On the Reduction of Deadlock Frequency by Limiting Message Injection in Wormhole Networks *

P. López[1], J. M. Martínez[1], J. Duato[1] and F. Petrini[2]

[1] Facultad de Informática, Universidad Politécnica de Valencia, Camino de Vera s/n, 46071 - Valencia, SPAIN, E-mail: {plopez,jmmr,jduato}@gap.upv.es

[2] Dipartimento di Informatica, Università di Pisa, Corso Italia 40, Pisa, ITALY, E-mail: petrini@di.unipi.it

Abstract. Recently, deadlock recovery strategies began to gain acceptance in networks using wormhole switching. In particular, progressive deadlock recovery techniques are very attractive because they allocate a few dedicated resources to quickly deliver deadlocked packets, instead of killing them. Deadlock recovery is based on the assumption that deadlocks are really rare. Otherwise, recovery techniques are not efficient.
In this paper, we propose the use of a message injection limitation mechanism that reduces the probability of deadlock to negligible values, even when fully adaptive routing is used. The main new feature is that it can be used with different message destination distributions. The proposed mechanism can be combined with any deadlock detection mechanism. In particular, we use the deadlock detection mechanism proposed in [21]. In addition, the proposed injection limitation mechanism considerably reduces performance degradation when the network reaches the saturation point.

Keywords: Wormhole switching, adaptive routing, deadlock recovery, message injection limitation.

1 Introduction

Wormhole [10] has become the most widely used switching technique for multicomputers [5, 16] and distributed shared-memory multiprocessors [19, 27], and it is also being used for networks of workstations [6, 15]. Wormhole switching only requires small buffers in the nodes through which messages are routed. Also, it makes the message latency largely insensitive to the distance in the network. See [24] for a detailed analysis of wormhole.

The main drawback of wormhole switching is that blocked messages remain in the network, therefore wasting channel bandwidth. In order to reduce the

* This work was supported by the Spanish CICYT under Grant TIC94–0510–C02–01

impact of message blocking, physical channels may be split into virtual channels by providing a separate buffer for each virtual channel and by multiplexing physical channel bandwidth. The use of virtual channels can increase throughput considerably by dynamically sharing the physical bandwidth among several messages [11]. However, it has been shown that virtual channels are expensive, increasing node delay considerably [9]. So, the number of virtual channels per physical channel should be kept small.

An alternative approach consists of using adaptive routing [14]. However, deadlocks may appear if the routing algorithms are not carefully designed. A deadlock occurs in an interconnection network when no message is able to advance toward its destination because the network buffers are full. As routing decisions must be taken in a few nanoseconds in wormhole networks, a practical way to avoid deadlock is to design deadlock-free routing algorithms. A simple and effective approach to avoid deadlocks consists of restricting routing so that there are no cyclic dependencies between channels [10]. A more efficient approach consists of allowing the existence of cyclic dependencies between channels while providing some escape paths to avoid deadlock, therefore increasing routing flexibility [12, 13].

However, deadlock avoidance techniques require dedicated resources to provide those escape paths. Usually, those dedicated resources are virtual channels, thus preventing the use of all the virtual channels for fully adaptive routing. Deadlock recovery strategies overcome this constraint but the cost associated with deadlock recovery can be high, specially when deadlocked messages are killed and injected again at a later time [26, 18]. Progressive deadlock recovery strategies, like Disha [2, 3, 4], are more efficient because they allocate a few dedicated resources to quickly deliver deadlocked packets, instead of killing them. Disha also uses dedicated resources to recover from deadlock. In this case, one central buffer per node is enough to route deadlocked messages towards their destination node. More recently, we have proposed a software-based progressive deadlock recovery strategy [21] thus eliminating the need for dedicated resources to recover from deadlock. Basically, when a message is presumed to be involved in a deadlock, it is absorbed at the current node and later reinjected into the network. The key issues in any deadlock recovery strategy are the deadlock detection mechanism and how often a deadlock is reached.

Progressive deadlock recovery techniques usually achieve a higher performance than deadlock avoidance techniques because they require less dedicated resources to handle deadlocks [3, 21]. However, both kinds of techniques may produce severe performance degradation when the network is close to saturation [12, 3]. This situation occurs when messages block cyclically faster than they are drained by using the escape paths or the deadlock recovery mechanism.

