An Efficient Fault-Tolerant Routing Scheme for Two Dimensional Meshes

Vara Varavithya, Jatin Upadhyay, and Prasant Mohapatra
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011
E-mail: *prasant@iastate.edu*

Abstract

Most of the previously proposed fault-tolerant wormhole routing algorithms for direct networks are extensions of the existing adaptive routing algorithms. These extensions are usually done by adding extra virtual channels to the network. Since these additional virtual channels are used for the fault tolerance purposes, they are not efficiently utilized when there are no faults in the network. In this paper, we propose a new fault-tolerant routing scheme for 2-dimensional mesh networks. It provides alternate physical paths to a message until it reaches the last dimension. While routing in the last dimension, if the message encounters a faulty node, it is stored completely at the adjacent node and is retransmitted. Using this routing scheme no adaptivity is sacrificed when there are no faults in the network and no additional virtual channels are required for fault tolerance. The performance of the proposed scheme is better than the previously proposed *fcube* algorithm which uses the same number of virtual channels. The effect of number and location of faults on the latency and the utilization is also studied.

1 Introduction

Message passing in multicomputers is implemented based on a switching technique coupled with a routing algorithm. Due to low latency and small buffer requirements, wormhole routing is preferred over virtual cut through and is widely used in recent multicomputers. The routing algorithm determines the path from a source to its destination. If the path between every pair of source and destination is predetermined, the algorithm is called a deterministic routing algorithm. For better system performance, it is however preferable that the algorithm adapts itself to the network traffic conditions and allows alternate paths. A plethora of adaptive routing algorithms have been proposed in the literature [7]. A good routing algorithm should not only reduce network congestion but should also be able to route in the presence of faulty nodes and channels. This necessitates development of adaptive fault-tolerant routing algorithms.

The objective of a fault-tolerant routing algorithm is to maximize the ability of the operational nodes to communicate with each other in presence of faults. This ability should be incorporated with the use of minimum redundancy. Issues in the design of fault-tolerant routing algorithms include avoidance of deadlock and livelock, low latency message delivery, high throughput, graceful performance degradation, and adaptation to a variety of traffic and fault patterns.

Previous work on fault-tolerant routing has been concentrated on augmenting the existing adaptive routing algorithms with fault tolerant capabilities. In [5], Ni has extended the partially adaptive turn model algorithm - negative first; to tolerate $n-1$ faults in n-dimensional meshes without using any virtual channels. For the low dimensional networks, the number of faults tolerated are very small. Dally and Akoi have presented an adaptive, non-minimal fault-tolerant algorithm based on dimensional reversal scheme [3]. Using their scheme, a message can tolerate any number of faults as long as it is in adaptive routing mode. However, once in deterministic mode, it cannot tolerate any faults. Linder and Harden have presented a fully adaptive routing algorithm that can tolerate at least one fault at the cost of 2^{n-1} virtual channels per physical channel in an n-dimensional mesh [6]. Planar adaptive scheme, proposed by Chien and Kim [2], limits the adaptivity to only two dimensions at a time and thereby reduces the number of virtual channels required to only 3. Boppana and Chalasani

have proposed a scheme called *f-cube* routing which adds fault tolerance to the widely used deterministic e-cube routing [1]. The scheme uses 4 virtual channels and tolerates any number of faults in the network as long as the source and the destination nodes are connected.

The primary disadvantage of all the above schemes is that they use additional virtual channels to make the existing algorithms fault-tolerant. Since the extra virtual channels are added from a fault tolerance point of view, they are not used at all when there are no faults in the network.

In this paper, we present a new scheme for fault-tolerant routing in two dimensional meshes using only two virtual channels. Message routing is done positive-first (PF) in one network layer and negative-first (NF) in the other. The two virtual channels are used for adaptivity as well as for fault tolerance. In the last dimension, the algorithm allows only one path to the destination. If there is a fault while routing in the last dimension, the message is completely consumed at one of the adjacent nodes and then retransmitted. The main advantage of the proposed scheme is that it tolerates multiple faults using only two virtual channels which is less than most of the existing fault-tolerant algorithms. In terms of network latency and throughput, the proposed scheme is observed to perform significantly better than the fcube algorithm [1] which also uses the same number of virtual channels. The effect of number and location of faults on the system performance is also discussed.

The rest of the paper is organized as follows. Section 2 discusses the fault model. Section 3 presents the proposed routing scheme for two dimensional meshes. Simulation results are presented in Section 5. Section 6 concludes the paper.

