A Very Efficient Distributed Deadlock Detection Mechanism for Wormhole Networks *

P. López, J. M. Martínez, J. Duato
Grupo de Arquitecturas Paralelas
Depto. Ingeniería de Sistemas, Computadores y Automática
Universidad Politécnica de Valencia
Camino de Vera s/n
46071 - Valencia, SPAIN
{plopez,jmmr,jduato}@gap.upv.es

Abstract

Networks using wormhole switching have traditionally relied upon deadlock avoidance strategies for the design of routing algorithms. More recently, deadlock recovery strategies have begun to gain acceptance. Progressive deadlock recovery techniques are very attractive because they allocate a few dedicated resources to quickly deliver deadlocked messages, instead of killing them. However, the distributed deadlock detection techniques proposed up to now detect many false deadlocks, especially when the network is heavily loaded and messages have different lengths. As a consequence, messages detected as deadlocked may saturate the bandwidth offered by recovery resources, thus degrading performance considerably. In this paper we propose an improved distributed deadlock detection mechanism that uses only local information, detects all the deadlocks, considerably reduces the probability of false deadlock detection and is not strongly affected by variations in message length and message destination distribution.

1. Introduction

Many research papers have been devoted to deadlock handling in interconnection networks. A deadlock occurs when some packets cannot advance toward their destination because all of them are requesting resources held by other packets in a cyclic way. The size of the buffers strongly influences the probability of reaching a deadlocked configuration. In wormhole switching, buffers are relatively small, being more deadlock-prone than packet switching. There are three strategies for deadlock handling [8]: deadlock prevention, deadlock avoidance and deadlock recovery.

In deadlock prevention all the required resources are reserved before starting packet transmission. This is the case for all the variants of circuit switching when the message header is allowed to backtrack. In deadlock avoidance, resources are requested as packets advance through the network, granting them to a packet only if the resulting global state is safe. A common technique consists of establishing an ordering between resources and granting resources to each packet in decreasing order. This is the case for routing algorithms that restrict routing so that there are no cyclic dependencies between channels [5]. A more efficient approach that also fits into this category consists of allowing the existence of cyclic dependencies between channels while providing some escape paths to avoid deadlock, therefore increasing routing flexibility [6, 7].

Deadlock recovery strategies allow the use of unrestricted fully adaptive routing, potentially outperforming deadlock avoidance techniques. However, these strategies require a deadlock detection mechanism and a deadlock recovery mechanism that is able to recover from deadlocks faster than they occur. Otherwise, performance will degrade considerably. If a deadlock is detected, the recovery mechanism is triggered, resolving the deadlock. Progressive recovery techniques deallocate buffer resources from other "normal" packets and reassign them to deadlocked packets for quick delivery. Regressive techniques deallocate resources from deadlocked packets by killing and later reinjecting them at the original source node (i.e., abort-and-retry). The frequency of packets recovering from deadlock plays an important role when considering the deadlock handling mechanism.

In [17, 15], the frequency of deadlock occurrence on k-ary n-cubes using true fully adaptive minimal routing was

*This work was supported by the Spanish CICYT under Grant TIC97–0897–C04–01.

measured, showing that deadlocks rarely occur when sufficient routing freedom is provided but are more likely to occur when the network is close to or beyond saturation. Moreover, the use of techniques that avoid network saturation such as the message injection limitation mechanisms proposed in [11, 12] reduces the probability of deadlock to negligible levels, even when fully adaptive routing is used with only a few virtual channels.

Deadlock detection is a key design issue in any deadlock recovery mechanism. The main goal of the deadlock detection mechanism is to realize this infrequent occurrence of deadlock by detecting only true deadlocks, not congestion disguised as "false" deadlock. This then effectively allows a variety of deadlock recovery techniques to be viable, particularly those that depend on infrequent invocation of the recovery mechanisms, such as the software-based progressive deadlock recovery strategy proposed in [13].

A deadlock detection mechanism should also have two important features. First, it should be simple without adding needless complexity to the network that could reduce performance and/or increase cost. Second, it should be implemented as a distributed mechanism, working only with local information available at each router.

Several heuristic deadlock detection mechanisms have been proposed in the literature. In [16] a packet is considered to be deadlocked when the time since it was injected is longer than a threshold. In [10], a deadlock is detected if the time since the last flit was injected exceeds a threshold. In both cases, deadlocks are detected at the source node. Deadlocks can also be detected at the node containing the header. In Disha [2, 3], deadlocks are detected at the node containing the header by measuring the time that the header is blocked.

