Adaptive Fault-Tolerant Wormhole Routing in 2D Meshes

Jipeng Zhou and Francis C.M. Lau
Department of Computer Science and Information Systems
The University of Hong Kong
Pokfulam Road, Hong Kong, P.R. China
E-mail:{jpzhou,fcmlau}@csis.hku.hk

Abstract

We present an adaptive fault-tolerant wormhole routing algorithm for 2D meshes. The main feature is that with the algorithm, a normal routing message, when blocked by some faulty processor, would detour along the f-polygons around the fault region. The proposed algorithm can tolerate convex faults with only three virtual channels per physical channel regardless of the overlapping of f-polygons of different fault regions. The proposed algorithm is deadlock-free.

Index Terms: *virtual channel, adaptive fault-tolerant routing, wormhole routing, disjoint fault-connected regions, deadlock freedom.*

1 Introduction

In multiprocessor systems, routing algorithms provide the mechanism for communication between processors. The efficiency of routing algorithms is important for achieving high performance in multiprocessor systems. There are two types of message routing: *deterministic routing* that uses only a single path from source to destination, and *adaptive routing* that allows more freedom in selecting message paths. Most commercial multiprocessor computers use deterministic routing because of its deadlock-freedom and ease of implementation. Adaptive routing on the other hand can reduce network latency and increase network throughput, and it can also tolerate more faults than deterministic routing. Adaptive routing, however, could be more prone to deadlock and/or livelock problems [12]. A deadlock occurs when a message waits for an event that will never happen; a livelock keeps a message moving indefinitely but not letting it reach the destination.

There exist some adaptive fault-tolerant wormhole routing algorithms [8][9][4][7][3][12][11] for mesh computers. Duato's design methodologies [4][5][6][7] have been used to design a fault-tolerant routing algorithm in meshes successfully. His adaptive algorithm can tolerate at least $n-1$ faulty channels in an n-dimensional mesh, which requires at least four virtual channels per physical channel. X. Lin et al. [10] proposed a message flow model in wormhole networks, and applied it to develop an adaptive wormhole routing algorithm for double Y-channel 2D meshes; their model is different from the turn model and can easily produce fault-tolerant routings. Chien and Kim [3] proposed the fault-tolerant PAR (planar adaptive routing) algorithm for n-dimensional meshes. Their algorithms can tolerate rectangle faults with no overlapping of f-rings. Su and Shin [12] proposed an adaptive fault-tolerant routing algorithm for n-dimensional meshes. Their algorithms can tolerate what is called a *disconnected rectangular block* in an n-dimensional mesh, where the distance between any two nodes in different faulty blocks is at least 3 in each dimension. Shih [11] improved the result of Su and Shin [12]; his algorithm can tolerate block faults, but the distance between any two nodes in different faulty blocks is at least 2 in each dimension. In [4], Duato describes a deadlock-free adaptive routing theory similar to that in [12], which can be used to develop adaptive fault-tolerant routing algorithms. Boppana and Chalasani proposed fault-tolerant routing algorithms for mesh networks [1][2], where deadlocks can be prevented by using four virtual channels per physical channel for deterministic and adaptive fault-tolerant routing; the faults in their model are rectangular [1] and sepcial convex [2] such as L, T or + (they call them concave and the solid fault model), but not including any boundary faulty nodes in the network. Our adaptive fault-tolerant wormhole routing algorithm can be contrasted with [1, 2, 4, 7, 3, 12] as shown in Table 1.

This paper is organized as follows. Section 2 describes the fault model. Section 3 presents the protocol of setting up the boundary routing paths of fault-connected regions. Section 4 proposes an adaptive fault-tolerant routing algorithm for 2D meshes. Section 5 concludes the paper.

