages stay in the network in case of contention, which is the case of wormhole.

* This work was supported by the Spanish CICYT under Grants TIC2000-1151-C07 and 1FD97-2129

B. Monien and R. Feldmann (Eds.): Euro-Par 2002, LNCS 2400, pp. 781–790.

Message throttling has been the most frequently used method to avoid network congestion. Several mechanisms have been already proposed, they have important drawbacks that will be analyzed in Section 2.

In this paper, we propose and evaluate a new mechanism to prevent network congestion that tries to overcome those drawbacks. The mechanism is based on locally estimating network traffic by using message transmission times and applying message throttling when congestion is presumed. The rest of the paper is organized as follows. In order to design an efficient mechanism, in Section 2 we describe what is expected from a good congestion control mechanism, criticizing previous approaches. Section 3 describes the new proposal. Performance evaluation results are presented in Section 4. Finally, some conclusions are drawn.

2 Features of a Good Congestion Control Mechanism

In order to design an efficient congestion control mechanism, in this section, we will describe the desirable features of such a mechanism.

First, the mechanism should be robust. As saturation point depends on network load and topology, a given mechanism may not always work properly. However, many of the previously proposed mechanisms have been analyzed for only one network size [1], [16] and for the uniform distribution of message destinations [4], [8], [7], [14] or do not achieve good results for different traffic patterns [15].

Second, the mechanism should not penalize network behavior when the network is not saturated, which is the most frequent situation [13]. However, some of the previous proposals increase message latency before the saturation point [4], [15], [7].

Finally, the new mechanism should not complicate network design. Some of the proposed mechanisms increase network complexity by adding new signals [14], [8] or even a sideband network [16], [15]. Others need to send extra information through the network [8].

3 Congestion Detection Based on Measuring Packets Transmission Time

This section describes the new mechanism proposed in this paper.

First, let us analyze the effect of traffic rate on message latency. With low traffic rate, messages flow smoothly through the network, only suffering the routing and switching delays at each traversed node. Indeed, physical channels will be seldom multiplexed into virtual channels.

As traffic increases, the probability that the message find busy channels also increases, thereby, multiplexing physical channels and slowing down message advance speed. Indeed, once one of the channels used by the message is multiplexed, the whole message advances at a lower speed. In this situation, message latency may increase until v times, v being the maximum number of virtual channels per physical channel.

If traffic continues to increase, the contention for the use of channels gets worse. Therefore, all the possible virtual output channels for reaching a given destination may

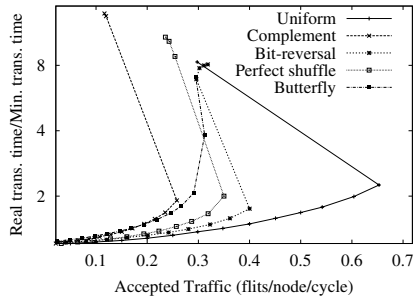

Fig. 1. Quotient between actual and minimum message transmission time versus accepted traffic. 8-ary 3-cube (512 nodes). TFAR routing with 3 virtual channels per physical channel and deadlock recovery. 16-flit messages.

be busy. In this situation, the message header stops, preventing also the advance of the remaining flits of the message. Channel buffers may allow data flits to continue flowing for some time, until they become full. In this case, message latency may considerably increase because of the combined effects of header contention and physical channel multiplexing.

Therefore, with low network load, message latency will be close to the minimum theoretical value, and it will increase as network load does. The mechanism proposed in this paper uses this idea in order to detect network congestion. In particular, the proposed method computes, for each virtual channel, the elapsed time between the arrival of a message header to the node and the transmission of its tail flit. This is the actual transmission time. The minimum value (what we call the theoretical transmission time) is the product of message size and the time required to transfer one flit across the link. Although the absolute value of actual transmission time depends on message destination distribution and message length, the quotient between this value and its minimum theoretical time does not strongly depend on network load, as Figure 1 shows. Indeed, the obtained values when the network is near to the saturation point are quite similar (around 2.0) for all the destination distributions considered. On the other hand, to make the mechanism independent on message length, this ratio is computed periodically, every time the time required to transfer a given number of flits (for instance 8 flits) has elapsed.

