Adaptive routing on the Recursive Diagonal Torus

Akira Funahashi Hideharu Amano

Keio University

3-14-1 Hiyoshi Yokohama 223, JAPAN

{funa, hunga}@aa.cs.keio.ac.jp

Abstract

Recursive Diagonal Torus, or RDT consisting of recursively structured tori is an interconnection network for massively parallel computers. By adding remote links to the diagonal directions of the torus network recursively, the diameter can be reduced within $log_2 N$ with smaller number of links than that of hypercube.

For an interconnection network for massively parallel computers, a routing algorithm which can bypass a faulty or congested node are essential. Although the conventional vector routing is a simple and near-optimal method, it can only use a deterministic path. In this paper, adaptive routing algorithms on RDT are proposed and discussed. The first algorithm is based on Duato's necessary and sufficient condition. With this method virtual channels are effectively used while paths with redundant routing steps are prohibited. Another algorithm based on the turn model is proposed. By prohibiting certain turns on RDT, it permits paths with additional hops. Both algorithms are proved to be deadlock free.

Interconnection Network, Adaptive Routing, Deadlock Avoidance, RDT

1 Introduction

Communication network is one of the critical components of a highly parallel multicomputer. Recently, multicomputers providing more than a thousand computation nodes are commercially available, and efforts have been exerted to implement Massively Parallel Computers (MPCs) with tens of thousands nodes. In these systems, the connection topology often dominates the system performance.

Instead of hypercube used in first-generation multicomputers, most recent machines take the 2-D or 3-D mesh (torus) network[1][2][3]. Although the diameter of a mesh network is large ($O(\sqrt{M})$ or $O(\sqrt[3]{M})$ for M nodes), it only requires four or six links per node unlike the hypercube which requires $log_2 M$ links per node. Architectural supports for fine grain processing [2] and the wormhole routing [4] enabled the use of networks with large diameter. Moreover, mesh networks are suitable for most scientific calculations including flow dynamics, QCD, and structural analysis.

However, in an MPC with more than ten thousands nodes, the large diameter of the mesh network is intolerable. A lot of connection topologies (De Bruijn[5], fat tree[6], Star graph[7] and others) have been proposed for such MPCs. Although these networks support a small diameter with a small degree, emulation of the mesh network is difficult. On the current machines with mesh structure, parallel computation algorithms and message handling algorithms have been studied and refined. To make the best use of them, a network including the mesh structure is advantageous even for future MPCs.

We proposed a novel class of networks called Recursive Diagonal Torus (RDT) [8], which consists of recursively structured mesh (torus) connection. It supports a smaller diameter and degree than that of the hypercube if the number of nodes is 1000-10000. Through the computer simulation, the bandwidth and latency are greatly improved compared with 2-D/3-D tori [8].

A simple routing algorithm called the vector routing was proposed for RDT[8], and a technique based on the e-cube routing[9] is applied for avoiding the deadlock[10]. The router chip providing the vector routing algorithm with multicasting was implemented for a massively parallel machine JUMP-1[11].

However, with these deterministic routings, congested or faulty nodes cannot be bypassed although such functions are essential in massively parallel machines. In this paper, deadlock-free adaptive routing algorithms on RDT are proposed and discussed in order to cope with this problem. In Section 2, the structure of RDT and the vector routing algorithm are introduced. An adaptive routing using minimal paths based on Duato's method is proposed in Section 3. More flexible routing algorithm based on the turn model is also proposed in Section 4.

2 Interconnection Network: RDT

Recursive Diagonal Torus (RDT) is a novel class of networks which consists of recursively structured mesh (torus) connections of meshes with different sizes in the diagonal directions. Here, interconnection structure of RDT is introduced.

2.1 Definitions of RDT

First, a two-dimensional square mesh (torus) is defined as the basis of RDT.

Definition 1 : **Base torus**

The base torus is a two-dimensional square array of nodes each of which is numbered with a two-dimensional number as follows:

$$\begin{array}{ccccc}
(0,0) & (1,0) & (2,0) & \cdots & (N{-}1,0) \\
(0,1) & (1,1) & (2,1) & \cdots & (N{-}1,1) \\
(0,2) & (1,2) & (2,2) & \cdots & (N{-}1,2) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
(0,N{-}1) & (1,N{-}1) & (2,N{-}1) & \cdots & (N{-}1,N{-}1)
\end{array}$$

where $N = n^k$. The n and k are natural numbers. The torus network is formed with four links between node (x, y) and neighboring four nodes:

$$(\mathrm{mod}(x \pm 1, N),\ y) \text{ and } (x,\ \mathrm{mod}(y \pm 1, N))$$

*This base torus is also called the rank-*0 *torus.* □

In order to reduce the diameter, the best way for the torus network is to provide bypass links in the diagonal direction. Assume that four links are added between a node (x, y) and nodes $(x \pm n, y \pm n)$. Then, the additional links form a new torus-like network. The direction of the new torus-like network is at an angle of 45 degrees to the original torus, and the grid size is $\sqrt{2}n$ times of the original torus. Here, this torus-like network is called the rank-1 torus. On the rank-1 torus, another torus-like network (rank-2 torus) can be made by providing four links in the same manner. Thus the rank-$(i{+}1)$ torus can be formatted on the rank-i torus in the same way. Figure 1 shows rank-1 and rank-2 tori when n is set to be 2. Our new network called RDT consists of such recursively formed tori.

Each odd rank torus-like network is a 2:1 rectangle one which provides spiral circular links as shown in Figure 2, while each even rank network is a common square nearest neighbor torus. Here, both types of networks, are called the "torus" or "tori" uniformly.

Definition 2 : **Upper rank torus**

Assume that the rank-i torus satisfies the following condition:

$$N_{x(r+1)} \times N_{y(r+1)} \geq 2$$

here,

$$N_{x(r+1)} = \frac{N_{yr}}{\mathrm{gcd}(N_{xr}, n)}$$

$$N_{y(r+1)} = \frac{N_{xr}}{2\mathrm{gcd}(N_{yr}, n)},$$

and, $N_{xr}, N_{yr}, N_{x(r+1)}$ and $N_{y(r+1)}$ are the sizes of the rank-r and rank-(r+1) in x and y axes.