The distributed deadlock detection mechanisms mentioned above may not detect deadlocks immediately. Also, they may indicate that a packet is deadlocked when it is simply waiting for a channel occupied by a long packet (i.e., false deadlock detection). Moreover, when several packets form a deadlocked configuration, it is sufficient to release the buffer occupied by only one of the packets to break the deadlock. However, because heuristic detection mechanisms operate locally and in parallel, several nodes may detect deadlock concurrently and release several buffers in the same deadlocked configuration. In the worst case, it may happen that all the packets involved in a deadlock release the buffers they occupy. These overheads suggest that deadlock recovery-based routing can benefit from highly selective deadlock detection mechanisms.

The most important limitation of heuristic deadlock detection mechanisms based on timeouts arises when packets have different lengths. The optimal value of the timeout for deadlock detection heavily depends on packet length unless some type of physical channel monitoring of neighboring nodes is implemented. When a packet is blocked waiting for channels occupied by long packets, the selected value for the timeout should be high in order to minimize false-deadlock detection. As a consequence, deadlocked packets have to wait for long until deadlock is detected. In these situations, latency becomes much less predictable. The poor behavior of current deadlock detection mechanisms considerably limits the practical applicability of deadlock recovery techniques.

In [13] we proposed a deadlock detection technique based on monitoring the activity of all the output physical channels that can be used by a given blocked message. It considerably reduces the number of false deadlock detections over crude timeouts. However, this mechanism is still sensitive to message length.

In this paper, we propose a very efficient mechanism for deadlock detection. The proposed technique is neither strongly affected by the message destination distribution nor the message length. In addition, it better distinguishes among true deadlocked messages and messages blocked for long, releasing only the buffer occupied by one of the packets that causes deadlock in most cases. The main contribution of the paper is the proposal of a new deadlock detection mechanism that considerably reduces the probability of false deadlock detection over previously proposed mechanisms. Thus, it effectively enables the use of simple recovery handling mechanisms and the exploitation of the flexibility of true fully adaptive routing.

The rest of the paper is organized as follows. Section 2 describes the mechanism proposed in [13] to detect deadlocks, on which the new mechanism is loosely based. This mechanism is described in Section 3. The new mechanism is evaluated for different message destination distributions and message lengths in Section 4, comparing the results with previous proposals. Finally, in Section 5, some conclusions are drawn.

2. Deadlock Detection Mechanism: Previous Approach

In this section we summarize the deadlock detection mechanism proposed in [13], highlighting its main drawbacks. This mechanism is the basis for the new one described in the next section.

The mechanism is distributed and uses a heuristic scheme in each node containing a blocked header. It is based on measuring the time that channels requested by blocked messages are inactive due to the current messages occupying them remaining blocked. Transmission activity is monitored in all the output channels that can be used by a given blocked message. A message is only presumed to be deadlocked if all the alternative virtual output channels that are requested by that message contain blocked messages.

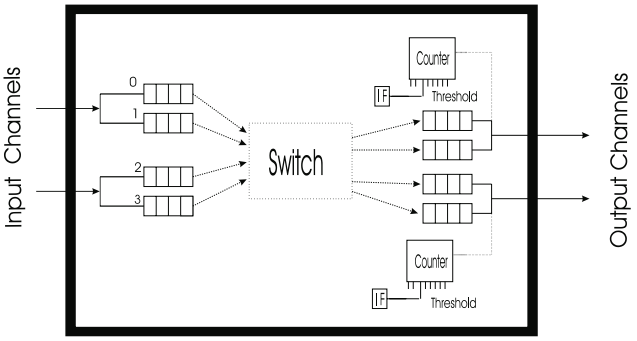

Figure 1. Implementation of our previous deadlock detection mechanism.

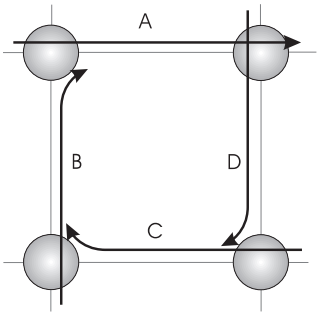

Figure 2. Messages B, C, and D are blocked (but not deadlocked).

It should be noted that when the routing algorithm uses all the virtual channels in each physical channel in the same way, it is only necessary to monitor activity in the physical channels. This is the case for true fully adaptive routing [17, 15, 13].