2 Preliminaries

For simplicity, we have made the following assumptions:

1. Faulty nodes are detectable by their neighbors. Since flits in wormhole routing are transferred between adjacent nodes through hardware handshaking, it is reasonable to assume that a node would know about its faulty neighbors.

2. The source node is notified if any of its messages are aborted. Most of the multicomputer systems have a scheme for acknowledging the successful transmission of messages. If the acknowledgment does not arrive, the source node may time out and retransmit the message.

3. A message arriving at its destination node is eventually consumed. This assumption made by many previous researchers [2, 4, 6] is based on the fact that the destination node does not block any incoming messages.

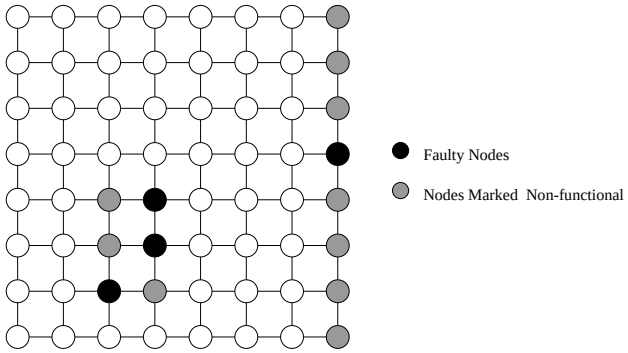

Figure 1: Reconfiguration of faulty regions.

In absence of any faults, a minimal routing algorithm would always route the message closer to its destination. If blocked by a fault, the message may sidetrack or backtrack and change its path. In our scheme, since we assume only local information about faults and since only two virtual channels are used, the message is not allowed to backtrack. Thus, there should not be any concave faulty regions in the network where the message may get trapped. If faulty regions are concave, they are extended so as to make them convex. The extension of the faulty region boundaries can be accomplished by marking some of the fully functional nodes in the network as a "non-functional" and routing messages to these nodes only if they are destined for them. Reconfiguration of the network so as to make the faulty region convex can be carried out distributedly at each node. Figure 1 shows an example of faults in the network and the reconfiguration of the faulty region to make them convex.

3 The Routing Scheme - Modified PFNF algorithm

The fault-tolerant routing scheme we propose is based on the Positive-First Negative-First (PFNF) algorithm presented in [8] for 2-dimensional mesh networks. Since the proposed scheme adds fault handling capability to the basic PFNF algorithm, we call it the *modified PFNF algorithm*.

3.1 The Modified PFNF Routing Algorithm

The modified PFNF routing scheme proposed is composed of mainly three components - the routing algorithm, the selection function and the fault handling steps. The routing algorithm provides a set of channels the message can take as its next hop and is responsible for deadlock avoidance. Depending upon the network traffic and faults, the selection function chooses one of these channels and routes the message along it. In case all the channels provided by the routing algorithm are blocked by faulty nodes, the selection function calls the fault handling routine which stores the entire message at one of its adjacent nodes. The message is subsequently retransmitted from that node.

The overall structure of the routing scheme is as shown below. Here, *routing_tag* refers to the vector of length n, the network dimension, where each element i in *routing_tag* is equal to

$$routing\_tag[i] = des\_addr[i] - curr\_node\_addr[i]. \quad (1)$$

```
Route(routing_tag) {
    virtual_channel_set VC;
    virtual_channel vc;
    VC = Routing_Algorithm(routing_tag);
    vc = Selection_Function(VC)
    Route along channel vc.
}
```

The Routing_Algorithm is the PFNF algorithm as shown in Figure 2. The algorithm uses the positive-first routing scheme in one virtual network and the negative-first scheme in the other virtual network. At each intermediate node, the algorithm always provides more than one virtual channels for the message to take as its next hop. The algorithm is proven to be more efficient than most of the existing adaptive routing algorithms [8].

Figure 3 (a) shows the Selection_Function algorithm. From the set of virtual channels provided by the Routing_Algorithm, it discards those which lead to the faulty nodes. From the remaining channels, one is chosen at random to route the message. If there are no such fault free channels available, it calls the Handle_Fault routine.

In the Handle_Fault routine shown in Figure 3 (b) the entire message in consumed at one of the adjacent nodes from where it is subsequently retransmitted. The adjacent node where the message is consumed should be farther away from the source node than the current node. This requirement is to avoid livelock situations as described later.