Links between node (x, y) and nodes (x', y'), (x'', y'') on the rank-i torus form the rank-$(i{+}1)$ torus where

$$x' = (x+n) - (N_{xr} - N_{yr}) \left\lfloor \frac{x+y}{N_{xr} - 2n} \right\rfloor - N_{xr} \left\lfloor \frac{x+n}{N_{xr}} \right\rfloor \left\lfloor \frac{N_{yr}}{N_{xr}} \right\rfloor$$

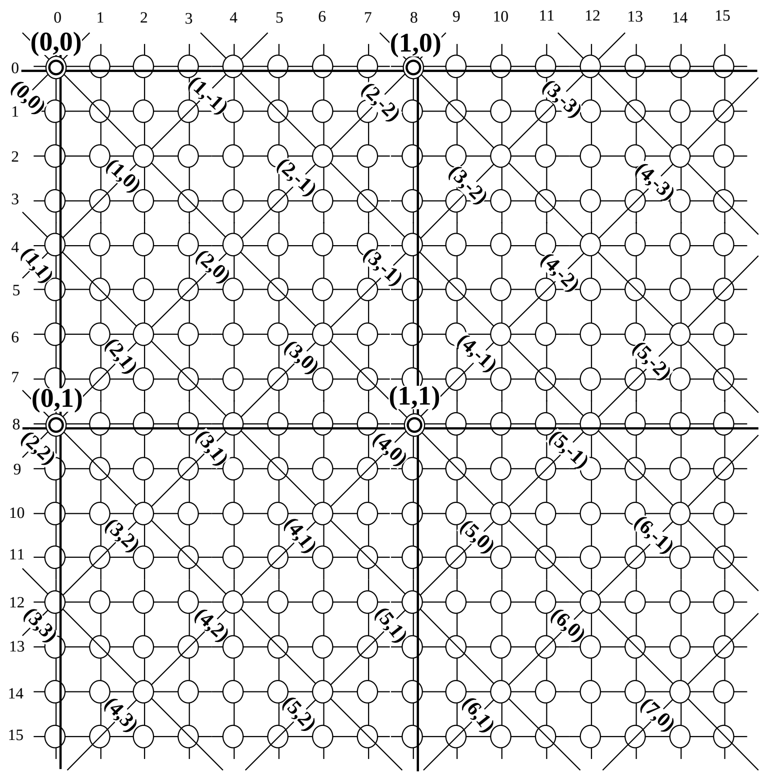

Figure 1: Upper rank tori

$$y' = (y+n) - (N_{xr} - N_{yr}) \left\lfloor \frac{x+y}{N_{xr} - 2n} \right\rfloor - N_{yr} \left\lfloor \frac{y+n}{N_{yr}} \right\rfloor \left\lfloor \frac{N_{yr}}{N_{xr}} \right\rfloor$$

$$x'' = (x+n) - (N_{xr} - N_{yr}) \left\lfloor \frac{x-y}{N_{xr} - 2n} \right\rfloor - N_{xr} \left\lfloor \frac{x+n}{N_{xr}} \right\rfloor \left\lfloor \frac{N_{yr}}{N_{xr}} \right\rfloor$$

$$y'' = (y-n) + (N_{xr} - N_{yr}) \left\lfloor \frac{x-y}{N_{xr} - 2n} \right\rfloor - N_{yr} \left\lfloor \frac{y-n}{N_{yr}} \right\rfloor \left\lfloor \frac{N_{yr}}{N_{xr}} \right\rfloor$$

Nodes on the rank-$(i+1)$ *torus can be identified by another two-dimensional number* (x_{i+1}, y_{i+1}). *The number is given in the following manner:*

1. *Let node* (0,0) *of the rank-i torus be* (0,0) *of the rank-*$(i+1)$.

2. *For an even rank torus, the axis is set as the same direction of the base torus. For an odd rank, the direction from* (x, y) *to* (x', y') *is set to be an x-axis of the rank-*$(i+1)$ *torus.*

3. *Give a two-dimensional number to each node on the rank-*$(i+1)$ *torus according to Definition 1.*

The rank-$(i+1)$ *torus is called as upper rank torus based on the rank-i. n is called the* **cardinal number**.
□

Note that there are several independent upper rank tori formed on a torus (This problem will be discussed in Section 2.2). By forming upper rank tori recursively on the base torus (Definition 1) according to Definition 2, Recursive Diagonal Torus (RDT) is defined.

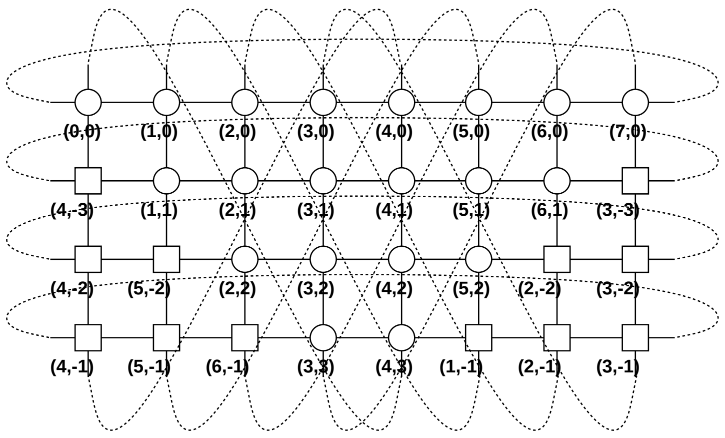

Figure 2: Structure of the rank-1 torus

Definition 3 : Perfect RDT
*Let upper rank tori form recursively according to Definition 2 on the base torus defined in Definition 1 as many as possible. A network in which every node has links to form all possible upper rank tori is called the perfect RDT (**PRDT**(n, R)) where n is the cardinal number and R is the maximum rank.* □

Although PRDT is unrealistic because of its large degree ($4(R + 1)$), it is important as the basis for establishing message routing algorithms of RDT theoretically.

Definition 4 : RDT
*Recursive Diagonal Torus **RDT**(n, R, m) is a class of networks in which each node has links to form the base (rank-0) torus and m upper tori (the maximum rank is R) with the cardinal number n.* □

According to this definition, the degree of RDT(n, R, m) can be shown as: $4(m + 1)$.

2.2 Torus assignment

Various structures of RDT can be formed by changing n and m. Although large n is sometimes advantageous in a large system, the cardinal number n is set to be 2 so as to enable easy implementation of algorithms based on binary tree or cube. Since an upper torus requires four links, RDT with large m requires too much hardware. Here, a system with ten thousand nodes (for example, array of 128 $\times$ 128 nodes or 256 $\times$ 256 nodes) is assumed, and m is set to be 1 (degree = 8). For this number of nodes, the maximum rank of upper tori is 4. Thus, RDT(2,4,1) is mainly treated here.

