88-DE

ISMM

MINI AND MICROCOMPUTERS

From Micros to Supercomputers

MIMI '88

Miami Beach, Florida, U.S.A.
December 14-16, 1988

Editor: M.H. Hamza

A Publication of
The International Society for
Mini and Microcompters - ISMM

ISBN 0-88986-107-2

ACTA PRESS

ANAHEIM * CALGARY * ZURICH

Deadlock Elimination in Networks with Hamiltonian Circle

Naphtali Rishe and Qiang Li

School of Computer Science
Florida International University–
The State University of Florida at Miami
University Park, Miami, FL 33199, USA

Abstract

Deadlock handling in a packet-switching network results in complicated software and heavy overhead. This paper presets a mechanism which can eliminate deadlock in a wide range of packet-switching networks with reasonable cost. The principle requirement from the network is the existence of a Hamiltonian circle.

Keywords: Deadlock, Packet-switching network, Hamiltonian circle, Ring network.

1 Introduction

Deadlock is a problem existing in most store-and-forward packet- switching networks. The effort made to solve the problem is normally very costy in terms of extra software, heavy overhead, etc. General solutions to the deadlock problem are known[6,5,8,3,2,4]. This paper proposes a method to eliminate the deadlocks with a reasonable cost. Two conditions are assumed:

1. A Hamiltonian circle exists in the network. (A Hamiltonian circle is a loop passing through each node exactly once)
2. A timeout criterion is available for each node of the network., i.e., when the node has waited for another node to accept a message for a given period of time, the sending node can assume the possibility of a deadlock and abort the message. (The assumption needs not be correct in all the cases. However, to avoid the network's slowdown, the assumption must be correct in most cases.)

An example of a network which satisfies condition (1) is the Hypercube network. The n-dimensional hypercube [9] is defined as follows: Each of the 2^n nodes is labeled from 0 to 2^{n-1} by a unique binary string of length n. Two nodes are connected iff they differ in exactly one bit position. Figure 1 shows a 3-dimensional hypercube. The thick arcs indicate one of the Hamiltonian circles. The sequence of the nodes in the circle is

(000, 001, 011, 010, 110, 111, 101, 100)

The above sequence forms a ring-shaped subnetwork. In general, Condition (1) suggests that there is a ring-shaped subnetwork which spans all the nodes of the network.

This research has been supported in part by a grant from the Florida High Technology and Industry Council.

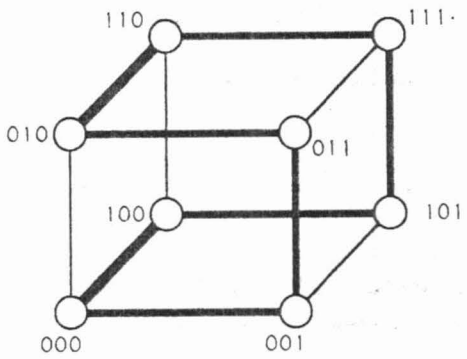

Figure 1: A 3-dimensional Hypercube

2 Deadlock prevention in a ring network

First we consider a trivial case of a network with Hamiltonian circle: the ring network, i.e., there are no arcs in addition to those in the Hamiltonian circle. A ring network in general can be either one-directional or two-directional. We will only concern with the one-directional ring network, since a two-directional ring can be considered as two one-directional rings physically overlapped if there are no messages being sent from node A to an adjacent node B and being redirected back to node A immediately by node B.

If each node in such a ring has at least two communication buffers of maximum message length, an efficient algorithm has been developed to guarantee a deadlock-free communication with very little cost. The details can be found in [7]. The following is a brief description of the algorithm. Figure 2 shows the block structure of a ring network. Note that each node of the ring has two buffers, namely, the communication buffer and the contingency buffer. They have different functions. Messages originated by the local processes or coming from the previous node in the ring will be put into the communication buffer first. In the case that the communication buffer is occupied, the newly generated message by the local processes will be rejected (held by the local processes) even when the communication buffer is empty. On the other hand, a message coming from the previous node will

