Adaptive and Fault-Tolerant Routing with 100% Node Utilization for Mesh Multicomputer

Sheng-De Wang* and Ming-Jer Tsai
Department of Electrical Engineering, National Taiwan University,
Taipei 106, TAIWAN

Abstract

In this paper, we propose an adaptive and deadlock-free routing algorithm to tolerate irregular faulty patterns using two virtual channels per physical link. It can improve the node utilization up to 100%. When a node becomes faulty or recovered, the central control unit constructs a directed path graph which is used for generating the intermediate nodes of the message path. Thus a message can be transmitted from sources or to destinations within faulty blocks via a set of "intermediate nodes". Our method requires the global failure information if the central control unit is not available.

1. Introduction

Message routing achieves inter-node communication in large-scale parallel computers. In a concurrent multicomputer, wormhole switching [9] is widely used; however, it suffers the deadlock problem when not well-designed in a fault environment. Thus, a reliable routing algorithm is supposed to be deadlock-free and fault-tolerant.

Glass and Ni [6] have proposed a partially-adaptive routing algorithm to tolerate $n-1$ faults in an n-dimensional mesh based on the turn model. In an $N \times N$ 2-dimensional mesh, Cunningham and Avresky [4] can improve the performance and provide fault tolerance for up to $N-1$ faults. Besides, Hadas and Brandt [7] have proposed an original-based routing algorithm to tolerate square faulty blocks. When a fault arises, their method has a hot-spot effect and will non-minimally route a message even if the message does not encounter any square faulty block.

Moreover, Linder and Harden [8] have proposed a fully-adaptive and fault-tolerant routing algorithm based on the concept of virtual interconnection networks. However, it requires exponential number of virtual channels per physical link and tolerates small number of faults. Dally and Aoki [5] have proposed a dynamic algorithm to remove cycles from packet wait-for graph instead of channel dependency graphs. Thereby, the virtual channel utilization can be considerably improved. In their algorithm, the number of faults tolerated and virtual channels used depends on the location of faults.

Using three virtual channels per physical link, Chien and Kim [3] have proposed a planar-adaptive routing algorithm to tolerate disconnected faulty blocks with distance of no less than two in at least one dimension. Using extra four virtual channels per physical link, Boppana and Chalasani [1] can enhance a fully-adaptive algorithm to tolerate disconnected faulty blocks with distance of no less than two in at least one dimension. Using three virtual channels per physical link, the algorithm [2] presented by Boura and Das provides full-adaptivity and fault-tolerance. It can tolerate disconnected faulty blocks with distance of at least three.

In addition, Su and Shin [10] have proposed an adaptive routing algorithm to tolerate disconnected faulty blocks with distance of three or more in at least one dimension using only two virtual channels per physical link which is so far the smallest. However, their algorithm can possibly reach a deadlock and has been improved to tolerate disconnected faulty blocks with distance of no less than two in at least one dimension [11].

As was stated above, many adaptive routing algorithms [1, 3, 10] are designed to tolerate a large number of faults by introducing rectangular faulty blocks. Thus, some good nodes are regarded as faulty ones and are prohibited from interchanging messages with the other good nodes. It implies that the node utilization may drastically degrade when the faulty nodes densely distribute in certain particular patterns. In this paper, using two virtual channels per physical link, we develop an adaptive and deadlock-free routing algorithm, by which two good nodes can communicate with each other if there is at least one path between them. Notation used in this paper are summarized in Table 1, where two nodes are connected if there is a link between them, and two sets S_i and S_j are connected if there is a node $A \in S_i$ and a node $B \in S_j$ such that nodes A and B are connected

*Corresponding author. E-mail: sdwang@hpc.ee.ntu.edu.tw.