/* All the following algorithms use local fault knowledge and all ports */

	fault model	virtual channels	faults tolerated	routing technique
Duato [4][7]	Arbitrary	4	Up to $n-1$ in nD meshes	adaptive
Chien and Kim [3]	Rectangles (non-overlapping f-ring only)	3	Multiple faulty rectangles	planar-adaptive
Su and Shin [12]	Disconnected rectangles	2	Multiple faulty rectangles	adaptive
Shih [11]	Rectangles	2	faulty rectangles	adaptive
Chalasani and Boppana [1] [2]	Special convex (of f-rings)	4	multiple convex regions (f-rings)	e-cube or adaptive
Our results	Convex (f-rings and f-chains)	3	multiple convex regions (f-rings and f-chains)	adaptive

Table 1. The comparison of techniques for fault-tolerant wormhole routing

2 Model of Faults

There are two different types of faults: either the entire processing element (PE) and its associated router can fail, or just a physical link may fail. When a physical link fails, all virtual channels on that particular physical link are marked as faulty. When a PE and its router fail, all physical links incident on the failed PE are also marked as faulty. In this paper, faulty PEs are considered a basic faulty element for simplicity. In order to find fault-connected regions, each PE should know the states (faulty or safe) of its enclosing PEs after running a diagnostic program. Two PEs (x_0, y_0) and (x_1, y_1) are called 4-neighbors if $|x_0-x_1|+|y_0-y_1| = 1$, 8-neighbors if $max\{|x_0-x_1|, |y_0-y_1|\} \leq 1$. Connectivity of faulty PEs is defined in terms of adjacency relation among faulty PEs.

Definition 1 *Given that C is the connectivity relation in a mesh M, for any pair of processors $p, q \in M$ and a given faulty processor set F, we have $(p, q) \in C$ if and only if $p, q \in F$ and there exists $a_1, a_2, \dots, a_n \in F$ such that $a_1 a_2 \dots a_n$ is a connected path in F, where $p = a_1, q = a_n$, and the connectivity is 8-connectivity, i.e., a_{i+1} is a 8-neighbor of a_i, $1 \leq i \leq n-1$.* □

It is clear that the relation C is an equivalence relation that partitions the faulty processors into disjoint equivalence classes. Each equivalence class is called a *fault-connected region*. Two fault-connected regions are given in Figure 1, with one consisting of a set of PEs, $\{(1,1), (1,2), (2,2), (3,2), (2,3)\}$, and another one of $\{(5,2), (5,3)\}$.

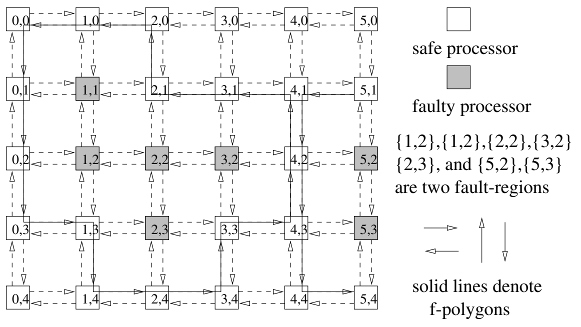

Figure 1. Two fault-connected regions and their f-polygons

Definition 2 *A fault-connected region F is convex if for every pair of PEs, $p, q \in F$, that are in the same 1D across section, all PEs between p and q in the same 1D cross section are also faulty.* □

In this paper, the convex fault-connected regions are used as the fault model, and overlapping of fault-connected regions at the boundary PEs is allowed. For example, two convex fault-connected regions are shown in Figure 1, where one contains no boundary node of the network, while the other one has, and the contours of the two fault regions have overlapping nodes, $(4,1)$, $(4,2)$ and $(4,3)$.

3 The Boundary of Fault-Connected Regions

For simplicity and easy description of the fault-tolerant wormhole routing algorithm and the proof of deadlock free-