The hardware required to implement the mechanism consists of a counter, a comparator and a single-bit latch associated with each physical output channel, as shown in Figure 1. The counter is incremented every clock cycle, and it is reset when a flit is transmitted across the physical channel. Thus, it contains the number of cycles that this channel is inactive (the number of cycles since last flit transmission across any of the virtual channels in that physical channel). If this time is greater than a given threshold, a one-bit flag (inactivity flag, IF) will be set indicating that the physical output channel is inactive for a timeout period. This flag is reset when a flit is transmitted across the physical channel. Every time a message is routed, if all the feasible output channels are busy, then the IFs associated with the corresponding physical output channels are checked. If all of these flags are set, then there is no activity through any of the feasible physical output channels, and the message is presumed to be involved in a deadlock. Note that blocked messages are repeatedly routed until a free channel is found or a deadlock is detected.

Although this mechanism improves detection over crude timeout mechanisms [13], it has several drawbacks. First, the threshold used in the comparison still depends on message length, leading to splitting messages into fixed-length packets which, in turn, may increase overhead [9]. Second, this mechanism will mark as deadlocked all the messages involved in a deadlock. Thus, recovery will be performed on all these messages if a concurrent recovery scheme is used [4, 13]. However, in order to break the cycle, it is sufficient to perform recovery on only one of the messages involved in the deadlock. Thus, this detection mechanism adds needless overhead to recovery. Third, this mechanism is unable to precisely distinguish between true deadlocked messages and messages that are simply waiting for a channel to be freed. For instance, in Figure 2, a configuration is shown in which three messages B, C, and D are blocked (but not deadlocked). As message A is advancing, the IF associated with the channel occupied by A is not set, and message B is not detected as deadlocked. However, as message B is blocked, the IF of the channel occupied by it will be set. As a consequence, message C will be detected as deadlocked. A similar situation may occur for message D. Although there was no true deadlock, the mechanism detected it, leading to false deadlock detection and recovery by two packets.

3. New Deadlock Detection Mechanism

In this section we present a new mechanism for deadlock detection that better distinguishes between true and false deadlocks. In addition, when a deadlock is detected, it marks as deadlocked only the message that is at the root of the tree of blocked messages in most cases, noticeably reducing the number of deadlocked packets that become eligible to recover, and, thus, deadlock recovery overhead. As a by-product, the mechanism should depend neither on message length nor on message destination distribution.

The basic idea for the new deadlock detection mechanism is as follows: When several messages block waiting for resources occupied by a message that is advancing, those messages form a tree (possibly with only one branch). The message that is advancing is the root of the tree. When the root releases the resources occupied by it, then the remaining messages will be able to advance and there is no deadlock. However, if the root blocks waiting for resources occupied by another message in the tree, then a deadlock occurs. Therefore, a deadlock detection mechanism should propagate information along the branches of the tree of blocked messages indicating that there is no deadlock. However, if the root message blocks, it should

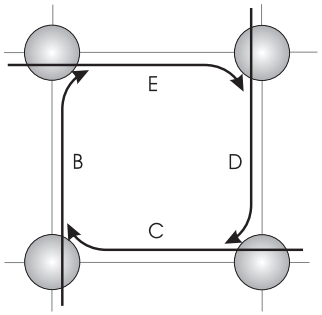

Figure 3. Messages E, B, C, and D are deadlocked.

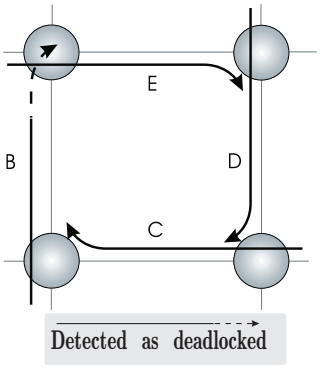

Figure 4. Message B triggers the recovery mechanism.

be able to detect whether the requested resource is held by another message in the tree.

The mechanism described above requires sending information along the channels occupied by messages in the tree. We propose an heuristic scheme that does not propagate that information. Instead, it uses the information provided by the flow control signals already used by the switching technique: if a message blocks waiting for resources occupied by a blocked message, it should not detect deadlock because the latter message is not the root of the tree. However, if a message blocks waiting for resources occupied by a non-blocked message which later becomes blocked, it should detect deadlock because the latter message could be the root. Let us show how this heuristic scheme works by using some examples.

Consider again the configuration shown in Figure 2. Message B is not marked as deadlocked because message A is advancing. Message A is in the root position of the tree. When message C requests a channel occupied by message B, it detects that B is not advancing. Message B was already blocked when C arrived. Thus, there is no need to mark C as deadlocked because there is another message that blocked previously. The same happens with message D. Hence, no deadlock is detected.