Performance degradation at the saturation point was studied in [12, 20]. In [12], it was shown that degradation can be removed by adding enough virtual channels per physical channel. However, this solution is expensive and may lead to an excessive number of virtual channel for most topologies. An alternative approach was proposed in [20]. It consists of limiting message injection when the network is heavily loaded, preventing message injection when the number of output virtual channels that are busy is higher than a given threshold. The main drawback of this mechanism is that the threshold strongly depends on the message destination distribution.

Very recently, the frequency of deadlock occurrence on k-ary n-cubes using a true fully adaptive minimal routing algorithm was measured [28, 25], showing that deadlocks only occur when the network is close to or beyond saturation.

In this paper we propose a mechanism to prevent network saturation. The proposed mechanism is more robust than previous proposals. It is almost independent of the traffic pattern in the network. First, this mechanism will avoid the performance degradation problem that typically occurs in both, deadlock avoidance and recovery techniques. Second, it will guarantee that the frequency of deadlock is really negligible. As a consequence, the time needed to recover from deadlock is not critical. Hence, deadlock recovery techniques that focus on making the common case fast can be efficiently used. Deadlock detection and recovery mechanisms enable the use of true fully adaptive routing.

Section 2 describes the mechanism proposed in [20] to prevent network saturation. It is the starting point for the new one. The new mechanism is described in Section 3. Performance evaluation results of the new mechanism for different message destination distributions are presented in Section 4. Finally, some conclusions are drawn.

2 Message Injection Limitation: First Approach

In this section, we will briefly describe the message injection limitation proposed in [20]. The main goal of the mechanism, as originally proposed, was to mitigate the performance degradation of the network when it reaches saturation. This problem can be stated as follows. Latency increases with network traffic until certain point, at which latency value goes up while traffic decreases. Average traffic matches flit injection rate until the performance degradation point is reached, then traffic noticeably decreases.

This behavior occurs both with deadlock avoidance and recovery strategies [12, 2]. Performance degradation is produced because these strategies allows cyclic dependencies between some channels provided that there are some escape paths or a recovery mechanism. When message injection rate becomes high, messages block cyclically faster than they are drained by the escape paths or the recovery mechanism, thus increasing latency and decreasing the number of messages delivered per time unit. It must be noticed that messages do not deadlock, but they spend a long time in the network.

Therefore, fully adaptive routing algorithms may degrade performance considerably when the network reaches the saturation point. One solution to this problem could be to increase the number of virtual channels [12], so that messages have a lower probability of being involved in cyclic dependencies. However, a high number of virtual channels could lead to a lower clock frequency [9].

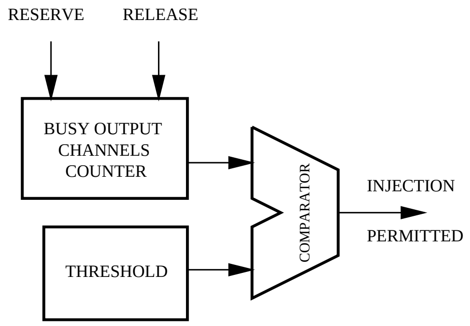


Fig. 1. Implementation of the message injection limitation mechanism

Another solution is to control network traffic, in order to guarantee that it is always under the performance degradation point. But traffic is a global quantity. Thus, it is not feasible to measure it in a distributed way. Fortunately, we have found that traffic can be estimated locally by counting the number of busy output channels at each node.

The message injection limitation mechanism proposed in [20] works as follows. Before injecting a message into the network, traffic is estimated by counting the number of busy output virtual channels at the source node, comparing it with a threshold. If traffic is lower than this threshold, the message is injected into the network. Otherwise, it is stored in the source queue. If threshold value is properly selected, it can be guaranteed that the performance degradation point will never be reached.

The implementation of this mechanism only requires one comparator and one counter associated with each router, as shown in Figure 1. The counter maintains the number of busy output channels. Its value is increased each time a successful routing is made and is decreased when the tail of a message leaves the node. This hardware is not in the critical path. Of course, this mechanism will increase the delay of those messages that are not injected into the network at once, but the average delay is lower than the one obtained by the same adaptive algorithm without injection limitation mechanism [21] because contention in the network is considerably reduced.