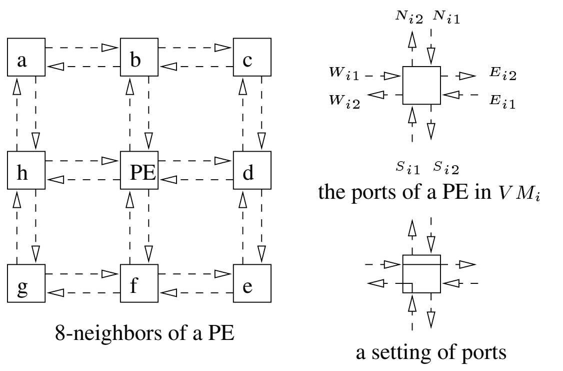

Figure 2. The eight neighbors and ports of a PE

dom, we can connect the safe PEs and channels around a fault region to form directional paths (ring or chain). These paths can be used to detour the routing messages when normal routings are blocked by faults. If a fault-connected region does not touch the boundary PEs of the mesh, then the safe PEs around it form a directional ring, which is called an *f-ring*. If a fault-connected region touches the boundary PEs of the mesh, then the safe PEs around it form a chain, which is called an *f-chain*. Each f-chain has two end PEs. The rings and chains around the fault-connected regions are *f-polygons*. The PEs on the f-polygons are adjacent to at least one faulty PE in the fault-connected regions.

Every PE except the boundary PEs of the mesh has eight neighbors, which are represented by the eight Boolean variables *a,b,c,d,e,f,g,h*. The PE communicates with its neighbors through its input ports $\{W_{i1}, E_{i1}, N_{i1}, S_{i1}\}$ and output ports $\{W_{i2}, E_{i2}, N_{i2}, S_{i2}\}$ in a virtual network VM_i, as shown in Figure 2. Each PE maintains three virtual channels which timeshare the bandwidth of each physical channel. The sum of all these virtual channels form three virtual networks, $VM_i, 1 \leq i \leq 3$ respectively. A Boolean variable is equal to 1 if the corresponding neighbor is a safe PE, to 0 if the corresponding neighbor is a faulty PE.

The boundary routing paths around a fault-connected region will be set up in two stages. First stage, the disjoint f-rings and f-chains are set using VM_i. Second stage, f-chains in opposite directions are set around fault-connected regions with boundary faulty PEs of the network—i.e., there are two f-chains in opposite directions around fault-connected regions with boundary faulty PEs of the network in VM_i.

Procedure 1: Setting up disjoint f-polygons
1. Each PE tests the state of its neighbors
2. Each PE sets the connection of its ports
according to Table 2

The disjoint f-polygons around the fault-connected regions can be set up according to Procedure 1 in VM_i. In constant time, every PE can detect the state of its eight neighbors, which is either safe or faulty. Each PE sets its port connections according to Table 2, where the capital letter pair in braces denotes the directional link between two ports of a PE; e.g., $\{W_{i1}, N_{i2}\}$ denotes a directional link from W_{i1} to N_{i2}, and $\{W_{i1}, N_{i2}\} = h\bar{a}b$ means that the connection $\{W_{i1}, N_{i2}\}$ (i.e., from port W_{i1} to port N_{i2}) is established if neighbors h and b are safe PEs and a is a faulty PE. After the connections of the ports are set up according to the Table 2, we have all f-rings in counter-clockwise orientation, and every f-chain has its two end PEs touching the boundary of the network, and the direction of the f-chain is from one end PE to the other. Figure 1 shows the f-polygons for some connected faulty regions, where the black box denotes the faulty processor, empty box denotes safe processor, solid lines are f-polygons—one is an f-ring, another is an f-chain.

If a fault-connected region contains the boundary PEs of the network, the f-polygon around it would form an f-chain, which is from one end (called chain head) to another end (called chain tail). In order to have a fault-tolerant algorithm, a special procedure is needed when the f-polygon of a fault-connected region forms a chain: another f-chain in the different direction in VM_i needs to be set.

Procedure 2: Setting up new f-chains
1. Each PE tests the state of its neighbors,
2. Each PE that satisfies Table 3 is the chain head, and the output port of it is given in Table 3,
3. Each chain head sends its coordinates to all PEs on the chain,
4. Each PE, receiving the head's messages, connects its ports according to Table 4.

