A Simple and Efficient Mechanism to Prevent Saturation in Wormhole Networks*

E. Baydal, P. López and J. Duato
Depto. Informática de Sistemas y Computadores
Universidad Politécnica de Valencia, Camino de Vera s/n, 46071 - Valencia, SPAIN
E-mail: {elvira,plopez,jduato}@gap.upv.es

Abstract

Both deadlock avoidance and recovery techniques suffer from severe performance degradation when the network is close to or beyond saturation. This performance degradation appears because messages block in the network faster than they are drained by the escape paths in the deadlock avoidance strategies or the deadlock recovery mechanism.

Many parallel applications produce bursty traffic that may saturate the network during some intervals [14, 8], significantly increasing execution time. Therefore, the use of techniques that prevent network saturation are of crucial importance. Although several mechanisms have been proposed in the literature to reach this goal, some of them introduce some penalty when the network is not fully saturated, require complex hardware to be implemented or do not behave well under all network load conditions. In this paper, we propose a new mechanism to avoid network saturation that overcomes these drawbacks.

1. Introduction

Wormhole [3] has become the most widely used switching technique. In the wormhole context, there are two strategies for deadlock handling [7]: deadlock avoidance and deadlock recovery.

Deadlock avoidance [3, 5] prevents deadlocks by restricting routing so that there are not cyclic dependencies between channels [3] or there are some escape paths to avoid deadlock [5]. On the other hand, deadlock recovery [9, 1, 13] do not impose routing restrictions, allowing the use of unrestricted fully adaptive routing. However, it requires a deadlock detection mechanism and a deadlock recovery mechanism.

Both deadlock handling schemes suffer from severe performance degradation when network traffic is close to or beyond the saturation point [5, 1]. Figure 1 illustrates the problem for a 8-ary 3-cube with a deadlock-recovery fully-adaptive routing and an uniform distribution of message destination with 16-flit messages. Notice that both latency and accepted traffic are dependent variables on offered traffic. Performance degradation appears because messages

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

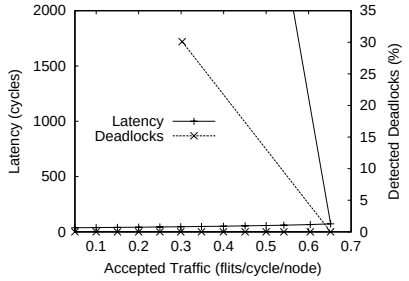

Figure 1. Effects of performance degradation.

cyclically block in the network faster than they are drained by the escape paths (deadlock avoidance) or by the deadlock recovery mechanism. Thus, latency and percentage of detected deadlocks increase and the number of messages delivered by time unit decreases. It is important to remark that with deadlock avoidance, messages do not really deadlock, but they spend a long time blocked in the network.

A solution to the problem is to reduce the probability of cyclic waiting in the network. This can be accomplished by splitting each physical channel into more virtual channels, increasing the number of routing options. However, this makes hardware more complex, possibly leading to a reduction in clock frequency [2].

Another approach to solve the problem is to use a mechanism to ensure that accepted network traffic does not exceeds a given maximum level. This mechanism is often known as congestion control mechanism. In this paper, we present a new technique to avoid network saturation.

The rest of the paper is organized as follows. Section 2 presents the motivation of this work and reviews some related work. Section 3 describes the new proposal. Performance evaluation results are presented in Section 4. Finally, some conclusions are drawn.

2. Motivation and related work

As stated above, both deadlock handling mechanisms suffer from performance degradation when network load is close to or beyond saturation. Although the interconnection network is not the bottleneck in most current systems, as processor clock frequency is increasing at a faster rate than network bandwidth, the interconnection network may

become a bottleneck within the next years [7]. On the other hand, in order to increase the cost-effectiveness of interconnection networks, more than one processor can be attached to each network router [14]. As a consequence, network load is increased. Finally, some studies of interconnection network behavior under the traffic generated by real applications [15, 8] show that network traffic is bursty and peak traffic may saturate the network. As a consequence, the interconnection network may reach the performance degradation point, potentially increasing application execution time. Hence, mechanisms that prevent network saturation may help in improving overall system performance.

