Brief Contributions

Lower Bounds on Communication Loads and Optimal Placements in Torus Networks

M. Cemil Azizoglu and Ömer Egecioglu

Abstract—Fully populated torus-connected networks, 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 occupy a subset of available nodes and a routing algorithm is specified among the processors placed. Analogous to multistage networks, it is desirable to have the total number of messages being routed through a particular edge in toroidal networks increase at most linearly with the size of the placement. To this end, we consider placements of processors which are described by a given placement algorithm parameterized by k and d: We show formally, that to achieve linear communication load in a d-dimensional k-torus, the number of processors in the placement must be equal to ck^{d-1} for some constant c. Our approach also gives a tighter lower bound than existing bounds for the maximum load of a placement for arbitrary number of dimensions for placements with sufficient symmetries. Based on these results, we give optimal placements and corresponding routing algorithms achieving linear communication load in tori with arbitrary number of dimensions.

Index Terms—Torus, routing, placement, load, bisection, interconnection network, edge separator.

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. Furthermore, they are scalable, as well as highly suitable for hardware implementation.

An important factor determining the efficiency of a parallel algorithm on a network is the efficiency of communication itself among processors. The network should be able to handle "large" number of messages without exhibiting degradation in performance. *Throughput*, the maximum amount of traffic which can be handled by the network, is an important measure of network performance [4]. The throughput of an interconnection network is in turn bounded by its *bisection width*, the minimum number of edges that must be removed in order to split the network into two parts, each with about equal number of processors [9].

In this paper, we consider the behavior of toroidal networks with bidirectional links under heavy communication load. We assume that the communication latency is kept minimum by routing the messages through only *shortest (minimal length) paths*. 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], and central to efficient implementation of high-level computing models, such as the PRAM and Bulk-Synchronous Parallel (BSP). In Valiant's BSP-model for parallel computation [15], for example, routing of h-relations, in which every processor in the network is the source and destination of at most h packets, forms the main communication primitive. The complete-exchange scenario that we investigate in this paper has been studied and shown to be useful for efficient routing of both random and arbitrary h-relations [8], [13], [14].

The d-dimensional k-torus is modeled as a directed graph where each node represents either a router or a processor-router pair, depending on whether or not a processor is attached at the node. Each edge represents a unidirectional communication link between two adjacent nodes. Hence, every node in the network is capable of message routing, i.e., directly receiving from and sending to its neighboring nodes.

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 $\frac{1}{2}k^d$ processors for each component of the bisection. Under the complete-exchange scenario, the number of messages passing through the bisection in both directions is $2(k^d/2)(k^d/2)$. Dividing by the bisection bandwidth, we find that there must exist 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. [4], [5] have proposed *partially populated tori*. In this model, the underlying network is toroidal, but only the nodes with an attached processor inject messages into the network. Only a (relatively small) subset of nodes (called a *placement*) possess attached processors, while the other nodes merely function as routing nodes. This is similar to the case of a multistage network: A multistage network with $k \times k$ switches (routing nodes) and $\log_k n$ stages serves n injection points, and utilizes $n \log_k n$ routing nodes [4].

In partially populated tori, a routing algorithm which utilizes shortest paths is specified together with the placement. The routing algorithm dictates routing paths between every processor pair. An *optimal placement* is a placement that achieves linear communication load on edges under the specified routing algorithm using the maximum number of processors possible. Achieving linear load would be irrelevant if we were to consider a specific placement in a torus with fixed parameters k and d. Hence, the placements we are interested in are actually descriptions of (or algorithms for) assigning processors to nodes in the whole class of tori, determined by varying values of k and d. We use the symbol P for placement to refer to the placement $P_{d,k}$ in the d-dimensional k-torus when the context is clear.

The notion of *resource placement* in general has been investigated by a number of researchers such as Bose et al. [6], Alverson et al. [1], Pitteli and Smitley [12]. Our aim is to find placements and routing algorithms which will enable efficient communication between processors and, at the same time, reduce the susceptibility of the network to link faults by reducing the number of messages relying upon a particular edge [4]. This is achieved by providing routing algorithms in which the number of minimal paths specified between pairs of processors in the placement is kept large, without compromising the linearity of load.