Let's analyze the relationship of the mechanism with deadlock recovery strategies. Recently, the frequency of deadlock occurrence on k-ary n-cubes using a true fully adaptive minimal routing algorithm was measured [28], showing that deadlocks only occur when the network is close to or beyond saturation. Thus, one way to effectively reduce the probability of reaching deadlocks is to prevent network saturation. This is exactly what the described message injection limitation mechanism does. In [21], it has been used combined with a deadlock detection mechanism to develop a new deadlock recovery strategy.

However, the behavior of the mechanism depends on the threshold used in the comparison. Although it can be easily selected for a given message destination distribution, different message destination distributions need different thresholds. If we use the threshold value that imposes the maximum restrictions, there will be a performance penalty when a looser limitation is really required. On the other hand, if we tune the threshold imposing a looser limitation than it is really required, the mechanism does not work, and the network will reach saturation.

3 New Message Injection Limitation Mechanism

The main goal of the new message injection limitation mechanism is to take into account the message destination distribution existing in the network at a given time and adapt to it.

The key idea is to dynamically update the threshold used in the comparison. To modify the threshold according to the message destination distribution we exploit the information obtained every time a message is injected. In particular, we have observed that the number of different useful dimensions that a newly injected message has to cross to reach its destination can be used to distinguish between different distributions. Effectively, when destinations are randomly distributed, the destination of a given message can be any node of the network. Thus, a set of injected messages may use every output channel at the source node. On the other hand, when a specific message destination distribution such as bit-reversal or complement is used, the destination of messages originated at a given node is always the same. Thus, messages will only use a subset of the output channels of the node. Destination distributions produced by parallel applications lie somewhere in between.

In addition, we will consider this information not only for the last injected message but for the last m injected messages. Figure 2 shows the hardware required to implement the mechanism. Each time a message generated at a given node is routed at that node, a bit set representing the physical channels that the message can use for routing is stored in a table. This history table contains information for the last m injected messages. A new bit set is obtained by computing the bitwise OR operation of the m entries in the table. This set represents the useful physical channels that have been used or could have been used (adaptive routing) by the m last messages originated at this node. The cardinal of this set is computed and used to calculate the threshold value of the message injection limitation mechanism. The threshold computation can be done by using a translation table.

An example will show how to build this table. Consider a bidirectional 3-D torus with true fully adaptive routing and 3 virtual channels per physical channel. Thus, there are 18 virtual output channels per node. For the uniform message destination distribution we have found that the optimal threshold for the injection limitation mechanism described in Section 2 was 9 busy virtual channels. With this distribution, different messages originating at each node can use any of the three dimensions. Taking into account that channels are bidirectional, there are 6 feasible physical output channels for this distribution. Thus,

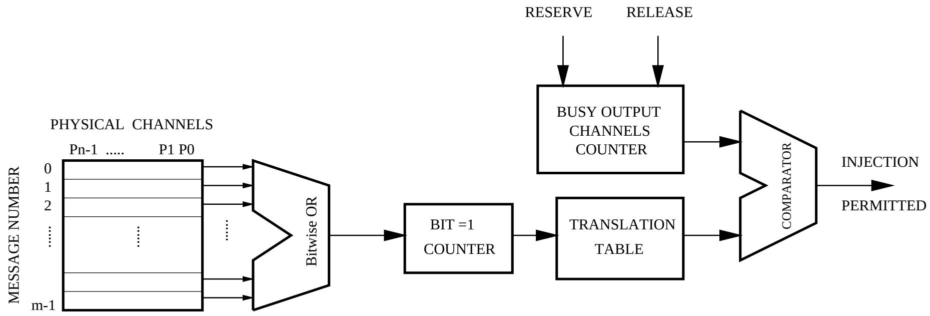


Fig. 2. Implementation of the new message injection limitation mechanism

Feasible physical output channels (POC)	Busy virtual output channels (Threshold Value, TV)
6	9
5	8
4	6
3	5
2	3
1	2

Table 1. Example of translation table to compute the threshold value.

the table should return a threshold of 9 for 6 feasible physical output channels. The returned value should decrease with the number of feasible physical output channels. For instance, in Table 1 the threshold is reduced proportionally to the number of feasible physical output channels.

The function corresponding to Table 1 is $TV = \lceil 1.5 * POC \rceil$. Less restrictive tables or functions can be employed to better match the message destination distributions that are likely to be used. In the extreme case, the number of busy virtual output channels would not depend on the number of feasible physical output channels, leading to the injection limitation mechanism described in Section 2.