All chain heads can be detected in step 1 and step 2 in Procedure 2. For PE $(5, 1)$ in Figure 1, after detecting the states of its neighbors, which makes $\ddot{d}\bar{f}h = 1$, PE $(5, 1)$ becomes the chain head; the port W_{i2} of $(5, 1)$ is the output port of the chain head according to Table 3, where the letter with two dots on top means that the corresponding neighbor does not exist (the boundary PE of the network is the only possible chain head candidate). Each chain head then sends its coordinates to all PEs on the chain in step 3 of Procedure 2, and all PEs, upon receiving this message, connect their ports according to Table 4 in step 4 of Procedure 2. This step sets up a new chain in VM_i. For simplicity, the original chain is called a chain in VM_{i1}, and the new chain is called a chain in VM_{i2} which has the same chain head as the f-chain in VM_{i1}. Such an f-chain is shown in Figure 3. The time cost of Procedure 1 is $O(1)$. For Procedure 2, the time costs of steps 1, 2, and 4 are also $O(1)$, and the cost of step 3 depends on the length of the chain, which is at most $O(m + n)$ for an $m \times n$ mesh.

$\{W_{i1}, N_{i2}\} = h\bar{a}b$	$\{E_{i1}, S_{i2}\} = d\bar{e}f$	$\{N_{i1}, E_{i2}\} = b\bar{c}d$	$\{S_{i1}, W_{i2}\} = f\bar{g}h$
$\{W_{i1}, E_{i2}\} = h\bar{b}d$	$\{E_{i1}, W_{i2}\} = d\bar{f}h$	$\{N_{i1}, S_{i2}\} = b\bar{d}f$	$\{S_{i1}, N_{i2}\} = f\bar{h}b$
$\{W_{i1}, S_{i2}\} = h\bar{b}\bar{d}f$	$\{E_{i1}, N_{i2}\} = d\bar{f}\bar{h}b$	$\{N_{i1}, W_{i2}\} = b\bar{d}\bar{f}h$	$\{S_{i1}, E_{i2}\} = f\bar{h}\bar{b}d$

Table 2. The ports' connections of PEs for f-polygon

boundary PE	on left edge	on right edge	on upper edge	on down edge
	$\{E_{i2}\} = \ddot{h}\bar{b}d$	$\{W_{i2}\} = \ddot{d}\bar{f}h$	$\{S_{i2}\} = \ddot{b}\bar{d}f$	$\{N_{i2}\} = \ddot{f}\bar{h}b$
	$\{S_{i2}\} = \ddot{h}\bar{b}\bar{d}f$	$\{N_{i2}\} = \ddot{d}\bar{b}\bar{f}h$	$\{W_{i2}\} = \ddot{b}\bar{d}\bar{f}h$	$\{E_{i2}\} = \ddot{f}\bar{b}\bar{h}d$
corner PE	up left corner	down right corner	up right corner	down left corner
	$\{S_{i2}\} = \ddot{h}\ddot{b}f\bar{d}$	$\{N_{i2}\} = \ddot{d}\ddot{f}b\bar{h}$	$\{W_{i2}\} = \ddot{b}\ddot{d}\bar{f}h$	$\{E_{i2}\} = \ddot{f}\ddot{h}\bar{b}d$