Let $\mathcal{E}_{max}$ denote the maximum load over all edges of placement P. Blaum et al. give the lower bound

$$\mathcal{E}_{max} \geq \frac{|P|-1}{2d}, \tag{1}$$

which means that, for $d=2$, $\mathcal{E}_{max} \geq |P|/4$ and, for $d=3$, $\mathcal{E}_{max} \geq |P|/6$. 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 $d=3$, along with corresponding routing algorithms. These placements

• *The authors are with the Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA 93106. E-mail: {azizoglu, omer}@cs.ucsb.edu.*

Manuscript received 13 May 1998; accepted 2 Dec. 1999. For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 106835.

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 placement that can achieve linear load has $O(k^{d-1})$ processors, let alone exactly 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, the average load on an edge would be

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

This seems to imply that linear load is at least possible for

$$| P | = ck^{d-1}. \tag{2}$$

This is a faulty argument, however, as we do not as yet know that the number of edges needed to split P into two equal-size pieces is the same as the bisection width of the whole torus. This necessitates the concept of *bisection width with respect to a placement* P, which we use to prove that the size of an optimal placement is as given in (2). This results in a lower bound of the form

$$\mathcal{E}_{max} \geq ck^{d-1} \tag{3}$$

for maximum load. Unlike (1), in (3), c is a constant independent of d. This is a tighter lower bound for the load for large d than the lower bound (1). We give a simple proof of this result for *uniform* placements in which the number of processors in each principal subtorus is the same. Finally, we give optimal uniform placements (called *linear placements*) achieving the lower bound (3) and corresponding routing algorithms (*Ordered Dimensional Routing (ODR)* and *Unordered Dimensional Routing (UDR)*) in tori with arbitrary number of dimensions. Of the two routing algorithms, ODR is simpler, but UDR provides fault tolerance by allowing more routes. We also show how to extend these to more general placements in tori that we refer to as *multiple linear placements*. These are optimal placements of size tk^{d-1}, where t is some arbitrarily picked integral constant.

Since there are two parameters k and d to consider in the analysis, it is worthwhile noting there are two distinct treatments of linearity that arise. In the requirement of linearity on the maximum load as a function of $| P |$ and the particular form of the size of an optimal placement that we prove, there are two constants in the quantities $c_1 | P |$ and c_2k^{d-1}. When either c_1 or c_2 depends on d, then the linearity is in terms of k only. The desirable case is the construction of placements in which neither constant depends on d.

The outline of the paper is as follows: Section 2 gives necessary definitions and the formal statement of the problem. In Section 3, a lower bound on the maximum load on an edge is given, which is also a generalization of the lower bound given by [4]. This bound, along with the notion of bisection width with respect to a placement, is used to get an upper bound on the number of processors in an optimal placement. We show in Section 4 that a torus can be split into two parts with respect to a placement which assigns an equal number of processors to each subtorus along a specific dimension by removing $4k^{d-1}$ edges. We use this to give a new lower bound on the maximum load which is independent of the dimension parameter. Finally, in Sections 5, 6, and 7, we define and analyze an important class of placements, called linear placements, which possess the property that, along any dimension, the number of processors assigned to each subtorus is the same, and give associated routing algorithms which achieve linear load and fault tolerance. Section 8 includes conclusions and some possible generalizations.

An extended abstract of some of the ideas in this paper can be found in [2].

2 Preliminaries and Problem Definition

We start out with the problem definition and follow it with a sequence of formal definitions and terminology that will be used in the rest of the paper.

2.1 Problem Definition

Our aim is to find *placements* and associated *routing algorithms* in the d-dimensional k-torus T_k^d that have *linear communication load* (in terms of the number of processors in the placement) on edges under the *complete exchange* scenario. Specifically, we would like to devise placements $P = P_{d,k}$ and corresponding routing algorithms A for the class of d-dimensional k-tori for which the maximum load $\mathcal{E}_{max} = c \mid P \mid$, for some constant c.