Consider now that message A leaves the channel it occupied, and that a newly arrived message E occupies that channel. Consider also that E requests a channel occupied by D and a true deadlock is formed, as shown in Figure 3. As stated above, neither message C nor message D become eligible to recover from deadlock although they are involved in the deadlock. Message E will not detect deadlock either, because message D was blocked prior to its arrival. Thus, the deadlock should be detected by message B. Remember that when message B arrived at the node containing its header, the message using the channel it requested was advancing. Now message E replaced message A, thus becoming the root of the tree. When E blocks and the time that message B is waiting exceeds a given threshold, a deadlock

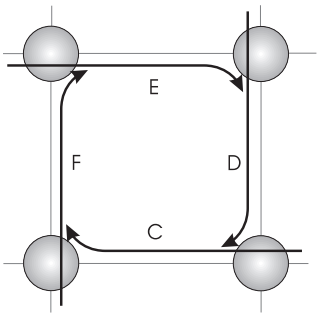

Figure 5. Another deadlock is formed.

will be detected and the recovery mechanism will be triggered only in the node containing the header of B (Figure 4). As a consequence, the deadlock is removed.

However, this is not enough. As shown in Figure 5, another message F may appear in the scenario, acquiring the channel that message B has just freed. If this message requests a channel occupied by message E, another deadlock is reached. Also, when message F blocks, message E was already blocked. Thus, message F will not detect the deadlock. However, when the first flit of F is transmitted across the channel requested by message C, this message detects the transmission and labels message F as the root. Thus, when message F blocks and a new deadlock is reached, message C will now detect deadlock and trigger the recovery mechanism.

It may happen that several messages involved in a deadlock block simultaneously. In this case, deadlock is detected by several messages, because they are blocked by another message that is still advancing. Though recovery is invoked, the overhead is higher that in the other cases. This case is expected to be infrequent because it requires several messages arriving simultaneously. This cannot happen in a congested network and, as indicated in [15, 17], most deadlocks

occur when the network is saturated.

It may also happen that several trees of blocked messages exist at a given time, and the root messages of some of those trees block waiting for resources occupied by messages in another tree. This case has the same consequence as when several messages block simultaneously: more than a single message will be labeled as deadlocked.

When several virtual channels per physical channel are considered, deadlock can only occur when every virtual channel in each physical channel involved in the deadlocked configuration is busy. This is only true when the routing algorithm uses all the virtual channels in the same way, as is the case for true fully adaptive routing [17, 15, 13]. Therefore, the deadlock detection hardware should only work when all the virtual channels in a given physical channel are busy.

The proposed deadlock detection mechanism can be implemented as follows (Figure 6). Each physical output channel has an associated counter and two one-bit flags: I (Inactivity) and DT (Deadlock Threshold). The counter is incremented every clock cycle that the channel is idle. Thus, the counter contains the number of cycles that this channel remains inactive. Note that the counter is only incremented if at least one virtual channel is occupied by a message. This value is continuously compared with two thresholds $t1$ and $t2$ ($t1 << t2$). If the counter value is greater than threshold $t1$, the I flag will be set. If the counter value is greater than threshold $t2$, the DT flag will be set. The counter and the flags are reset when a flit is transmitted across the physical channel. Note that a physical channel may be split into several virtual channels. However, this implementation assumes that all the virtual channels in each physical channel can be used in the same way, as mentioned above.

The threshold $t1$ is set to a very low value (for instance, only one clock cycle). Thus, the I flag will be set as soon as none of the messages is advancing across the channel. This flag is used for the purpose of testing if the messages occupying the virtual channels requested by a newly arrived message are already blocked. In other words, by testing the I flags the router knows if the newly arrived message is the first in a branch of the tree of blocked messages. The result of this test is stored into another one-bit flag G/P (Generate/Propagate), which is associated with each physical input channel. The threshold $t2$ will be used to detect deadlocked messages. The associated DT flag is equivalent to the inactivity flag of the mechanism proposed in [13] and summarized in Section 2. The value of $t2$ must be properly tuned to achieve the best results (see Section 4).

This hardware works as follows. On the first unsuccessful attempt at routing a message, if any virtual channel of the physical input channel containing the message header is free then this message cannot produce deadlock yet. Thus, the G/P flag is set to P. This is the flag value that prevents

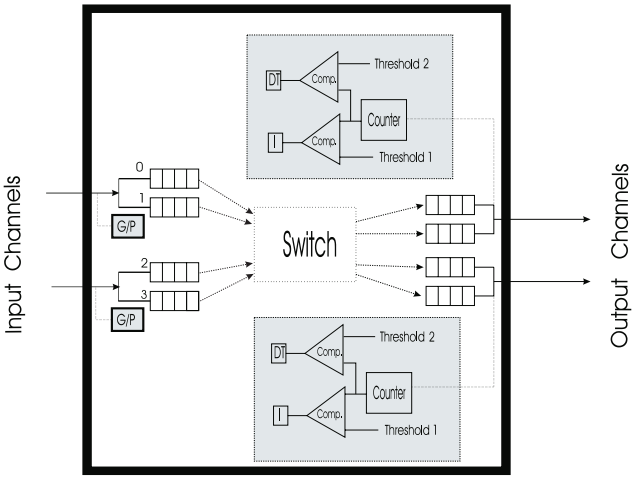

Figure 6. Implementation of the new deadlock detection mechanism.