In RDT(2,4,1), one of upper rank tori is assigned to each node. Thus, the structure of RDT(2,4,1) also varies with the rank of tori which is assigned to each node. This assignment is called the *torus assignment.* Various torus assignment strategies can be selected considering the traffic of the network. If the local traffic

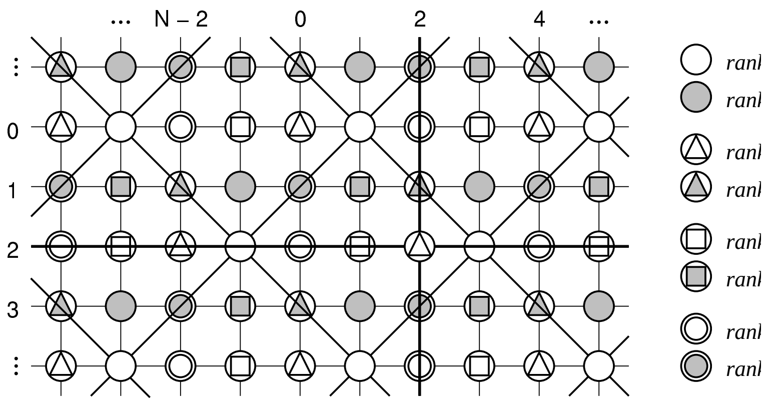

Figure 3: Torus assignment for the RDT(2,4,1)/α

is large, the number of nodes which have low ranks should be increased. Figure 3 shows an assignment adopted in a massively parallel machine JUMP-1[12].

In this assignment which is called RDT(2,4,1)/α a node has eight links, four for the base (rank-0) torus and four for rank (1-4) torus (most links for upper rank tori are omitted in Figure 3). With the cardinal number $n = 2$, 8 independent rank-1 tori can be formed on the base torus. Two tori are used directly as the rank-1 torus, and other rank-2 tori are formed on two rank-1. Similarly, two rank-1 tori are used in forming rank-3 tori, and two rank-2 tori for rank-4 tori.

2.3 The vector routing

The vector routing is an assignment independent of routing algorithm which represents the route of a message with a combination of unit vectors each of which corresponds to each rank of tori. Although this routing algorithm is proposed for PRDT, it can be applied to any assignment with a small modification[10].

On the torus structure, a vector from a source node to the destination node is represented with a vector $\vec{A} = a\vec{x_0} + b\vec{y_0}$ where $\vec{x_0}$ and $\vec{y_0}$ are unit vectors of the base (rank-0) torus. The goal of the routing algorithm is to represent the vector $\vec{A}$ with a combination of vectors each of which corresponds to a unit vector of each rank of torus (Figure 4(a)).

First, the direction of the unit vector corresponding to each rank torus must be defined. Here, the direction of the unit vector for each rank torus is changed clockwise at an angle of 45 degrees as shown in Figure 4(b). That is, the unit vectors of rank-(i+1) torus $\vec{x}_{i+1}, \vec{y}_{i+1}$ can be represented with the unit vectors of rank-i $\vec{x_i}, \vec{y_i}$ as follows:

$$\vec{x}_{i+1} = n\vec{x_i} + n\vec{y_i} \tag{1}$$

$$\vec{y}_{i+1} = -n\vec{x_i} + n\vec{y_i} \tag{2}$$

First, the target vector $a\vec{x_0} + b\vec{y_0}$ is represented with a combination of $\vec{x_1}$,$\vec{y_1}$,$\vec{x_0}$ and $\vec{y_0}$ as follows:

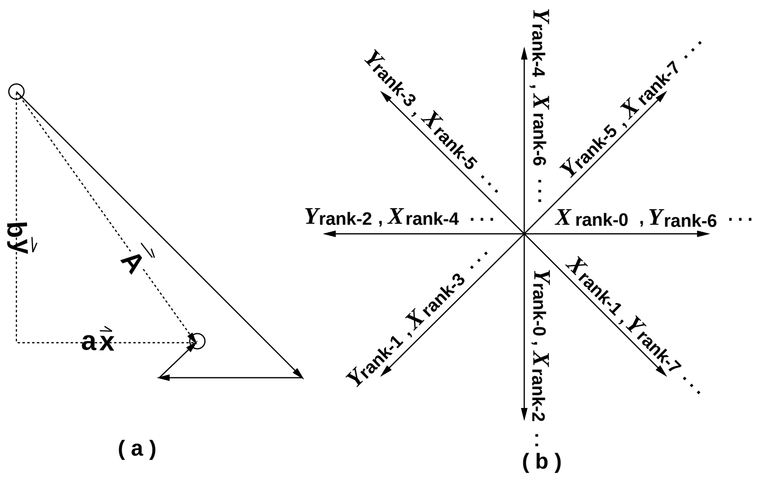

Figure 4: Directions or the coordinate axes

$$a\vec{x_0} + b\vec{y_0} = g\vec{x_1} + f\vec{y_1} + j\vec{x_0} + k\vec{y_0} \qquad (3)$$

Here, we select maximum g and f in order to use the upper torus as possible. From equations (1) and (2), maximum integers for g and f are represented as follows:

$$g = \frac{a+b}{2n}, f = -\frac{a-b}{2n}$$

In order to minimize j and k corresponding to the remaining unit vectors of the rank-0 torus (thus, the required message transfers using the rank-0 torus), the integer divisor used here is rounded to the nearest whole number (If the remainder is greater than n, increment the divisor).

Thus, j and k are represented with g and f:

$$a\vec{x_0} + b\vec{y_0} = g(n\vec{x_0} + n\vec{y_0}) + f(-n\vec{x_0} + n\vec{y_0}) + j\vec{x_0} + k\vec{y_0}$$

$$a = ng - nf + j, b = ng + nf + k$$

$$j = a - ng + nf, k = b - ng - nf$$

.

Then, $g\vec{x_1} + f\vec{y_1}$ are represented with a combination of vector $\vec{x_2}$, $\vec{y_2}$, $\vec{x_1}$, and $\vec{y_1}$ in the same manner. By iterating this process to the maximum rank, vectors for message routing are obtained.