Definition 1 (d-Dimensional k-Torus). *The d-dimensional k-torus is a directed graph $T_k^d = (V, E)$, with vertex set*

$$V = \{\vec{a} \mid \vec{a} = (a_1, a_2, \ldots, a_d),\ a_i \in \mathbb{Z}_k\}, \tag{4}$$

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_k^d 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_k^d are also referred to as *links*. Any fixed value of an a_i in (4) defines a subgraph of T_k^d isomorphic to a lower dimensional torus T_k^{d-1}. These are called the *principal subtori* of T_k^d.

Definition 2 (Placement). *A* placement *P of processors in $T_k^d = (V, E)$ is a subset of V. A placement is assumed to be given by a placement description (algorithm) which can generate it for any k and d.*

We use the term *node* for a generic element of the vertex set of T_k^d. A node with a processor attached is simply called a *processor*. A placement P is called *uniform* if each principal subtorus of T_k^d contains the same number of processors of P.

Definition 3 (Routing Algorithm). *Let P be a placement in T_k^d. A* routing algorithm *A is a subset $\mathcal{C}^A_{\vec{p}\to\vec{q}}$ of the set of all shortest paths between $\vec{p}$ and $\vec{q}$ for every pair $\vec{p}, \vec{q} \in P$ (See Fig. 1).*

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 $\mathcal{C}^A_{\vec{p}\to\vec{q}}$ is selected randomly with uniform probability.

For any link l, we denote the set of paths in $\mathcal{C}^A_{\vec{p}\to\vec{q}}$ going through l by $\mathcal{C}^A_{\vec{p}\to l\to\vec{q}}$, and use the following definition of load as given in [4].

Definition 4 (Load). *Given a placement P in a T_k^d along with a routing algorithm A, the* load *of an edge l is defined as*

$$\mathcal{E}(l) = \sum_{\substack{\vec{p},\vec{q}\in P \\ \vec{p}\neq\vec{q}}} \frac{| \mathcal{C}^A_{\vec{p}\to l\to\vec{q}} |}{| \mathcal{C}^A_{\vec{p}\to\vec{q}} |}. \tag{5}$$

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,*

$$\mathcal{E}_{max} = \max_{l\in E} \mathcal{E}(l).$$

Considering (5) for $\mathcal{E}(l)$, the more paths the routing algorithm provides between any two processors, the smaller the load on any edge that is used to route messages between these processors. In addition, availability of a large number of choices means better fault tolerance.