deadlock detection. If all the virtual channels of the physical input channel are busy (i.e., the current message is the latest in arriving across the physical channel), all the feasible output channel I flags are tested. There are two cases:

- If one or more I flags are not set, it means that some messages are advancing across the corresponding channels. Thus, the G/P flag is set to G, indicating that those messages could be the root of the tree of blocked messages.
- If all I flags are set (I=1), it means that the corresponding channels are busy and the messages occupying them are blocked. A possible deadlock may be forming but the current message is not waiting on the root of the tree. Thus, the G/P flag is set to P.

For every unsuccessful successive attempt at routing the blocked message, the DT flags associated with all the feasible output channels and the G/P flag associated with the physical input channel containing the header are checked. The following cases exist:

- If one or more DT flags are not set, it means that channels are busy but some messages advanced across them during the last $t2$ clock cycles. Thus, no deadlock configuration exists yet. Hence, the message should wait until an output channel becomes free.
- If all the DT flags are set (DT=1) and the flag G/P=G, it means that the threshold $t2$ has been exceeded, and the current message is presumed to be involved in a deadlock. Because the G/P flag is set to G, that message is the first of a branch in the tree of possibly deadlocked messages. Therefore, the message is marked as

	Injection rate (flits/cycle/node)															
	0.428				0.471				0.514				0.600 (saturated)			
M. Size	s	l	L	sl	s	l	L	sl	s	l	L	sl	s	l	L	sl (*)
Th 2	.055	.191	.295	.299	.199	.662	1.08	1.03	.605	2.37	4.61	4.86	26.0	30.5	33.4	36.0
Th 4	.000	.014	.025	.033	.023	.043	.088	.094	.100	.205	.335	.736	13.1	7.75	6.64	13.4
Th 8	.000	.003	.010	.005	.007	.011	.026	.036	.020	.095	.115	.355	8.58	5.07	3.95	9.87
Th 16	.000	.003	.010	.005	.004	.007	.026	.024	.000	.072	.115	.260	5.45	4.42	3.83	8.32
Th 32	.000	.002	.010	.005	.000	.005	.023	.013	.000	.050	.110	.155	2.96	3.24	3.66	5.87
Th 64	.000	.000	.010	.001	.000	.004	.021	.005	.000	.012	.090	.038	1.71	1.63	3.30	3.20
Th 128	.000	.000	.005	.001	.000	.002	.018	.000	.000	.002	.070	.008	1.24	.350	2.50	1.57
Th 256	.000	.000	.005	.000	.000	.000	.005	.000	.000	.000	.045	.000	.840	.020	1.27	1.01
Th 512	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.005	.000	.400	.000	.290	.680
Th 1024	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.002	.000	.110	.000	.020	.290

Table 1. Percentage of messages detected as possibly deadlocked for the previously proposed detection mechanism. True fully adaptive routing algorithm with 3 virtual channels per physical channel. Uniform distribution of message destinations.

deadlocked in order to activate the deadlock recovery mechanism.

- If all the DT flags are set (DT=1) and the flag G/P=P, it also means that the current message is presumed to be involved in a deadlocked configuration but it is not the first message in a branch. Therefore, it is not marked as deadlocked. It must wait until the deadlock is broken because some input channel of another node will have its G/P flag set to G.

When any of the messages that were occupying the virtual channels of an input physical channel with the G/P flag set to G is successfully routed, the corresponding G/P flag should be reset to P, because the last message that arrived at the physical channel is no longer waiting on the root of the tree. The same happens when any of these virtual channels is freed.

Finally, in order to deal with the situation presented in Figure 5, when an I flag is reset because one of the messages that occupy its associated output channel advances, the G/P flags of those channels containing messages waiting for that output channel should be set to G. A simple implementation of this mechanism consists of changing all the P flags in a router to G when any I flag is reset. This solution may lead to an increase in the number of false deadlocks detected with respect to a more selective change. We are currently studying this issue.

