Adaptive Routing Based on Deadlock Recovery

Nidhi Agrawal[1] and C.P. Ravikumar[2]

[1] Hughes Software Systems, Sector18, Electronic City, Gurgaon, INDIA
[2] Deptt. of Elec. Engg., Indian Institute of Technology, New Delhi, INDIA

Abstract. This paper presents a deadlock recovery based fully adaptive routing for any interconnection network topology. The routing is simple, adaptive and is based on calculating the probabilities of routing at each node to neighbors, depending upon the static and dynamic conditions of the network. The probability of routing to the i^{th} neighbor at any node is a function of the traffic and distance from the neighbor to the destination. Since with our routing algorithm deadlocks are rare, deadlock recovery is a better solution. We also propose here two deadlock recovery schemes. Since deadlocks occur due to cyclic dependencies, these cycles are broken by allowing one of the messages involved in deadlock to take an alternate path consisting of buffers reserved for such messages. These buffers can be centralized buffers accessible to all neighboring nodes or can be set of virtuals. The performance of our algorithm is compared with other recently proposed deadlock recovery schemes. The *2-Phase* routing is found to be superior compared to the other schemes in terms of network throughput and mean delay.

1 Introduction

Interprocessor communication is the bottleneck in achieving high performance in Message Passing Processors (MPPs). Various routing algorithms exist in the literature for wormhole routed[4] interconnection networks. Most routing algorithms achieve deadlock-freedom by restricting the routing, for example, Turn Model[5] router for hypercubes and meshes ensures deadlock-freedom by prohibiting certain turns in the routing algorithm. Many routing techniques achieve deadlock-freedom and adaptivity through the use of virtual channels [3,4]. Not only do virtual channels increase hardware complexity, they also reduce the speed of the router. Chien[2] has shown that virtual channels adversely affect the router performance, and multiplexing more than 2 to 3 virtual channels over one physical channel results in poor router performance. From the discussion above, it is clear that achieving deadlock-freedom and adaptivity at the cost of increased router complexity and increased delays is not a good solution.

In this paper we propose fully adaptive routing based on deadlock recovery. The algorithm breaks deadlocks without adding significant hardware. We introduce the notion of *Routing Probability* at each node. $P_i(j,k)$ denotes the probability of routing a message at node i to a neighboring node k, destined for j. Entirely chaotic routing can be achieved by selecting $P_i(j,k) = 1/d_i$. On

the other hand, a deterministic routing algorithm can be modeled by selecting $P_i(j,l) = 1$ for some $1 \leq l \leq d_i$, and $P_i(j,k) = 0$, $\forall k \neq l$; here d_i is the degree of node i. In this paper we propose a simple heuristic to calculate the routing probabilities at any node depending on the distance from the destination and/or congestion at the neighboring node. Our algorithm dynamically computes the probabilities, thus providing adaptivity. A message is declared to be deadlocked at any node i, if the message header has to wait at node i for more than a predefined time called *TimeOut*. Since deadlocks are infrequent, it is more logical to recover from deadlocks rather than avoiding. Various schemes for deadlock recovery are available in the literature[1, 6]. Compressionless routing[6] is a deadlock recovery scheme based on killing the deadlocked packets and transmitting again. This scheme requires *padding flits* to be attached with the message when the message length is less than the network diameter. Moreover, messages that are killed and rerouted suffer large delays. A deadlock recovery scheme called *Disha*[1] uses a single flit buffer at each node. The disadvantage of this scheme is that at any time only one message can be routed onto deadlock-free buffers, thus only one of the deadlocks we can recover from at a time; furthermore the token based mutual exclusion on the central virtual network is implemented by adding additional hardware.

We propose here two schemes for deadlock recovery. Our first scheme makes use of multiple central buffers; on the other hand our second scheme, also called *2-Phase Routing* divides the network into two virtual networks. Both the schemes eliminate the disadvantages associated with the previously proposed recovery schemes. Rest of the paper is organized as follows : Next section describes the routing algorithm. Section 3 describes the proposed deadlock recovery schemes. The results of simulation are given in Section 4. Section 5 concludes the paper.

