Computers Math. Applic. Vol. 23, No. 1, pp. 3–15, 1992
Printed in Great Britain.

0097-4943/92 $5.00 + 0.00

ADAPTIVE AND FAULT-TOLERANT ROUTING ALGORITHMS FOR HIGH PERFORMANCE 2D TORUS INTERCONNECTION NETWORK

TING-WEI HOU
Department of Engineering Science, National Cheng Kung University
Tainan City 70101, Taiwan, R.O.C.
S. R. TSAI
Computer Systems Lab., Department of Electrical Engineering
National Cheng Kung University, Tainan City 70101, Taiwan, R.O.C.
L. M. TSENG
Department of Electrical Engineering, National Central University
Chongli, Taiwan, R.O.C.

(Received February 1990)

Abstract—Adaptive and fault-tolerant schemes for routing messages in a 2D torus interconnection network for distributed memory multi-computers (message passing concurrent computers) are presented. For the adaptive scheme, two new techniques, channel switching and dimension switching, are developed and proved deadlock-free. For the fault-tolerant scheme, a message can be rerouted to a virtual destination, which in turn sends the message to the real destination. This scheme can tolerate all single faults and many multiple faults, and is deadlock-free. The two routing schemes are suitable for the high performance virtual cut-through and wormhole routing. The required hardware overhead for realizing the fault-tolerant scheme is small and no time penalty is paid in the fault-free case.

1. INTRODUCTION

Some commercial and experimental distributed memory multi-computers (message passing concurrent computers) with a large number of processors are currently available or under development. Linear speedup in excess of 100 Mflops has been achieved, using 1024 node hypercube [1]. A vital component of these systems is the interconnection network that enables the processing nodes to communicate among themselves. The performance of the system depends to a great extent on the internode communication possible. In particular, the failure of a component in the interconnection network can frequently bring down the entire system unless the network can tolerate such failures.

An important class of the network topology for the message passing concurrent computers is the torus (k-ary n-cube). Hypercubes, toroidal meshes, and rings are examples of the torus network. In the first generation of these machines, hypercube topology was preferred. But low dimensional torus (2D or 3D) are widely used in the recent development of these machines, such as TRC [2], Horizon [3], and iWarp [4].

Academic studies justify the choice of a 2D torus. Vitányi [5] assumes that interconnection topologies must be embedded in 3D space and derives lower bounds on total wire length of a few graph topologies. It has been shown that the mesh connected architecture is the most cost effective. Dally [6] analyzes k-ary n-cube, including rings, tori, and hypercubes, under the assumptions that (1) networks must be embedded into the plane, (2) wire density is held constant

This research work was supported by the grants from National Science Committee of Republic of China under contract NSC-77-0408-E-00 and NSC-77-0204-E-006-03.

for networks with the same number of nodes, (3) wormhole routing is used, (4) the channel delay is constant or is logarithmically or linearly proportioned to the wire length. He compares various networks with $N = k^n$ nodes and concludes that low dimensional torus outperforms hypercubes, and a 256 node 2D torus is optimal.

In the recent development of high speed interprocessor communication for distributed memory multi-computers, hardware acceleration techniques, i.e., dedicated hardware routers which enforce virtual cut-through [7] or wormhole routing [6], are adopted. For example, wormhole routing is enforced in iWarp, TRC, and iPSC/2 [8], and virtual cut-through in ComCoBB [9]. Instead of storing each message in a processing node before start transmitting to the next node in the store-and-forward routing, virtual cut-through and wormhole routing send out the message before it is received completely at the communication channel. The delay due to unnecessary buffering in front of an idle channel is avoided. As a result, the message latency is insensitive to the distance in the message passing network and the network topology appears to be a logically fully connected network and consequently the mapping of application is greatly simplified. The major difference between traditional store-and-forward, virtual cut-through and wormhole routing is that: store-and-forward buffers each message, virtual cut-through buffers a message only as it is blocked, and the wormhole routing does not buffer any message in the intermediate channels even when it is blocked.

There are three approaches in designing high performance distributed routing algorithms. The first approach is **deterministic routing**. It is generally proved dead-lock-free, such as the e-cube algorithm [8], the virtual channel algorithm [10], and NDF [11]. The major drawback of this approach is that only one path exists from any source node to a destination node. Thus a fault in a channel causes all the paths that visit the channel to fail. The effect of such faults cannot be ignored in large systems because of the large number of components involved.

The Connection Machine [12], HEP [13], and Horizon [3] use the second approach, termed **desperation routing**, to route messages around congestion or to bypass faulty channels. However, the above implementations for desperation routing have not proved to be deadlock and livelock free [11]. The hyperswitch [14] and the virtual network [15] use the third approach, named **adaptive routing**. The former keeps the "history" of a message within the header to heuristically prune out the known faulty or congested node. This method is not proved deadlock-free and the history concept is not directly applicable to the 2D torus. The latter is proved deadlock-free. However, it is not regular, and not well scalable as the 2D torus is considered. In addition, the faulty conditions are not considered. In summary, none of the existing routing schemes is deadlock-free, adaptive, and fault-tolerant and can be directly applicable to the 2D torus.

Therefore, the adaptive and the fault-tolerant deadlock-free routing schemes for the 2D torus are presented. The approach is suitable for virtual cut-through and wormhole routing. It adaptively selects an available deadlock-free path. A message is rerouted to another node if there is no alternative path left because of the failure of the succeeding channel. We present the adaptive deadlock-free scheme for 2D torus and apply the scheme on 2D mesh in Section 3, after introducing preliminaries and background in Section 2. Fault-tolerant routing for the single fault model is presented in Section 4. The correctness of the schemes is proved and illustrated with examples in corresponding sections respectively.

2. PRELIMINARY AND BACKGROUND

In the graph representation of an interconnection network for message passing concurrent computers, nodes denote the processors and directed edges represent the communication channels. The construction of the interconnection topology is embedded in channels, which enforce the routing algorithms. Nodes communicate with each other by sending/receiving messages. A path from a source node to a destination node is the ordered list of channels visited by a message. An eligible path is the path determined by the routing function.