This ratio may be used to detect network congestion. When it is greater than a given threshold $u1$, the message using the virtual channel is having problems to advance, then we mark the physical channel containing the virtual channel as congested during some interval $u2$.

The information about physical channel status -congested or not- is used when new messages have to be injected into the network. Only the status of useful channels[1] will be considered. In particular, if any of the useful channels to route the new message is congested, message throttling will be applied. On the other hand, the proposed mecha-

[1] A physical channel is useful to route a message if it is a minimal route between the current node and the destination node

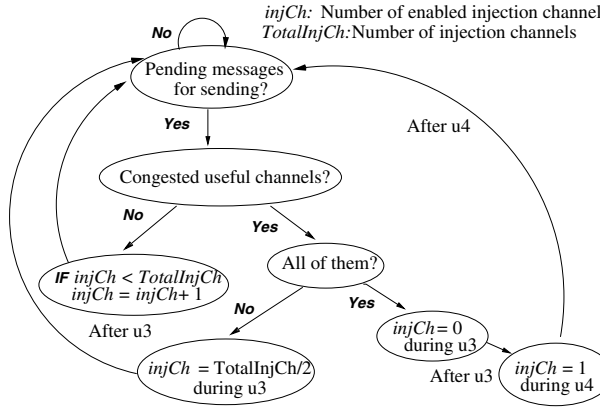

Fig. 2. Operation of the congestion control mechanism based on message transmission time.

nism tries to consider the saturation level of the network, in order to apply congestion control measures in a step-by-step fashion. In particular, different restrictions are applied depending on the number of congested useful channels, reducing accordingly the number of injection channels[2]. If some of the useful physical channels are congested (but not all of them), we reduce the number of injection channels by half (e.g. from 4 to 2 injection channels) during an interval time $u3$. If all the useful channels are congested, the measures have to be more restrictive. In this case, newly generated messages cannot be injected during $u3$ cycles. After that the performed actions depend on the level of detected congestion. In the former case (soft congestion), the status of useful physical channels is analyzed again, proceeding in the same way as we described. In the latter case (serious congestion), we will enable one injection channel during an interval $u4$, regardless of the network status, in order to prevent starvation. After $u4$, if there are pending messages in the injection queue, we will analyze again the status of the useful physical channels. Finally, if congestion is not longer detected, the number of injection channels is progressively increased. Figure 2 shows the behavior of the mechanism.

The proposed mechanism has the advantage of being able to detect the congestion in a remote way, but by using only local information. Moreover, all the nodes that are along the path followed by the message that finds congestion will detect the problem. The sender node of the message may do not detect the problem if the message tail has already left the node when the header flit reach the congested area. However, if the congestion situation is persistent, the next sent messages will every time increase their transmission times in nearer areas. Hence, the sender node will also detect the problem.

4 Evaluation

In this section, we will evaluate by simulation the behavior of the proposed congestion control mechanism. The evaluation methodology used is based on the one proposed in

[2] This is equivalent to reduce the bandwidth associated to message injection at each node

[5]. The most important performance measures are latency (time required to deliver a message, including the time spent at the source queue) and throughput (maximum traffic accepted by the network). Accepted traffic is the flit reception rate. Latency is measured in clock cycles, and traffic in flits per node per cycle.

4.1 Network Model

Message injection rate is the same for all nodes. Each node generates messages independently, according to an exponential distribution. Destinations are chosen according to the *Uniform, Butterfly, Complement, Bit-reversal,* and *Perfect-shuffle* traffic patterns, which illustrate different features. Uniform distribution is the most frequently used in the analysis of interconnection networks. The other patterns take into account the permutations that are usually performed in parallel numerical algorithms [9]. For message length, 16-flit and 64-flit messages are considered.

The simulator models the network at the flit level. Each node has a router, a crossbar switch and several physical channels. Routing time, transmission time across the crossbar and across a channel are all assumed to be equal to one clock cycle. Each node has four injection/ejection channels. Although most commercial multiprocessors have only one injection/ejection channel, previous works [8], [6], [2] have highlighted that the bandwidth available at the network interface may be the bottleneck to achieve a high network throughput.