Most previously proposed techniques [4, 10, 12, 16] are based on monitoring network traffic and apply message throttling when it exceeds a given level.

In [10, 12] network traffic is locally estimated by counting the number of busy virtual output channels in each node. A newly generated message is injected only if this number is lower than a given threshold. Both mechanisms use dynamic thresholds. In [10] the new threshold is computed according to a guess of the message destination distribution currently in the network. In [12] the new threshold is computed by counting the number of virtual output channels when the network is just entering saturation.

Other mechanisms are based on timers. In [9, 16], it is measured the time that a message header is stopped. If it exceeds a threshold, the mechanism presume that the network is congested. Having detected congestion, [16] uses some special signals to inform the rest of nodes that they must stop message injection. So, congestion control is global and all of the network nodes take part on it. In [9], if a message stops more time than the threshold, the sender kills and retransmits it later. To ensure that the header reaches its destination before the last flit has been injected into the network, short messages have to be padded.

Although these mechanisms are able to prevent network saturation, they have some drawbacks. Although they should work properly for any network load condition, many of the mechanisms have been analyzed only with the uniform distribution of message destinations [4, 9, 16]. Others do not achieve good results for each individual traffic pattern considered [10] or strongly depend on message size [9, 16]. Moreover, the new mechanism should not penalize the network when it is not saturated. However, some of the proposals increase message latency before the saturation point [4]. Finally, the new mechanism should not increase network complexity or generate new problems. However, some of the mechanisms add new signals [9, 16], padding [9] or increase node complexity [10, 12]. Finally, in [12] some nodes may begin to apply strict restrictions before others do, which may produce starvation.

3. A new injection limitation mechanism

The new mechanism is based on measuring network traffic prior to inject a message. In [10, 12], it was found that there is a useful correlation between the number of busy (or free) virtual output channels and accepted network traffic. Thus, traffic is locally estimated by counting the number of busy virtual output channels. However, in order to better distinguish among different message destination distributions, only those output channels that are useful to forward a message towards its destination are considered. This information is obtained by executing the routing function for the message.

The new technique is based on the assumption that the routing algorithm tries to minimize virtual channel multiplexing in order to avoid its negative effects. Many adaptive routing algorithms work in this way [6, 13]. Thus, busy virtual channels tend to be distributed among all the physical channels of the router. In other words, the number of busy virtual channels will tend to be the same for all the physical channels of the router. As network load is increased, the number of busy virtual channels per physical channels will also increase. Intuitively, when the last virtual channel of one physical channel starts to be used, network traffic is becoming high. Remember that only those channels that are useful to route a message are considered. To support this intuitive idea, we have performed an analysis of the percentage of routing occurrences which have at least one free virtual channel in all the physical channels that are useful to forward the message towards its destination. Figure 2 (a) curve) shows the results. As can be seen, the condition is satisfied in almost all the routings for low injection rates. However, as traffic increases, the number of routings which have at least one free virtual channel in all the feasible output channels is reduced. Therefore, some correlation exists between the completely used physical channels at a node and the network traffic in the node area.

Taking this correlation into account, the basic idea of the new mechanism is the following. Before injecting a newly generated message, the routing function is executed and the number of free virtual channels of all useful physical output channels is checked. If all of them have at least one free virtual channel, injection of the message is permitted. Otherwise, it is forbidden. Notice that as the method only considers those physical channels that are useful to route the message, it does not matter that some network areas are congested if they are not likely to be used by the message. On the other hand, if the channels that forward a message towards its destination are congested, the mechanism should prevent injection regardless of the other channels are free.