A 2D torus is a synonym for k-ary 2-cube. Each node has a 2 digit radix k address. A node is denoted as N_{xy} or N_δ, where δ stands for the two digit address, xy. The subscript x represents the position of the node in the X (horizontal) dimension and y represents its position in the Y (vertical) dimension. For example, N_{12} in a 4-ary 2-cube of Figure 1 is at the position 1 in the X dimension and the position 2 in the Y dimension. A **relative destination** of a message

is defined as the node with the same address as the destination node in some dimension of the torus. For example, refer to Figure 1, if N_{22} sends a message to N_{00}, a relative destination of the message is N_{02} in the X dimension and a relative destination in Y dimension is N_{20}.

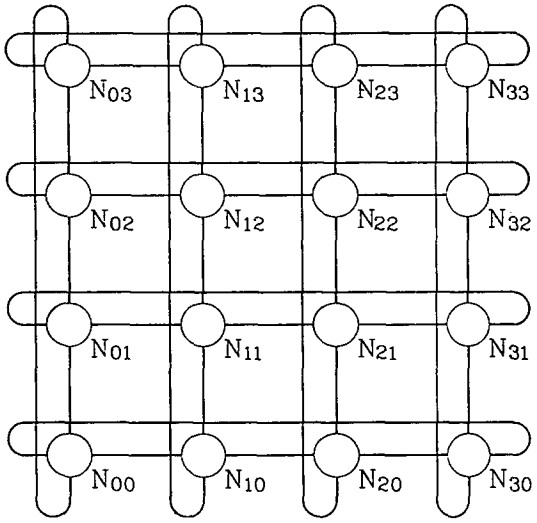


Figure 1. 4-ary 2-cube.

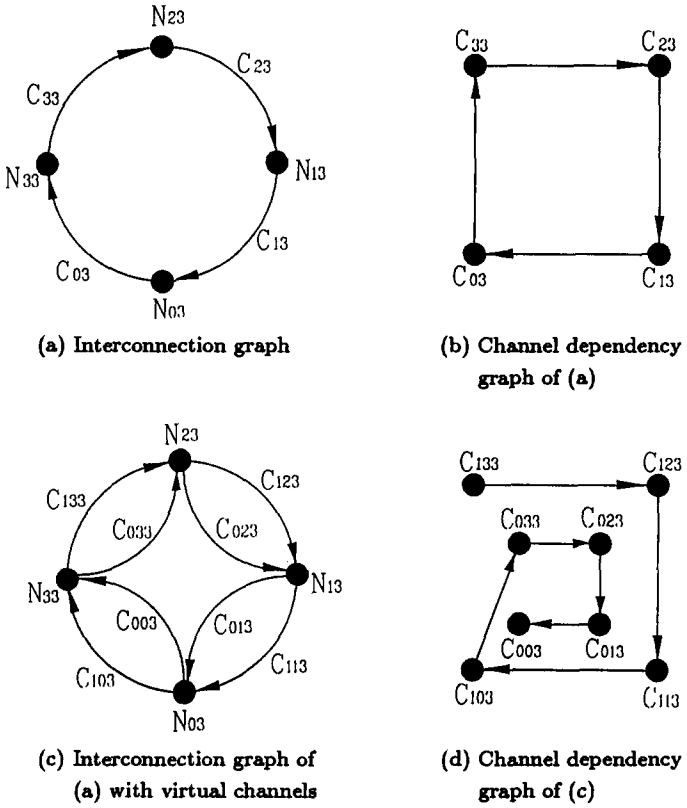
Being deadlock-free is an important feature of high speed communication networks. Many algorithms have been proposed for the store-and-forward routing. The deadlock-free routing algorithms for cut-through routing are identical to those of store-and-forward routing because they both require buffering. However, these algorithms are not applicable to wormhole routing. The original and the most well known solution to the problem of deadlock-free wormhole routing, which is also applicable to cut-through routing, is Dally and Seitz's virtual channel approach [10]. In their approach, a routing function selects the next channel on the path to the destination according to the current channel and the destination node of a message. As the routing function is realized in each channel, the channels form a directed channel dependency graph, where the vertices of the dependency graph are the channels of the interconnection network and the edges are pairs of channels connected by the routing function. For example, Figure 2(a) shows a unidirectional ring, which is the uppermost ring of Figure 1 along the X dimension, and Figure 2(b) shows its channel dependency graph.

Because of the inherent cycles in the channel dependency graph, such as in Figure 2(b), it is possible that cyclic waiting, i.e., deadlock, would occur. By splitting the physical channels, which form cycles, into groups of virtual channels, and restricting the routing of messages to follow a strictly ordered path, the approach can remove cycles in the channel dependency graph.

Consequently, the interconnection is deadlock-free. For example, each channel in Figure 2(a), can be split into high virtual channels, $C_{103}, \ldots, C_{133}$, and low virtual channels, $C_{003}, \ldots, C_{033}$, as shown in Figure 2(c). (The channels in this case are denoted as $C_{v\delta}$, where δ is the node address, $v = 1$ selects the high channel, $v = 0$ the low channel.) The most important issue now is to restrict the routing to form an acyclic channel dependency graph.

As the virtual channel concept is applied to a torus, Dally and Seitz first develops a routing function in [10], and then modifies the routing function for hardware implementation in the TRC [2]. To simplify the following presentation and to keep the implementation in mind, we introduce the modified routing function in [2] for 2D torus in the following paragraphs. Consider the routing function for Figure 2(c), the routing function can be stated as: A message begins on high virtual channel. A message remains on high virtual channel until it reaches its relative destination or address 0 in the corresponding dimension. After a message crosses address 0 it is routed on low virtual channel. The low channel at address 0 is not used. Therefore, the obtained channels, shown in Figure 2(d), are totally ordered by their subscripts:

$$C_{133} > C_{123} > C_{113} > C_{103} > C_{033} > C_{023} > C_{013} > C_{003}.$$



(a) Interconnection graph

(b) Channel dependency graph of (a)

(c) Interconnection graph of (a) with virtual channels

(d) Channel dependency graph of (c)

Figure 2. Avoids deadlock by adding channels and restricting routing. (1D torus, the upper most ring of Figure 1 along X dimension.)