Table 1. Summary of Notation

VIN_i	the virtual interconnection network i ($i = 1, 2$).
$VC_{i,j}$	the virtual channel in dimension i of VIN_j.
$num(VC_{i,j})$	the channel number of virtual channel $VC_{i,j}$.
N_S	the source node.
N_D	the destination node.
N_C	the current node.
N_I	the intermediate node.
N_R	the node receives the message from an intermediate node.
B_i	the faulty block i.
M_0	the interconnection network outside all faulty blocks.
M_i	the sub-mesh i ($i > 0$) contained in a faulty block.
S_{-1}	a set consists of faulty nodes.
S_i	a set consists of good nodes in M_i ($i \geq 0$).
V_i	vertex i in the directed path graph.
$G_{i,j}$	a set consists of the entry (w_i, dim, dir), where $w_i \in S_i$ is an intermediate node to enter S_j (S_i and S_j are connected) for the message with header in M_i, and the message is routed to w_i via VIN_{dir} if $S_i \neq S_0$, then is routed out from w_i via $VC_{dim,dir}$.
$T_{i,j}$	a set consists of the entry (w_i, dim, dir), where $w_i \in S_i$ is an intermediate node to reach $N_D \in S_j$ (S_i may or may not connect to S_j) for the message with header in M_i, and the message is routed to w_i via VIN_{dir} if $S_i \neq S_0$, then is routed out from w_i via $VC_{dim,dir}$.

nodes.

The rest of this paper is organized as follows. In the next section, we first present a method to construct the directed path graph which is used to generate the intermediate node of the message path. In section 3, we propose an adaptive and deadlock-free routing algorithm to tolerate irregular faulty patterns using two virtual channels per physical link. In section 4, we conclude this paper.

2. Construct the directed path graph

In this section, algorithm 2.1 is presented to construct the directed path graph which is used to generate the message path in the next section. Before describing it, Definitions 1 and 2 are needed.

Definition 1 (Safe/Unsafe Node) *For an n-dimensional mesh, a good node is called an unsafe node if it connects to at least two faulty/unsafe nodes, and is called a safe node otherwise.*

Definition 2 (Ancestor, Common Ancestor, Least Common Ancestor) *In the directed path graph, if there is a directed path from vertex A to vertex B, then vertex A is called an ancestor of vertex B. If vertex A is an ancestor of both vertices B and C, then vertex A is called the common ancestor of vertices B and C. For two vertices B and C, common ancestor A is called the least common ancestor if no other common ancestor is located in each directed shortest path from vertex A to vertex B and in each directed shortest path from vertex A to vertex C.*

Algorithm 2.1 /* When a node becomes faulty or recovered, algorithm 2.1 is executed by the central control unit. And, the central control unit will send $T_{i,*}$ to all nodes in S_i after algorithm 2.1 is completed. */

1 Let the directed path graph be an empty graph.

2 Add vertex V_0 to the directed path graph and set x to 1.

3 For all faulty blocks B_1 to B_p do /* construct the directed path graph, and compute $G_{k,j}$ and $G_{j,k}$ for each directed edge (V_k, V_j). */

3.1 Find sub-meshes $M_x, M_{x+1}, \cdots, M_{x+q-1}$ in B_i such that

(1) Each good node in B_i belongs to exactly one M_j ($x \leq j \leq x+q-1$).

(2) For each M_j ($x \leq j \leq x+q-1$), the number of faulty nodes in M_j is less than the dimension of M_j.

(3) For each M_j ($x \leq j \leq x+q-1$), there exists $M_0 = M_{a_1}, M_{a_2}, \cdots, M_{a_b} = M_j$ such that (1) S_{a_c} and $S_{a_{c+1}}$ are connected and (2) $a_c < a_{c+1}$, where $M_{a_2}, M_{a_3}, \cdots, M_{a_b}$ are all in B_i.

3.2 For $j = x$ to $x+q-1$ do

3.2.1 Add vertex V_j to the directed path graph.

3.2.2 For each S_k ($0 \leq k < j$) connected to S_j do

3.2.2.1 Add a directed edge (V_k, V_j) to the directed path graph.

3.2.2.2 Let $G_{k,j}$ and $G_{j,k}$ be empty sets.

3.2.2.3 For each pair of connected nodes $w_k \in S_k$ and $w_j \in S_j$ do

3.2.2.3.1 $G_{k,j} = G_{k,j} \cup (w_k, dim, 1)$.

3.2.2.3.2 $G_{j,k} = G_{j,k} \cup (w_j, -dim, 2)$.

/* dim denotes the directed dimension where the link from node w_k to node w_j is located. */

3.3 $x = x + q$.

4 For $j = 0$ to $x - 1$ do /* compute $T_{j,k}$ for $0 \leq j \leq x-1, 0 \leq k \leq x-1$, and $j \neq k$, where x is the number of vertices in the directed path graph. */

4.1 For $k = 0$ to $x - 1$ $(k \neq j)$ do

4.1.1 If vertex V_k is an ancestor of vertex V_j in the directed path graph, then

4.1.1.1 $T_{j,k}$ is set to $\emptyset$.