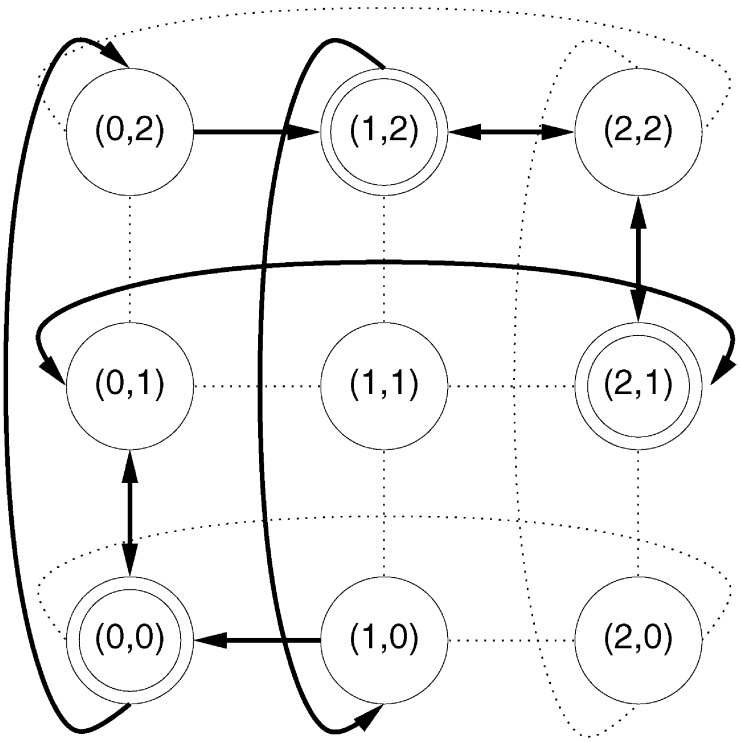

Fig.1. A placement of three processors on T_3^2. Among the links, the ones on specified shortest paths between the processors are highlighted.

We shall consider algorithms which use minimal (shortest) paths. Minimal paths on a toroidal network are associated with the notion of *cyclic distance* and *Lee distance* which we define next.

Definition 6 (Cyclic Distance, Lee Distance). *Given three integers, i, j, and k, the cyclic distance between i and j modulo k is given by*

$$min\{i - j(mod\ k), j - i(mod\ k)\},$$

where the equivalence classes modulo k are taken to be $0, 1, \ldots, k-1$. The Lee distance *between two nodes $\vec{p}, \vec{q} \in T_k^d$ is the sum of the cyclic distances between the coordinates of $\vec{p}$ and $\vec{q}$.*

The Lee distance between $\vec{p}, \vec{q} \in T_k^d$, is the length of a shortest path between $\vec{p}$ and $\vec{q}$ on the torus [6], [10].

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 $T_k^d = (V, E)$ is the minimum number of edges which must be removed from E in order to split V into two parts, each of which contains an equal (within one) number of processors in P.*

We denote by $\partial_b P$ a minimal cardinality set of edges of T_k^d which needs to be removed to bisect P. Thus, $|\ \partial_b P\ |$ is the bisection width with respect to the placement P.

Definition 9 (O, Ω). *We use the notation $f(k) = O(g(k))$ to mean $f(k) \leq cg(k)$ for some constant $c > 0$ whenever $k \geq k_0$. $f(k) = \Omega(g(k))$ if, for some constant $c > 0$, $f(k) \geq cg(k)$ for infinitely many values of k.*

For $d = 2, 3$, Blaum et al. [4], [5] 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_k^d (for varying k and d) without compromising linear load on edges?*

Another important issue is *fault tolerance*. Specifically, the routing algorithm should provide multiple routing paths between each pair of processors so that, if any of the links fails, the network will remain functional by routing the messages through paths which do not include the defective link. Consequently, we also address the following problem: *Is it possible to construct optimal placements which are, at the same time, fault- tolerant?*

In the following sections, we analyze lower bounds for maximum load and study the questions posed above.

3 A General Lower Bound for Maximum Load

We start out with an important lemma which will prove to be a useful tool in the subsequent sections. The lower bound for maximum load originally given by Blaum et al. [4] is

$$\mathcal{E}_{max} \geq \frac{|\ P\ | - 1}{2d}. \tag{6}$$

The following lemma gives a more general form of (6).

Lemma 1. *Let P be a placement in a $T_k^d = (V, E)$. Let $S \subset P$ and denote by ∂S the set of all edges each connecting a node in S with another node not in S. Then,*

$$\mathcal{E}_{max} \geq \frac{2\ |\ S\ |\ (|\ P\ | - |\ S\ |)}{|\ \partial S\ |}. \tag{7}$$

Proof. The total number of messages exchanged between processors in S and processors in $P - S$, in either direction, is $2\ |\ S\ |\ (|\ P\ | - |\ S\ |)$ under all-to-all personalized communication scenario. Also, these messages must go through one of the edges in ∂S. The average number of messages going through an edge in ∂S is $2\ |\ S\ |\ (|\ P\ | - |\ S\ |)/\ |\ \partial S\ |$ and the lemma follows. □

It is easy to see that (7) reduces to (6) if the set S is taken to contain only one processor, i.e., $|S| = 1$ and $|\partial S| = 4d$. The lower bound (7) is valid independent of the routing algorithm used. Another interesting form of (7) that we shall subsequently make use of is obtained when the set S consists of half of the processors in P, i.e.,

$$\mathcal{E}_{max} \geq \frac{2\left(\frac{|P|}{2}\right)^2}{|\partial_b P|}. \tag{8}$$

Note that, in this case, ∂S becomes $\partial_b P$, which is the bisection width of T_k^d 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.