Hence, there is no cycle in the channel dependence graph when the virtual channels are added and the routing is restricted in a strictly decreasing order, as shown in Figure 2(d), and then the routing algorithm is deadlock-free.

Both the deterministic routing functions, R_d and R' in [2] are briefly reviewed in the remainder of this section. For a channel of N_δ in some dimension, i.e., $C_{v\delta}$, the determination of an eligible next channel for a message, destined for $N_{\delta'}$, is defined by R_d. Formally the routing function R_d in the d^{th} dimension (for the $2D$ torus, R_1 is in X dimension, where $d = 1$, and R_0 in Y dimension, $d = 0$) is:

$$R_d(C_{v\delta}, N_{\delta'}) = \begin{cases} C_{1(\delta - k^d)}, & \text{if } (v = 1 \wedge (\text{dig}(\delta, d) \neq 0), \\ C_{0(\delta - k^d)}, & \text{if } (v = 0 \vee ((v = 1) \wedge \text{dig}(\delta, d) = 0)), \\ C_0, & \text{if } (\text{dig}(\delta, d) = \text{dig}(\delta', d)). \end{cases} \quad (1)$$

Where C_0 denotes that the message has reached its relative destination in the current dimension. The function dig (δ, d) extracts the d^{th} digit of δ, and k is the radix of the torus. For example, dig $(13, 0) = 3$. The subtraction, $\delta - k^d$ decrements the d^{th} digit of δ modulo k. For example, if $k = 3$, then $12 - 3^0 = 11$ and $12 - 3^1 = 02$. If (dig (δ, d) = dig (δ', d)) is true, then the message reaches its relative destination in the dimension.

It is trivial that Lemma 1 is true.

LEMMA 1. *R_d (i.e., R_1 and R_0) correctly routes a message to its relative destination in each dimension of a 2D torus.*

Virtual channels are added to a 2D torus to avoid the cycles along X and Y dimensions. Each of the X and Y physical channels is separated into two virtual channels. The routing function, R', acts the same as the previous 1D case, except that a message is first routed in the X dimension, then in the Y dimension. That is, R_1 is first applied. As the message reaches

its relative destination in the X dimension, R_0 is applied to route it to the relative destination in the Y dimension. However, to keep the routing in decreasing order of channel subscripts, Dally and Seitz [10] adds another digit, the dimension digit d, to distinguish the channels along the two dimensions. A channel $C_{v\delta}$ at node N_δ is now denoted as $C_{dv\delta}$, $d = 1$ stands for the X dimension, and $d = 0$ for the Y dimension, v selects the virtual channel, and δ is the node address. For example, the high channel of N_{22} in the Y dimension is denoted as C_{0122}, where the first digit 0 means the channel in the Y dimension, the second digit 1 means the high channel, and the third and the fourth digits 22 means the node N_{22}. The routing function for 2D torus can be restated and defined formally as:

$$R'(C_{dv\delta}, N_{\delta'}) = \begin{cases} C_{d1(\delta - k^d)}, & \text{if } (v = 1 \wedge (\mathrm{dig}(\delta, d) \neq 0)), \\ C_{d0(\delta - k^d)}, & \text{if } (v = 0 \vee ((v = 1) \wedge \mathrm{dig}(\delta, d) = 0)), \\ C_{01(\delta - k^d)}, & \text{if } (\mathrm{dig}(\delta, 1) = \mathrm{dig}(\delta', 1)), \\ C_0, & \text{if } ((\mathrm{dig}(\delta, 1) = \mathrm{dig}(\delta', 1)) \wedge (\mathrm{dig}[\delta, 0] = \mathrm{dig}(\delta', 0)). \end{cases} \tag{2}$$

For example, in Figure 1, if a message is sent from the source node N_{22} to the destination node N_{00}, the message first enters the network on channels C_{1122}. Via its next channel, C_{1112}, determined by R', the message reaches its relative destination in the X dimension, node N_{02}. Then, it switches into the Y dimension, travels through channel C_{0102}, C_{0101}, and finally arrives at its destination node N_{00}. It must be pointed out that the subscripts of the channels the message has traveled are in a decreasing order, i.e., $C_{1122} > C_{1112} > C_{0102} > C_{0101}$.

This routing approach is **deterministic** because there is only one path from any source node to any destination node. The path is predetermined by the routing function. Refer to Figure 3, if we consider the case that N_{12} is now sending a message to N_{02}, which occupies C_{1112}. At this time if N_{22} sends a message to N_{00}, the message will be blocked, waiting for C_{1112}. It will have to wait until C_{1112} is released. In addition, if C_{1112} turns out to be faulty, the message sent from N_{22} to N_{00} will wait indefinitely on C_{1112}, while holding C_{1122}. Furthermore, all the paths that pass C_{1112} will be hung. To solve this problem the adaptive and the fault tolerant routing schemes are required to improve the efficiency and the fault tolerance of the network.

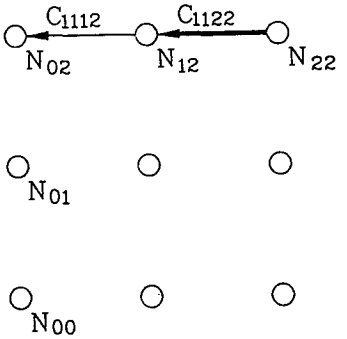


Figure 3. Blocking condition as N_{12} communicates with N_{02} and N_{22} with N_{00}.