This algorithm is specified with a simple C program fragment as follows:

Algorithm A: The simple vector routing

```
for (rank=0; rank<MAX_RANK; rank++) {
    g=div_(2n)(a+b); f=div_(2n)(-(a-b));
    vector[rank].x= a-(ng-nf);
    vector[rank].y= b-(ng+nf);
    a=g; b=f;
}
vector[MAX_RANK].x=g; vector[MAX_RANK].y=f;
```

where div_(2n) means division with $2n$ rounding to the nearest integer. The routing vectors for each rank are obtained in the array $vector[rank]$.

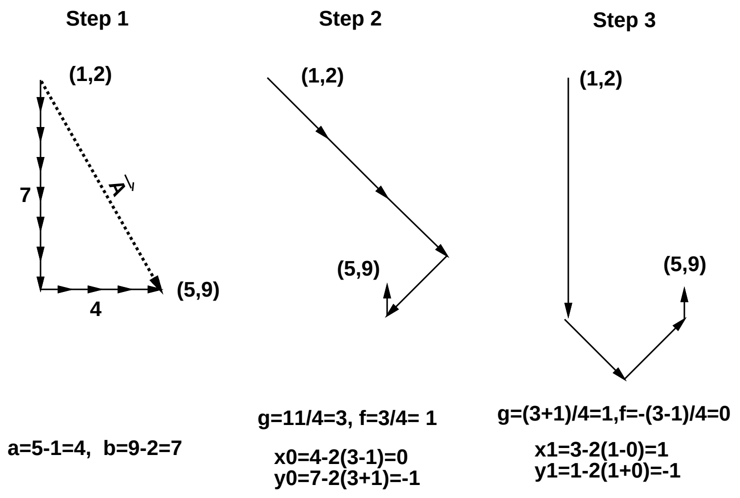

Figure 5: An example of the vector conversion

Figure 5 shows an example of a vector from (1,2) to (5,9) converted into a combination of unit vectors of rank-0, rank-1, and rank-2.

The vector routing is useful for bypassing a faulty or congested node, as alternative routes can be easily obtained by changing the order of vectors. Since this routing is deterministic, the bypassing path must be calculated with the knowledge on the location of faulty or congested node. However, it is difficult to know the location of congested nodes for other distant nodes.

Adaptive routing is a technique to select the route of packet dynamically. When a packet encounters a faulty or congested node, the packet can select another bypassing route. However, we must not forget that adaptive routing have a possibility of deadlock. There are a lot of researches on deadlock free adaptive routing techniques[4]. These techniques are classified into two methods: using only minimal paths, and using alternative paths with additional routing steps. The former method does not require extra routings while the latter can use alternative routes more flexibly. Here, both methods on RDT are discussed.

First, deadlock free adaptive routings with minimal routes are proposed for RDT based on Duato's protocol. Then, another algorithm which permits redundant routing steps is proposed based on the turn model.

3 Adaptive routing with minimal paths

3.1 Duato's protocol in the k-ary n-cube

Duato states a general theorem defining a criterion for deadlock freedom and then uses the theorem to propose a fully adaptive, profitable, progressive protocol[13], called Duato's protocol (DP). The theorem states that by separating virtual channels on a link into restricted and unrestricted partitions, a fully adaptive routing can be performed and yet be deadlock-free. This is not restricted to a particular topology or routing algorithm. Cyclic dependencies between channels are allowed, provided that there exists a connected channel subset free of cyclic dependencies.

Simple description of Duato's protocol is as follows.

a. Provide that every packet can always find a path toward its destination whose channels are not involved in cyclic dependencies(escape path).

b. Guarantee that every packet can send to any destination node using escape path and the other path which cyclic dependency is broken by escape path.

By selecting these two routes a. and b. adaptively, it can prevent deadlocks.

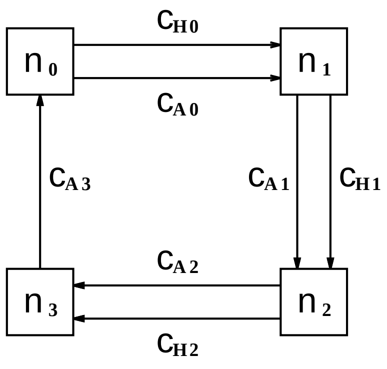

Figure 6: Example network for deadlock-free adaptive routing.

With the following steps, Duato achieved a deadlock-free adaptive routing algorithms for k-ary n-cube[14].

Theorem 1 *The adaptive routing algorithm using Duato's protocol for k-ary n-cube is deadlock-free.* □

Proof

1. First, consider a unidirectional ring(Figure6) with four nodes. C_{Ai} channels can be used to forward packets to all the destinations, but C_{Hi} channels can only be used if the destination is higher than the current node. This routing algorithm is deadlock-free, as shown in [13].

2. Second, let this routing algorithm extend for bidirectional rings. When a packet starts crossing channels in one direction following a minimal path, it cannot turn and continue in the opposite direction. Thus,

the use of bidirectional channels does not introduce any additional channel dependency and the routing algorithm is deadlock-free.

3. Third, for k-ary n-cubes, channels can only be used crossing dimensions in ascending order. Inside each dimension, the algorithm for bidirectional rings is used. Like the e-cube routing[9], dimensions are crossed only in ascending order. This means that there is no additional cyclic dependency and this routing algorithm is also deadlock-free.

   By these steps, escape path C_1 is provided.

4. Then, add a new virtual channel C_F(*Fully adaptive*) which is used for fully adaptive routing, crossing dimensions in any order following a minimal path. There are two methods for providing fully adaptive virtual channel C_F:

□

In this algorithm, only minimal paths can be used in order to satisfy the second step.

3.2 Applying Duato's protocol on PRDT

Here, we apply this routing algorithm for PRDT.

Definition 5 : Duato's protocol on PRDT

1. *Provide an escape path C_1 on a torus of PRDT as well as the case for the k-ary n-cube.*

2. *Next, the order of rank usage is restricted. Let X_i and Y_i be channel of each dimension in the rank i torus. Use the channel in the X first and descending order of the rank. That is, for PRDT(2,4), the channel is used in the following order*
   $X_3 \rightarrow Y_3 \rightarrow X_2 \rightarrow Y_2 \rightarrow X_1 \rightarrow Y_1$
   We refer this escape path C_1'.