However, in some cases -not many- a physical channel can become completely free, while another in the same node has still all its virtual channels occupied. In this situation, traffic in the node area is not really saturated, but the described mechanism will prevent injection of messages, increasing message latency. Figure 2 shows the percentage of routing occurrences that satisfy both conditions individually and any of them. As can be seen, the second condition alone is a worse indicator of congestion. However, both rules combined (the first one OR the second one) improve congestion detection. Thus, the final mechanism allows injection if, after applying the routing function, at least one virtual channel of all the useful physical channels is free or at least one physical channel have all its virtual chan-

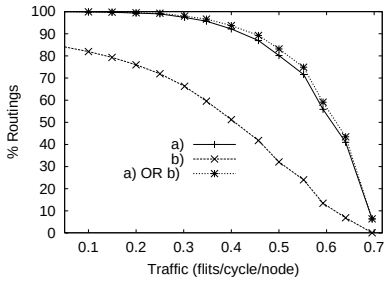

Figure 2. Percentage of routings with: a) All useful physical output channels with at least one free virtual channel. b) At least one useful physical channel completely free.

nels completely free. Notice that, contrary to previous approaches, there is not any threshold to adjust in this mechanism. This will noticeably simplify the implementation. However, the ability of the mechanism to adapt to different message destination distribution is achieved by considering only those channels returned by the routing function.

To illustrate the method, consider a bidirectional k-ary 3-cube network with 3 virtual channels per physical channel. A message generated according to a uniform distribution of message destination may use, at most, the total number of physical channels of the node, 6. The first rule will allow injection if all of them have at least one virtual channel free. Therefore, injection will be permitted if 6 or more virtual channels, fairly distributed among physical channels, are free. The second rule will permit injection if any physical channel is completely free, with no virtual channel occupied. Otherwise, if both conditions are not satisfied, injection will be forbidden. On the other hand, a more specific traffic pattern such as the *butterfly* one, that swaps the most and least significant bits of the source node to compute the destination node, will only use physical channels in two dimensions. In this case, injection is allowed if the number of free virtual channels is, at least, 2, one in each physical channel, or at least one of the channels has its 3 virtual channels free.

Summing up, the new proposal follows the next steps before injecting a new message in the network. First, the routing function is applied to obtain the useful channels that the message may use to reach its destination. Second, the routing information is merged with the status (free, completely free or busy) of the physical output channels. A physical channel is considered to be free if at least one of its virtual channels is free, completely free if all its virtual channels are free and busy in any other case. Next, information about all useful channels is combined. If all useful physical channels are free or any of them is completely free, the message can be injected. Otherwise, the message is queued. Finally, messages in the pending message queue have higher priority than messages generated later.

Figure 3 shows the implementation of the mechanism. The routing function must be applied before injecting the

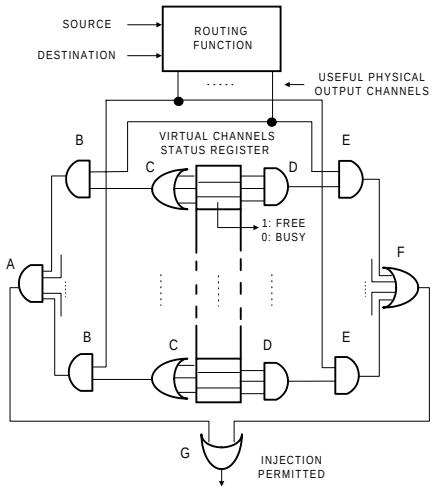

Figure 3. Hardware required for the new injection limitation mechanism.

newly generated message into the network. This requires replicating some hardware, but with a simple routing algorithm (see Section 4.1), this should not be a problem. The routing function returns the useful physical output channels[1]. In parallel, two logical operations are performed on the virtual channels status register. First, C gates detect if there is at least one free virtual channel in each physical channel[2]. Simultaneously, D gates detects if all the virtual channels of a physical channel are free. Next, this information is combined with the result of the routing function (B and E gates) in order to consider only the useful channels. Finally, A and F gates applies the first and second rules, respectively, for all the useful physical channels and G gate allows injection if any of them are satisfied.