Table 3. The detected chain head

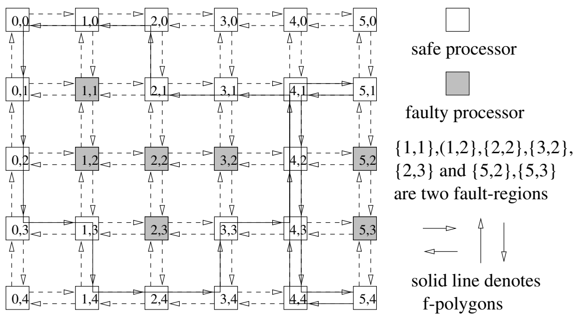

Figure 3. Overlapped f-ring and f-chain in VM_i

4 An Adaptive Fault-Tolerant Routing Algorithm

The deadlock-free adaptive fault-tolerant routing algorithm for 2D meshes we are going to present needs to use three virtual channels per physical channel in order to tolerate the convex faults in our model with possible overlapping of PEs across f-polygons. The virtual networks are VM_1, VM_2, and VM_3, and the preprocessing in Section 3 is needed for the PEs on the f-chain to get information about its chain head when there exist boundary faulty PEs. For description convenience, the message routings according to the current PE (x_c, y_c) and the destination (x_d, y_d) are divided into four types: WE, EW, SN and NS routings, where WE (from west to east) routing is taken if $x_c < x_d$, EW (from east to west) routing if $x_c > x_d$, NS (from north to south) routing if $x_c = x_d$ and $y_c < y_d$, and SN (from south to north) routing if $x_c = x_d$ and $y_c > y_d$. WE and EW routings may be changed into SN or NS routings, but not vice versa. In wormhole routing, the header flit contains the address of the destination, and each PE applies the fault-tolerant routing to direct movement of the header flit. It is assumed that each PE knows the status of the PEs incident upon it. A *normal* message routing is along the channel on the minimal routing path. When a normal routing is blocked by a faulty PE, the routing is detoured along the f-polygon around the fault-region—called misrouting.

With algorithm AFTR, when there exists an outgoing channel in a shortest path from C to D which is not blocked by any fault region, a normal adaptive routing channel is used to route the message according to Table 5. According to Table 5, if channels are available on the shortest path from C to D in the corresponding qualified virtual network, each routing selects an outgoing channel to forward the message. If all qualified channels are busy, then each routing will wait for the channels on the shortest path from C to D in the corresponding denoted virtual networks. When a normal minimal adaptive routing is blocked by a fault-region, meaning that the destination is at least one more hop away from the current PE, the message is misrouted along the f-polygon of the fault-region until $d_c < d_{flag}$—i.e., the routing header has moved at least one hop closer to the destination.

If a normal adaptive routing is blocked by a fault-region with an f-ring, i.e., the f-polygon is an f-ring, the misrouting channel is determined by Table 6, where if channels along the f-ring in corresponding qualified virtual network are available, the message is forwarded along one of these channels; otherwise the routing will wait for the channel on the f-ring in the corresponding virtual network. Since each PE on the f-ring has one input port and one output port along the f-ring, the misrouting message goes forward from an input channel to an output channel along the f-ring, and so Table 6 only gives the virtual network where the f-ring is in. When a normal adaptive routing is blocked by a fault-region with f-chains, since each PE on the f-chains has two possible misrouting directions along two different f-chains in each virtual network VM_i (except the ends of the chain), and only one outgoing channel needs to be selected, the misrouting direction is determined by Table 7.

$\{N_{i1}, W_{i2}\} = h\bar{a}b$	$\{S_{i1}, E_{i2}\} = d\bar{e}f$	$\{E_{i1}, N_{i2}\} = b\bar{c}d$	$\{W_{i1}, S_{i2}\} = f\bar{g}h$
$\{E_{i1}, W_{i2}\} = h\bar{b}d$	$\{W_{i1}, E_{i2}\} = d\bar{f}h$	$\{S_{i1}, N_{i2}\} = b\bar{d}f$	$\{N_{i1}, S_{i2}\} = f\bar{h}b$
$\{S_{i1}, W_{i2}\} = h\bar{b}\bar{d}f$	$\{N_{i1}, E_{i2}\} = d\bar{f}\bar{h}b$	$\{W_{i1}, N_{i2}\} = b\bar{d}\bar{f}h$	$\{E_{i1}, S_{i2}\} = f\bar{h}\bar{b}d$