The fault handling steps using the proposed scheme can be illustrated as shown in Figure 4. Figure 4(a) shows the case where the message encounters a faulty node, while it still has to route in both the directions. The path it takes is the one not blocked by the fault. Figure 4(b) shows the case where the fault is in the last dimension of routing. As shown in the figure, the message is completely stored in the adjacent node and is subsequently retransmitted. The possible alternate paths the message may take after retransmission are also shown in the figure.

```
Routing_Algorithm(routing_tag)
{
    virtual_channel_set VC
    In virtual network VN1, /* PF Algorithm */
        If (∃ routing_tag[i] > 0){
            For all dimension i
                If (routing_tag[i] > 0) add vc
                in + direction of dimension i to VC
        }
        else{
            For all dimension i
                If(routing_tag[i] < 0) add vc
                in − direction of dimension i to VC
        }

    In virtual network VN2, /* NF Algorithm */
        If (∃ routing_tag[i] < 0){
            For all dimension i
                If (routing_tag[i] < 0) add vc
                in − direction of dimension i to VC
        }
        else{
            For all dimension i
                If(routing_tag[i] ≠ 0) add vc
                in + direction of dimension i to VC
        }
    return(VC)
}
```

Figure 2: The PFNF Routing Algorithm.

As mentioned earlier, if blocked by a faulty node in the last dimension, the routing scheme consumes the entire message in one of the adjacent nodes. Since every node acts as an infinite sink for incoming messages, once the head flit moves to the adjacent node, the entire message is absorbed completely and it does not create an indefinite blocking for any other message in the network. Hence, the fault handling strategy in the modified PFNF routing scheme does not create any additional dependencies.

Theorem 1: The modified PFNF algorithm is dead-

lock free.

Proof:

Lemma 1: The basic PFNF algorithm is deadlock free.

The proof of this lemma can be found in [8].

Lemma 2: The Fault handling in the algorithm does not create any additional dependency.

Thus, using Duato's theorem [4] and based on Lemma 1 and Lemma 2, the modified PFNF algorithm is deadlock free. A detailed proof is reported in [9]. □

```
Selection Function(virtual_channel_set VC) {
      virtual_channel_set next_channels;
      For each channels j in VC {
            if (sink_node(j) != faulty)
                  add j to the set next_channels;
      }

      if(set next_channels == empty)
            call Handle_Fault() ;
      else
            return a channel from the set next_channels
}
```

(a)

```
Handle_Fault() {
      if(∃ a neighbor node such that it is farther than
            the message source than the current node) {
            route and consume the message at that node.
      }
      else {
            The destination is unreachable from this node.
            Abort the message.
      }
}
```

(b)

Figure 3: (a) The Selection Function (b) Fault handling Routine.

Theorem 2: The modified PFNF routing algorithm is livelock free.

Proof:

The basic PFNF algorithm is minimal and hence livelock free. In the fault handling steps, the message is consumed in the adjacent node only if by doing so, the message moves farther away from the source than its current position. If that is not possible, the message is aborted. Thus, at each step, whether it is a normal minimal routing or a fault handling step, the message always moves farther away from its source node. Since, the distances in the network are finite, the message always reaches its destination or is aborted. Thus, the modified PFNF algorithm is livelock free. □

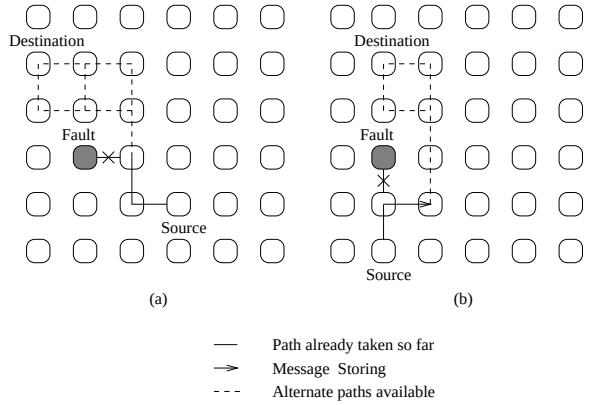

Figure 4: Fault-tolerant routing in modified PFNF algorithm.

4 Simulation Study

We have compared the performance results of our algorithm with that of the previously proposed fcube scheme [1]. Only the fcube algorithm is compared since it also uses the same number of virtual channels (two) for 2-dimensional meshes. All the other reported schemes that can tolerate multiple faults require more number of virtual channels.