As can be seen, the hardware required to implement the mechanism is very simple. Only some logic gates are required. As the mechanism does not need any threshold, there is neither need for registers nor comparators. Finally, notice that, although this hardware may add some delay to injected messages, it does not reduce clock frequency because it is not on 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 is based on the one proposed in [6, 13]. The most important performance measures are not only latency (time required to deliver a message, including the time spent at the source queue) and throughput (maximum traffic accepted by the network) but also the percentage of detected

[1]This implementation assumes that all the virtual channels of a physical channel can be used in the same way by a message, which is the case for True Fully Adaptive Routing (see Section 4.1).

[2]This implementation assumes 3 virtual channel per physical channel.

deadlocks in the network. Traffic is the flit reception rate. Latency is measured in clock cycles, and traffic in flits per node per cycle. The percentage of detected deadlocks is measured as the ratio between the number of messages detected as deadlocked and the total number of sent messages.

4.1. Network model

Each node has four injection/ejection channels, a router, a crossbar switch and several physical channels. Routing time and transmission time across the crossbar and a channel are all assumed to be equal to one clock cycle. Physical channels may be split into up to three virtual channels, each one with a four-flit buffer. Deadlock recovery is used. The routing algorithm is a True Fully Adaptive (TFAR) [13], which 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 FC3D mechanism proposed in [11], with a threshold of 32 cycles. The recovery mechanism is the software-based proposed in [13]. The new mechanism described in Section 3 is used to prevent network saturation.

Message injection rate is the same for all nodes. Each node generates messages independently, according to an exponential distribution. Destinations are chosen according to this communication patterns: *Uniform, Butterfly, Complement, Bit-reversal,* and *Perfect-shuffle*. For message length, 16-flit and 64-flit messages are considered. Finally, we have evaluated the performance of the new mechanism proposed on a bidirectional 8-ary 3-cube network (512 nodes).

4.2. Performance comparison

In this section, we will analyze the behavior of the new mechanism proposed in section 3, which will be referred to as **ALO** (At Least One). In addition, we will also evaluate the behavior of some previous techniques for comparison purposes. In [10] and [12] we proposed two mechanisms to prevent network congestion. We will refer to them as (**Linear Function**) and (**DRIL**), respectively. Finally, results without any injection limitation mechanism (**No-Lim**) are also shown.

First, we will focus on fairness. With low traffic rates (not shown) message injection limitation mechanisms do not impose any restriction, so the behavior is the same for all mechanisms. On the contrary, with high traffic rates, message injection mechanisms are working. As the injection rate is the same for all nodes in the network, the average number of messages sent by each node should be roughly the same. Figure 4 shows the differences in the number of messages sent per node (in %) for each analyzed mechanism for the uniform distribution of message destinations with 64-flit messages. Other message patterns or message lengths have a similar shape. While the **Linear Function** and **ALO** mechanisms allows fairness, **DRIL** mechanism presents large differences in sent messages for some nodes. This is due to the fact that nodes compute its own threshold in different moments (see [12]). The ones that compute threshold first reduces injection and average traffic is reduced, so the rest of nodes computes a less restricted threshold value. The differences in sent messages per node with **ALO** are less than 13% in the worst cases, while with **Linear Function** this value grows up to 21% and with **DRIL** mechanism some of the nodes have injected about 60% less messages than the rest and even one of them is under 80%.

Figures 5 and 6 show the average message latency, standard deviation of latency and percentage of detected deadlocks for the analyzed mechanisms for the uniform message destination distribution and for 16-flits and 64-flits messages, respectively. As we can see, any of the three mechanisms helps in preventing network saturation and performance degradation. However, the **ALO** mechanism introduces the lowest latency penalty and achieves the highest throughput. Notice that latency penalty is due to the increased time that messages stay at the source nodes before being injected. Also, any of the message injection limitation mechanism reduces the percentage of detected deadlocks to negligible values (0.06% in the worst case).