Concerning deadlock handling, we use software-based deadlock recovery [12] and a True Fully Adaptive routing algorithm (TFAR) [13,12] with 3 and 4 virtual channels per physical channel. This routing algorithm allows the use of any virtual channel of those physical channels that forwards a message closer to its destination. In order to detect network deadlocks, we use the mechanism proposed in [10] with a deadlock detection threshold equal to 32 cycles.

We have evaluated the performance of the proposed congestion control mechanism on a bidirectional k-ary n-cube. In particular, we have used the following network sizes: 256 nodes (n=2, k=16), 512 nodes (n=3, k=8), and 4096 nodes (n=3, k=16).

4.2 Performance Comparison

In this section, we will analyze the behavior of the mechanism proposed in section 3, which will be referred to as **Trans-t** (*Transmission time*). For comparison purposes, we will also evaluate the behavior of the mechanism proposed in [16], which will be referred to as **Self-Tuned**. In this mechanism, nodes detect network congestion by using global information about the total number of full buffers in the network. If this number surpasses a threshold, all nodes apply message throttling. The use of global information requires to broadcast data among all the network nodes. A way of transmitting this control information is to use a sideband network with a far from negligible bandwidth [16]. To make a fair comparison, as the mechanism proposed in this paper does not need to exchange control messages, the bandwidth provided by the sideband network should be considered as additional available bandwidth in the main interconnection network. However, in the results that we present we do not consider this fact. If this additional bandwidth were considered, the differences, not only in throughput but also in latency,

between **Self-Tuned** and the new mechanism would be greater than the ones shown. Moreover, results without any congestion control mechanism (**No-Lim**) are also shown.

First of all, the **Trans-t** mechanism has to be tuned. We have found that the most critical threshold is $u1$ (the threshold used for comparing the quotient of transmission times). As once congestion is detected the injection bandwidth is dynamical adjusted, the other thresholds have a high tolerance variations. The other thresholds tolerate bigger variations. In the case of $u2$ and $u4$, if they are very high, congestion control measures may be applied more time than needed. Therefore, if the network is not longer congested, message latency increases. On the other hand, if they are too low, the status of physical channel is unnecessarily checked many times. Concerning $u3$, if it is too low, it does not apply enough injection limitation restrictions to reduce congestion and the mechanism does not work. After several experiments, the thresholds $u2$, $u3$, and $u4$ have been established in 8, 16 and 8 clock cycles, respectively.

With respect to threshold $u1$, we used a value equal to the number of virtual channels per physical channel as the starting point. The explanation is simple. When congestion appears for the first time, many physical channels will be completely multiplexed in virtual channels. Therefore, the message transmission time across each virtual channel will be roughly equal to the theoretical minimum value multiplied by the multiplexing degree. However, we must consider also that physical channels may not be fully multiplexed and the contention experimented by message header. Hence, the optimal value for this threshold can slightly change. The results show that, for a given network topology, the same value of threshold $u1$ is well suited for different message destinations and message sizes. The design factors that impact threshold $u1$ are the topology radix (k), and the number of virtual channels per physical channel. For a given k-ary n-cube, the optimal threshold increases with the number of virtual channels. On the contrary, when k increases, the optimal threshold decreases. The justification is simple. The higher the number of virtual channels per physical channel, the lower the network congestion, thereby injection policy can be more relaxed. On the contrary, increasing k (the number of nodes per dimension) without increasing the number of dimensions, there is a higher number of paths which shared links among them, which exacerbates congestion. Hence, a more strict injection policy is required (a lower threshold should be used).

Figure 3 shows the average message latency versus traffic for different $u1$ threshold values for the uniform and perfect-shuffle traffic patterns for a 8-ary 3-cube (512 nodes). As it can be seen, the lowest threshold values lead to apply more injection limitation than necessary. As a consequence, message latency is increased due to the fact that messages are waiting at the source nodes. On the other hand, the highest threshold value allows a more relaxed injection policy and trends saturating the network. In this case, a good $u1$ threshold value is 4. Table 1 shows the optimal thresholds found for other topologies and number of virtual channels.

Once tuned the new mechanism, we can compare it with other proposals. Figures 4 through 7 show some of the results obtained, for different message destination distributions, network and message sizes.