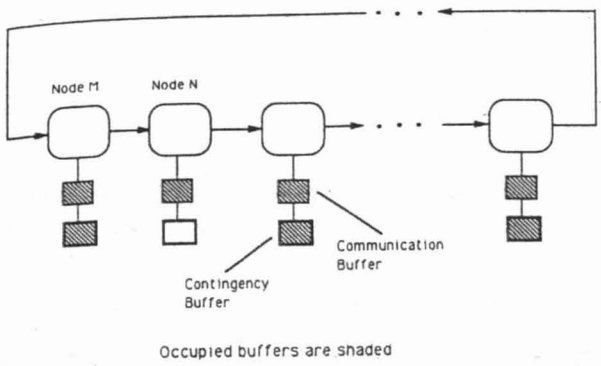

Figure 2: The Deadlock-free Ring Network

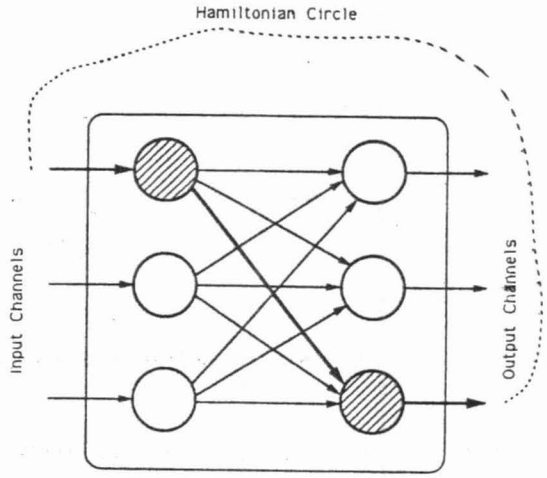

Figure 3: Communication Processes in a Node. The buttons represent the communication processes

be put into the contingency buffer if the communication buffer is full and the contingency buffer is empty. If both buffers are full, the message will be rejected (held by the previous node). If the contingency buffer is full when the communication buffer becomes empty, the message in the contingency buffer will be moved to the communication buffer before any new message can be accepted.

With such two-buffer architecture, the ring network is deadlock-free. Deadlock can only happen when both buffers of each node of the ring are full. We can prove that this can not happen by contradiction. Suppose there was a time instance t such that the last empty contingency buffer, say the contingency buffer of node N shown in Figure 2, was filled at t, and from t on, all buffers of the ring were full. Note that, if node M is the node immediately in front of node N, the contingency buffer of node N can only be filled by a message from the previous node, M. Therefore, for the contingency buffer of node N to be filled, there must be a message being sent from M to N. But then, M will have its contingency buffer empty. Therefore, the time instance t does not exits.

This algorithm introduces almost no overhead for the deadlock prevention since the algorithm works the same way as a normal buffered store-forward network except that the locally generated messages can not fill the last buffer left.

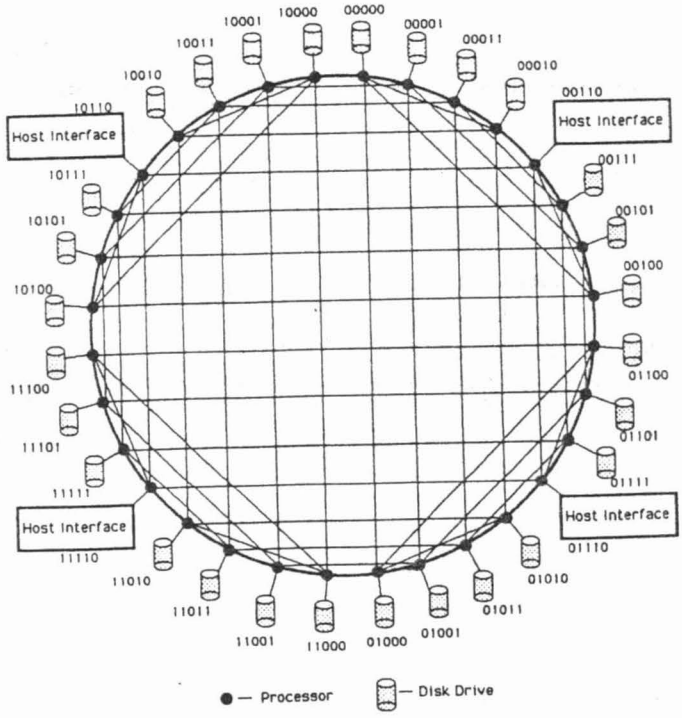

Figure 4: LSDM Hypercube Network. 32 nodes are depicted.

3 Algorithm for deadlock elimination in arbitrary networks with Hamiltonian circle

Most of the networks in the real world have far more complicated topologies than the ring network. Even though a ring, that is, a Hamiltonian circle, can be identified in most cases, it makes no sense to force all the messages to be sent through the ring in order to avoid the deadlock.

An efficient way of eliminating deadlock is as follows. Each node in the communication network is identified as a node in the Hamiltonian circle. Each communication channel connecting to a node is handled by an independent process on the node. Figure 3 shows a node with three input channels and three output channels. There is one process for each channel. The processes communicate through internal mechanism, e.g., shared memory. The thick arcs indicate that the path is a part of the Hamiltonian circle. The processes which handle the channels which are part of the Hamiltonian circle (the shaded processes) are different from the other processes. In addition to the normal protocol, those processes have to handle some special messages called contingency messages. Under the normal circumstances, the messages flow through the network according to the fastest routing algorithm as if there is no possible deadlock. The additional mechanism comes to play only when a deadlock is suspected. Suppose the nodes n_1 and n_2 are directly connected and n_1 desires to send a message to n_2. If n_2 is refusing to accept the message for the timeout period, the node n_1 assumes that the message may be causing a deadlock. Node n_1 will then abort the message and remove it from the buffer attached to the channel linked to n_2. Then, a contingency message is constructed, which has an envelope put on the message indicating that it should be routed through the Hamiltonian circle. The envelope also contains the originator and destination of the message and some other information. Since the Hamiltonian circle is deadlock-free, the message is guaranteed to be delivered.