The threshold value $t2$ plays an important role in the detection mechanism. Thus, it must be properly tuned. To be marked as deadlocked, a message must exceed threshold $t2$ (DT=1) and be the first message in one of the branches of the tree of blocked messages (G/P=G). Remember that the second condition means that some messages were advancing across the requested channels when the message arrived at the current node. If this situation changes and all those messages block, threshold $t2$ is reached and a deadlock is detected at once. As threshold $t2$ measures the time relative to the last flit transmission, the consequence is that threshold $t2$ should not depend on message length and destination distribution. We will show this behavior in Section 4.

4. Evaluation of the New Mechanism

The evaluation methodology used is based on the one proposed in [13]. We will measure the number of deadlocks detected using the new mechanism for different load conditions, comparing it with the number of deadlocks detected with previously proposed mechanisms. In particular, we have considered several message destination distributions (uniform, with locality, bit-reversal, perfect-shuffle, butterfly and a hot-spot pattern obtained by modifying the uniform distribution in such a way that 5 % of the messages are destined for the same node) and message lengths (16-flit messages (s), 64-flit messages (l), 256-flit messages (L) and a hybrid load (sl) composed of 60 % of 16-flit and 40 % of 64-flit messages). Finally, taking into account that deadlocks are more frequent at high loads, several network traffic loads near saturation have been analyzed.

4.1. Network Model

The network model considered consists of a four port architecture [14], a true fully adaptive router [17, 15, 13] with three virtual channels per physical channel, a crossbar switch and channels. Routing time and transmission time across the crossbar and a channel are all assumed to be equal to one clock cycle. There is also a four-flit buffer associated with each virtual channel. The deadlock detection

	Injection rate (flits/cycle/node)															
	0.428				0.471				0.514				0.600 (saturated)			
M. Size	s	l	L	sl	s	l	L	sl	s	l	L	sl	s	l	L	sl (*)
Th 2	.000	.021	.055	.028	.015	.069	.123	.086	.045	.097	.555	.513	2.40	3.75	4.33	3.92
Th 4	.000	.000	.005	.001	.001	.005	.000	.002	.000	.002	.125	.045	.830	.551	.412	.900
Th 8	.000	.000	.000	.000	.000	.001	.000	.002	.000	.000	.005	.020	.417	.283	.178	.560
Th 16	.000	.000	.000	.000	.000	.000	.000	.001	.000	.000	.005	.010	.205	.218	.168	.447
Th 32	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.005	.006	.069	.138	.159	.280
Th 64	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.005	.001	.035	.054	.132	.100
Th 128	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.002	.000	.027	.011	.084	.040
Th 256	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.002	.000	.015	.002	.037	.030
Th 512	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.005	.000	.009	.017
Th 1024	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.007

Table 2. Percentage of messages detected as possibly deadlocked for the new detection mechanism. True fully adaptive routing algorithm with 3 virtual channels per physical channel. Uniform distribution of message destinations.

	Injection rate (flits/cycle/node)											
	1.429			1.571			1.857 (saturated)			2.000 (saturated)		
M. Size	s	l	sl	s	l	sl	s	l	sl	s	l	sl
Th 2	.002	.000	.015	.012	.007	.020	.030	.037	.052	.050	.049	.052
Th 4	.000	.000	.007	.002	.000	.010	.013	.012	.018	.013	.019	.018
Th 8	.000	.000	.007	.000	.000	.005	.007	.011	.017	.009	.017	.017
Th 16	.000	.000	.002	.000	.000	.000	.003	.006	.009	.005	.013	.009
Th 32	.000	.000	.002	.000	.000	.000	.000	.004	.004	.001	.005	.004
Th 64	.000	.000	.002	.000	.000	.000	.000	.001	.001	.000	.000	.001
Th 128	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 3. Percentage of messages detected as possibly deadlocked for the new detection mechanism. True fully adaptive routing algorithm with 3 virtual channels per physical channel. Uniform distribution of message destinations with locality.

technique is the one proposed in Section 3, with threshold $t1$ equal to 1 clock cycle. Finally, the message injection limitation mechanism proposed in [11] is used in order to avoid the performance degradation of the network when it reaches saturation and also to decrease the effective deadlock frequency. This mechanism allows injection of a newly generated message into the network only if the number of busy virtual output channels at the current node does not exceed a given threshold.