Of course, the new mechanism still depends on the number of network dimensions and the number of virtual channels per physical channel, but these parameters are fixed in the design process.

When the routing control unit is reset, the first item in the history table is set in such a way that it indicates that all physical output channels are in use (all "1"). Thus, the mechanism will apply the less restrictive limitation until the history table fills out. After that, the mechanism will limit the injection according to the physical output channel usage.

Finally, it must be noticed that the update of the history table can be done after computing the routing algorithm. Moreover, the calculation of the threshold value can also be done between message routing operations. Thus, this hardware is not in the critical path.

4 Performance Evaluation

In this section, we will evaluate by simulation the behavior of the new injection limitation mechanism. The evaluation methodology used is based on the one proposed in [12]. The most important performance measures are latency (time required to deliver a message, including the time spent in the source queue) and throughput (maximum traffic accepted by the network). Traffic is the flit reception rate. Latency is measured in clock cycles. Traffic is measured in flits per node per cycle.

Taking into account the sizes of current multicomputers and the studies about the optimal number of dimensions [1], we have evaluated the performance of the new injection mechanism on a bidirectional 8-ary 3-cube network (512 nodes).

4.1 Network Model

Our simulator models the network at the flit level. Each node in the network consists of a processor, its local memory, a routing control unit, a switch, and several channels. A four port architecture [22] is considered.

The routing control unit computes the output channel for a message as a function of its destination node, the current node and the output channel status. The routing algorithm can use any minimal path to forward a message towards its destination. In addition, several virtual channels per physical channel can be used. In other words, all virtual channels in all the feasible directions can be used. This algorithm is usually referred to as True Fully Adaptive Routing algorithm [28, 25, 21]. We assume that it takes one clock cycle to compute the routing algorithm. The routing control unit can only process one message header at a time. It is assigned to waiting messages in a demand-slotted round-robin fashion (including those messages generated in the local processor). When a message gets the routing control unit but it can not be routed because all the alternative output channels are busy, it must wait in the input buffer until its next turn. The deadlock recovery strategy proposed in [21] is used. In this strategy, deadlocked messages are removed from the network at the current node, and injected again at a later time. The new message injection limitation mechanism described in Section 3 is used. Although we have not performed an in-depth analysis of the influence of the size m of the history table on performance, it seems that it does not affect significantly. We have used a value of m equal to 30 (5 times the number of output physical links). We also tried with m equal to 24 and 36, obtaining similar results.

The switch is a crossbar. So, it allows multiple messages to traverse it simultaneously without interference. It is configured by the routing control unit each time a successful routing is made. We assume that it takes one clock cycle to transfer a flit across the switch.

Physical channels can be split into several virtual channels. Virtual channels are assigned to the physical link using a demand-slotted round-robin arbitration scheme. Each virtual channel has an associated buffer with capacity for four flits. We assume that it takes one clock cycle to transfer a flit across a physical channel.

4.2 Message Generation

Message traffic and message length depend on the applications. For each simulation run, we considered that message generation rate is constant and the same for all the nodes. Once the network has reached a steady state, the flit generation rate is equal to the flit reception rate (traffic). We have evaluated the full range of traffic, from low load to saturation. On the other hand, we have considered three kinds of message distribution: uniform, with locality, and specific.

In the first case, message destination is randomly chosen among all the nodes. This pattern has been widely used in other performance evaluation studies [11, 8, 7]. For the message distribution with locality, the destination node is uniformly distributed inside a cube with side two centered at the source node, modeling short distance traffic. Finally, we have considered specific communication patterns between pairs of nodes. These patterns have been selected taking into account the permutations that are usually performed in parallel numerical algorithms [23, 17]:

- Bit-reversal. The node with binary coordinates $a_{n-1}, a_{n-2}, ..., a_1, a_0$ communicates with the node $a_0, a_1, ..., a_{n-2}, a_{n-1}$.
- Perfect-shuffle. The node with binary coordinates $a_{n-1}, a_{n-2}, ..., a_1, a_0$ communicates with the node $a_{n-2}, a_{n-3}, ..., a_0, a_{n-1}$ (rotate left one bit).
- Butterfly. The node with binary coordinates $a_{n-1}, a_{n-2}, ..., a_1, a_0$ communicates with the node $a_0, a_{n-2}, ..., a_1, a_{n-1}$ (exchange the most and least significant bits).