Table 4. The ports' connections of a PE for f-chain in VM_i

Since the fault-regions are convex, and the preprocessing in Section 3 lets the PE know that the misrouting is along the f-ring or f-chain, Table 7 guarantees that the misrouting along the f-chain only uses the channels on a chain in virtual network VM_1, VM_2, or VM_3, and so Table 7 only gives the virtual network in which the misrouting would forward the message along a chain of the fault-region.

```
Algorithm: AFTR (Adaptive Fault-Tolerant Routing)
/* Let $S = (x_s, y_s)$ and $D = (x_d, y_d)$ be the addresses */
/* of the source and the destination resp., $C = (x_c, y_c)$ */
/* be the current PE, where the routing header is on.*/
/* Let $d_c = |x_c - x_d| + |y_c - y_d|$ be the distance from */
/* $C$ to $D$. Initially, $x_c = x_s, y_c = y_s$ */
1. If $x_c = x_d$ and $y_c = y_d$,
2.    consume routing message and return
   End If
3. If outgoing channels in shortest paths from $C$ to $D$
      are not blocked by faults
4.    determine an outgoing channel in shortest paths
      from $C$ to $D$ according to Table 5
   End If
5. If outgoing channels in shortest paths from $C$ to $D$
      are blocked by a fault-region
6.    let $d_{flag} = d_c$
7.    determine routing channel along the f-ring according
      to Table 6
8.    determine routing channel along the f-chain according
      to Table 7
9.    If $x_c < x_d$ (WE routing) or $x_c > x_d$ (EW routing)
10.      the routing header goes forward along the
         f-polygon until $d_c < d_{flag}$ or $x_c = x_d$
      End If
11.   If $x_c = x_d$ and $y_c > y_d$ (SN routing) or
         $x_c = x_d$ and $y_c < y_d$ (NS routing)
12.      the routing header goes forward along the
         f-polygon until $d_c < d_{flag}$ and $x_c = x_d$
      End If
   End If
```

The proof of deadlock-freedom of the adaptive fault-tolerant algorithm AFTR is decomposed into three parts: first, there is no cyclic dependency (forming a waiting cycle) among routing channels when the qualified virtual network is the same as the denoted virtual network for the corresponding routings, and when there is no faulty boundary PE of the network; second, there is no cyclic dependency among routing channels when the qualified virtual network is the same as the denoted virtual network for the corresponding routings, and when there exist faulty boundary PEs of the network; third, algorithm AFTR is deadlock-free.

4.1 The Proofs of Part 1 and Part 2

We skip the proofs of part 1 and part 2, which are the same as the those in a related paper [13].

Lemma 1 *There is no cyclic dependency among channels of WE routings in VM_1.*

Lemma 2 *There is no cyclic dependency among channels of EW and SN routings in VM_2.*

Lemma 3 *There is no cyclic dependency among channels of NS routings in VM_3.*

Theorem 1 *There is no cyclic dependency among routing channels by using Algorithm AFTR when the qualified virtual network is the same as the denoted virtual network for the corresponding routings, and when there is no faulty boundary PE of the network.*

Lemma 4 *There is no cyclic dependency among channels of NS routings in VM_3, when there are faulty boundary PEs of network.*

Lemma 5 *There is no cyclic dependency among channels of WE routings in VM_1 and VM_3 and NS routing in VM_3, when there are faulty boundary PEs of network.*

Lemma 6 *There is no cyclic dependency among channels of EW and SN routings in VM_2 and VM_3 and NS routings in VM_3, when there are faulty boundary PEs of network.*

Theorem 2 *There is no cyclic dependency among routing channels by using algorithm AFTR when the qualified virtual network is the same as the denoted virtual network for corresponding routings, and when there may exist faulty boundary PEs of the network.*

Routing type	channels in qualified virtual networks	channels in denoted networks
WE routing	in VM_1	in VM_1
EW routing	in VM_1 or VM_2	in VM_2
SN routing	in VM_1,VM_2 or VM_3	in VM_2
NS routing	in VM_1,VM_2 or VM_3	in VM_3

Table 5. The strategy for channel selection of normal message routing

Routing type	channels in qualified virtual networks	channels in denoted networks
WE routing	in VM_1	in VM_1
EW routing	in VM_1 or VM_2	in VM_2
SN routing	in VM_1 or VM_2	in VM_2
NS routing	in VM_1,VM_2 or VM_3	in VM_3

Table 6. The strategy for channel selection of misrouting along f-rings

4.2 The Proof of Deadlock Freedom of Algorithm AFTR

Theorem 3 *The adaptive fault-tolerant routing algorithm AFTR is deadlock-free.*

Proof: We will prove that the routing channels for all message routings are available according to the adaptive fault-tolerant routing algorithm AFTR.