3. ADAPTIVE ROUTING SCHEME

The routing functions R_d and R' in [2] reduce the number of possible paths that a message may take from one which may be very large to a single path, thus disallowing any adaption to local traffic conditions. The major theme of the virtual channel approach is to restrict the subscripts of an eligible path into a decreasing order, which makes the channel dependency graph acyclic. Hence the network is free from deadlock. To improve the adaptiveness of the routing function, two techniques, channel switching and dimension switching, are devised to choose the next channel. The channel subscripts of all paths determined by the two techniques are still in a decreasing order, hence the channel dependency graph is acyclic, and the network is deadlock-free.

3.1 Channel Switching

Reviewing the original idea of virtual channels, as shown in Figure 2(d), more carefully we can see that there are other paths within the channel dependency graph. A new technique, called

channel switching can be incorporated to improve the virtual channel approach, which then improves efficiency and fault-tolerance. Suppose some channel receives the header of a message, the next channels determined by the routing function is blocked, and there is a free low channel of the adjacent node lying in the same direction. Then the message can be transferred to the low channel, even though there is no directed edge in the channel dependency graph. However, some restriction must be made to prevent the network from deadlock.

LEMMA 2. *(channel switching) A message waiting for a high channel of an adjacent node can switch to a low channel of the same adjacent node along the same dimension, as long as it would not cross the node 0 (i.e., N_{0y} in X dimension, N_{x0} in Y dimension) in that dimension before reaching its relative destination. This method is called channel switching. Channel switching will not cause deadlock.*

PROOF. Since a message is only allowed to switch from a high channel to a low channel and it will reach its relative destination before reaching the sink, low channel of node 0, of the dependency graph in the dimension, the subscripts it traveled are still partially ordered. Therefore the channel dependency graph is acyclic and channel switching will not cause deadlock. ∎

Channel switching can be incorporated within R_d and R' to enhance their capability. Refer to Figure 3 again, if channel switching is embedded into R', and C_{1012} in the X dimension is free. Then the message can switch to C_{1012}. Hence it is not blocked, as shown in Figure 4.

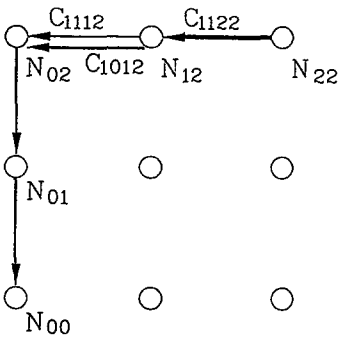


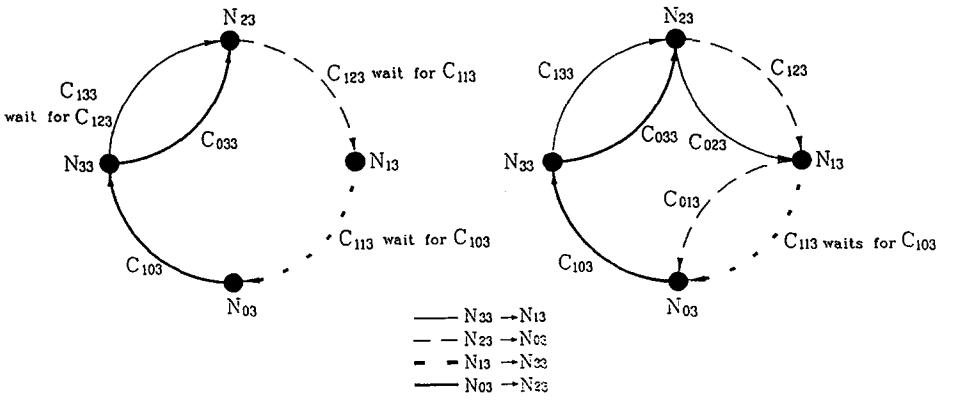
Figure 4. Nonblocking condition as N_{12} communicates with N_{02} and N_{22} with N_{00}.

If channel switching is associated with R_d, the enhanced routing function, S_d (i.e., S_1 in the X dimension, and S_0 in the Y dimension), can be defined as:

$$S_d(C_{v\delta}, N_{\delta'}) \begin{cases} C_{1(\delta - k^d)}, & \text{if } (v = 1 \wedge (\mathrm{dig}(\delta, d) \neq 0)), \\ C_{0(\delta - k^d)}, & \text{if } (v = 0 \vee ((v = 1) \wedge \mathrm{dig}(\delta, d) = 0) \vee \\ & \qquad ((C_{v1(\delta - k^d)} \text{ not available}) \wedge (\Delta\delta\delta' \leq \Delta\delta\phi))), \\ C_0, & \text{if } (\mathrm{dig}(\delta, d) = \mathrm{dig}(\delta', d)) \end{cases} \tag{3}$$

where $\Delta\delta\delta'$ denotes the relative distance between N_δ and $N_{\delta'}$ along dimension d, $\Delta\delta\phi$ denotes the relative distance between N_δ and node 0 in the d^{th} dimension.

Channel switching is useful for local communications. Considering the 1D torus in Figure 2, if each node simultaneously sends a message to its neighbor two hops away, such as N_{03} to N_{23}, N_{33} to N_{13}, N_{23} to N_{03}, and N_{13} to N_{33}. If R_1 is applied for the routing, at the first time C_{103} is occupied by the message sent to N_{23}, and C_{133}, C_{123}, C_{113} are occupied by the messages accordingly. At the second time, only one message is sent to its destination, i.e., the message sent to N_{23}. Other messages are blocked waiting for their next channels to be released. This is shown in Figure 5(a). However, if S_1 is used, three messages can reach their destinations. The message waiting for C_{123} can switch to C_{023}, the message waiting for C_{113} can switch to C_{013}, but the message waiting for C_{103} is blocked, as shown in Figure 5(b). In this case, S_1 improves the efficiency of R_1 by a factor of two.



(a) Deterministic routing (b) Channel switching

Figure 5. Comparison between deterministic routing and channel switching.