3. *Add a new virtual channel C_F(Fully adaptive) which is used for the fully adaptive routing, following a minimal path.*

Algorithm: D-A *Provide the virtual channel C_F directly for the escape channel C_1'. In C_F, each direction of $+X$ and $+Y$ in odd rank and even rank must be the same direction. In the vector routing, the unit vector for each rank torus is changed clockwise at an angle of 45 degree as represented in function(1) and function(2), the unit vector for odd rank torus must be same direction with the unit vector for rank 0 torus $(\vec{x}_0, \vec{y}_0)$ and the unit vector for even rank torus must be the same with the one for rank 1 torus$(\vec{x}_1, \vec{y}_1)$.*

Algorithm: D-B *Provide the virtual channel C_{Fn} not for C_1' but for C_1 in each rank. C_{Fn} channels can cross dimensions in any order following a minimal path, but must cross ranks in descending order.*

□

Figure 7 illustrates the fully adaptive virtual channel C_F in Algorithm D-A. Since C_F is directly assigned to the escape path C'_1, the C_F itself must be a minimal routing. This means that a packet must not use the opposite direction which used in the past.

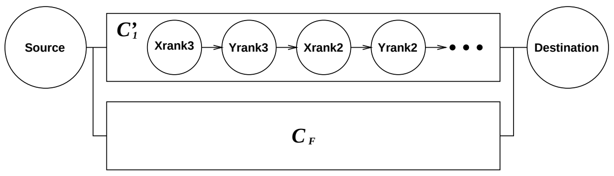

Figure 7: Channels using Algorithm D-A

On the other hand, the fully adaptive path is assigned to the escape path C_1 of each rank in Algorithm D-B(Figure8). Therefore, there is no restriction for using unit vector, while the order of using ranks is restricted.

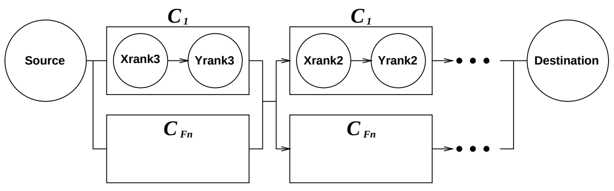

Figure 8: Channels using algorithm D-B

Figure9 illustrates the possible path and impossible path for algorithm D-A and D-B. The path (b) which uses rank 2 before rank 3 is allowed in the algorithm D-A while it is prohibited in the algorithm D-B, since the rank is not be used in the descending order. On the contrary, path (c) in which the unit vectors of rank 1 and rank 3 are directed opposite to each other is prohibited in the algorithm D-A but allowed in the algorithm D-B.

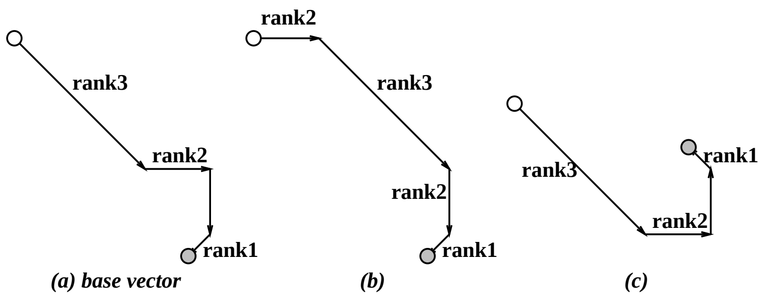

Figure 9: Examples of vectors in algorithm D-A and D-B

Theorem 2 *Algorithm D-A is deadlock-free.* □

Proof Since the order of the rank is the same as that of the e-cube routing[9], the escape path $C'1$ is deadlock free. In C_F, the opposite direction which used in the past is prohibited, and so C_F is a minimal path. From theorem 1, Algorithm D-A is deadlock-free. □

Theorem 3 *Algorithm D-B is deadlock-free.* □

Proof C_1 is the same escape path used in theorem 1, and is deadlock free. C_{Fn} is a minimal path in each torus. From theorem 1, Algorithm D-B is deadlock-free in each rank of torus. Since the order of used rank is the same as the e-cube routing, C_1 nor C_{Fn} in any rank does not cause a cycle each other. Therefore, Algorithm D-B is deadlock-free.□

4 Adaptive routing based on the turn model

Although Duato's protocol is powerful approach for bypassing the congestion, only minimal paths can be used. For selecting paths with additional steps, another adaptive routing based on the turn model[15] is proposed here.

4.1 Turn model for Two-Dimensional Meshes

Deadlock in the wormhole routing is caused by message packets waiting for each other in a cycle. The turn model proposed by Glass is a method which prevents deadlock by prohibiting certain turns.

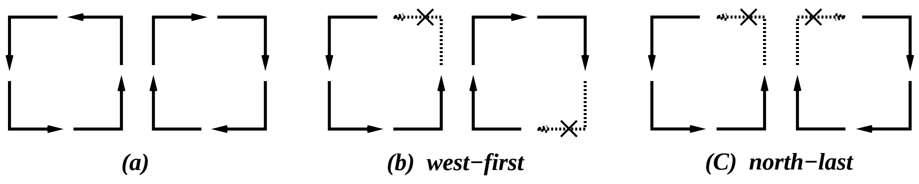

Figure 10: The turn model for two-dimensional meshes.

For two-dimensional meshes, Figure10(a) shows the possible turns and simple cycles. Deadlock can be prevented by prohibiting only one turn from each cycle, as shown in Figure10(b),(c). These routing algorithms are called the west-first routing algorithm and north-last routing algorithm, respectively. Although this model is for a simple mesh network without cyclic links, it is easily used in the torus by introducing extra channels like the e-cube routing.

4.2 The turn model for RDT

Here, we extend the turn model for two-dimensional meshes of the RDT. The possible turns in RDT are expressed in Figure11. As shown in Figure11, there are eight different directions in the RDT, so there exists sixteen 45-degree turns, sixteen 90-degree turns and sixteen 135-degree turns.