4.1.1.2 For each directed shortest path P_{up} from vertex V_k to vertex V_j do.

4.1.1.2.1 Let (V_a, V_j) be a directed edge in P_{up}. Then, $T_{j,k}$ is set to $T_{j,k} \cup G_{j,a}$.

4.1.2 Else if vertex V_j is an ancestor of vertex V_k in the directed path graph, then

4.1.2.1 $T_{j,k}$ is set to $\emptyset$.

4.1.2.2 For each directed shortest path P_{down} from vertex V_j to vertex V_k do

4.1.2.2.1 Let (V_j, V_a) be a directed edge in P_{down}. Then, $T_{j,k}$ is set to $T_{j,k} \cup G_{j,a}$.

4.1.3 Else if vertices V_j and V_k has no common ancestor in the directed path graph, then $T_{j,k}$ is set to $\emptyset$.

4.1.4 Else

4.1.4.1 $T_{j,k}$ is set to $\emptyset$.

4.1.4.2 For each least common ancestor V_{lca} of vertices V_j and V_k do

4.1.4.2.1 For each directed shortest path P_{up} from vertex V_{lca} to vertex V_j do

4.1.4.2.1.1 Let (V_a, V_j) be a directed edge in P_{up}. Then, $T_{j,k}$ is set to $T_{j,k} \cup G_{j,a}$.$\square$

An example is shown in Figure 1(a), where 36 faulty nodes spread over a 21 × 21 mesh. Based on Definition 1, three faulty blocks B_1, B_2, and B_3 are formed by the deactivating method. After algorithm 2.1 is completed, we have

1 B_1 finds sub-mesh M_1, B_2 finds sub-mesh M_2, and B_3 finds sub-meshes M_3 and M_4 due to step 3.1.

2 The directed path graph is constructed as shown in Figure 1(b) due to step 3.2.1.

Table 2. $T_{j,k}$ obtained for the injured mesh shown in Figure 1(a).

$j \setminus k$	0	1	2	3	4
0	-	$G_{0,1}$	$G_{0,2}$	$G_{0,3}$	$G_{0,3}$
1	$G_{1,0}$	-	$G_{1,0}$	$G_{1,0}$	$G_{1,0}$
2	$G_{2,0}$	$G_{2,0}$	-	$G_{2,0}$	$G_{2,0}$
3	$G_{3,0}$	$G_{3,0}$	$G_{3,0}$	-	$G_{3,4}$
4	$G_{4,3}$	$G_{4,3}$	$G_{4,3}$	$G_{4,3}$	-

3 For each directed edge (V_k, V_j) in the directed path graph, $G_{k,j}$ and $G_{j,k}$ is obtained due to step 3.2.2, where $G_{0,1} = \{$((6,16),-1,1), ((7,16),-1,1), ((8,-16),-1,1), ((9,15),-0,1), ((9,14),-0,1), ((9,13),-0,1)$\}$, $G_{1,0} = \{$((6,15),1,2), ((7,15),1,2), ((8,15),1,2), ((8,15),0,2), ((8,14),0,2), ((8,13),0,2)$\}$, $G_{0,2} = \{$((11,10),-1,1), ((12,10),-1,1)$\}$, $G_{2,0} = \{$((11,9),1,2), ((12,9),1,2)$\}$, $G_{0,3} = \{$((3,8),-1,1)$\}$, $G_{3,0} = \{$((3,7),1,2)$\}$, $G_{3,4} = \{$((3,7),-1,1)$\}$, $G_{4,3} = \{$((3,6),1,2)$\}$.

4 $T_{j,k}$ ($0 \leq j \leq 4$, $0 \leq k \leq 4$, $j \neq k$) is obtained as shown in Table 2 due to step 4, where $T_{j,0}$ is set to $G_{j,0}$ for $1 \leq j \leq 3$ and $T_{4,3}$ is set to $G_{4,3}$ due to step 4.1.1, $T_{0,k}$ is set to $G_{0,k}$ for $1 \leq k \leq 3$ and $T_{3,4}$ is set to $G_{3,4}$ due to step 4.1.2, $T_{j,k}$ $(j \neq k)$, excluding $T_{3,4}$, is set to $G_{j,0}$ for $1 \leq j \leq 3, 1 \leq k \leq 4$ and $T_{4,k}$ is set to $G_{4,3}$ for $k = 1, 2$ due to step 4.1.4.