3.2 Dimension Switching

Note that because of the unidirectional nature of R_d and R', a message sent by a node to its right (up) neighbor in the $X(Y)$ dimension will travel $(k-1)$ channels. For example, in Figure 1 if N_{11} sends a message to N_{21}, which is topologically adjacent to N_{11}, the message will travel through C_{111}, C_{101} and C_{031} to N_{21} along X dimension.

In general, a message sent to another node in a torus may enter the network in one of the four quadrants, $(+X,+Y)$, $(-X,+Y)$, $(-X,-Y)$, and $(+X,-Y)$. The choice of $+X$ or $-X(+Y$ or $-Y)$ depends on the relative distance of the message in the direction. Take R' as a reference function. It is for the $(-X,-Y)$ quadrant. A message enters the network in the $-X(-Y)$ direction if the relative distance is less than $[k/2]$ along the direction defined by R'. Otherwise it is in the $+X(+Y)$ direction. The message remains in the determined quadrant until it reaches its destination. Four routing functions, one for each corresponding quadrant, are required to implement a fully bi-directional torus. The difference of the routing functions is in the labelling of channel subscripts.

Recall that the label of a channel, determined by R', C_{dvxy} denotes a channel at N_δ, where $\delta = xy$. This is for the $(-X,-Y)$ quadrant. As $(-X,+Y)$ quadrant is considered, a channel is labelled as $C_{dvx\bar{y}}$, where $\bar{y}$ is the $(k-1)$'s complement of the digit y. Consequently, $C_{dv\bar{x}\bar{y}}$ for $(+X,+Y)$ and $C_{dv\bar{x}y}$ for $(+X,-Y)$ quadrant. It is easy to derive the routing functions for the other quadrants by modifying the representation of δ and δ' in (2) according to the above rule. It is obvious that in each quadrant the routing is in decreasing order.

Observe that v and δ suffice to number the subscripts of channels in each dimension, such as the case in Equation (1) and Equation (3). Instead of adding another digit, the dimension digit d as in Equation (2), an adaptive function, which can adaptively choose R_1 and R_0 along X and Y dimension, respectively, can be defined. To prove the adaptive routing function deadlock-free is to prove the routing is in a partial order of channel subscripts.

Since there are 4 deterministic routing functions for the 4 quadrants in a bi-directional torus, there are four adaptive routing functions for each quadrant, too. The subscripts of channels are labelled as that of the deterministic routing function except that the d digit is removed. That is, a channel is denoted as $C_{v\delta}$ along some dimension. Note that δ is different in each quadrant, just as the case of deterministic routing shown above. For example, the high channels of N_{00} of a 4-ary 2-cube in Figure 1 along X direction are C_{133} in the $(+X,+Y)$ quadrant, C_{103} in $(-X,+Y)$, C_{100} in $(-X,-Y)$, and C_{103} in $(-X,+Y)$.

Since the characteristic of the four adaptive routing functions for each quadrant are similar, only the routing function T for the $(-X,-Y)$ quadrant is proved formally. The other three can also be proved in the same manner. Let T be the adaptive routing function which applies R_d (or S_d) in both dimensions of a 2D torus. Let the two-tuple $[\alpha, \beta]$ denotes the subscripts of eligible next channels that can be adaptively chosen by T upon receiving a message sent to $N_{\delta'}$,

where α is determined by R_1 in the X direction and β by R_0 in the Y dimension. If $\alpha \neq 0$ and $\beta \neq 0$, either X or Y direction is eligible. If $\alpha = 0$ and $\beta \neq 0$, the message must be sent along Y dimension. Similarly if $\alpha \neq 0$ and $\beta = 0$, T must choose X dimension. Since $\alpha = 0$ or $\beta = 0$ identifies that the message has reached its relative destination in the dimension, the message has reached its destination as $\alpha = 0$ and $\beta = 0$.

LEMMA 3. *The adaptive routing function T correctly routes a message to its destination.*

PROOF. Since in each dimension $\mathrm{Rd}(S_d)$ routes a message to its relative destination, T correctly routes a message to its destination in a 2D torus. ∎

LEMMA 4. *The adaptive routing function T is deadlock-free.*

PROOF. For a channel at a node N_δ, let $[\alpha, \beta]$ denote the subscripts of eligible next channels, determined by R_d (or S_d), for a message destined to $N_{\delta'}$, where α is in the X dimension and β in the Y dimension. Let $[\alpha', \beta']$ be the subscripts of eligible next channels of the same message if it is sent 1-hop from N_δ in the X dimension. Similarly $[\alpha'', \beta'']$ in the Y dimension. According to the definition of T, we have

(1) If $\alpha = 0$, $\beta \neq 0$ then the message must be sent to the Y dimension. Since R_0 (or S_0) restricts the routing to be strictly decreasing, hence $\alpha'' = \alpha = 0$, and $\beta > \beta''$.
(2) Similarly, if $\beta = 0$, $\alpha \neq 0$ then $\alpha > \alpha'$, and $\beta = \beta' = 0$
(3) If $\alpha \neq 0$, and $\beta \neq 0$ then either X or Y dimension can be chosen. Because $R_d(S_d)$ restricts the routing in decreasing order, it is true that $\alpha > \alpha'$ and $\beta > \beta'$ in the X dimension and $\alpha > \alpha''$ and $\beta > \beta''$ in the Y dimension.