For message length, 16-flit messages were considered. This value is considered as typical in other interconnection network evaluation studies [1, 26].

4.3 Performance Comparison

In this section we compare the results obtained by the routing algorithm considering the new injection limitation mechanism with different translation tables. We will also show the results obtained with the previous message injection limitation mechanism described in Section 2 (using 9 busy virtual channels as the threshold) and without injection limitation, for comparison purposes. Note that

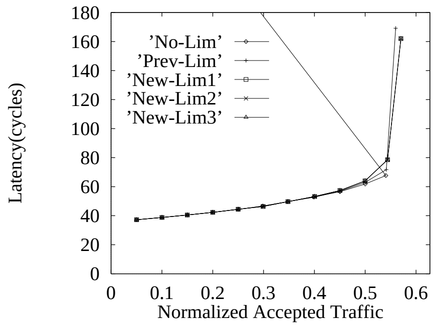


Fig. 3. Average message latency versus traffic for a 512-node 3-D torus with true fully adaptive routing (3 virtual channels per physical channel). Uniform distribution of message destinations.

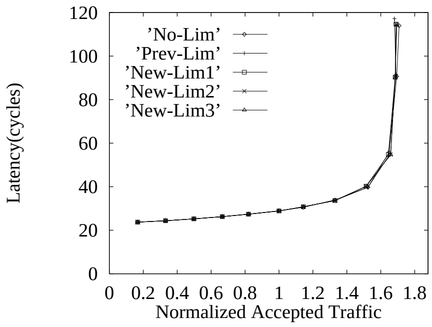


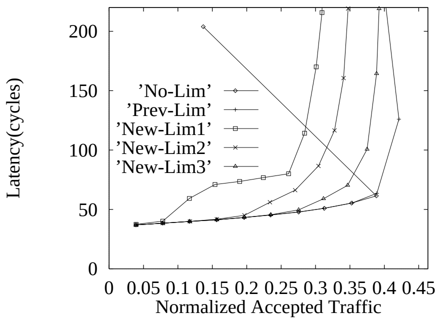
Fig. 4. Average message latency versus traffic for a 512-node 3-D torus with true fully adaptive routing (3 virtual channels per physical channel). Uniform distribution of message destinations with locality.

the average latency plotted in all the figures includes the time spent by messages at their source nodes due to the injection limitation mechanism.

For the new injection limitation mechanism we tested different functions in order to obtain the threshold value (TV). We have seen that a linear function obtains good results. In order to define a linear function, we need a pair of points. The first point is given by the TV returned for the maximum number of feasible physical output channels (TV equal to 9 for 6 feasible physical output channels). The second point is given by the TV returned when there is only one feasible physical output channel. We tried three linear functions with different slopes. The first function is the most restrictive one. It was described in Section 3. The other ones impose looser restrictions to inject messages into the network. These functions are the following:

1. $TV = \lceil 1.5 * POC \rceil$. It will referred to as *New-Lim1*.
2. $TV = \lceil 1.2 * POC + 1.8 \rceil$. It will referred to as *New-Lim2*.
3. $TV = \lceil 0.8 * POC + 4.2 \rceil$. It will referred to as *New-Lim3*.

Figure 3 shows the comparison for the uniform distribution of message destinations. As we can see, the routing algorithm without the injection limitation mechanism reaches the performance degradation point. In addition, many deadlocks were detected. On the other hand, the behavior of the previous and new message injection limitation mechanisms is almost identical. This is due to the fact that the new mechanism limits in the same way as the previous one when every output physical channel has been requested by some of the last m injected messages. There are no differences between the three functions used in the new limitation mechanism, because all of them return the same threshold value.



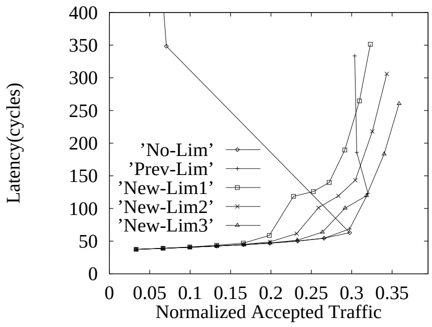


Fig. 5. Average message latency versus traffic for a 512-node 3-D torus with true fully adaptive routing (3 virtual channels per physical channel). Bit-reversal pattern for message destinations.