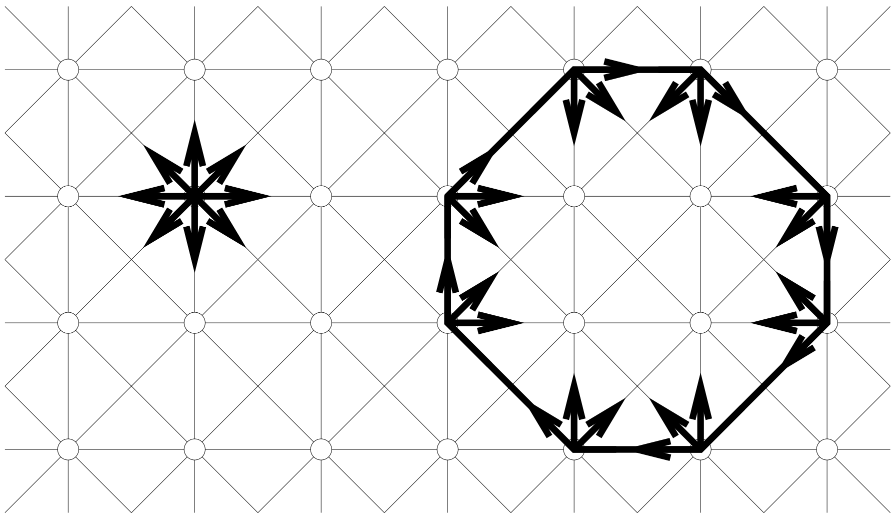
Figure 11: The possible turns and simple cycles in RDT.

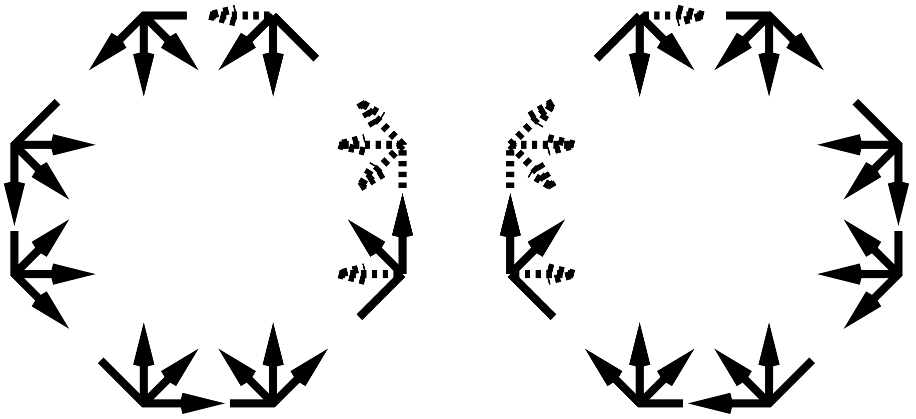
Figure 12: The first step to the north-last routing on RDT.

Here, like the north-last routing algorithm for two dimensional mesh, the right top turns and left top turns of cycles are prohibited as shown in Figure12.

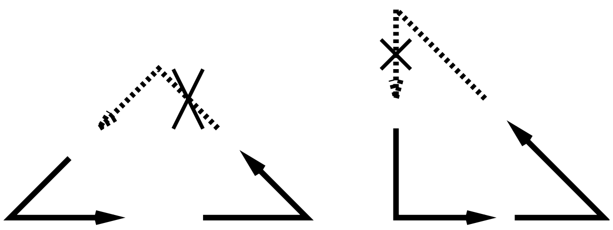

Figure 13: Particular types of cycles.

However, these restrictions are not sufficient for RDT. Cycles without left top turns or right top turns are still possible as shown Figure13. In order to break such triangle cycles, dotted turns shown in Figure13 must be prohibited. As a result, the following turns are prohibited in RDT.

Definition 6 : **North-last routing for PRDT**

North-last routing for PRDT is a routing in which fourteen turns shown in Figure14 are prohibited. A packet transfer through cyclic links is also prohibited. □

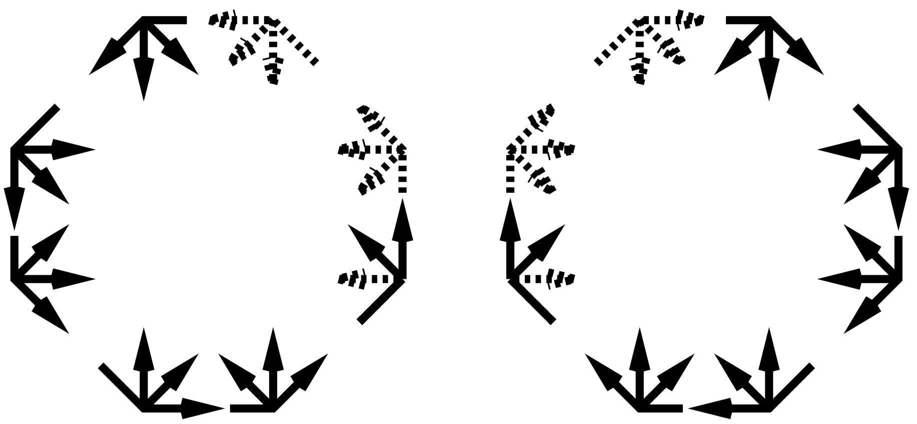

Figure 14: The turn model for RDT (2).

As well as the turn model for two dimensional torus, cyclic links can be used by introducing an extra channel for the e-cube routing. Also, this routing can be directly applied for any type of RDT including RDT(2,4,1)/α.

For showing that the proposed north-last routing algorithm for RDT is deadlock free, the channel numbering method by Dally and Seitz[9] is applied. In this method, channels in the direct network is numbered so that every packet is transferred along channels with strictly increasing (or decreasing) numbers. If such a numbering is possible, it shows that there is no cyclic path between buffers in channels.

Theorem 4 *The north-last routing for RDT is deadlock-free.* □

Proof Assuming that the size of the base torus of RDT is $m \times n$. Assign two dimensional number of channel from a node (x, y) according to its direction as shown in Figure15, and let the unique number of the channel be $c_x \times m + c_y$.

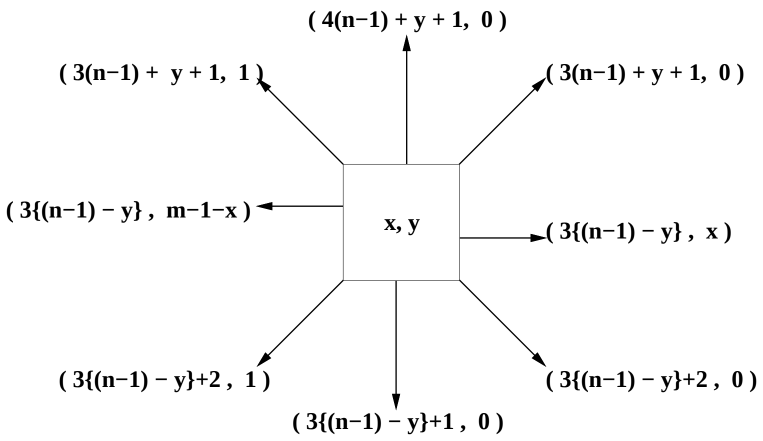

Figure 15: Numbering of the channels leaving each node (x, y) for the north-last routing algorithm for RDT.