According to (1)–(3), it is true for the same message, $\alpha \geq \alpha'$ and $\beta \geq \beta'$ ($\alpha \geq \alpha''$, $\beta \geq \beta''$). Therefore we can lay down that $[\alpha, \beta] \geq [\alpha', \beta']$ and $[\alpha, \beta] \geq [\alpha'', \beta'']$. The result is the same as the theorem in [16] which states that the cardinal product of two partially ordered sets is also partially ordered. Furthermore each path determined by the routing function T is performed in strictly decreasing order of the two-tuple channel subscripts. Thus the channel dependency graph is acyclic and T is deadlock-free. ∎

For example, consider the example in Figure 3. If R' is adopted, the eligible channel at N_{22} is C_{1122}. If T is adopted, the subscripts of eligible channels are denoted at [122, 122], which implies that the high channels in both dimensions are eligible. If the X direction is chosen, at N_{12} the subscripts are [112, 112]. Since the high channel of N_{12} is occupied by another message, the message can switch to the Y dimension. It flows to N_{11}. The subscripts become [111, 111]. Suppose it switches to X dimension again, it reaches N_{01}. At N_{01}, the subscripts are [0, 101]. Since it reaches its relative destination in X dimension, the message must switch to the Y dimension. Finally it arrives at N_{00}. This path is marked by bold lines in Figure 6. The two-tuples [122, 122], [112, 112], [111, 111], and [0, 101] are in decreasing order. In each dimension, the subscripts are also in strictly decreasing order. For example, $122 > 112 > 111 > 101$. In addition, all 6 eligible paths that may be chosen from N_{22} to N_{00} are derived by solid and bold lines in Figure 6. Take another example, one eligible path from N_{00} to N_{22} which is routed in the $(+X, +Y)$ quadrant, is through N_{10}, N_{20}, N_{21}. The two-tuples are [133, 133], [123, 123], [0, 113], and [0, 112].

In general, suppose a message will travel m hops and n hops in the X and Y dimension, respectively. If dimension switching is enforced, there are $\binom{m+n}{n}$ paths. In addition, if channel switching is embedded into dimension switching, the number of paths is raised at best to $(m+1)(n+1)\binom{m+n}{n}$. Take the example in Figure 6, there are 6 paths from N_{22} to N_{00}, if dimension switching is applied. As channel switching is considered, there are 54 paths.

The 2D torus routing function can be directly applicable to 2D meshes with little modification. In 2D meshes, virtual channels are not necessary because channels are partially ordered in $(+X, +Y)$, $(-X, +Y)$, $(-X, -Y)$, and $(+X, -Y)$ quadrants. One implementation of deterministic routing for 2D meshes is NDF [11]. As adaptive routing is considered for 2D meshes, the degraded routing function T, without channel switching, is of the same characteristic as the virtual network [15], which also requires 4 virtual networks. The alterntive paths from a source to a destination determined by both adaptive routing schemes are the same for 2D meshes.

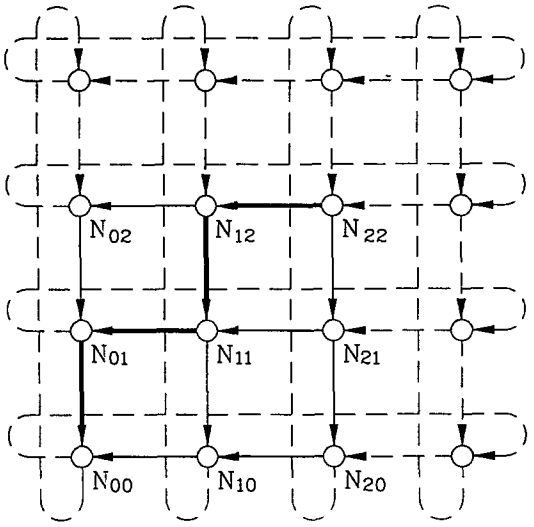
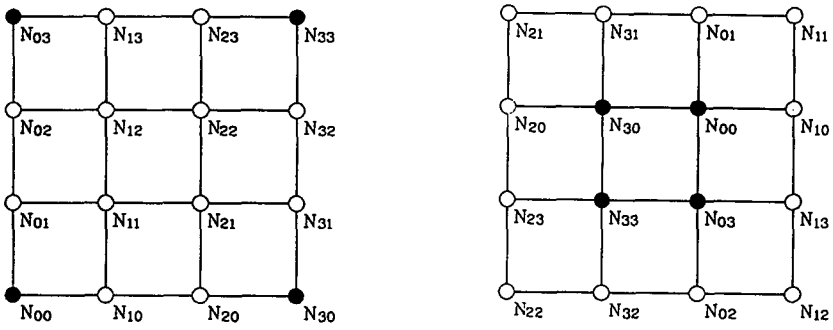


Figure 6. Alternative paths from N_{22} to N_{00} by dimension switching.

Though the hardware complexities of both schemes are comparable under the same assumptions in a 2D torus [15,17], the virtual network approach has the same number of paths from a source to a destination as dimension switching (i.e., $\binom{m+n}{n}$). However, if channel switching is embedded into dimension switching, there is at best $(m+1)(n+1)$ times the number of paths for that of the virtual network. In addition, the virtual network is not so well regular as our scheme. To enforce a 2D torus, the virtual network approach requires to overlap two meshes, totally eight virtual networks, to build a torus. As Figure 7(a) and Figure 7(b) shows, a normal mesh and a complementary mesh are necessary to support the connectivity of the torus. Consider the implementation of the virtual networks, especially the complementary mesh. The connection (wiring) of the virtual networks to the node processors is not regular. Particularly, adding new nodes, such as a new row or column, to a torus would require more overhead than our scheme. The wiring effort to reconfigure the connection between virtual networks and processors and the connection between the normal mesh and complementary meshes is more complex. In other words, the virtual networks are less "scalable," which is an important characteristic of distributed memory multi-computers, than the regular interconnection supported by our scheme.



(a) Normal 4 × 4 mesh

(b) Complementary 4 × 4 mesh

Figure 7. Construction of a 4-ary 2-cube by two 4 × 4 meshes.

In summary, suppose a message will travel m hops in the X dimension and n hops in the Y dimension to reach its destination node, there is only 1 path if R' is applied for the routing. If dimension switching, T, is applied, the number of eligible alternative paths is raised to $\binom{m+n}{n}$. If channel switching is adopted, the number of alternative paths is raised at most to $(m+1)(n+1)\binom{m+n}{n}$.

