

# Lower Bounds on Communication Loads and Optimal Placements in Torus Networks

M. Cemil Azizoglu and Ömer Eğecioğlu  
Department of Computer Science  
University of California at Santa Barbara  
{azizoglu, omer}@cs.ucsb.edu

## Abstract

Fully-populated tori, where every node has a processor attached, do not scale well since load on edges increases superlinearly with network size under heavy communication, resulting in a degradation in network throughput. In a partially-populated network, processors are placed on a subset of available nodes, and a routing algorithm is specified among the processors.

Analogous to multistage networks, it is desirable to have the total number of messages being routed through a particular edge increase at most linearly with the size of the placement on torus networks. Recently, Blaum, Bruck, Pi-farré, and Sanz investigated placements and provided both a lower bound, and optimal placements in the cases of 2 and 3-dimensional  $k$ -tori, consisting of  $k$  and  $k^2$  processors, respectively.

In this paper we show formally that to achieve linear load in a  $d$ -dimensional  $k$ -torus, the number of processors in the placement must be  $O(k^{d-1})$ . We use this to construct a tighter lower bound for the maximum load of a placement for 4 or more dimensions. Based on these results, we give optimal placements and their corresponding routing algorithms in tori with arbitrary number of dimensions.

## 1. Introduction

Meshes and torus based interconnection networks have been utilized extensively in the design of parallel computers in recent years [6]. This is mainly due to the fact that these families of networks have topologies which reflect the communication pattern of a wide variety of natural problems. *Throughput*, the maximum amount of traffic which can be handled by the network, is an important measure of network performance [1]. The throughput of an interconnection network is in turn bounded by its *bisection width*, the minimum number of edges that must be removed in or-

der to split the network into two parts each with about equal number of processors [3].

We consider the behavior of torus networks with bidirectional links under heavy communication load where routing is done through only *shortest (minimal length) paths* between processors. In particular, we are interested in the scenario where every processor in the network is sending a message to every other processor (also known as *complete exchange* or *all-to-all personalized communication*). This type of communication pattern is central to numerous parallel algorithms such as matrix transposition, fast Fourier transform, distributed table-lookup, etc. [7].

The  $d$ -dimensional  $k$ -torus is modeled as a directed graph where each node represents either a router or a processor-router pair, and each edge represent a communication link between two adjacent nodes. Hence, every node, with or without a processor attached, is capable of handling (sending, receiving, etc.) messages. A fully-populated  $d$ -dimensional  $k$ -torus where each node has a processor attached, contains  $k^d$  processors. Its bisection width is  $4k^{d-1}$  ( $k$  even), which gives  $k^d/2$  processors on each component of the bisection. Since every processor is communicating with every other processor, the number of messages passing through the bisection in either direction is  $2(k^d/2)(k^d/2)$ . Dividing by the bisection bandwidth, there exists an edge in the bisection with a load  $\geq k^{d+1}/8$ . This means that unlike multistage networks, the maximum load on a link is not linear in the number of processors injecting messages into the network. To alleviate this problem, Blaum et al. [1, 2] have proposed *partially-populated tori*. In this model, processors are attached to a (relatively small) subset of nodes in the network, called a *placement*. A routing algorithm which utilizes shortest paths is specified together with the placement. An *optimal placement* is a placement that achieves linear load on edges using maximum number of processors possible.

Let  $\mathcal{E}_{max}$  denote the maximum load over all the edges

for the placement  $P$ . Blaum et al. give the lower bound

$$\mathcal{E}_{max} \geq (|P| - 1)/(2d). \quad (1)$$

If  $|P|$  is constrained to be of the form  $k^i$ , then they also give placements of sizes  $k$  for  $d = 2$  and  $k^2$  for  $d = 3$ , together with routing algorithms. These placements are *optimal* in the sense that the two lower bounds are actually achieved by the placements.

How do we justify that in general a maximal size placement that can achieve linear load is  $O(k^{d-1})$ ? If the placement has size  $ck^{d-1}$  for some constant  $c$ , then mimicking the case of the fully-populated  $d$ -dimensional  $k$ -torus,

$$2(ck^{d-1}/2)(ck^{d-1}/2)/(4k^{d-1}) = O(k^{d-1}).$$

This seems to imply that linear load is at least possible for  $|P| = ck^{d-1}$ . This is a faulty argument however, as we do not know a priori that number of edges needed to split  $P$  into two equal size pieces is the same as the bisection width of the whole torus. This may push the size of an optimal placement above or below  $k^{d-1}$ .