Another example is shown in Figure 2(a), where 2 faulty blocks B_1 and B_2 form in 21 × 21 injured mesh. After algorithm 2.1 is completed, we have

1 B_1 finds sub-meshes $M_1 \sim M_7$, and B_2 finds sub-meshes $M_8 \sim M_9$.

2 The directed path graph is constructed as shown in Figure 2(b).

3 For each directed edge (V_k, V_j) in the directed path graph, $G_{k,j}$ and $G_{j,k}$ is obtained.

4 $T_{j,k}$ ($0 \leq j \leq 9$, $0 \leq k \leq 9$) is obtained as shown in Table 3.

In step 3.1, each faulty block B_i can find sub-meshes by a deterministic algorithm. Thus, our method requires the global failure information if the central control unit is not available.

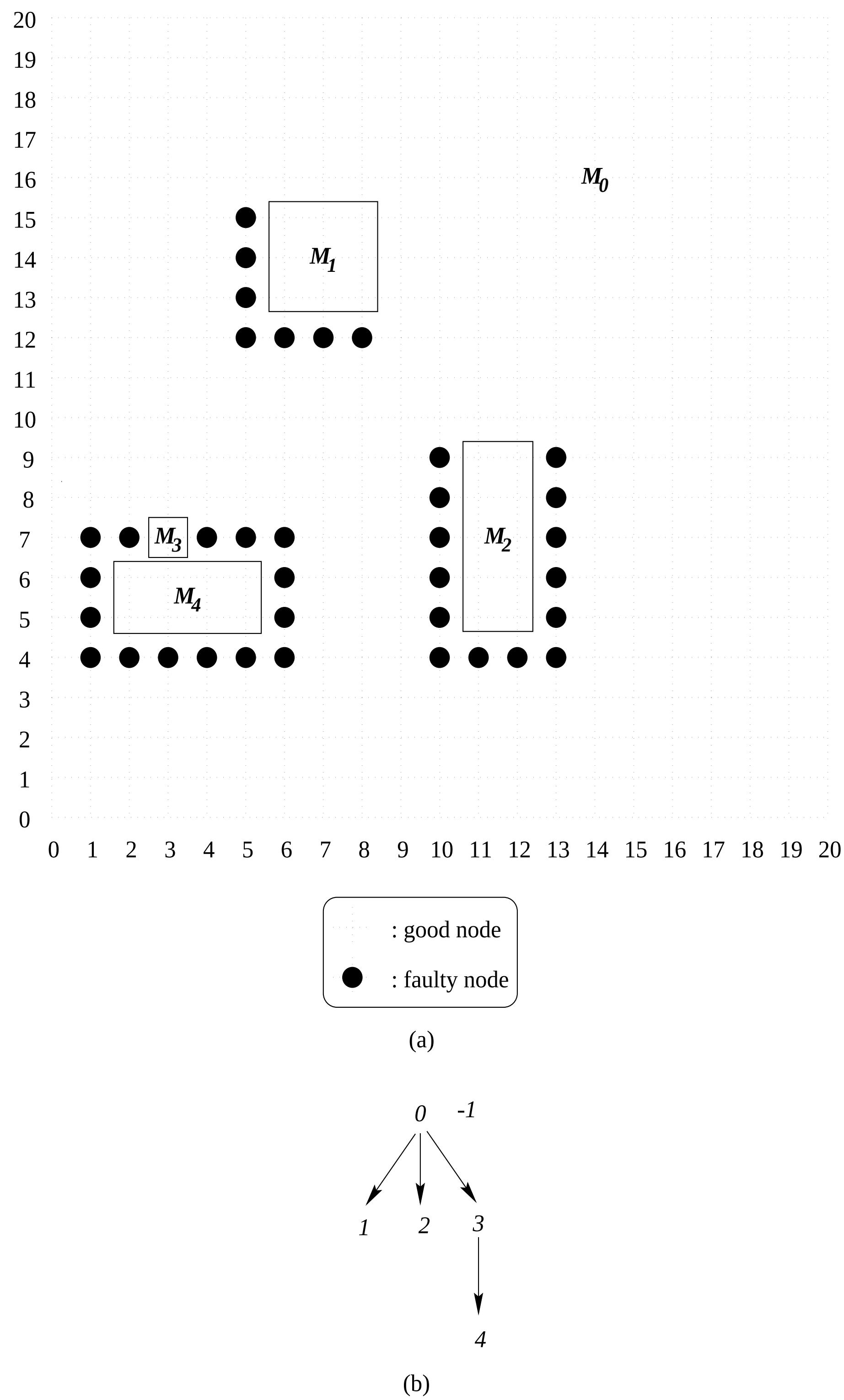

Figure 1. (a) shows a 21×21 injured mesh, where 3 faulty blocks $B_1 \sim B_3$ are formed by Definition 1, and $M_0 \sim M_4$ are decided after the completion of algorithm 2.1. (b) shows the directed path graph constructed by algorithm 2.1.

Table 3. $T_{j,k}$ **obtained for the injured mesh shown in Figure 2(a).**

$j \setminus k$	0	1	2	3	4
	5	6	7	8	9
0	-	$G_{0,1}$	$G_{0,2}$	$G_{0,1} \cup G_{0,2}$	$G_{0,4}$
	$G_{0,2} \cup G_{0,4}$	$G_{0,6}$	$G_{0,7}$	$\emptyset$	$G_{0,9}$
1	$G_{1,0}$	-	$G_{1,2}$	$G_{1,3}$	$G_{1,4}$
	$G_{1,2} \cup G_{1,3} \cup G_{1,4}$	$G_{1,4}$	$G_{1,2}$	$\emptyset$	$G_{1,0}$
2	$G_{2,0}$	$G_{2,1}$	-	$G_{2,3}$	$G_{2,0} \cup G_{2,1}$
	$G_{2,5}$	$G_{2,0} \cup G_{2,1}$	$G_{2,7}$	$\emptyset$	$G_{2,0}$
3	$G_{3,1} \cup G_{3,2}$	$G_{3,1}$	$G_{3,2}$	-	$G_{3,1}$
	$G_{3,5}$	$G_{3,1}$	$G_{3,5}$	$\emptyset$	$G_{3,1} \cup G_{3,2}$
4	$G_{4,0}$	$G_{4,1}$	$G_{4,0} \cup G_{4,1}$	$G_{4,1}$	-