4. FAULT-TOLERANT ROUTING SCHEME

Though adaptive routing improves the fault tolerance capability of the network by alternative paths, it is still possible that a message cannot reach its destination because of channel failures. Take a trivial example, in Figure 8 a message sent from N_{22} to N_{02} may travel along C_{022}, and C_{012} in the X dimension. If C_{012} fails, the message will be blocked indefinitely unless the routing scheme can be accommodated with the faulty condition.

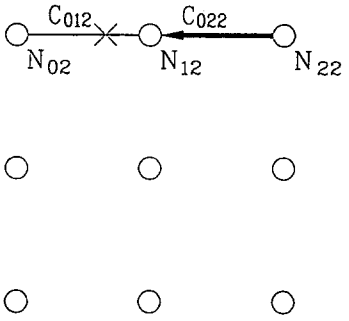


Figure 8. An example of one faulty channel.

Various fault-tolerant schemes have been proposed for hypercubes [18,19]. But there is no existing scheme for the 2D torus as wormhole and deadlock-free routing are considered. In this section we propose a fault-tolerant routing scheme that requires little hardware overhead for implementing the scheme and bring forth no time overhead in normal operation for a 2D torus message passing interconnection network. The basic idea is to route the message to an intermediate destination, called **virtual destination,** in the event that all the possible succeeding channel(s) is (are) faulty. The message is rerouted to the real destination in a second pass by the virtual destination node.

Because of the occurrence of faults, a message may be either a **normal** message or a **rerouted** message. A normal message is a message that would not pass the faulty channel. Rerouted messages are those with the reroute tag set. It is assumed that the header of the rerouted message carries sufficient information to declare itself as a rerouted message, including the reroute tag, the real destination address, and the virtual destination address.

In terms of the single fault model, i.e., there is only one faulty channel at the same time, the rerouting procedure is slightly different depending on the routing schemes. Only the adaptive routing functions are discussed in this section. The details of the single fault rerouting procedures for deterministic routing can refer to our other paper [20]. The major advantage of the fault-tolerant scheme, either deterministic or adaptive, is that it requires low hardware cost because the rerouting steps would only add a few control states to the router. In addition, it causes no overhead in normal operation.

Two problems must be solved in designing the fault-tolerant routing scheme. One is to choose the virtual destination from which the rerouted message would not traverse the faulty channel again. The other is to ensure the path traveled by the rerouted message will not cause deadlock. Three steps are developed to solve the above two problems.

Step 1. For a message M (destined for $N_{\delta'}$) arriving at some channel of N_δ, if it has to be rerouted because there is no eligible fault-free succeeding channels, the channel will either

(1) (wormhole routing) send the message to N_δ, or
(2) (virtual-cut through) accept the message into a buffer.

The node (channel) that accepts the message is called the rerouting component. The rerouting component generally will set the reroute tag, select a virtual destination node, and then send the message, which becomes a rerouted message now, to the virtual destination.

Step 2. For a message M (destined for $N_{\delta'}$) arriving at some channel $C_{v\delta}$ of N_δ, where $\delta = xy$. $x(y)$ denotes the address of the node in the $X(Y)$ dimension in one of the four quadrants. If there is a fault in the succeeding channel and there is no alternative succeeding channels, then

the rerouting component will route the message to the virtual destination $N_\mu(\mu = x'y')$. The value of x' and y' is determined by:

$$x' = (x - 1) \bmod k, \quad y' = (y - 1) \bmod k. \tag{4}$$

Though there are many possible candidates, which guarantee that the rerouted message will not travel the faulty channel again, the virtual destination node chosen by Step 2 is the nearest one to the rerouting component.

As a rerouted message arrives at its virtual destination, Step 3 is performed.

Step 3. For a rerouted message M (virtual destination N_μ, real destination $N_{\delta'}$) arriving at some channel of N_μ, then the channel will either

(1) (wormhole routing) send the message to N_μ, which in turn will reset the reroute tag, strip off the virtual destination address, and resend the message to the real destination $N_{\delta'}$, or

(2) (virtual cut-through) receive the message into a buffer, reset the rerouting tag, strip off virtual destination address, and resend the message to $N_{\delta'}$.

As the rerouting procedures are invoked, the message M is removed from the network three times. One is by the rerouting component, another is the virtual destination, and the other is the real destination. Since all the channels occupied by M are released at the rerouting component and the virtual destination, therefore cyclic waiting condition is avoided. In addition, all the paths that M travels, i.e., from the source to the rerouting component, from the rerouting component to the virtual destination, and from the virtual destination to real destination, are determined by the deadlock-free routing algorithm. Therefore the fault-tolerant routing scheme for the single fault model is deadlock-free.

If channel switching is considered, there are at least two channels available to the adjacent node in either X or Y dimension, except the sink, the low channel of node at position 0 of the dimension. Therefore a virtual destination node is not necessary, the message can be rerouted directly to the real destination. Step 2 can be enhanced to Step 2'.

Step 2'. For a message M (destined for $N_{\delta'}$) arriving at some channel of node N_δ, in the $X(Y)$ dimension, if there is a fault in the succeeding channel and there is no alternative succeeding channels. If the current node is not at the position 0 in the same dimension of the faulty channel, then sends the message directly to $N_{\delta'}$. The rerouting component will route the message to the real destination N_δ. Otherwise, route to the virtual destination node as Equation (4) defined.

Refer to the fault condition in Figure 8 again, as Step 2 is performed, the message will be resent to N_{01}. As Step 2' is applied, the message will be resent directly to its destination. Both cases are shown in Figure 9.

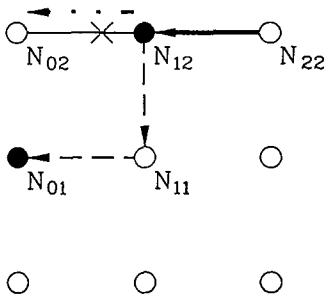


Figure 9. An example of fault tolerant routing.