In this paper, we introduce the concept of bisection width with respect to a placement  $P$ , and use its properties to prove that in a  $d$ -dimensional  $k$ -torus, the size of an optimal placement is  $\Theta(k^{d-1})$ . Given a placement  $P$  of maximal size, we also prove that there exists an edge separator of size  $\Theta(k^{d-1})$  which splits the torus into two components with  $\Theta(k^{d-1})$  processors of  $P$  on each side. This gives a lower bound of the form

$$\mathcal{E}_{max} \geq ck^{d-1} \quad (2)$$

for maximum load. In (2)  $c$  is a constant independent of  $d$ . This is a tighter lower bound for the load for large parameters than the lower bound (1).

Finally, we give optimal placements achieving the lower bound (2) and their corresponding routing algorithms in tori with arbitrary number of dimensions.

## 2. Preliminaries and Problem Definition

In this section, we formalize the terminology used and give necessary definitions.

**Definition 1 {Torus}**: The  $d$ -dimensional  $k$ -torus is a directed graph  $T_d^k = (V, E)$ , with vertex set

$$V = \{\vec{a} \mid \vec{a} = (a_1, a_2, \dots, a_d), a_i \in \mathbb{Z}_k\}$$

where  $\mathbb{Z}_k$  denotes the integers modulo  $k$ , and edge set

$$E = \{(\vec{a}, \vec{b}) \mid \exists j \text{ such that } a_j \equiv b_j \pm 1 \pmod{k}, \text{ and}$$

$$a_i = b_i \text{ for } i \neq j, 1 \leq i \leq d\}.$$

$T_d^k$  has a total of  $k^d$  nodes. Each node has two neighbors in each dimension, for a total of  $2d$  neighbors. Directed edges of  $T_d^k$  are also referred to as *links*.

**Definition 2 {Placement}**: A placement  $P$  of processors in  $T_d^k = (V, E)$  is a subset of  $V$ .

We use the term *node* for a generic element of the vertex set of  $T_d^k$ . A node with a processor attached is simply called a *processor*.

**Definition 3 {Routing Algorithm}**: Let  $P$  be a placement in  $T_d^k$ . A routing algorithm  $A$  is a subset  $\mathcal{C}_{\vec{p} \rightarrow \vec{q}}^A$  of the set of all shortest paths between  $\vec{p}$  and  $\vec{q}$  for every pair  $\vec{p}, \vec{q} \in P$ .

The routing algorithm  $A$  is used to deliver packets from  $\vec{p}$  to  $\vec{q}$ : When  $\vec{p}$  needs to communicate with  $\vec{q}$ , a shortest path in  $\mathcal{C}_{\vec{p} \rightarrow \vec{q}}^A$  is selected randomly with uniform probability.

For any link  $l$ , we denote the set of paths in  $\mathcal{C}_{\vec{p} \rightarrow \vec{q}}^A$  going through  $l$  by  $\mathcal{C}_{\vec{p} \rightarrow l \rightarrow \vec{q}}^A$ .

**Definition 4 {Load}**: Given a placement  $P$  in a  $T_d^k$  along with a routing algorithm  $A$ , the load of an edge  $l$  is defined as

$$\mathcal{E}(l) = \sum_{\substack{\vec{p}, \vec{q} \in P \\ \vec{p} \neq \vec{q}}} \frac{|\mathcal{C}_{\vec{p} \rightarrow l \rightarrow \vec{q}}^A|}{|\mathcal{C}_{\vec{p} \rightarrow \vec{q}}^A|} \quad (3)$$

**Definition 5 {Maximum Load}**: The maximum value of  $\mathcal{E}(l)$  for a network with placement  $P$  and a routing algorithm  $A$  is called the maximum load and denoted by  $\mathcal{E}_{max}$ . Thus  $\mathcal{E}_{max} = \max_{l \in E} \mathcal{E}(l)$ .

**Definition 6 {Cyclic Distance, Lee Distance}**: Given integers,  $i, j$  and  $k$ , the cyclic distance between  $i$  and  $j$  modulo  $k$  is

$$\min\{i - j \pmod{k}, j - i \pmod{k}\}$$

where the equivalence classes modulo  $k$  are taken to be  $0, 1, \dots, k - 1$ .