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

4.2. Frequency of Deadlock Detection

In this section, we will show the percentage of messages detected as deadlocked using the new deadlock detection mechanism (NDM) proposed in this paper for threshold values ranging from 2 (Th 2) to 1024 (Th 1024) clock cycles and several load conditions. We have also evaluated the behavior of the previously proposed deadlock detection mechanism (PDM) [13] and timeout-based mechanisms for comparison purposes. However, considering that PDM already reduced the frequency of false deadlock detection over crude timeouts by a factor of 10 [13], we have not included comparisons against crude timeouts. On the other hand, due to the lack of space, we will only show the results for the PDM for the uniform distribution of message destinations. The conclusions obtained by analyzing this distri-

	Injection rate (flits/cycle/node)											
	0.352			0.386			0.421			0.451 (saturated)		
M. Size	s	l	sl	s	l	sl	s	l	sl	s	l	sl
Th 2	.004	.006	.013	.011	.013	.065	.129	.041	.292	.638	.346	1.14
Th 4	.001	.000	.003	.001	.001	.005	.024	.000	.041	.148	.038	.223
Th 8	.000	.000	.000	.000	.000	.002	.003	.000	.012	.041	.005	.090
Th 16	.000	.000	.000	.000	.000	.002	.001	.000	.009	.026	.004	.070
Th 32	.000	.000	.000	.000	.000	.002	.001	.000	.007	.009	.001	.043
Th 64	.000	.000	.000	.000	.000	.001	.000	.000	.003	.002	.000	.019
Th 128	.000	.000	.000	.000	.000	.000	.000	.000	.001	.000	.000	.002
Th 256	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 4. Percentage of messages detected as possibly deadlocked for the new detection mechanism. True fully adaptive routing algorithm with 3 virtual channels per physical channel. Bit-reversal pattern of message destinations.

	Injection rate (flits/cycle/node)											
	0.214			0.250			0.286			0.320 (saturated)		
M. Size	s	l	sl	s	l	sl	s	l	sl	s	l	sl
Th 2	.000	.000	.002	.003	.006	.010	.095	.060	.118	.581	.571	.887
Th 4	.000	.000	.000	.000	.000	.000	.020	.010	.020	.292	.177	.304
Th 8	.000	.000	.000	.000	.000	.000	.015	.000	.013	.167	.122	.208
Th 16	.000	.000	.000	.000	.000	.000	.010	.000	.009	.117	.107	.169
Th 32	.000	.000	.000	.000	.000	.000	.000	.000	.006	.073	.090	.124
Th 64	.000	.000	.000	.000	.000	.000	.000	.000	.004	.032	.061	.089
Th 128	.000	.000	.000	.000	.000	.000	.000	.000	.003	.014	.035	.053
Th 256	.000	.000	.000	.000	.000	.000	.000	.000	.000	.003	.013	.020
Th 512	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.004	.006
Th 1024	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 5. Percentage of messages detected as possibly deadlocked for the new detection mechanism. True fully adaptive routing algorithm with 3 virtual channels per physical channel. Perfect-shuffle traffic pattern of message destinations.

bution are also applicable to the other message destination distributions analyzed.

Table 1 shows the percentage of packets invoking the recovery mechanism for the PDM. Tables 2 through 7 show the same results for the NDM for all the message destination distributions analyzed. It must be noticed that in some cases several actual deadlocks have been detected. These cases have been marked with (*) in the tables.

As we can see, the NDM considerably reduces the number of false deadlock detections over the PDM. In particular, this number is reduced on average by a factor of 10, which implies improving detection over crude timeout by two orders of magnitude. This reduction leads to an important decrease in the overhead of deadlock recovery.

On the other hand, the behavior of the PDM is affected by message size. When the network is below saturation, the threshold value has an approximately linear dependence with message size. Hence, in order to reduce false deadlock detection, message size dependent thresholds must be used. When the network is deeply saturated, the PDM shows an odd behavior, requiring a large threshold to avoid false deadlock detection for some cases. In the case of the NDM, the threshold required is less dependent on message length, even when the network is deeply saturated. However, the percentage of false deadlock detections increases when the network is saturated. This behavior is almost similar for all the message destination distributions analyzed, with the exception of the hot-spot one, where the percentage of false deadlock detections increases just before entering saturation. The reason is that around the hot-spot zone,

	Injection rate (flits/cycle/node)											
	0.107			0.118			0.129			0.139 (saturated)		
M. Size	s	l	sl	s	l	sl	s	l	sl	s (*)	l	sl (*)
Th 2	.007	.006	.089	.033	.015	.300	.296	.092	1.22	2.70	.920	4.60
Th 4	.000	.000	.006	.000	.000	.032	.030	.004	.261	.885	.116	1.94
Th 8	.000	.000	.000	.000	.000	.004	.005	.001	.102	.437	.026	1.38
Th 16	.000	.000	.000	.000	.000	.003	.000	.000	.084	.298	.018	1.23
Th 32	.000	.000	.000	.000	.000	.002	.000	.000	.063	.191	.015	1.03
Th 64	.000	.000	.000	.000	.000	.001	.000	.000	.029	.103	.011	.785
Th 128	.000	.000	.000	.000	.000	.001	.000	.000	.013	.075	.004	.420
Th 256	.000	.000	.000	.000	.000	.000	.000	.000	.004	.067	.000	.230
Th 512	.000	.000	.000	.000	.000	.000	.000	.000	.002	.065	.000	.155
Th 1024	.000	.000	.000	.000	.000	.000	.000	.000	.002	.065	.000	.145