Fig. 6. Average message latency versus traffic for a 512-node 3-D torus with true fully adaptive routing (3 virtual channels per physical channel). Perfect-shuffle pattern for message destinations.

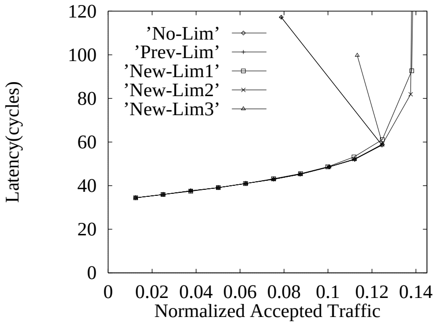


Fig. 7. Average message latency versus traffic for a 512-node 3-D torus with true fully adaptive routing (3 virtual channels per physical channel). Butterfly pattern for message destinations.

In case of uniform distribution of message destination with locality (Figure 4), all the limitation mechanisms achieve similar results. Indeed, injection is not limited even when the network reaches saturation. There is no performance degradation and there are no deadlocks in any case. Taking into account that messages traverse very short distances, the probability of being involved in cyclic dependences is very low.

Figure 5 shows the results for the bit-reversal traffic pattern. As can be seen, message injection limitation is compulsory. Otherwise, the network will suffer performance degradation and deadlocks. The previous injection limitation mechanism achieves the best results, with a small performance degradation with very high load. The new limitation mechanisms achieve worse results because they impose more restrictions to inject a message into the network. Among these, the *New-Lim3* is the best, as it is the one that limits less the injection of messages.

Figure 6 shows the results for the perfect-shuffle traffic pattern. Again, the network without injection limitation reaches the saturation point and degrades performance drastically. Although the previous message injection limitation mechanism reduces the performance degradation, the problem is not completely eliminated. Any of the new mechanisms definitively solves the problem. Again, the *New-Lim3* is the best, as it is the one that imposes looser restrictions to message injection.

Finally, Figure 7 shows the results for the butterfly traffic pattern. If we do not impose restrictions to message injection, performance degradation and deadlock are reached. With the previous message injection limitation, the problem is not solved. The problem is reduced, but not completely solved, with the *New-Lim3* mechanism. Only the *New-Lim1* and *New-Lim2* mechanisms which restrict message injection more drastically are able to prevent network saturation.

The new message injection limitation mechanism prevents network saturation in all of the traffic patterns analyzed in this section. In some cases, it is too restrictive and it achieves poorer results than the previously proposed message limitation mechanism described in Section 2 tuned with the proper threshold. However, the new mechanism is able to deal with several message destination distributions, dynamically adapting to the requirements of the traffic pattern in the network.

Among the translation tables analyzed in this section, *New-Lim3* seems to be the best choice. The small performance degradation observed with the butterfly permutation should not harm in practice because it would require a parallel application injecting messages only according to this traffic pattern and driving the network beyond saturation.

5 Conclusions

In this paper we have proposed a new message injection limitation mechanism to prevent network saturation. By doing so, the performance degradation of adaptive routing algorithms with cyclic dependencies between channels is avoided and, most importantly, the frequency of deadlock is reduced to negligible values.

The proposed mechanism consists of using a history table to keep track of the physical channels requested by the last set of messages injected at each node. The number of requested channels is converted into a threshold by using a translation table, which in turn is compared with the number of busy output virtual channels at that node in order to decide whether message injection should be permitted or not.

The main goal of the proposed mechanism is robustness. Unlike previous proposals, it is almost independent of the message destination distribution. The proposed mechanism needs to be implemented in the routing control unit but it does not affect clock frequency, because it is not in the critical path.

The new message injection limitation mechanism has been evaluated by simulation under several traffic patterns that presumably are representative of real programs. Although it does not achieve the best results for each individual traffic pattern, is able to prevent network saturation and achieve performance results very close to the best algorithm for each case.

As for future research, we plan to analyze alternative injection limitation mechanisms, looking for simpler and more effective ways to reduce the frequency of deadlock in order to make deadlock recovery really feasible. Also we plan to use traffic patterns generated by parallel applications to test those mechanisms.