	$G_{4,5}$	$G_{4,6}$	$G_{4,5} \cup G_{4,6}$	$\emptyset$	$G_{4,0}$
5	$G_{5,2} \cup G_{5,4}$	$G_{5,2} \cup G_{5,3} \cup G_{5,4}$	$G_{5,2}$	$G_{5,3}$	$G_{5,4}$
	-	$G_{5,4}$	$G_{5,7}$	$\emptyset$	$G_{5,2} \cup G_{5,4}$
6	$G_{6,0}$	$G_{6,4}$	$G_{6,0} \cup G_{6,4}$	$G_{6,4}$	$G_{6,4}$
	$G_{6,4}$	-	$G_{6,7}$	$\emptyset$	$G_{6,0}$
7	$G_{7,0}$	$G_{7,2}$	$G_{7,2}$	$G_{7,5}$	$G_{7,5} \cup G_{7,6}$
	$G_{7,5}$	$G_{7,6}$	-	$\emptyset$	$G_{7,0}$
8	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	$\emptyset$	$\emptyset$	$\emptyset$	-	$\emptyset$
9	$G_{9,0}$	$G_{9,0}$	$G_{9,0}$	$G_{9,0}$	$G_{9,0}$
	$G_{9,0}$	$G_{9,0}$	$G_{9,0}$	$\emptyset$	-

3. Adaptive and fault-tolerant routing with 100% node utilization

In this section, algorithm 3.1 is proposed to route a message using two virtual channels per physical link. It uses Glass and Ni's algorithm [6] to route the message inside a faulty block, and uses algorithm RRFB2 [11] to route the message outside a faulty block. It is capable of tolerating irregular faulty patterns by transmitting the message from sources or to destinations within faulty blocks via multiple "intermediate nodes". In algorithm 3.1, the message header format is ($header.N_D$, $header.N_I$, $header.dim$, $header.dir$). $header.N_D$ records the address of N_D, $header.N_I$ records the address of N_I, $header.dim$ records the directed dimension along which the message is routed out from N_I, and $header.dir$ records the virtual interconnection network $VIN_{header.dir}$ used to route the message from N_C to N_R if $N_C \notin S_0$. To allow clearer exposition, we assume $N_S \in S_p, N_C \in S_q, N_D \in S_r$ in algorithm 3.1.