2 Adaptive Routing

2.1 Algorithm

Various heuristics may be used to calculate the routing probabilities $P_i(j,k)$ introduced in Section 1. In this section we propose a simple heuristic as shown below :

$$\forall k \in Nbr(i), \qquad P_i(j,k) = \frac{\frac{1}{W_k}}{\sum_k \frac{1}{W_k}} \tag{1}$$

Here $Nbr(i)$ is the set of immediate neighbors of node i and W_k is a weight assigned to the neighbor k which is a linear function of the distance $\delta(j,k)$ and the congestion at the neighboring node k denoted as Q_k. The weight W_k is assigned as follows :

$$W_k = \alpha * \delta(j,k) + \beta * Q_k \tag{2}$$

Each node is assumed to have the knowledge of the fault-status of its neighbor and the congestion at each neighboring node. The congestion at any node k,

also denoted by Q_k, is the average amount of time a header has to spend at k normalized against a timeout period $TimeOut$. To measure the value of Q_k, node k makes use of as many timers as there are neighbors. If a header waits for more than $TimeOut$ period, the message is declared to be deadlocked.

For most regular interconnection networks, the normalized distance function $\delta()$ is easy to evaluate. The scaling factors α and β are non-negative real constants, which dictate the nature of the routing algorithm; by selecting $\beta = 0$, we obtain distance-optimal routing. Similarly, by selecting $\alpha = 0$, we obtain a hot-potato routing algorithm. We found optimum values of the ratio $\frac{\alpha}{\beta}$, by simulation, for uniform and bit-reversal traffic (see Section 4).

2.2 Detection of deadlocks

The selection of $TimeOut$ interval greatly affects the router performance. If the $TimeOut$ period is very large, the deadlocked messages remain in the network and blocking many other messages; on the other hand if the $TimeOut$ period is small, false deadlocks are declared. Whenever node i receives a message header H, a timer $T(i, H)$ starts counting the number of clock cycles the header has to wait before being forwarded. If for any header H, $T(i, H)$ exceeds $TimeOut$ at node i, then header is declared to be *deadlocked*. The optimum $TimeOut$ interval can be found by simulations (see Section 4). We also measured the frequency of deadlocks due to the proposed routing algorithm. It is observed that under maximum load less than 5% messages are deadlocked for a 64 node hypercube and $TimeOut = 16$ clocks.

3 Deadlock-Recovery

3.1 Recovery Scheme 1

After detecting a deadlock, the deadlock cycle is broken by switching one of the messages involved in the deadlock cycle to the central buffers kept aside at each node for routing deadlocked messages. Each node say i has $K+1$ central buffers numbered $0, 1, 2..., K$. These buffers are accessible by all neighboring nodes of i. The k^{th} central buffers of all the nodes form a central virtual network VN_k, $0 \leq k \leq K$. On any central virtual network, only one message is allowed to travel at a time. Thus, at most K deadlocks can be recovered simultaneously.

The permission to break deadlocks is given in a "round-robin" fashion to all the nodes. This may be implemented using K *tokens* corresponding to each central virtual network. Token i corresponds to the central virtual network VN_i. These tokens are nothing but packets of one flit size, which rotate on VN_0 along a cycle including all healthy nodes of the network. Each token carries the address of the next node in the cycle. Further details on token management and implementation are given below:

- Token synchronization: The tokens are synchronized with the router clock i.e. each token remains at a node for one clock cycle while circulating. The worst

case waiting time to capture a token is $(N * (Avg.Dist.)/K)$ clock cycles, where N is the number of nodes and $Avg.Dist.$ is the average distance of the network.

- Capturing a token: When a node detects a deadlock and concurrently receives a token j along the central buffer 0 carrying the address of the node, the token is captured by the node.
- Regenerating a token: In order to make sure that messages do not get deadlocked on central virtual networks, a captured token j at a node N_i is regenerated after $\delta(N_i, D)$ clock cycles, ensuring that there is at most one packet on the network, where D is the destination node.