In all the cases, simulations finish after receiving 500,000 messages, but only the last 300,000 ones are considered to calculate average latencies. As we can see, the new mechanism (**Trans-t**) avoids the performance degradation in all the cases. Indeed, it always

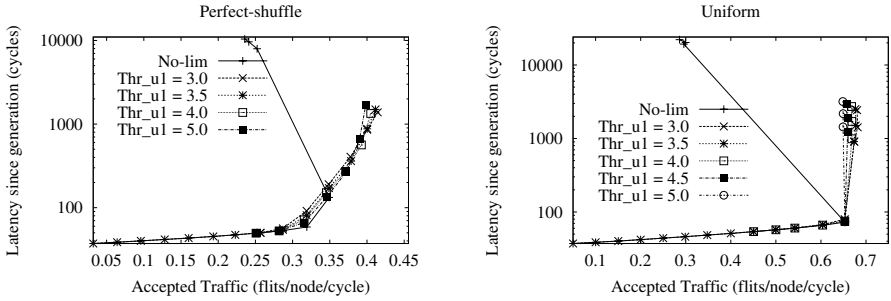

Fig. 3. Average message latency vs. accepted traffic for different $u1$ threshold values for the **Trans-t** mechanism. 8-ary 3-cube (512 nodes). 16-flit messages. 3 virtual channels per physical channel.

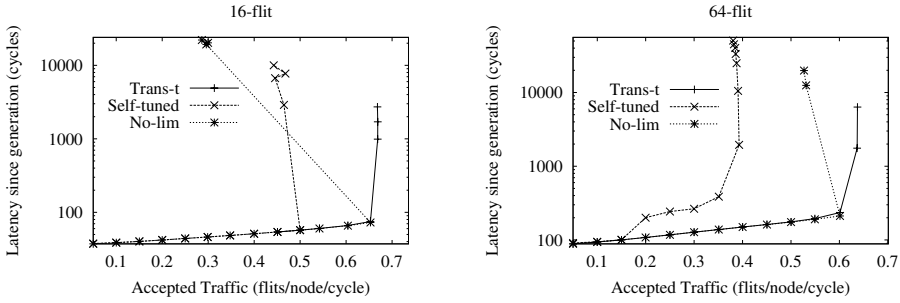

Fig. 4. Average message latency vs. accepted traffic. Uniform distribution of message destinations. 8-ary 3-cube (512 nodes). 3 virtual channels per physical channel. $u1 = 4$.

Table 1. Optimal $u1$ threshold values for different topologies and number of virtual channels per physical channel.

Nodes	Nr. of vc	$u1$	Nr. of vc	$u1$
512 ($8 \times 8 \times 8$)	3	4	4	5
256 (16×16)	3	2.5	4	3.75
4096 ($16 \times 16 \times 16$)	3	2.25	4	3.5
1024 (32×32)	3	2	4	3.25

improves network performance by increasing the throughput achieved when no congestion control mechanism is used. On the other hand, although the **Self-Tuned** mechanism helps in alleviating network congestion, it strongly reduces network throughput and increases network latency with low and medium loads.

We have also used a bursty load that alternates periods of high message injection rate, with periods of low traffic. In this case, we inject a given number of messages into the network and simulation goes on until all messages arrive to their destinations. Figure 8 shows the results for a 2-ary 16-cube (256 nodes) with a uniform distribution

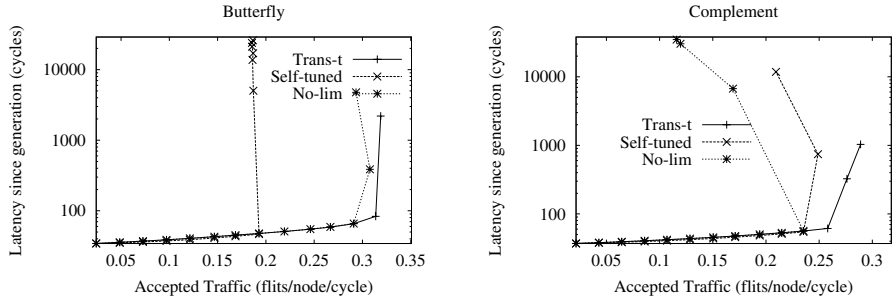

Fig. 5. Average message latency vs. accepted traffic. 16-flit messages. 8-ary 3-cube (512 nodes). 3 virtual channels per physical channel. $u1 = 4$.