Algorithm 3.1 /* Assume $N_S \in S_p, N_C \in S_q$, and $N_D \in S_r$. */

1 If $N_C = N_D$, then exit.

2 Else if $N_C = N_I$, then the message is routed via $VC_{header.dim,header.dir}$.

3 Else

3.1 If $(N_C = N_S)$ or $(N_C = N_R)$, then Set_Message_Header.

3.2 If $S_q = S_r$, then

3.2.1 If $N_C \in S_0$, then route the message to N_D via VIN_1 and VIN_2 by an adaptive deadlock-free routing algorithm in [10].

3.2.2 Else route the message to N_D via $VIN_{header.dir}$ by an adaptive deadlock-free routing algorithm in [6].

3.3 Else

3.3.1 If $N_C \in S_0$, then route the message to N_I via VIN_1 and VIN_2 by an adaptive deadlock-free routing algorithm in [10].

3.3.2 Else route the message to N_I via $VIN_{header.dir}$ by an adaptive deadlock-free routing algorithm in [6].

Set_Message_Header **1** If $(N_C = N_S)$ and ($T_{p,r} = \emptyset$ or $N_D \in S_{-1}$), then exit.

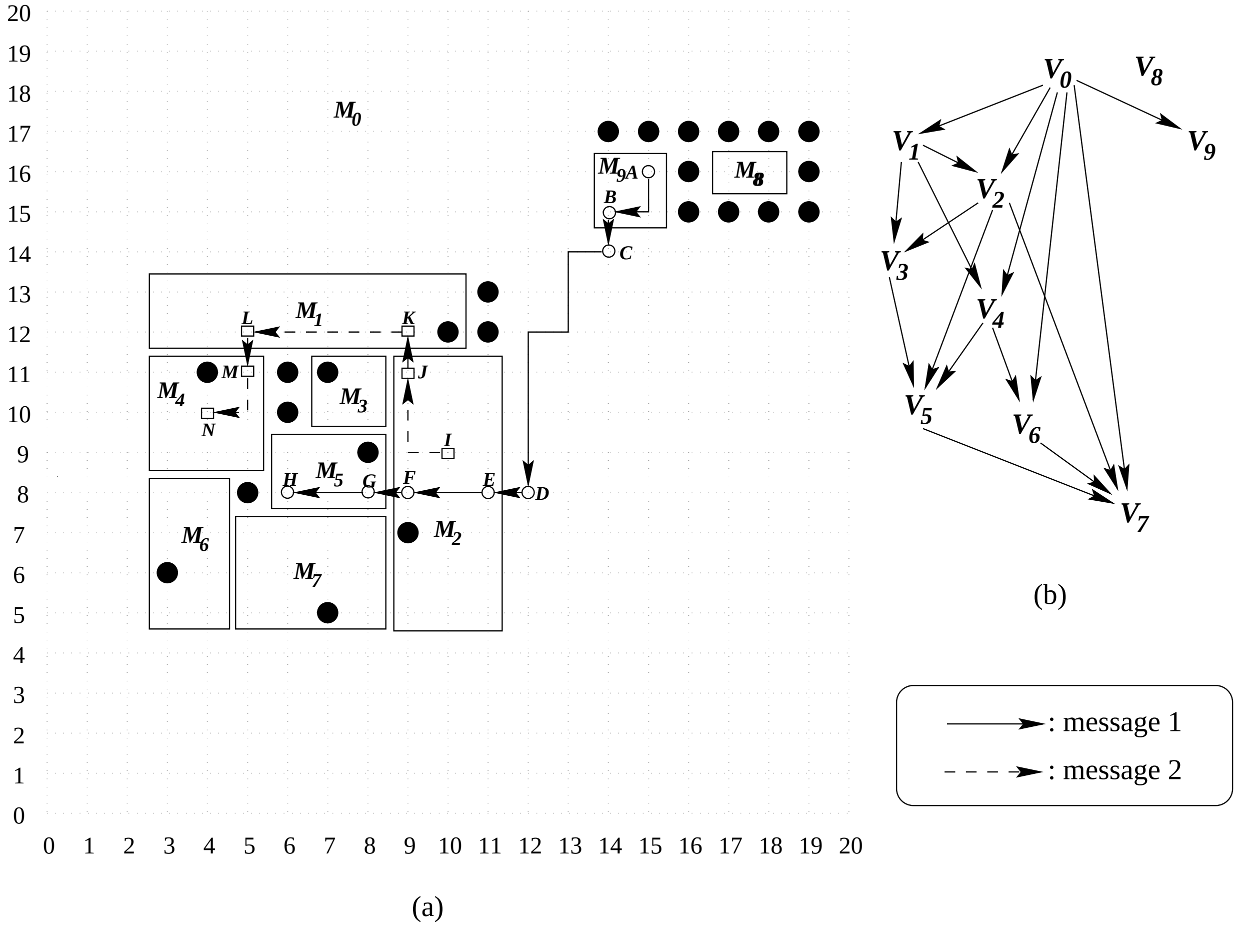

Figure 2. (a) shows a 21×21 injured mesh, where two messages 1 and 2 are routed by algorithm 3.1. (b) shows the directed path graph constructed by algorithm 2.1.

2 Else if ($N_C = N_S$) and ($S_q = S_r$), then set $header.dir$ to 1 or 2.

3 Else if ($S_q \neq S_r$), then update the message header by

- **(1)** selecting one element of $T_{q,r}$, say (w_i, dim, dir), such that the sum of the distance between nodes N_C and w_i and the distance between nodes w_i and N_D is the smallest.
- **(2)** setting $header.N_I$, $header.dim$, and $header.dir$ to w_i, dim, and dir, respectively.