Since the size of the base torus of RDT is $m \times n$, the range of the possible channel number (c_x, c_y) is represented by the following equations.

$$\begin{aligned} 0 \leq \quad & c_x \quad \leq 5(n-1) \\ 0 \leq \quad & c_y \quad \leq m-2 \end{aligned}$$

In RDT, there are eight possible input directions. As shown in Figure 16, all possible output channel numbers are larger than the number of input channel. In other words, the packet transfer to an output channel whose number is less than input channel is prohibited by the Definition 2 within the range shown in the above equations.

Therefore, channels are used in the increasing order on RDT.□

Figure17 shows an example of routing on the 4×4 RDT. The blocked channels are bypassed with a path consisting of channels in increasing order. This figure also shows that the number of permitted output channel is lager than that of input channel.

5 Conclusion

Two adaptive routing algorithms on RDT are proposed and discussed. The first algorithm is based on Duato's necessary and sufficient condition. Based on this method, we proposed two derivatives: Algorithm D-A and D-B. Algorithm D-A must not use the opposite direction which is used in the past, while Algorithm D-B can use any direction but must cross ranks in descending order. These algorithms can only use minimal paths. These methods are proved to be deadlock-free based on Duato's protocol.

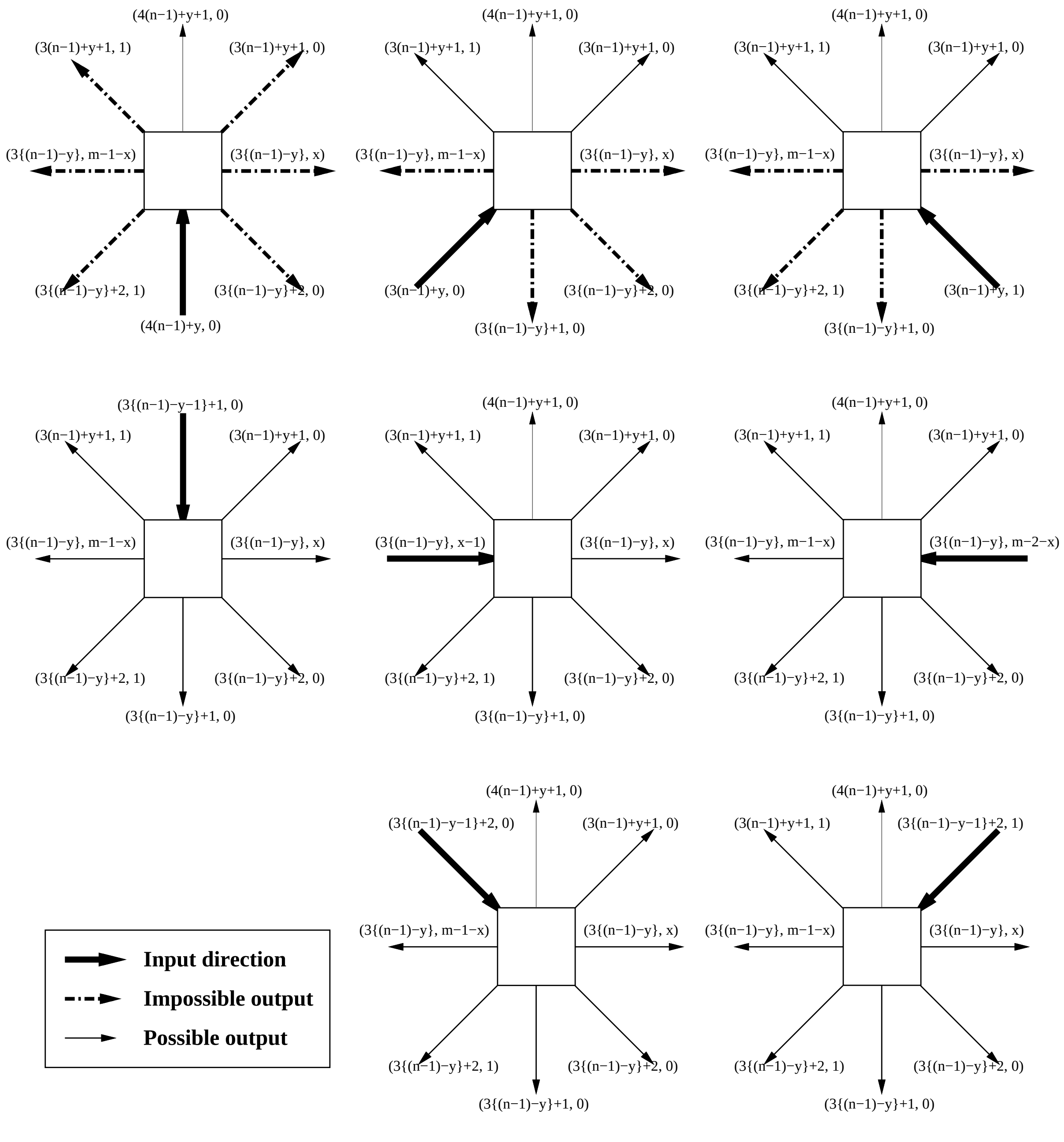

Figure 16: The possible output channels for each input channel.