Some mechanism should be provided to issue a time-out interrupt if a message does not arrive to its destination within a reasonable time. The time-out mechanism is useful for both single fault and multiple fault model. For multiple fault model, all steps can be applied. But a message may be rerouted more than once because a rerouted message may encounter a faulty component which will make it to be rerouted. However, some combination of faults cause the message to be rerouted infinitely in the network, such as sending a message to an unreachable node, i.e., a node whose channels are all faulty. Alternatively, once the rerouting rules are invoked, a signal of fault is issued, which causes a repair to be scheduled. The system meanwhile operates in the degraded mode until the fault is actually repaired.

5. CONCLUSION

This paper presents schemes for routing messages in a 2D torus interconnection network for message passing concurrent computers. The adaptive and the fault-tolerant routing schemes can be applied independently or be combined to enforce a more powerful routing scheme. Though the algorithms developed are limited to 2D torus, they can be used to construct higher dimensional torus topology, since the 2D torus can be used as a building block for higher dimensional ones [6].

Though wormhole and virtual cut-through routing supports high performance communication for distributed memory multicomputers, there is a variety of consideration of their implementation. Based on wormhole or virtual cut-through routing, our approach contributes most to enhance the high overall efficiency of the interconnection network by the fully utilization of the inherently redundant paths in the 2D torus topology, while retaining the deadlock-free property of each path. Suppose a message will travel m hops in the X dimension and n hops in the Y dimension to reach its destination. If deterministic routing is applied, there is only one eligible path. As our dimension switching is enforced, the number of eligible paths is raised to $\binom{m+n}{n}$, and if our channel switching is associated with dimension switching, the number is at most $(m+1)(n+1)\binom{m+n}{n}$. However, the policy for choosing a channel among all possible succeeding ones for a message at a certain channel is not discussed in the adaptive routing scheme. Two policies may be considered. One is to choose the first available channel. The other may take the Horizon approach, which tries to maximize the message's degree of freedom, i.e., to choose the dimension which is the farthest from its relative destination.

Another contribution is the fault tolerant routing scheme, which is deadlock-free, requires low hardware cost in implementation, and causes no overhead in normal operation. Faults in the network bring forth extra latency and blocking, but the connectivity is not affected for the considered faults. Thus the approach allows the system to be kept operational until a repair can be scheduled. It suits for systems where graceful degradation is preferred and the tolerance of a fault is essential.

REFERENCES

1. J.L. Gustafson, G.R. Montry and R.E. Benner, Development of parallel methods for a 1024-processor hypercube, *SIAM J. Sct. Stat. Comput.* **9** (4), 609–638 (1988).
2. W.J. Dally and C.L. Seitz, The torus routing chip, *Distributed Computing* **1**, 187–196 (1986).
3. F. Pittelli, and D. Smitley, Analysis of a 3D toroidal network for a shared memory architecture, In *Proc. Supercomputing'88*, pp. 42–47, (1988).
4. S. Borkar, R. Cohn and G. Cox *et al.*, iWARP: An integrated solution to highspeed parallel computing, In *Proc. Supercomputing '88*, pp. 330–339, (1988).
5. P.M.B. Vitányi, Locality, communication, and interconnect length in multi-computers, *SIAM J. Comput.* **17**(4), 659–672 (1988).
6. W.J. Dally, A VLSI architecture for concurrent data structure, Ph.D. Thesis, Dept. Comput. Sci., California Instit. Tech., (1986).
7. R. Kermain and L. Kleinrock, Virtual cut-through: A new computer communication switching technique, *Computer Networks* **3**, 267–286 (1979).
8. S. Nugent, The iPSC/2 Direct Connect Communications technology, In *Proc. 3rd Conf. on Hypercube Concurrent Computers and Applications*, (1988).
9. Y. Tamir and G.L. Frazier, High-performance multi-queue buffers for VLSI communication switches, Presented at the *15th Annual Conf. on Comp. Arch.*, pp. 343–354, (1988).
10. W.J. Dally and C.L. Seitz, Deadlock-free message routing in multi-processor interconnection networks, *IEEE Trans. on Comp.* **36** (3), 547–553 (1987).
11. W.J. Dally and P. Song, Design of a self-timed VLSI multi-computer communication controller, In *Proc. IEEE Int'l Conf. on Computer Design*, pp. 230–234, (1987).
12. W.D. Hillis, *The Connection Machine*, MIT Press, (1985).
13. H.F. Jordan, Performance measurements on HEP—a pipelined MIMD computer, In *Proc. 14th Symp. on Comp. Arch.*, pp. 207–212, (1983).
14. E. Chow, H. Madan, J. Peterson, D. Grunwald and D. Reed, Hyperswitch network for the hypercube computer, In *Proc. of the 15th Int'l Symp. on Comp. Arch.*, pp. 90–99, (1988).
15. C.R. Jesshope, P.R. Miller and J.T. Yantchev, High performance communications in processor networks, *IEEE 16th Annual Int'l Symp. on Comp. Arch.*, pp. 150–157, (1989).
16. T. Donnellan, *Lattice Theory*, Pergamon Press Ltd., pp. 41–45, (1968).
17. T.-W. Hou, W.G. Lin, M.R. Tsai and L.M. Tseng, An adaptive deadlock-free router for torus interconnected multi-computers, In *Proc. National Computer Symposium, Taipei, Taiwan*, pp. 714–722, (1989).

18. Y. Lan, Interprocessor communication in distributed memory multi-processors, Ph. D. Thesis, Dept. Comput. Sci., Michigan State Univ., (1988).
19. H.P. Katseff, Incomplete hypecubes, *IEEE Trans. on Comp.* **3** (5), 604–608 (1988).
20. M.R. Tsai, T.-W. Hou, L.M. Tseng and C.K. Shieh, Incomplete low dimensional torus, In *Proc. National Computer Symposium, Taipei, Taiwan*, pp. 816–825, (1989).