References

1. A. Agarwal, "Limits on interconnection network performance", *IEEE Transactions on Parallel Distributed Systems*, vol. 2, no. 4, pp. 398–412, Oct. 1991.
2. Anjan K. V. and T. M. Pinkston, "DISHA: A deadlock recovery scheme for fully adaptive routing," in *Proceedings of the 9th International Parallel Processing Symposium*, April 1995.
3. Anjan K. V. and T. M. Pinkston, An efficient fully adaptive deadlock recovery scheme: DISHA," in *Proceedings of the 22nd International Symposium on Computer Architecture*, June 1995.
4. Anjan K. V., T. M. Pinkston and J. Duato, "Generalized theory for deadlock-free adaptive routing and its application to Disha Concurrent," in *Proceedings of the 10th International Parallel Processing Symposium*, April 1996.
5. W. C. Athas and C. L. Seitz, "Multicomputers: Message-passing concurrent computers," *IEEE Computer*, vol. 21, no. 8, pp. 9–24, August 1988.
6. 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.
7. R. V. Boppana and S. Chalasani, "A comparison of adaptive wormhole routing algorithms," in *Proceedings of the 20th International Symposium on Computer Architecture*, May 1993.
8. A. A. Chien and J. H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors," in *Proceedings of the 19th International Symposium on Computer Architecture*, May 1992.
9. A. A. Chien, "A cost and speed model for k-ary n-cube wormhole routers," in *Proceedings of Hot Interconnects'93*, August 1993.
10. W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C–36, no. 5, pp. 547–553, May 1987.
11. W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, March 1992.
12. J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 12, pp. 1320–1331, December 1993.

13. J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 10, pp. 1055–1067, October 1995.
14. P. T. Gaughan and S. Yalamanchili, "Adaptive routing protocols for hypercube interconnection networks," *IEEE Computer*, vol. 26, no. 5, pp. 12–23, May 1993.
15. R. Horst, "ServerNet deadlock avoidance and fractahedral topologies," in *Proceedings of the International Parallel Processing Symposium*, pp. 274–280, April 1996.
16. *Paragon XP/S Product Overview.* Intel Corporation, Supercomputer Systems Division, Beaverton, OR, 1991.
17. J. Kim, A.Chien, "An evaluation of the planar/adaptive routing," in *Proc. 4th IEEE Int. Symp. Parallel Distributed Processing*, 1992.
18. J. H. Kim, Z. Liu and A. A. Chien, "Compressionless routing: A framework for adaptive and fault-tolerant routing," in *Proceedings of the 21st International Symposium on Computer Architecture*, April 1994.
19. D. Lenoski, J. Laudon, K. Gharachorloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz and M. Lam, "The Stanford DASH multiprocessor," *IEEE Computer*, vol. 25, no. 3, pp. 63–79, March 1992.
20. P. López and J. Duato, "Deadlock-free adaptive routing algorithms for the 3D-torus: Limitations and solutions," in *Proceedings of Parallel Architectures and Languages Europe 93*, June 1993.
21. J.M. Martínez, P. López, J. Duato and T.M. Pinkston, "Software-Based Deadlock Recovery Technique for True Fully Adaptive Routing in Wormhole Networks," to appear in *Proceedings 1997 International Conference Parallel Processing*, August 1997.
22. P.K. McKinley, H. Xu, A. Esfahanian and L.M. Ni, "Unicast-based multicast communication in wormhole-routed networks," in *Proceedings 1992 International Conference Parallel Processing*, August 1992.
23. P.R. Miller, "Efficient communications for fine-grain distributed computers," Ph.D Thesis, Southamptom University, 1991.
24. L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, vol. 26, no. 2, pp. 62–76, February 1993.
25. T.M. Pinkston and S. Warnakulasuriya, "On Deadlocks in Interconnection Networks", in *Proceedings 24th International Symposium on Computer Architecture*, June 1997.
26. D. S. Reeves, E. F. Gehringer and A. Chandiramani, "Adaptive routing and deadlock recovery: A simulation study," in *Proceedings of the 4th Conference on Hypercube, Concurrent Computers & Applications*, March 1989.
27. S. L. Scott and G. Thorson, "The Cray T3E networks: adaptive routing in a high performance 3D torus," in *Proceedings of Hot Interconnects IV*, August 1996.
28. S. Warnakulasuriya and T.M. Pinkston, "Characterization of deadlocks in interconnection networks," in *Proceedings of the 11th International Parallel Processing Symposium*, April 1997.