3.2 Recovery Scheme-2

The recovery scheme-2 , also called *2-phase routing* discussed in this section, provides simultaneous recovery from all the deadlocks. Each physical channel is divided into two virtual channels. The network can be viewed as two virtual networks : *the Adaptive Virtual Network*(AVN) and *the Deadlock-free Virtual Network*(DVN). All the messages are initiated in AVN. Routing is carried out without any restrictions following the routing algorithm of Section 2. Whenever any message gets deadlocked, the routing enters phase 2 and the message is switched over to DVN. The routing procedure followed on DVN can be any deadlock-free routing for example, e-cube routing for hypercube, X-Y routing and negative-first routing for meshes and k-ary n-cubes.

We describe here briefly, a deadlock-free, Hamiltonian path based routing to route on DVN, which is partially adaptive and works for various topologies like hypercubes, k-ary n-cubes, Star graphs, meshes etc. The nodes of DVN are numbered according to Hamiltonian number denoted as $Hamilt(i)$. The DVN is further partitioned into two subnetworks *the Higher Virtual Subnetwork* (HSN) and *the Lower Virtual Subnetwork* (LSN). In HSN, there is an edge from any node i to any other node j, iff $Hamilt(i) < Hamilt(j)$. There is an edge from i to j in the lower network iff $Hamilt(i) > Hamilt(j)$. Each of these subnetworks is acyclic, thus routing in each of these networks is deadlock-free [7]. The routing function $\Re_1(\Re_2)$ of Equation3(4) corresponds to routing in LSN(HSN) from i to j, where $E_L(E_H)$ is edge set of LSN(HSN).

$$\Re_1(i,j) = k : (i,k) \in E_L, Hamilt(i) > Hamilt(k) \geq Hamilt(j) \quad (3)$$

$$\Re_2(i,j) = k : (i,k) \in E_H, Hamilt(i) < Hamilt(k) \leq Hamilt(j) \quad (4)$$

The routing functions defined by Equations 3 and 4 are deterministic. In a modification of the routing function of Equation 3, if we allow a message to switch from LSN to HSN, when a faulty or congested node is encountered, without permitting the message to return to LSN, the routing function $\Re_1$ continues to remain deadlock-free (see Theorem 1). A similar argument holds for the function $\Re_2$. The resulting partially adaptive routing is called *Adapt-Hamilt*.

Theorem 1. *The adaptive routing* Adapt-Hamilt *is deadlock-free.*

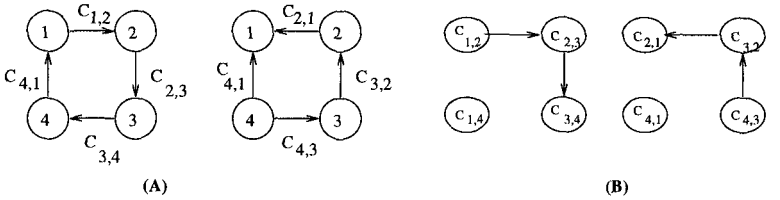

Fig. 1. (A) HSN and LSN for 2-dimensional Hypercube (B) CDG for $\Re_1$ and $\Re_2$

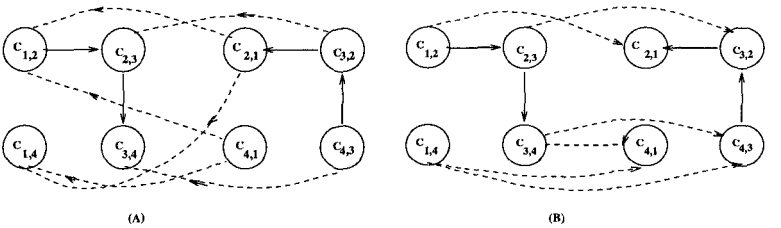

Fig. 2. CDG for adaptive routing in a 2-d Hypercube

Proof. We have seen that the routing function $\Re_1$ of Equation 3 is deadlock-free, implying that the channel dependency graph(CDG) induced by this function is acyclic [4](see Figure 1). The modified routing introduces additional edges in the CDG. These additional edges are of the form (x_h, y_l) as shown in Figure 2, where x_h is a node of the higher network CDG and y_l is a node of the lower network CDG. The adaptive routing function *Adapt-Hamilt*, if allows the additional edges of the form (x_h, x_l) then edges (x_l, y_h) are not permitted. Hence the theorem. □

4 Performance Study

We conducted experiments to measure the performance of our adaptive routing with various deadlock recovery schemes on a 256 node hypercube. The performance metric considered are the *throughput* and the *mean delay*. *Throughput* of the network is defined as the mean number of messages going out of the network per node per clock cycle. *Mean delay* is the mean number of clock cycles elapsed from the time the first flit of the message enters the network to the time the last flit leaves the network. It may be noted that throughput and latency are measured for various applied loads, thus throughput-latency graphs do not represent a function.