- If the channels for NS routings (normal or misrouting along f-ring) in VM_1, VM_2 or VM_3 are available, the routing messages are forwarded along selected channels; if all channels for NS routings in VM_1, VM_2 and VM_3 are busy, there may be a waiting cycle, where the NS routings only wait for the channels in VM_3; since normal SN and NS routings use different routing channels in VM_3, one does not change into another, and no other routings use channels in VM_3 except normal SN routings and routings in part 1 and part 2, and so according to the results of part 1 and part 2, the channels, which NS routings wait for, are available eventually, and NS routings should forward the messages to their destinations.

- If the channels for normal SN routings in VM_1, VM_2 or VM_3 (SN misroutings along f-ring in VM_1 or VM_2) are available, the routing messages are forwarded along selected channels; if all channels for normal SN routings in VM_1, VM_2 and VM_3 (SN misroutings along f-ring in VM_1 and VM_2) are busy, the SN routings (normal and misrouting along f-ring) only wait for the channels in VM_2; since normal SN and NS routings use different routing channels in VM_3 and one does not change into another, no other routings use channels in VM_3 except NS routings and routings in part 1 and part 2; since NS routings should reach the destinations, so according to the results of part 1 and part 2, the channels, which SN routings wait for, are available eventually, and SN routings should forward the messages to their destinations too.

- If the channels for normal EW routings (misroutings along f-rings) in VM_1 or VM_2 are available, the routing messages are forwarded along selected channels; if all channels for normal EW routings (misroutings along f-rings) in VM_1 and VM_2 are busy, the EW routings (normal and misrouting) only wait for the channels in VM_2, and no other routings use channels in VM_2 except NS routings and routings in part 1 and part 2; since NS routings should reach the destinations, so according to the result of part 1 and part 2, the channels, which EW routings wait for, are available eventually.

- Since WE normal routing only uses channels in VM_1, WE misroutings along c-ring may use channels in VM_3; no other routings (i.e., EW, SN and NS routings) are changed into WE routings; since the channels for EW, SN and NS routings are available eventually, and with the results of part 1 and part 2, the routing channels for WE routings (normal and misroutings) are available eventually too.

Therefore, the algorithm AFTR is deadlock-free. □

5 Conclusion

In this paper, an adaptive fault-tolerant wormhole routing algorithm is proposed for 2D meshes. The proposed algorithm can tolerate convex fault-regions regardless of overlapping of PEs on the f-polygons, and only three virtual

/* The mesh is $n \times m$, (x_s, y_s) is source, (x_d, y_d) is destination,
(x_c, y_c) is the current PE, (x_h, y_h) is the chain head */