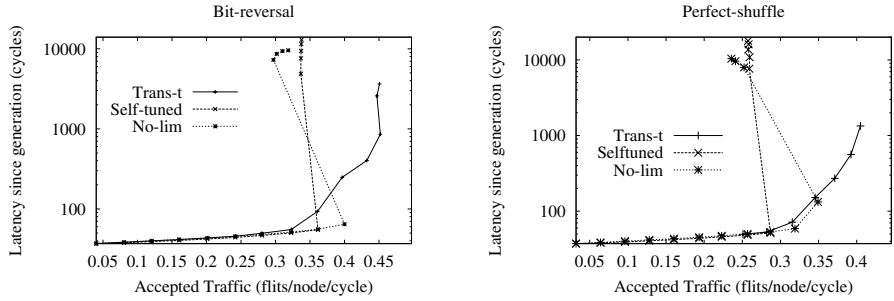

Fig. 6. Average message latency vs. accepted traffic. 16-flit messages. 8-ary 3-cube (512 nodes). 3 virtual channels per physical channel. $u1 = 4$.

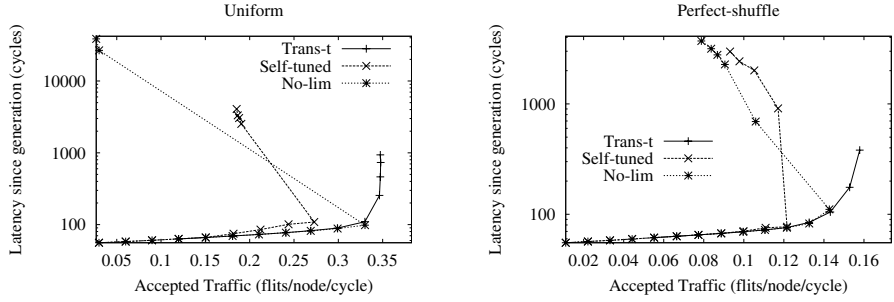

Fig. 7. Average message latency vs. accepted traffic. 16-flit messages. 16-ary 3-cube (4096 nodes). 3 virtual channels per physical channel. $u1 = 2.25$.

of message destinations, 3 virtual channels per physical channel, 400,000 messages generated at a rate of 0.34 flits/node/cycle (high load period) and 200,000 messages at 0.23 flits/node/cycle. These loads are applied alternatively twice. As we can see, with the **Trans-t** mechanism the network accepts the injected bursty traffic without problems.

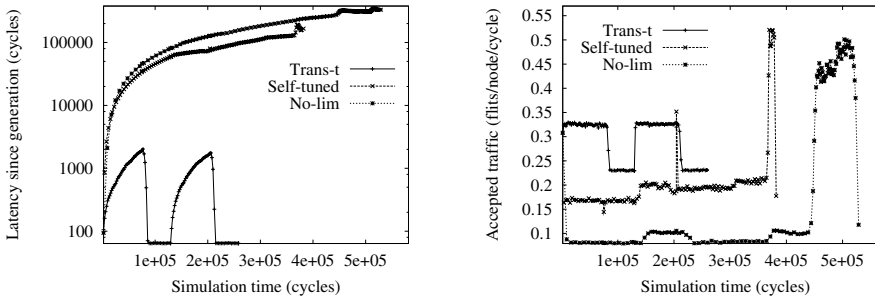

Fig. 8. Performance with variable injection rates. Uniform distribution of message destinations. 16-flits messages. 16-ary 2-cube (256 nodes), 3 virtual channels per physical channel. $u1 = 2.5$.

On the contrary, when no congestion control mechanism is applied, as soon as the first burst is applied into the network, congestion appears. As a consequence, latency strongly increases and accepted traffic falls down. Lately, after some time injecting at the low rate, network traffic starts recovering but the arrival of a new traffic burst prevents it. Congestion only disappears in the last period of time, when no new messages are generated. Concerning the **Self-Tuned** mechanism, we can see that it excessively limits the injection rate, significantly reducing the highest value of accepted traffic and increasing the time required to deliver all the injected messages. This time is another performance measure that is strongly affected by the presence of network congestion. As Figure 8 shows, **Trans-t** delivers the required number of messages in half the time than **No-Lim**, while **Self-Tuned** achieves an intermediate value between both of them.