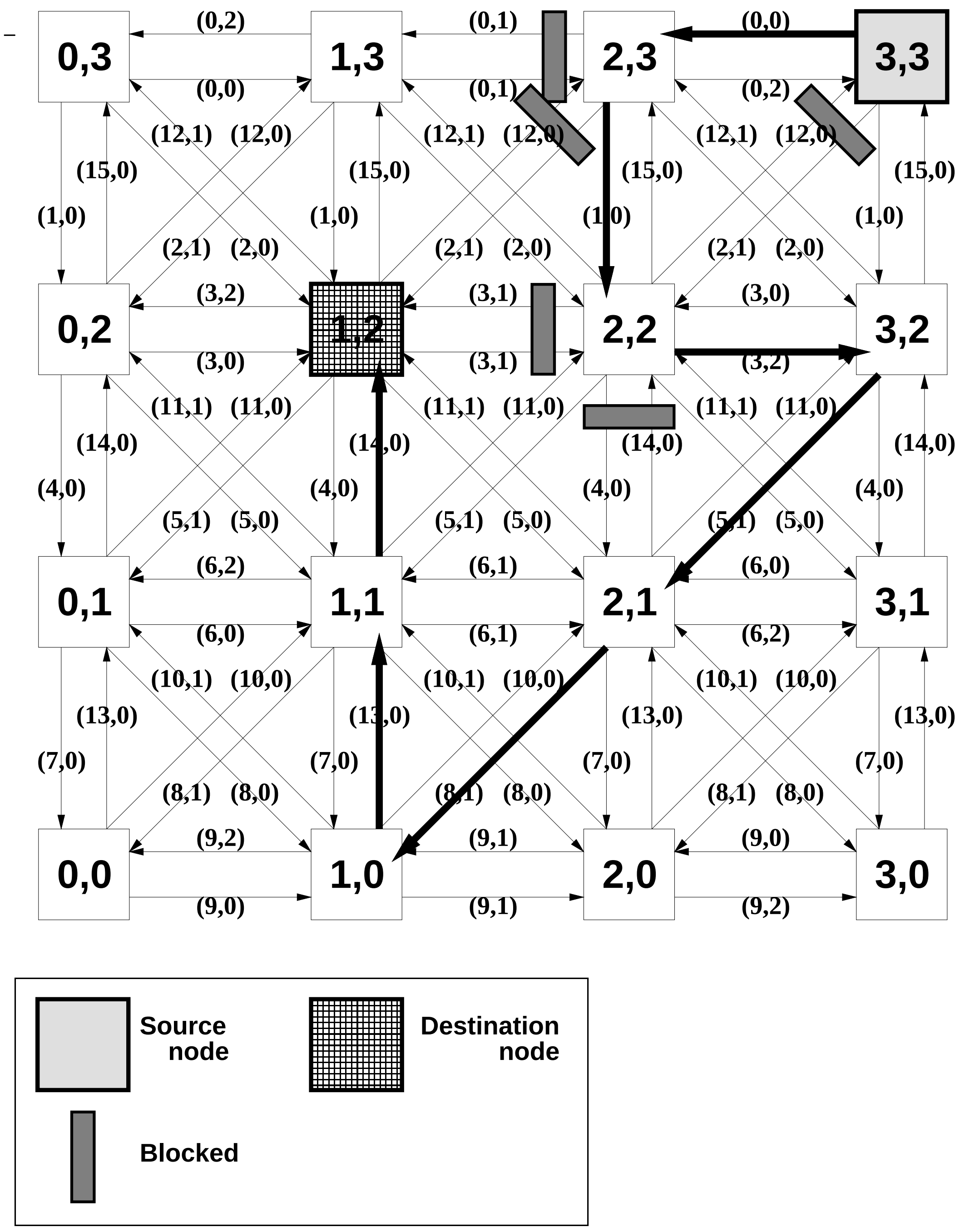

Figure 17: Example of north-last routing for RDT$(m = 4, n = 4)$.

Another algorithm called north-last routing is proposed based on the turn model. By prohibiting fourteen turns on RDT, cycles which may cause the deadlock are not formed, and alternative paths with additional hops are permitted. The north-last routing for RDT is also proved to be deadlock-free by numbering the channels.

Although this paper focuses on theoretical aspects, a simulation study which demonstrates the effect of the proposed routing algorithm is required. A simulator which can compare the proposed routing algorithms and vector routing is now under development. We will improve our algorithms based on the simulation results.

References

[1] *Paragon XP/S Product Overview.* Intel Corp., 1991.

[2] W. J. Dally A. Chien S. Fiske W. Horwat J. Kenn M. Larivee R. Lethin P. Nuth and S. Wills. The J-machine: A Fine-Grain Concurrent Computer. In *IFIP 11th Computer Congress*, pages 1147–1153, August 1989.

[3] H. Ishihata T. Horie S. Inano T. Shimizu S. Kato and M. Ikesaka. Third Generation Message Passing Computer AP1000. In *International Symposium on Supercomputing*, pages 46–55, November 1991.

[4] L. M. Ni and P. K. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Transactions on Computers*, February 1993.

[5] J. C. Bermond and C. Peyret. De Bruijn and Kautz Networks: A Competitor for the Hypercube. *North-Holland*, pages 279–293, 1989.

[6] C. E. Leiserson. Fat-trees: Universal Networks for Hardware-Efficient Supercomputing. *IEEE Transactions on Computers*, 34(10):892–901, October 1985.

[7] S. B. Akers, D. Harel, and B. Krishnamurthy. The Star Graph: An attractive alternative to the n-cube. *Proceeding of the International Conference on Parallel Processing*, 17(21):393–400, 1987.

[8] Y. Yang, H. Amano, H. Shibamura, and T. Sueyoshi. Recursive diagonal torus: An interconnection network for massively parallel computers. *Proceedings of IEEE SPDP*, 1993.

[9] W. J. Dally and C. L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers*, 36(5):547–553, May 1987.

[10] Y. Yang and H. Amano. Message Transfer Algorithms on the RDT. *IEICE Transaction on Information and Systems*, 79(2), 1996.

[11] H. Nishi, K. Nishimura, K. Anjo, H. Amano, and T. Kudoh. The JUMP-1 router chip: The versatile router for supporting distributed shared memory. *Proceedings of International Phoenix conference on computers and communications*, 1996.

[12] T. Tanaka and et.al. *The Massively Parallel Processing System JUMP-1.* IOS Press, 1996.

[13] J. Duato. A Necessary And Sufficient Condition For Deadlock-Free Adaptive Routing In Wormhole Networks. *Proceedings of the International Conference on Parallel Processing*, 1:142–149, 1994.

[14] J. Duato. A Necessary And Sufficient Condition For Deadlock-Free Adaptive Routing In Wormhole Networks. *IEEE Transaction on Parallel and Distributed Systems*, 6(10), 1995.

[15] C. J. Glass and L. M. Ni. Maximally Fully Adaptive Routing in 2D Meshes. *Proceedings of ISCA92*, pages 278–287, 1992.