The Lee distance between  $\vec{p}, \vec{q} \in T_d^k$  is the sum of the cyclic distances between the coordinates of  $\vec{p}$  and  $\vec{q}$  [6].

**Definition 7 {Bisection Width}**: The Bisection width of a graph is the minimum number of edges which must be removed in order to split the node set into two parts of equal (within one) cardinality.

**Definition 8 {Bisection Width with respect to a Placement}**: The bisection width with respect to a placement  $P$  of  $T_d^k = (V, E)$  is the minimum number of edges which must be removed from  $E$  in order to split  $V$  into two parts each of which containing an equal (within one) number of processors in  $P$ .

We denote the set of edges of  $T_d^k$  which needs to be removed to bisect  $P$  as described by  $\partial_b P$ . Thus  $|\partial_b P|$  is the bisection width with respect to the placement  $P$ .

**Definition 9**  $\{\alpha$ -separator Width with respect to a Placement $\}$ : *The  $\alpha$ -separator width with respect to a placement  $P$  in  $T_d^k$  is the minimum number of edges which must be removed in order to split the graph into two parts containing (approximately)  $\alpha|P|$  and  $(1 - \alpha)|P|$  processors, for  $0 < \alpha < 1$ .*

We denote the set of edges in an  $\alpha$ -separator by  $\partial_\alpha P$ . Thus  $|\partial_\alpha P|$  is the  $\alpha$ -separator width of  $T_d^k$  with respect to  $P$ . Note that when  $\alpha = 1/2$ ,  $\partial_\alpha P$  and  $\partial_b P$  are equivalent.

### 2.1. Problem Definition

Our aim is to find *placements* and associated *routing algorithms* in the  $d$ -dimensional  $k$ -torus  $T_d^k$  that have *linear message load* (in number of processors in the placement) on edges under the *complete exchange* scenario.

For  $d = 2, 3$ , Blaum et al. [1, 2] have investigated placements with  $k^{d-1}$  processors. Evidently, placements with provably maximum possible number of processors are desirable. This raises another important question which we shall address: *what is the maximum number of processors a placement could have on  $T_d^k$  without compromising linear load on edges?*

Another issue is *fault tolerance*. Specifically, the routing algorithm should provide multiple routing paths between each pair of processors. Consequently we also address the following problem: *is it possible to construct optimal placements which are at the same time fault tolerant?*

### 3. A General Lower Bound for $\mathcal{E}_{max}$

The expression in (1) is the lower bound for maximum load originally given by Blaum et al. [1]. The following lemma gives a general lower bound for maximum load.

**Lemma 1** *Let  $P$  be a placement in a  $T_d^k = (V, E)$ , also, let  $S \subset P$  and  $\partial S$  be the set of all edges each connecting a node in  $S$  with another node not in  $S$ . Then*

$$\mathcal{E}_{max} \geq \frac{2|S|(|P| - |S|)}{|\partial S|} \quad (4)$$

It is easy to see that (4) reduces to (1) if the set  $S$  is taken to contain only one processor, i.e.,  $|S| = 1$  and  $|\partial S| = 4d$ . The lower bound (4) is valid independent of the routing algorithm used. Another interesting form of (4) that we shall subsequently make use of is obtained when the set  $S$  consists of half of the processors in  $P$ , i.e.,

$$\mathcal{E}_{max} \geq \frac{2(\frac{|P|}{2})^2}{|\partial_b P|} \quad (5)$$

Note that in this case  $\partial S$  becomes  $\partial_b P$ , which is the bisection width of  $T_d^k$  with respect to placement  $P$ . Next we give an upper bound on the size of  $\partial_b P$ , which we then use to calculate the maximum number of processors an optimal placement can contain.