Each node generates messages with a poisson distribution. Two kinds of traffic patterns are generated: uniform traffic and bit-reversal traffic. In uniform traffic the destinations are generated uniformly with each node being equally probable; on the other hand in bit-reversal traffic pattern, the destination node address is obtained by reversing the bits of the source node address, for example source $s_1, s_2, ..., s_n$ sends a message to the destination node $D = s_n, s_{n-1}, ..., s_2, s_1$. Messages are 8 flits long. The results are taken for 4000 messages out of which

the information for first 2000 messages is discarded (warm-up period). It takes one clock cycle time to transfer a flit across a physical channel. The header is assumed to be one flit long. The experiment is repeated several times by varying the seed and the average value is plotted.

4.1 Finding optimum *TimeOut* and $\frac{\alpha}{\beta}$

For fine tuning *TimeOut* period for each type of traffic, we conducted experiments. The network performance is measured for various *TimeOut* periods. Figure 3 shows the *throughput* versus *latency* curves for various *TimeOut* periods under bit-reversal traffic pattern. It may be observed that latency increases slowly initially upto throughput value of 0.02 messages per node per clock cycle and then there is a sharp increase. Since the highest throughput is achieved with $TimeOut = 16$, it is chosen as the optimum value. Similarly, for uniform traffic the optimum *TimeOut* value is also found to be 16. The plot is not shown due to the lack of space.

For both type of traffic model, we found optimum value of the ratio $\frac{\alpha}{\beta}$. Figure 4 shows the performance of the routing algorithm for various values of $\frac{\alpha}{\beta}$ for uniform traffic. The optimum value of $\frac{\alpha}{\beta}$ is the value which gives highest peak performance. The throughput reaches its maximum value upto a certain applied load then starts reducing; this is the unstable condition of the network. From the Figure 4 it can be noticed that the optimum value is $\frac{\alpha}{\beta} = 10$, because for this value the highest throughput is achieved before saturation. Similarly for bit-reversal traffic the optimum value of $\frac{\alpha}{\beta}$ is found to be 5, as shown in figure 5.

4.2 Comparison of various schemes

We compared the performance of the proposed routing algorithm for various deadlock recovery schemes. Both the proposed recovery schemes are compared with the existing deadlock recovery scheme *disha*. Figure 6 shows a comparison of the three recovery schemes under uniform traffic conditions. The throughput value saturates at 10% with both *disha* and scheme-1 and at 21% with *2-phase* routing. It is also observed that by adding more and more central virtual buffers (or tokens) the performance of scheme-1 improves. In figure 6, the performance of the scheme-1 is measured for 3 central virtual buffers at each node.

We analyze the performance of the proposed routing in the presence of node faults. It is observed that there is a negligible degradation in the performance routing for various fault conditions. The plots are not shown here due to the lack of space.

The performance is also compared for the three recovery schemes under bit-reversal traffic. Figure 7 shows the comparison. It may be observed that the recovery scheme-1 outperforms the recovery scheme *disha*, and the *2-phase* routing performs far better than both the scheme-1 and *disha*. Figure 7 shows that after saturation the mean delay increases sharply, however the throughput value does not increase. In this case saturation occurs only at 6%.

5 Conclusion

We have presented a fully adaptive, restriction-free routing algorithm. Since deadlocks are rare we considered deadlock recovery instead of traditional *deadlock avoidance*. We have presented two deadlock recovery schemes, the recovery scheme-1 and *2-phase* routing. The recovery scheme-1 provides simultaneous recovery from K deadlocks, where K is the number of central buffers at each node; on the other hand *2-phase* routing provides the simultaneous recovery from all the deadlocks. The performance of both the proposed recovery schemes is compared with the recovery scheme disha [1]. It is observed that the recovery scheme-1 outperforms the recovery procedure of [1], under both uniform and bit-reversal traffic. Also the *2-phase* routing performs far superior than both disha and the recovery scheme-1.