As shown in Figure 2(a), message 1 is routed from node $A(15,16)$ to node $H(6,8)$, and message 2 is routed from node $I(10,9)$ to node $N(4,10)$. For message 1, the header is updated to $((6,8),(14,15),-1,2)$ due to step 3.1.3. Thus, it is routed to node $B(14,15)$ via VIN_2 by an adaptive deadlock-free routing algorithm in [6] due to step 3.3.2. Then it is routed to node $C(14,14)$ via $VC_{-1,2}$ due to step 2. When message 1 reaches node C, the header is updated to $((6,8),(12,8),-0,1)$ due to step 3.1.3. Then it is routed to node $D(12,8)$ via VIN_1 and VIN_2 by an adaptive deadlock-free routing algorithm in [10] due to step 3.3.1, and is routed to node $E(11,8)$ via $VC_{-0,1}$ due to step 2. And so on, message 1 is routed to node $F(9,8)$ via VIN_1 [6] due to step 3.3.2, is routed to node $G(8,8)$ via $VC_{-0,1}$ due to step 2, and is finally sent to node H via VIN_1 by [6] due to step 3.2.2. For message 2, it is first routed to J via VIN_2 by [6] due to step 3.3.2, is routed to node K via $VC_{1,2}$ due to step 2, is routed to node L via VIN_1 by [6] due to step 3.3.2, is routed to node M via $VC_{-1,1}$ due to step 2, is finally sent to node N via VIN_1 by [6] due to step 3.2.2.

Theorem 1 *Algorithm 3.1 is deadlock-free with two virtual channels per physical link.*

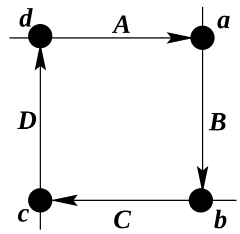

Figure 3. A waiting cycle of four messages A, B, C, and D.

PROOF. The proof is divided into three parts: (P1) we assign each virtual channel x one channel number $num(x)$, (P2) we prove that a message can always find a virtual channel to use for each one hop in a non-decreasing order of channel numbers, (P3) A waiting cycle is not a real deadlock.