**Theorem 1** *Any subgraph  $P$  of the  $d$ -dimensional  $k$ -torus  $T_d^k$  has bisection width  $O(k^{d-1})$ .*

The constant in  $O(k^{d-1})$  of Theorem 1 is no larger than  $6d$  when we consider directed edges. As a particular case of the theorem we have

**Corollary 1** *The  $d$ -dimensional  $k$ -torus  $T_d^k$  has bisection width of at most  $6dk^{d-1}$  with respect to any placement  $P$ . I.e.  $|\partial_b P| \leq 6dk^{d-1}$  for any placement  $P$ .*

### 3.1. Maximum Placement Size

An upper bound for the maximum number of processors an optimal placement can now be obtained by substituting the bound for  $|\partial_b P|$  given in corollary 1 into the inequality (5), while at the same time insuring that  $\mathcal{E}_{max} = O(|P|)$ , i.e. the load remains linear in the number of processors in the placement.

$$\begin{aligned} \mathcal{E}_{max} &\geq \frac{2(\frac{|P|}{2})^2}{|\partial_b P|} \geq \frac{2(\frac{|P|}{2})^2}{6dk^{d-1}} \\ &\Rightarrow c_1|P| \geq \frac{2(\frac{|P|}{2})^2}{c_2k^{d-1}} \Rightarrow |P| \leq ck^{d-1} \end{aligned}$$

for  $c_2 = 6d$  and  $c = 2c_1c_2$  for some constant  $c_1$ . That is, the size of an optimal placement in  $T_d^k$  is  $O(k^{d-1})$ . Thus, we are justified in seeking placements which have  $ck^{d-1}$  processors, for some constant  $c$ .

### 4. An Improved Lower Bound for $\mathcal{E}_{max}$

From corollary 1, we know that the bisection width of  $T_d^k$  with respect to a placement  $P$  is no larger than  $6dk^{d-1}$ . The lower bound on maximum load that one can obtain using this result is a function of dimension  $d$ , however. We can show that given a placement  $P$  on  $T_d^k$  with  $|P| = \Theta(k^{d-1})$ , it is possible to divide the network into two parts each having  $\Theta(|P|)$  processors by removing  $O(k^{d-1})$  edges, where the constants involved in  $\Theta(|P|)$  and  $O(k^{d-1})$  are independent of  $d$ .

**Theorem 2** *Relative to a placement  $P$  of size  $\Theta(k^{d-1})$ ,  $T_d^k$  has an edge separator  $\partial_\alpha P$  of size  $O(k^{d-1})$  which splits  $T_d^k$  into two parts each with  $\Theta(k^{d-1})$  processors (specifically,  $\alpha k^{d-1}$  and  $(1 - \alpha)k^{d-1}$  processors, respectively, for some  $\alpha$ ,  $0 < \alpha < 1$ ), where the constants are independent of  $d$ .*

Therefore given a placement  $P$  of size  $c_1 k^{d-1}$ , it is possible to split  $T_d^k$  into two parts having  $\alpha|P|$  and  $(1-\alpha)|P|$  processors by removing at most  $c_2 k^{d-1}$  edges, for constants  $c_1, c_2$ , and  $\alpha, 0 < \alpha < 1$ . We use this result to establish a lower bound on load which shows that the lower bound  $\mathcal{E}_{max} \geq (|P| - 1)/2d$  becomes too small as the parameter  $d$  grows larger. Taking  $|P| = c_1 k^{d-1}$ ,  $|S| = \alpha|P|$  and  $|\partial S| = c_2 k^{d-1}$  in (4), we have

$$\mathcal{E}_{max} \geq \frac{2\alpha|P|(1-\alpha)|P|}{c_2 k^{d-1}} \geq c k^{d-1}$$

where  $c = 2\alpha(1-\alpha)c_1^2/c_2$ . Note that the constant  $c$  is independent of parameter  $d$ . Hence, this lower bound comes to characterize the quantity  $\mathcal{E}_{max}$  more closely than (1) as the parameter  $d$  grows. We will use this lower bound to gauge the optimality of the placements and routing algorithms that we give next.

## 5. Linear Placements

We have established that optimal placements have  $\Theta(k^{d-1})$  processors. In this section, we introduce the notion of a *linear placement*.

**Definition 10** A placement  $P$  on  $T_d^k$  which satisfies

$$P = \{\vec{p} \mid c_1 p_1 + c_2 p_2 + \dots + c_d p_d \equiv c \pmod{k}\} \quad (6)$$

where  $c \in \mathbb{Z}_k$ , and at least one of  $c_i \in \mathbb{Z}_k$  is relatively prime to  $k$  is called a *linear placement*.