routing types	f-chain types	the head of f-chain	channel
WE routing $x_c < x_d$	E_{i2}	$y_c \geq y_h$ and $y_h \neq n-1$	in VM_{11}
		$y_c < y_h$ or $y_h = n-1$	in VM_{32}
	S_{i2}	$y_c \geq y_h$	in VM_{11}
		$y_c < y_h$	in VM_{32}
	W_{i2}	$y_d > y_h$ or $y_h = 0$	in VM_{11}
		$y_d \leq y_h$ and $y_h \neq 0$	in VM_{32}
	N_{i2}	$y_d > y_h$	in VM_{11}
		$y_d \leq y_h$	in VM_{32}
EW routing $x_c > x_d$	E_{i2}	$y_d \geq y_h$ and $y_h \neq n-1$	in VM_{32}
		$y_d < y_h$ or $y_h = n-1$	in VM_{21}
	S_{i2}	$y_d \geq y_h$	in VM_{32}
		$y_d < y_h$	in VM_{21}
	W_{i2}	$y_c \leq y_h$ and $y_h \neq 0$	in VM_{21}
		$y_c > y_h$ or $y_h = 0$	in VM_{32}
	N_{i2}	$y_c > y_h$	in VM_{32}
		$y_c \leq y_h$	in VM_{21}
NS routing $x_c = x_d$ and $y_c < y_d$	E_{i2}	$x_d \geq x_h$	in VM_{32}
		$x_d < x_h$	in VM_{31}
	S_{i2}	$x_c > x_h$ or $x_h = 0$	in VM_{32}
		$x_c \leq x_h$ and $x_h \neq 0$	in VM_{31}
	W_{i2}	$x_c > x_h$	in VM_{32}
		$x_c \leq x_h$	in VM_{31}
	N_{i2}	$x_d \geq x_h$ and $x_h \neq m-1$	in VM_{32}
		$x_d < x_h$ or $x_h = m-1$	in VM_{31}
SN routing $x_c = x_d$ and $y_c > y_d$	E_{i2}	$x_c \geq x_h$	in VM_{21}
		$x_c < x_h$	in VM_{32}
	S_{i2}	$x_d > x_h$ or $x_h = 0$	in VM_{21}
		$x_d \leq x_h$ and $x_h \neq 0$	in VM_{32}
	W_{i2}	$x_d > x_h$	in VM_{21}
		$x_d \leq x_h$	in VM_{32}
	N_{i2}	$x_c \geq x_h$ and $x_h \neq m-1$	in VM_{21}
		$x_c < x_h$ or $x_h = m-1$	in VM_{32}

Table 7. The channel selection strategy for misrouting along f-chains

channels per physical channel are required. The objective of fault-tolerant routing is to maximize the ability of the good processors in a direct network to communicate with each other when there are faulty processors. Ideally, all of the good nodes, when they are connected, would be able to communicate with each other, regardless of the number and positions of the faulty processors and channels. There may be any number of fault-regions in the faulty mesh as long as the network remains connected, and the fault-regions may be of any form, not only convex; so more general fault-tolerant routing algorithms need to be discovered in future research.

References

[1] R.V. Boppana and S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks," *IEEE Trans. Computers,* 44(7): 848–864, July 1995.

[2] S. Chalasani and R.V. Boppana, "Communication in multicomputers with nonconvex faults," *IEEE Trans. Computers,* 46(5): 616–622, May 1997.

[3] A.A. Chien and J.H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors," *Proc. 19th Ann. Int'l Symp. Computer Architecture*, pp. 268–277, May 1992.

[4] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Trans. Parallel and Distributed Syst.,* 4(12): 1320–1331, Dec. 1993.

[5] J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks," *Proc. Int'l Conf. Parallel Processing*, Aug. 1994.

[6] J. Duato, "A theory to increase the effective redundancy in wormhole networks," *Parallel Processing Letters,* 4(1& 2): 125–138, 1994.

[7] J. Duato, "A theory of fault-tolerant routing in wormhole networks," *Proc. 1994 Int'l Conf. Parallel and Distributed System,* pp. 600–607, Dec. 1994.

[8] C.J. Glass and L.M. Ni, "Fault-tolerant wormhole routing in meshes," *Proc. IEEE 23rd Int'l Symp. Fault-Tolerant Computing* pp. 240–249, 1993.

[9] C.J. Glass and L.M. Ni, "Fault-tolerant wormhole routing in meshes without virtual channels," *IEEE Trans. Parallel and Distributed Systems,* 7(6): 620–636, June 1996.

[10] X. Lin, P.K. Mckinley, and L.M. Ni, "The message flow model for routing in wormhole-routed networks," *IEEE Trans. Parallel and Distributed Systems,* 6(7): 755–760, July, 1995.

[11] J.D. Shil, "Adaptive fault-tolerant wormhole routing algorithms for hypercube and mesh interconnection networks," *Proc. 11th Parallel Processing Symposium*, pp. 333–340, April 1997.

[12] C.C. Su and K.G. Shin, "Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes," *IEEE Trans. Computers,* 45(6): 666–683, June 1996.

[13] J.P. Zhou and F.C.M. Lau, "Fault-tolerant wormhole routing in 2D meshes," *Proc. 2000 International Symposium on Parallel Architectures, Algorithms and Networks*, Dec. 2000.