5 Conclusions

In this paper, we propose a new mechanism (**Trans-t**) based on message throttling to avoid network congestion. This mechanism estimates network traffic by using only local information. In particular, the relationship between the actual and the minimum theoretical transmission time of messages sent across channels is used. The transmission time of a message is the elapsed time between the transfer of its header and tails. If this quotient exceeds a threshold for a given channel, the mechanism assumes that there is congestion in that direction. Although the threshold has to be empirically tuned, it does neither strongly depend on message destination distribution nor on message size, although it depends on the topology radix k (number of nodes per dimension) and the number of virtual channels per physical channel. This is not a problem, as these design parameters are fixed once the machine is built. The information about channels status -congested or not- is considered every time a node tries to inject a new message into the network, adjusting the injection bandwidth depending on the number of congested physical channels that are useful to route the message.

The mechanism has been evaluated for different network loads and topologies. The evaluation results show that the mechanism is able to avoid performance degradation in all the analyzed conditions, outperforming recent proposals, increasing network through-

put and reducing message latency. On the other hand, it does not introduce any penalty for low and medium network loads, when none congestion control mechanism is required. Finally, as it is based only on local information, it does neither require extra signaling nor control message transmission.

References

1. E. Baydal, P. López and J. Duato, "A Simple and Efficient Mechanism to Prevent Saturation in Wormhole Networks", in *14th. Int. Parallel & Distributed Processing Symposium*, May 2000.
2. E. Baydal, P. López and J. Duato, "Avoiding network congestion with local information", *4th. Int. Symposium High Performance Computing*, May 2002.
3. T. Callahan and S.C . Goldstein, "NIFDY: A Low Overhead, High Throughput Network interface", *Proc. of the 22th Int. Symposium on Computer Architecture*, June 1995.
4. W. J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels", *IEEE Trans. on Parallel and Distributed Systems, vol. 4, no. 4, pp. 466–475*, April 1993.
5. J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks", *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 12, pp. 1320–1331, December 1993.
6. J. Duato, S. Yalamanchili and L.M. Ni, *Interconnection Networks: An Engineering Approach*, IEEE Computer Society Press, 1997.
7. C. Hyatt and D. P. Agrawal, "Congestion Control in the Wormhole-Routed Torus With Clustering and Delayed Deflection" *Workshop on Parallel Computing, Routing, and Communication*, June 1997, Atlanta, GA.
8. J. H. Kim, Z. Liu and A. A. Chien, "Compressionless routing: A framework for Adaptive and Fault-Tolerant Routing", in *IEEE Trans. on Parallel and Distributed Systems*, Vol. 8, No. 3, 1997.
9. F. T. Leighton, "Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes", *San Mateo, CA, USA, Morgan Kaufmann Publishers*, 1992.
10. P. López, J.M. Martínez and J. Duato, "A Very Efficient Distributed Deadlock Detection Mechanism for Wormhole Networks", Proc. of High Performance Computer Architecture Workshop, February 1998.
11. P. López, J.M. Martínez and J. Duato, "DRIL: Dynamically Reduced Message Injection Limitation Mechanism for Wormhole Networks", *1998 Int. Conference Parallel Processing*, August 1998.
12. J.M. Martínez, P. López and J. Duato, "A Cost–Effective Approach to Deadlock Handling in Wormhole Networks", *IEEE Trans. on Parallel and Distributed Processing*, pp. 719–729, July 2001.
13. T.M. Pinkston and S. Warnakulasuriya, "On Deadlocks in Interconnection Networks", *24th Int. Symposium on Computer Architecture*, June 1997.
14. A. Smai and L. Thorelli, "Global Reactive Congestion Control in Multicomputer Networks", *5th Int. Conference on High Performance Computing*, 1998.
15. M. Thottetodi, A.R. Lebeck and S.S. Mukherjee, "Self-Tuned Congestion Control for Multiprocessor Networks", *Technical Report CS-2000-15*, Duke University, November 2000.
16. M. Thottetodi, A.R. Lebeck and S.S. Mukherjee, "Self-Tuned Congestion Control for Multiprocessor Networks", *Proc. of High Performance Computer Architecture Workshop*, February 2001.