Figures 7 to 10 display the results for the permutation-based traffic patterns analyzed. As results for 16-flits and 64-flits messages are qualitatively the same, we will only show the results for 16-flit messages. As we can see, message injection is mandatory to avoid severe performance degradation and keep low the percentage of detected deadlocks. The **ALO** injection limitation mechanism allows usually the highest network throughput. From the latency point of view, the **ALO** injection limitation mechanism achieves a good behavior for all message patterns, although in some cases another mechanism obtains slightly lower latencies. However, when **ALO** is outperformed by other mechanisms, its behavior is close to the mechanism that reaches the best results.

Concerning the percentage of detected deadlocks, all the message injection limitation mechanisms helps in considerably reducing them. If any message injection limitation mechanism is not applied, the number of detected deadlocks when the network enters saturation is very high. We have obtained deadlock detection rates greater than 70%, 35% and 20% (not shown in the figures) for complement, perfect-shuffle and bit-reversal traffic patterns, respectively. The **ALO** achieves the lowest deadlock detection rate for all distributions but the bit-reversal one and the uniform one with 64-flit messages.

In summary, the **ALO** mechanism presents a good behavior for all the message destinations and message lengths analyzed. This results, together with its simple implementation suggest that the **ALO** mechanism outperforms cost-effectiveness of previous proposals.

5. Conclusions

In this paper, we have proposed a new message injection limitation (ALO) mechanism to prevent network saturation. By using this mechanism, the number of cyclic dependencies that may result in deadlocks is considerably reduced. As a consequence, the performance degradation that typi-

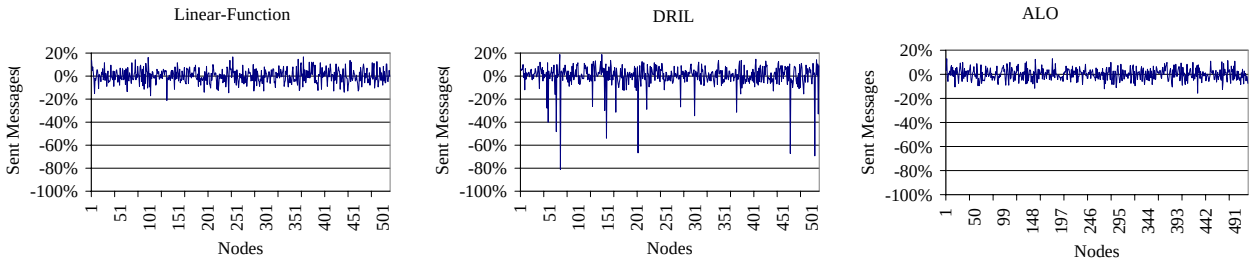

Figure 4. Differences in sent messages per node for Linear Function, DRIL and ALO. Uniform distribution of message destinations. 64-flit messages. Traffic rate of 0.65 flits/cycle/node.

cally occurs in both deadlock avoidance and recovery techniques is removed.

The proposed mechanism uses only information about the status (free or busy) of the useful virtual output channels (the ones that are provided by the routing algorithm) to estimate network traffic. Message injection is allowed only if any of the useful physical channels is completely free (none of its virtual channels is busy), or all the useful physical channels are partially free (at least one of its virtual channels is free). The proposed mechanism needs to be implemented in the routing control unit but it is very simple and does not affect clock frequency, because it is not on the critical path. In addition, it does not need any adjustment.

The evaluation results show that the mechanism is able to avoid performance degradation for all the traffic patterns analyzed, eliminating the lack of fairness found on other mechanisms. On the other hand, the new mechanism outperforms the previous proposals for almost all message destination distributions, increasing throughput and reducing latency and deadlock detection rate. However, when any of the previous mechanisms outperform the new one, its behavior is close to the mechanism that reaches the best results. Finally, its implementation is much simpler than any of the previous approaches.

References

[1] Anjan K. V. and T. M. Pinkston, An efficient fully adaptive deadlock recovery scheme: DISHA," in *Proc. of the 22nd Int. Symp. on Computer Architecture*, June 1995.