Apparently, no message will wait forever in such a system, i.e., it is deadlock-free.

4 An application example

The problem whose solution is presented in this paper has arisen from our current research on a massively parallel database machine to be composed of thousands processors (INMOS Transputers[1]) and thousands of small secondary storage units, e.g., 20 Megabyte disk drives. Each of the disk drives is exclusively connected to a processor. The processors are connected into a hypercube network. Figure 4 shows a reduced version of the system. The circle in the figure is a Hamiltonian circle. An adaptive routing algorithm is used in the network. Deadlock could be possible in the network. With the algorithm discussed above, the deadlock can be eliminated.

5 Relaxation of the requirement

Strictly speaking, we do not need a Hamiltonian circle in the network, but merely a circle which visit every node at least once and does not use any arc more than once. (The strict Hamiltonian circle visits every node exactly once). This is because several independent processes serve different channels and thus have separate buffers for those channels. Thus, a message can flow through a node twice through different processes and buffers, which would simulate two separate nodes in a real Hamiltonian circle. Figure 5 shows a node which is visited twice in a loop.

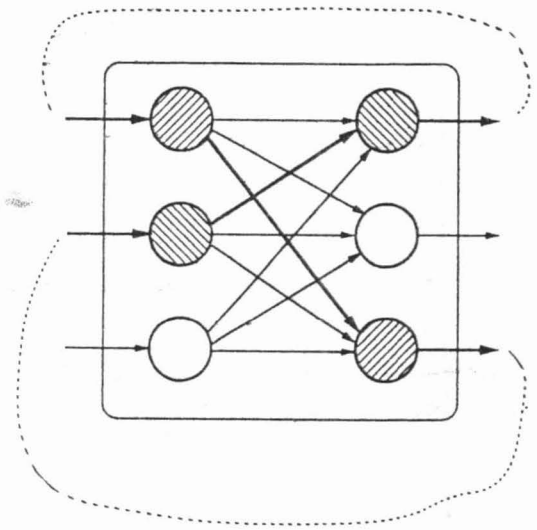

Figure 5: Relaxation of Hamiltonian circle requirement by making a Hamiltonian circle of processes rather than processors

6 Conclusion

The modified routing algorithm is deadlock-free at the expense of routing inefficiently only a small subset of messages, while the bulk of messages are routed very efficiently. The length of the waiting period before timeout is a critical factor. A too short waiting time can cause congestion on the Hamiltonian circle while a too long waiting time will result in a slow resolution of a deadlock situation. The actual timeout period can be fine tuned according to the characteristics of a particular system.

To avoid flooding the Hamiltonian circle, the system should be designed to work far from the possible deadlock situation. Normally, more communication buffers on each nodes can make the network more resilient, but it may result in a longer average time for a message to stay in the system. Again, compromise has to be made.

Many of the networks widely used in communications between large numbers of processors contain Hamiltonian circles. Also, when a network does not have a Hamiltonian circle, it can normally be modified or expanded to contain one with a reasonable cost.

Acknowledgement

The authors gratefully acknowledge the advice of David Barton, Nagarajan Prabhakaran, Doron Tal, and Wan-lai Chen.

References

[1] INMOS Corporation. *Transputer Architecture Reference Man-*

ual. INMOS Corporation, Bristol, U.K., 1986.

[2] D. Gelernter. A DAG-based algorithm for prevention of store-and-forward deadlo ck in packet networks. *IEEE Transactions on Computers*, C-30:709–715, Oct. 1981.

[3] I.S. Gopal. Prevention of store-and-forward deadlock in computer networks. *IEEE Transactions on Communications*, COM-33(12):1258–1264, 1985.

[4] K.D. Gunther. Prevetion of deadlocks in packet-switched data transport system. *IEEE Transactions on Communications*, COM-29:512–524, April 1981.

[5] J.W. Havender. Avoiding deadlock in multi-tasking systems. *IBM Systems Journal*, 74–84, 2 1968.

[6] E.G. Coffman Jr., M.J. Elphick, and A. Shoshani. System deadlocks. *Computer Survey*, 3(2):67–78, June 1971.

[7] Q Li, W.B. Feild, and D. Klein. Implementation of a transputer ring network and a deadlock prevention algorithm. In *The 3rd US Occam User Group Meeting*, Chicago, IL, Sept 1987.

[8] A. Shoshani and E.G. Coffman. Prevention, detection, and recovery from system deadlocks. In *Proc. of 4th Annual Princeton Conference on Information Science and Systems*, March 1970.

[9] L.G. Valiant and G.J. Brebner. Universal schemes for parallel communication. In *STOC, ACM Conference Proceedings*, Milwaukee, 1981.