Proposition 1. *Any subset P of nodes of the d-dimensional k-torus can be bisected by removing $O(k^{d-1})$ edges of T_k^d. In particular, any subgraph of T_k^d has bisection width $O(k^{d-1})$.*

Proof. See the Appendix. □

The constant in $O(k^{d-1})$ of Proposition 1 is no larger than $6d$ when we consider directed edges. Thus, we have the following corollary.

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

Remark. Although the assertion of Proposition 1 appears intuitive because of its geometric nature for $d = 2$ and $d = 3$, it is easy to come up with examples of general graphs for which the bisection width with respect to a given subgraph can become arbitrarily far from that of the original graph. As an example, two copies of the complete graph K_{2n} on $2n$ nodes joined by a single edge has bisection width 1. Its subgraphs with $2n$ nodes have bisection widths ranging from 1 to $\Omega(n^2)$, depending on how evenly the $2n$ nodes are distributed among the two copies of K_{2n}.

3.1 Maximum Placement Size

An upper bound for the maximum number of processors an optimal placement can contain can now be obtained by substituting the bound for $|\partial_b P|$ given in Corollary 1 into inequality (8), while, at the same time, insuring that $\mathcal{E}_{max} = c_1 |P|$ for some constant c_1, i.e., the load remains linear in the number of processors in the placement.

$$\mathcal{E}_{max} \geq \frac{2\left(\frac{|P|}{2}\right)^2}{|\partial_b P|} \geq \frac{2\left(\frac{|P|}{2}\right)^2}{6dk^{d-1}} \;\Rightarrow\; c_1 |P| \geq \frac{2\left(\frac{|P|}{2}\right)^2}{6dk^{d-1}} \;\Rightarrow\; |P| \leq c_2 k^{d-1} \tag{9}$$

for $c_2 = 12dc_1$. That is, the size of an optimal placement in T_k^d is ck^{d-1}, where $c = c_2$ is a constant of our choice. However, using this analysis, we are not able to have both c_1 and c_2 independent of dimension d, as c_1 and c_2 are related by $c_2 = 12dc_1$. Therefore, one of the linearity conditions imposed by $\mathcal{E}_{max} = c_1 |P|$ and $|P| = c_2 k^{d-1}$ can be in terms of the parameter k only. For a fixed dimension d, this may be satisfactory, but we are interested in varying the parameter d as well as k.

4 An Improved Lower Bound for Maximum Load

From Corollary 1, we know that the bisection width of T_k^d with respect to a placement P is no larger than $6dk^{d-1}$. The lower bound on maximum load that one can obtain using inequality (7) is a function of dimension d, however. This means as d gets larger, the lower bound on the maximum load gets smaller.

We can establish a tighter lower bound for the maximum load by eliminating the dependence of the multiplicative constant on d. The proof of this is particularly simple for uniform placements. Intuitively, uniform placements are among the class of placements that distribute the processors rather evenly throughout the torus. We shall describe a number of such placements in Section 5.

Specifically, we show that, given a uniform placement P on T_k^d with $|P| = ck^{d-1}$, it is possible to divide the torus into two parts, each having $\frac{1}{2}|P|$ processors, by removing $4k^{d-1}$ edges.