4.1 Simulation Environment

The simulations were conducted on a 16×16 mesh with fixed message length of 20 flits. Packet generation rate is assumed to be Poisson in nature with an exponential distribution of inter-arrival time. The length of generation queue at each node is kept 16 packet buffers, i.e., a node can have at most 16 messages pending to be transmitted before it blocks any further generation of packets. The number of flit buffers associated with each virtual channel is assumed to be one. The simulations were carried out for 150,000 packets. The first 40,000 to 60,000 packets were discarded from statistics to reduce the transient effects. Since the main purpose of the simulations was to study the effects of the network faults, we have considered only the uniform traffic pattern. Under uniform traffic, a node sends messages to every other node in the mesh with equal probability.

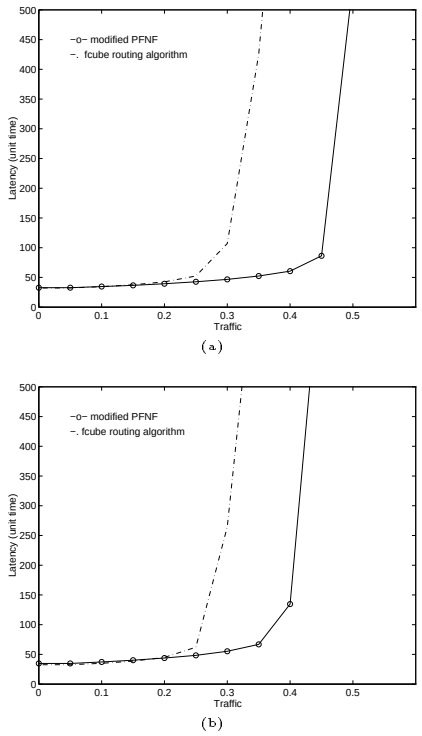

Figure 5: Comparison between the fcube algorithm and the modified PFNF algorithm for (a) One faulty node (b) Three faulty nodes in the network.

The performance of the proposed fault-tolerant scheme was studied in terms of the average communication latency and the average throughput of the network. The communication latency is the mean time from the message generation to the time when the tail reaches the destination. The throughput is defined as the average number of messages routed per unit time. To understand how the proposed scheme affects the traffic distribution in the network, we also measure the average flit buffer utilization at each node. All the above parameters are studied against the offered network traffic and various fault patterns and locations. The network traffic is defined as the ratio of the average traffic generated by a node to the average bandwidth available per node. Faults in the network are injected randomly and in case of faulty nodes forming a concave region, additional nodes are marked non-functional according to the algorithm presented in Section 2 to make the faulty regions convex.

4.2 Results and Discussion

Figures 5(a) and (b) show the comparison between the modified PFNF scheme and the fcube algorithm for one and three faulty nodes in the networks. It is observed from the graphs that the modified PFNF algorithm outperforms fcube scheme for all traffic loads. The fcube routing routing results in around 30-50% higher latency than the proposed modified PFNF scheme. The traffic at which the modified PFNF scheme saturates is also around 0.45% compared to only 0.3% of the fcube scheme. The reason for the poor performance of the fcube algorithm is that even though it uses two virtual channels, it is deterministic in nature. Modified PFNF algorithm provides more adaptivity for the same hardware used and this adaptivity compensates for the additional delay of storing the message at the intermediate nodes.

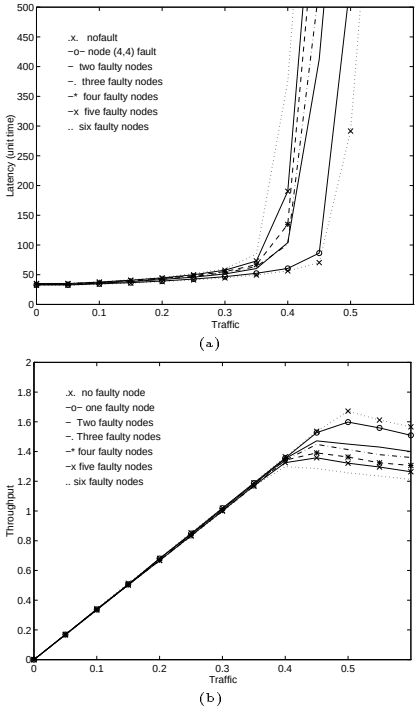

Figure 6: Latency and Throughput results for various number of faults.