For P1, let virtual channel x be output from a node in S_i $(i \geq 0)$. Then, $num(x)$ is set to i if virtual channel x is in VIN_1, and $num(x)$ is set to $-i$ if virtual channel x is in VIN_2. For P2, suppose a message, whose destination $N_D \in S_r$, is sent to N_C via virtual channel y and will be sent out from N_C via virtual channel x. We need to show $num(x) \geq num(y)$. If $N_C \neq N_R$, then $num(x) = num(y)$. Thus, we consider $N_C \in S_q$ receives the message from an intermediate node $N_I \in S_t$ ($S_t \neq S_q$) via virtual channel y. Three cases are discussed: (C2.1) vertex V_t is an ancestor of vertex V_r in the directed path graph, (C2.2) vertex V_r is an ancestor of vertex V_t in the directed path graph, and (C2.3) vertices V_r and V_t has least common ancestor V_{lca}. For C2.1, (V_t, V_q) is a directed edge in the directed shortest path P_{down} from vertex V_t to vertex V_r due to step 4.1.2 of algorithm 2.1. It implies $t < q \leq r$ and virtual channels x and y are all in VIN_1. Thus $num(y) = t < q = num(x)$. For C2.2, (V_q, V_t) is a directed edge in the directed shortest path P_{up} from vertex V_r to vertex V_t due to step 4.1.1 of algorithm 2.1. It implies $r \leq q < t$ and virtual channels x and y are all in VIN_2. And $num(y) = -t < -q = num(x)$. For C2.3, (V_q, V_t) is a directed edge in the directed shortest path P_{up} from vertex V_{lca} to vertex V_t due to step 4.1.3 of algorithm 2.1. If $V_q = V_{lca}$, then virtual channel y is in VIN_2 and virtual channel x is in VIN_1. Thus, $num(y) = -t < q = num(x)$. If $V_q \neq V_{lca}$, then virtual channels x and y are all in VIN_2. And, $num(y) = -t < -q = num(x)$. For P3, consider a waiting cycle as shown in Figure 3, where message A (resp. B, C, D) holds virtual channel d (resp. a, b, c) and requests a (resp. b, c, d). By P2, we have $num(a) \geq num(d) \geq num(c) \geq num(b) \geq num(a)$. It implies $num(a) = num(d) = num(c) = num(b)$. Thus, virtual channels a, b, c and d are in a M_i. If $i = 0$, then messages are routed by an adaptive and deadlock-free routing algorithm presented in [10]. Otherwise, messages are routed by an adaptive and deadlock-free routing algorithm presented in [6]. Thus, the waiting cycle is not a real deadlock. □

4. Conclusion

In a concurrent multicomputer, a reliable routing algorithm requires deadlock-freedom and fault-tolerance. Many researchers [1, 3, 10] proposed adaptive and deadlock-free routing algorithms using a certain number of virtual channels per physical link. But the shape of the tolerated faults should be rectangular. Thus, some good nodes need to be seen as faulty nodes and prohibited from interchanging messages with the other good nodes. In this paper, using two virtual channels per physical link, we propose an adaptive routing algorithm to tolerate irregular faulty patterns. Thus the node utilization is increased up to 100%. Our method requires a central control unit or the global information of the node state.

References

[1] R. V. Boppana and S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks," *IEEE Trans. Comput.*, vol. 44, pp. 848-864, 1995.

[2] Y. M. Boura and C. R. Das, "Fault-tolerant routing in mesh networks," *International Conference on Parallel Processing*, pp. 106-109, 1995.

[3] A. A. Chien and J. H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors," *19th Annual International Symposium on Computer Arch.*, pp. 268-277, 1992.

[4] C. M. Cunningham and D. R. Avresky, "Fault-tolerant adaptive routing for two-dimensional meshes," *1st IEEE Symposium on High Performance Computer Arch.*, pp. 122-131, 1995.

[5] W. J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Trans. Parallel Distributed Syst.*, vol. 4., pp. 466-475, 1993.

[6] C. J. Glass and L. M. Ni, "Fault-tolerant wormhole routing in meshes," *23rd International Symposium on Fault-Tolerant Computing.*, pp. 240-249, 1993.

[7] R. L. Hadas and E. Brandt, "Original-based fault-tolerant routing in the mesh," *1st IEEE Symposium on High Performance Computer Arch.*, pp. 102-111, 1995.

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

[9] C. Seitz et al., *Wormhole Chip Project Report*, Winter 1985.

[10] C. C. Su and K. G. Shin, "Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes," *IEEE Trans. Comput.*, vol. 45, pp. 666-683, 1996.

[11] M. J. Tsai, "Adaptive Fault-Tolerant Routing for Multicomputers," submitted to *IEEE Trans. Comput.*.