For simplicity, we will use placements where  $c_1 = c_2 = \dots = c_d = 1$ , even though our analyses apply to linear placements in general form (i.e. (6)) provided that  $c_1$  and  $c_d$  are relatively prime to  $k$ . Note that there are exactly  $k^{d-1}$  processors satisfying the expression  $p_1 + p_2 + \dots + p_d \equiv c \pmod{k}$  for any specific  $c \in \mathbb{Z}_k$ . Originally, linear placements of this form were used in three dimensional tori by Blaum et al. [1, 2], where they were called *shifted diagonal placements*.

We can also specify placements of size  $tk^{d-1}$  where  $t$  is a fixed integer less than  $k$ . For instance, the placement  $P = P_1 \cup P_2 \cup \dots \cup P_t$  where

$$\begin{aligned} P_1 &= \{\vec{p} \mid p_1 + p_2 + \dots + p_d \equiv 0 \pmod{k}\} \\ P_2 &= \{\vec{p} \mid p_1 + p_2 + \dots + p_d \equiv 1 \pmod{k}\} \\ &\vdots \\ P_t &= \{\vec{p} \mid p_1 + p_2 + \dots + p_d \equiv t-1 \pmod{k}\} \end{aligned}$$

has  $tk^{d-1}$  processors. We shall call such placements *multiple linear placements*.

We would like to point out that linear placements themselves do not guarantee the linearity of the load on edges.

We still need to find routing algorithms which enable communication between pairs of processors in a way that yields linear load. To this end, we consider two classes of routing algorithms and the analysis of the load in each case both for linear and multiple linear placements: *Ordered Dimensional Routing (ODR)* and *Unordered Dimensional Routing (UDR)*.

## 6. The Ordered Dimensional Routing Algorithm (ODR)

Given a placement  $P$  on  $T_d^k$  to route a packet from  $\vec{p} = (p_1, p_2, \dots, p_d)$  to  $\vec{q} = (q_1, q_2, \dots, q_d)$ , both in  $P$ :

**for**  $i := 1$  **to**  $d$  **do**

*Correct*  $p_i$  *in the direction of shortest cyclic distance.*

In other words, the routing path will include the nodes:

$$\begin{aligned} \vec{p} &\rightarrow (q_1, p_2, \dots, p_d) \rightarrow (q_1, q_2, p_3, \dots, p_d) \\ &\rightarrow \dots \rightarrow (q_1, q_2, \dots, q_{d-1}, p_d) \rightarrow \vec{q}. \end{aligned}$$

Note that if  $k$  is odd,  $|C_{\vec{p} \rightarrow \vec{q}}^{ODR}| = 1$ , i.e. there is only 1 path specified by the ODR algorithm for any given  $\vec{p}$  and  $\vec{q} \in P$ . However, when  $k$  is even the ODR algorithm may result in multiple paths between some pairs of processors in the placement. To aid in the analysis, we use the following (restricted) version which ensures the existence of only one canonical routing path between any given pair of processors regardless of the parity of  $k$ .

**for**  $i := 1$  **to**  $d$  **do**

**begin**

**if** *there is more than one way of correcting*  $p_i$  **then**

*Pick the path correcting*  $p_i$  *in the (+) direction*  $\pmod{k}$ ;

*Correct*  $p_i$  *in the direction of shortest cyclic distance*

**end**

Thus if there are two choices for some  $p_i, q_i$  coordinate pair, the algorithm routes through  $p_i + 1 \pmod{k}, p_i + 2 \pmod{k}, \dots, q_i$ .

**Theorem 3** Given a linear placement  $P = \{\vec{p} \mid p_1 + p_2 + \dots + p_d \equiv 0 \pmod{k}\}$  in  $T_d^k = (V, E)$ , Ordered Dimensional Routing Algorithm results in linear load on edges.

The proof of this theorem actually gives that regardless of the parity of  $k$ , for a linear placement  $P$  with ODR,  $|P| = k^{d-1}$  and

$$\mathcal{E}_{max} = \frac{k^{d-1}}{8} + o(k^{d-1}).$$

For multiple linear placements with ODR we have

**Theorem 4** *Multiple linear placements along with ODR algorithm on  $T_d^k$  results in linear load on edges.*

For multiple linear placements with  $tk^{d-1}$  processors, the proof of the theorem gives

$$\mathcal{E}_{max} \leq t^2 k^{d-1}$$

which is linear in  $|P|$  for constant  $t$ .

The shortcoming of having only one path between a pair of processors of ODR results in limited fault-tolerance of the network. We remedy this by the UDR algorithm in the next section.