Table 6. Percentage of messages detected as possibly deadlocked for the new detection mechanism. True fully adaptive routing algorithm with 3 virtual channels per physical channel. Butterfly traffic pattern of message destinations.

the network is actually saturated.

Unlike the PDM, the NDM allows the choice of a constant and low threshold value, even when the network is deeply saturated, which leads to a quick detection of actual deadlocks while maintaining a low number of false deadlock detections. For instance, a threshold equal to 32 clock cycles achieves a false deadlock detection rate in the worst case (saturation) lower than 0.16 % of the messages transmitted during the simulation. The only exception is the hotspot distribution which raises to 0.26 %.

5. Conclusions

In this paper we proposed an improved deadlock detection mechanism that detects all true deadlocks while considerably reducing the probability of detecting false deadlocks over previous approaches. Indeed, that probability is reduced with respect to mechanisms based on timeouts by two orders of magnitude. The proposed mechanism uses simple hardware that is not in the critical path and takes into account only local information. Once the threshold has been tuned, this detection mechanism works correctly regardless of message length and message destination distribution. In addition, when a deadlocked configuration is reached, the mechanism tries to detect as deadlocked as few messages as possible in each cycle of blocked messages, noticeably reducing recovery overheads. In most cases, a single message is detected as deadlocked.

To the best of our knowledge, this is the only distributed deadlock detection mechanism proposed up to now that only uses local information and is relatively insensitive to message distribution and message length. This detection mechanism is the key to make deadlock recovery feasible in practice, therefore benefiting from the higher performance of deadlock recovery-based routing algorithms.

Acknowledgments

The authors thank Dr. T.M. Pinkston and the anonymous referees for their helpful and valuable suggestions that have improved the presentation and technical contents of this paper.

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. J. Dally and C. L. Seitz, “Deadlock-free message routing in multiprocessor interconnection networks,”

	Injection rate (flits/cycle/node)											
	0.0628			0.0707			0.0786			0.0862		
M. Size	s	l	sl	s	l	sl	s	l	sl	s (*)	l (*)	sl
Th 2	.008	.005	.010	.040	.007	.022	.140	.110	.120	.506	.442	.422
Th 4	.003	.002	.006	.035	.003	.018	.110	.090	.107	.456	.417	.395
Th 8	.003	.000	.004	.020	.003	.018	.100	.087	.101	.390	.400	.358
Th 16	.002	.000	.002	.015	.003	.013	.065	.077	.083	.320	.377	.335
Th 32	.001	.000	.001	.000	.003	.007	.020	.052	.060	.203	.347	.260
Th 64	.000	.000	.000	.000	.000	.002	.000	.032	.029	.090	.282	.267
Th 128	.000	.000	.000	.000	.000	.000	.000	.007	.010	.035	.167	.077
Th 256	.000	.000	.000	.000	.000	.000	.000	.005	.001	.016	.065	.017
Th 512	.000	.000	.000	.000	.000	.000	.000	.000	.000	.013	.010	.000
Th 1024	.000	.000	.000	.000	.000	.000	.000	.000	.000	.005	.002	.000

Table 7. Percentage of messages detected as possibly deadlocked for the new detection mechanism. True fully adaptive routing algorithm with 3 virtual channels per physical channel. Hot-spot traffic pattern of message destinations.

IEEE Transactions on Computers, vol. C–36, no. 5, pp. 547–553, May 1987.

[6] 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.

[7] 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.

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

[9] V. Karamcheti and A. A. Chien, "Do faster routers imply faster communication?," *Proceedings of the Workshop on Parallel Computer Routing and Communication*, pp. 1–15, May 1994.

[10] 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.

[11] 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.

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

[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 International Conference Parallel Processing*, Aug. 1997.

[14] 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*, Aug. 1992.

[15] T.M. Pinkston and S. Warnakulasuriya, "On Deadlocks in Interconnection Networks",*the 24th International Symposium on Computer Architecture*, June 1997.

[16] 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.

[17] S. Warnakulasuriya and T.M. Pinkston, "Characterization of deadlocks in interconnection networks," in *Proceedings of the 11th International Parallel Processing Symposium*, April 1997.