The network latency and throughput results of the modified PFNF algorithm for different number of faults in the network are shown in Figure 6. The average latency increases with increase in number of faults. For the first few faults, the increase in latency is significant, however as the number of faults increases, the network connectivity decreases and the performance degradation is minimal. The same trend is observed in the network throughput results.

Figure 7 plots the average buffer utilization at each node when there are two faults in the network. The

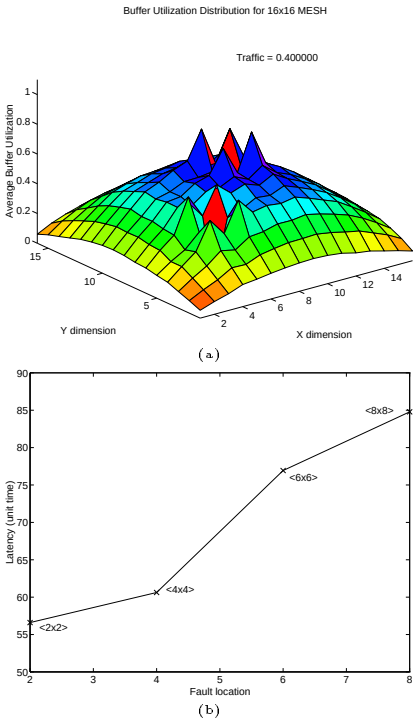

Figure 7: (a) Buffer Utilization near the faulty nodes. (b) Effect of location of faults on the network latency.

faulty nodes are at the location $< 4, 4 >$ and $< 8, 8 >$ in the 16×16 mesh, i.e., at the center of the mesh and at the center of one of the quadrants. The graphs represents the network traffic produced by the routing scheme when the actual traffic generated is uniform across all the nodes. Since the messages blocked by the faulty nodes are either routed around it or are completely stored in its adjacent nodes, the neighboring nodes of the faulty nodes are becoming hot spots. Thus, there are four spikes each around the two nodes $< 8, 8 >$ and $< 4, 4 >$ as is seen in Figure 7(a).

Figure 7(b) studies the effect of the location of faulty nodes on the network latency and the throughput. As is evident from the figure, fault at the center of the mesh results in much higher increase in the network latency than a fault in one of the quadrants. Since mesh network is not symmetric at its edges, there are more number of messages passing through the center of the mesh than through the edges and corners. Faults near the center of the mesh affect more number of messages and hence results in more degradation in performance. The effect of faults on the system performance goes on reducing as the fault moves away from the center.

5 Conclusions

A new strategy for fault-tolerant routing in two dimensional meshes is proposed by incorporating fault-tolerance in the PFNF routing algorithm. The modified PFNF scheme performs better than the previously proposed fcube scheme which also uses the same number of virtual channels. Several other factors, such as the effect of faults on the buffer utilization, effect of fault location and the number of faults on the system performance are also analyzed in this paper.

References

[1] R. V. Boppana and S. Chalasani, "Fault-tolerant routing with non-adaptive wormhole algorithms in mesh networks," *Intl. Conference of Supercomputing*, pp. 693-702, Dec. 1994.

[2] A. A. Chien and J. H. Kim, "Planar adaptive routing: low-cost adaptive networks for multiprocessors," *Intl. symposium on Computer Architecture*, pp. 268-277, May 1992.

[3] W. J. Dally and H. Akoi, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Trans. on Parallel and Distributed Systems*, pp. 466-475, April 1993.

[4] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole network," *IEEE Trans. on Parallel and Distributed Systems*, Dec. 1993.

[5] C. J. Glass and L. M. Ni, "Fault-Tolerant Wormhole Routing in Meshes," *Intl. Symposium on Fault-Tolerant Computing*, pp. 240-249, 1993.

[6] D. H. Linder and J. C. Harden, "An Adaptive and fault tolerant wormhole routing strategy for k-ary n cubes," *IEEE Trans. on Computers*, vol. 40, pp. 2-12, Jan. 1991.

[7] P. Mohapatra, "Wormhole Routing Techniques in Multicomputer Networks," Technical Report, Department of Electrical and Computer Engineering, Iowa State University, 1995.

[8] J. Upadhyay, V. Varavithya, and P. Mohapatra, "Efficient and Balanced Routing in Two-Dimensional Meshes," *Intl. Symposium on High Performance Computer Architecture*, pp. 112-122, Jan. 1995.

[9] V. Varavithya, "Wormhole Routing Algorithms for Mesh Interconnection Networks," *Master's Thesis*, Dept. of Electrical and Computer Engineering, Iowa State University, 1994.