[2] A. A. Chien, "A cost and speed model for k-ary n-cube wormhole routers," in *Proc. of Hot Interconnects'93*, August 1993.

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

[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 and P. López, "Performance evaluation of adaptive routing algorithms for k-ary n-cubes," in *Parallel Computer Routing and Communication*, K. Bolding and L. Snyder (ed.), Springer-Verlag, pp. 45–59, 1994.

[7] J. Duato, S. Yalamanchili and L.M. Ni, *Interconnection Networks: An Engineering Approach*, IEEE Computer Society Press, 1997.

[8] J. Flich, M. P. Malumbres, P. López and J. Duato, "Performance Evaluation of Networks of Workstations with Hardware Shared Memory Model Using Execution-Driven Simulation," *1999 Int. Conf. Parallel Processing*, September 1999.

[9] J. H. Kim, Z. Liu and A. A. Chien, "Compressionless routing: A framework for adaptive and fault-tolerant routing," in *Proc. of the 21st Int. Symp. on Computer Architecture*, April 1994.

[10] P. López, J.M. Martínez, J. Duato and F. Petrini, "On the Reduction of Deadlock Frequency by Limiting Message Injection in Wormhole Networks," in *Proc. of Parallel Computer Routing and Communication Workshop*, June 1997.

[11] P. López, J.M. Martínez and J. Duato, "A Very Efficient Distributed Deadlock Detection Mechanism for Wormhole Networks," in *Proc. of High Performance Computer Architecture Workshop*, February 1998.

[12] P. López, J.M. Martínez and J. Duato, "DRIL: Dynamically Reduced Message Injection Limitation Mechanism for Wormhole Networks," *1998 Int. Conf. Parallel Processing*, August 1998.

[13] J.M. Martínez, P. López, J. Duato and T.M. Pinkston, "Software-Based Deadlock Recovery Technique for True Fully Adaptive Routing in Wormhole Networks," *1997 Int. Conf. Parallel Processing*, August 1997.

[14] J.F. Martínez, J. Torrellas and J. Duato, "Improving the Performance of Bristled CC-NUMA Systems using Virtual Channels and Adaptivity," *1999 ACM Int. Conf. on Supercomputing*, June 1999.

[15] F. Silla, M.P. Malumbres, J.Duato, D. Dai, and D.K. Panda, "Impact of Adaptivity on the Behavior of Networks of Workstations under Bursty Traffic," *1998 Int. Conf. on Parallel Processing*, August 1998.

[16] A. Smai and L. Thorelli , "Global Reactive Congestion Control in Multicomputer Networks", *5th Int. Conf. on High Performance Computing*, 1998.

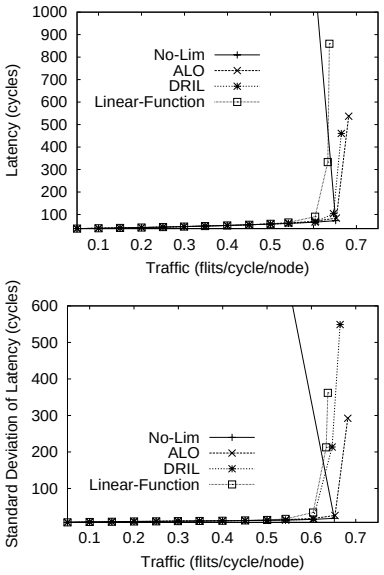

Figure 5. Average message latency and standard deviation of message latency versus traffic. Uniform distribution of message destinations. 16-flit messages.