## 7. Unordered Dimensional Routing (UDR)

*Unordered Dimensional Routing (UDR)*, provides multiple paths between each pair of processors. The algorithm is as follows: To route a packet from  $\vec{p} = \text{to } \vec{q} =$ , both in  $P$

```

for  $i := 1$  to  $d$  do
  begin
    Select a number  $j$  from the set  $\{1, 2, \dots, d\}$  that has not
    been used before;
    Correct  $p_j$  in the direction of shortest cyclic distance
  end

```

As in ODR, a dimension is corrected completely before another is selected. Unlike ODR, however, the order in which the dimension to be corrected next is picked is arbitrary. This algorithm thus provides multiple paths for each pair of processors and improves the fault-tolerance of the system. If  $\vec{p}$  and  $\vec{q}$  are two processors differing in  $s$  dimensions, then there will be  $s!$  different paths from  $\vec{p}$  to  $\vec{q}$  in UDR, i.e.  $|\mathcal{C}_{\vec{p} \rightarrow \vec{q}}^{UDR}| = s!$ . We can show that UDR algorithm results in linear load in edges.

**Theorem 5** *Given a linear placement  $P = \{\vec{p} \mid p_1 + p_2 + \dots + p_d \equiv 0 \pmod{k}\}$  in  $T_d^k = (V, E)$ , Unordered Dimensional Routing Algorithm results in linear load on edges.*

The maximum load for UDR satisfies

$$\mathcal{E}_{max} \leq 2^{d-1} k^{d-1}$$

which is linear in  $|P| = k^{d-1}$  for any fixed  $d$ .

Similarly, for multiple linear placements with UDR we have

**Theorem 6** *Multiple linear placements along with UDR algorithm on  $T_d^k$  results in linear load on edges.*

## 8. Conclusion

Following the work of Blaum, Bruck, Pifarré, and Sanz [1, 2], we have considered communication in partially-populated torus networks in terms of placements of processors and associated routing algorithms. We have provided lower bounds for the maximum load under the all-to-all communication scenario, and found bounds on the size of an optimal placement. We have shown that arbitrary placements can be bisected by removing a set of edges of the same order as the bisection width of the torus. We then provided optimal placements of size  $\Theta(k^{d-1})$  on the  $d$ -dimensional  $k$ -torus using what we call linear and multiple linear placements, and gave load analyses of each under two different routing algorithms.

There are some interesting properties of placements still to be resolved. Among these are the characterization of optimal placements in terms of restrictions to subtori and an extensive analysis of the properties of edge separators of tori relative to optimal placements.

## References

- [1] Mario Blaum, Jehoshua Bruck, Gustavo D. Pifarré, and Jorge L. Sanz. *On Optimal Placements of Processors in Tori Networks*. Proc. of the 8<sup>th</sup> IEEE Symp. on Parallel and Distributed Processing, pages 552-555, Oct. 1996.
- [2] Mario Blaum, Jehoshua Bruck, Gustavo D. Pifarré, and Jorge L. Sanz. *On Optimal Placements of Processors in Fault-Tolerant Tori Networks*. preprint, 1997.
- [3] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays· Trees· Hypercubes*. Morgan Kaufmann Publishers, San Mateo, California, 1992.
- [4] Lionel M. Ni and Philip K. McKinley. *A Survey of Wormhole Routing Techniques in Direct Networks*. Computer, Vol. 26 (2), pp. 62-76, Feb. 1993.
- [5] Myung M. Bae and Bella Bose. *Resource Placement in Torus-Based Networks*. IEEE Transactions on Computers, Vol. 46 (10), pp. 1083-1092, 1997.
- [6] Bella Bose, Bob Broeg, Younggeun Kwon and Yaagoub Ashir. *Lee Distance and Topological Properties of  $k$ -ary  $n$ -cubes*. IEEE Transactions on Computers, Vol. 44 (8), pp. 1021-1030, August 1995.
- [7] Yu-Chee Tseng, Ting-Hsien Lin, Sandeep K. S. Gupta, Dhableswar K. Panda. *Bandwidth-optimal Complete Exchange on Wormhole-routed 2D/3D Torus Networks: A Diagonal-Propagation Approach*. IEEE Transactions on Parallel and Distributed Systems, Vol. 8 (4), pp. 380-396, April 1997.