References

1. K.V.Anjan and T.M.Pinkston, "Disha: A deadlock Recovery Scheme for Fully Adaptive Routing", *Proc. of the 19th Int. Parallel Processing Symposium*, 1995.
2. A.A.Chien, "A Cost and Speed Model for k-ary n-cube Wormhole Routers", *IEEE Transactions on Parallel and Distributed Systems*, Vol.9, No.2, pages 150 – 162, 1998.
3. W.Dally and H.Aoki, "Deadlock-free Adaptive Routing in Multicomputer Networks using Virtual Channels", *IEEE Transactions on Parallel and Distributed Systems*, Vol.4, No.4,1993, pages 466 – 475.
4. W.Dally and C.Seitz, "Deadlock-free message routing in multiprocessor interconnection networks", *IEEE Transactions on Computers*, C-36(5), 1987, pages 547 – 553.
5. C.J.Glass and L.M.Ni, "The turn model for Adaptive routing", *Proc. of the 19th Int. Symp. on Comp. Architecture*, 1992, pages 278 – 287.
6. J.Kim, Z.Liu and A.Chien, "Compressionless Routing: A Framework for Adaptive and Fault-tolerant Routing", *IEEE Transactions on Parallel and Distributed Systems*, Vol.8, No.3, pages 229 – 244, 1997.
7. X.Lin, P.K.McKinley and L.M.Ni, "Deadlock-free multicast wormhole routing in 2D mesh multicomputers", *IEEE Trans. on Parallel and Distributed Systems*, Vol.5, no.8, 1994, pages 793 – 804.

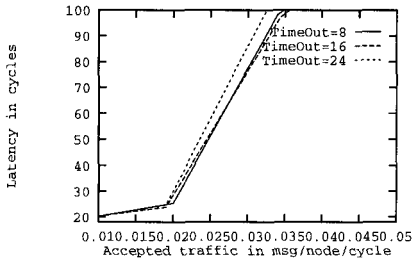

Fig. 3. Tuning *TimeOut* interval for bit-reversal traffic

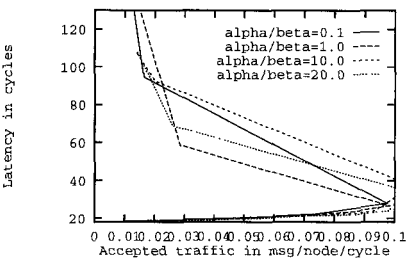

Fig. 4. Tuning $\frac{\alpha}{\beta}$ for unifrom traffic

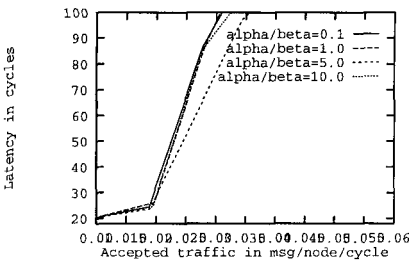

Fig. 5. Tuning $\frac{\alpha}{\beta}$ for bit-reversal traffic

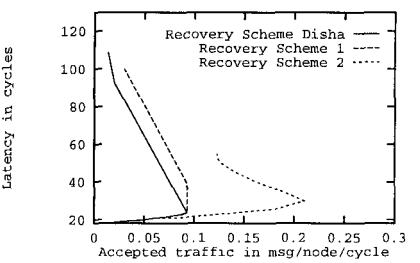

Fig. 6. Throughput Vs. mean delay for the three recovery schemes under uniform traffic

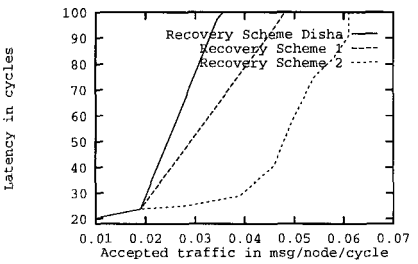

Fig. 7. Throughput Vs. mean delay for the three recovery schemes under bit-reversal traffic