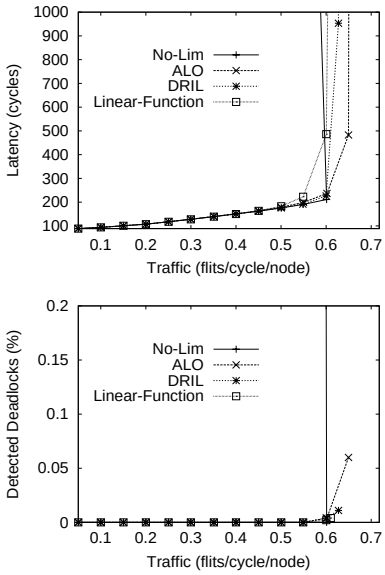

Figure 6. Average message latency and percentage of detected deadlocks versus traffic. Uniform distribution of message destinations. 64-flit messages.

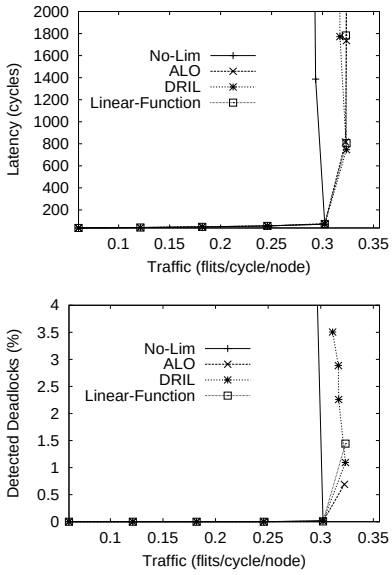

Figure 7. Average message latency and percentage of detected deadlocks versus traffic. Butterfly traffic pattern. 16-flit messages.

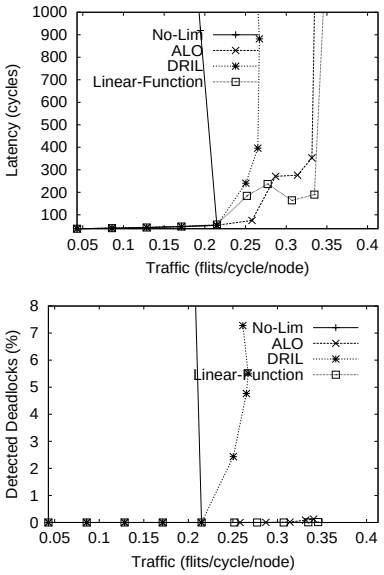

Figure 8. Average message latency and percentage of detected deadlocks versus traffic. Complement traffic pattern. 16-flit messages.

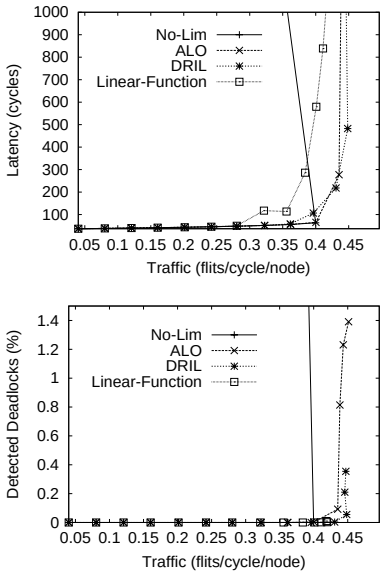

Figure 9. Average message latency and percentage of detected deadlocks versus traffic. Bit-reversal traffic pattern. 16-flit messages.

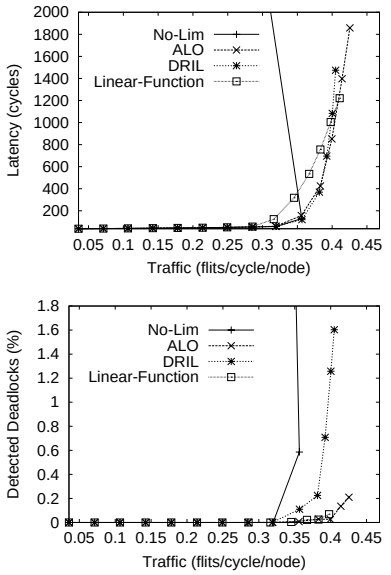

Figure 10. Average message latency and percentage of detected deadlocks versus traffic. Perfect-shuffle traffic pattern. 16-flit messages.