Theorem 1. *With respect to a uniform placement P of size ck^{d-1}, T_k^d has a bisection width $\partial_b P$ of size $4k^{d-1}$.*

Proof. We give the proof for k even. Pick an arbitrary dimension. Each of the k copies of $(d-1)$-dimensional k-subtori along this dimension has ck^{d-2} processors. Number the subtori along this dimension consecutively from 0 to $k-1$. We remove the $2k^{d-1}$ links between subtori labeled 0 and 1 and the additional $2k^{d-1}$ links between subtori labeled $\frac{1}{2}k$ and $1 + \frac{1}{2}k$. This bisects T_k^d into two parts each with $\frac{1}{2}ck^{d-1}$ processors. □

It is possible to generalize Theorem 1 to a larger class of placements than uniform placements by imposing weaker restrictions on the families $P_{d,k}$. For example, it suffices to assume only a single dimension along which an equal number of processors are assigned to each principal subtorus. We do not address possible generalizations of this problem here.

By Theorem 1, given a uniform placement P of size ck^{d-1}, it is possible to split T_k^d into two parts having $\frac{1}{2}|P|$ processors each by removing at most $4k^{d-1}$ edges. We can use this result to establish a lower bound on load which shows that the lower bounds given by inequalities (6) and (9) for $\mathcal{E}_{max}$ become too small as the parameter d grows larger.

Taking $|P| = ck^{d-1}$, $|S| = \frac{1}{2}|P|$, and $|\partial S| = 4k^{d-1}$ in (7), we have

$$\mathcal{E}_{max} \geq \frac{2\left(\frac{ck^{d-1}}{2}\right)^2}{4k^{d-1}} = \frac{c^2 k^{d-1}}{8},$$

where the constant c is independent of parameter d. Hence, this lower bound comes to characterize the quantity $\mathcal{E}_{max}$ more closely than (6) or (9) 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 in Section 3 that optimal placements have ck^{d-1} processors. In this section, we introduce the notion of a *linear placement* in which the coordinates of each processor in the placement satisfy a particular type of linear equation over $\mathbb{Z}_k$.

Definition 10. *A placement P on T_k^d which satisfies*

$$P = \{\vec{p} \mid c_1 p_1 + c_2 p_2 + \cdots + c_d p_d \equiv c \,(mod\ k)\}, \tag{10}$$

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 $c_1 = c_2 = \ldots = c_d = 1$. Note that there are exactly k^{d-1} processors satisfying the expression $p_1 + p_2 + \cdots + p_d \equiv c \,(mod\ k)$ for any specific $c \in \mathbb{Z}_k$. A special case of linear placements is the *shifted diagonal placement* used by Blaum et al. [4], [5] for the case $d = 3$.

We can also specify placements of size tk^{d-1}, where t is a fixed integer less than k. For instance, the placement

$$P = P_1 \cup P_2 \cup \ldots \cup P_t,$$

where

$$\begin{aligned} P_1 &= \{\vec{p} \mid p_1 + p_2 + \cdots + p_d \equiv 0 \ (mod\ k)\} \\ P_2 &= \{\vec{p} \mid p_1 + p_2 + \cdots + p_d \equiv 1 \ (mod\ k)\} \\ &\vdots \\ P_t &= \{\vec{p} \mid p_1 + p_2 + \cdots + p_d \equiv t-1 \ (mod\ k)\}, \end{aligned}$$

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

Note that both linear placements and multiple linear placements where all coefficients c_i are relatively prime to k (e.g., all $c_i = 1$) are uniform placements, i.e., they assign an equal number of processors to each principal subtorus. This is because the equation

$$c_1p_1 + c_2p_2 + \cdots + c_dp_d \equiv c \ (mod\ k)$$

that we use to define such placements has exactly k^{d-2} solutions for any fixed value of p_l, $1 \le l \le d$.

Remark. We would like to point out that linear (and multiple linear) placements themselves do not guarantee the linearity of the load on edges. *Linear*, only refers to the fact that the coordinates of the processors in the placement satisfy a linear equation over $\mathbb{Z}_k$. We still need to construct routing algorithms which enable communication between pairs of processors in a way that yields load that is linear in $\mid P \mid$.

In Sections 6 and 7, we specify different routing algorithms and analyze their maximum communication load on edges. As we have mentioned earlier, the routing algorithms will use minimal (shortest) paths between processors. To deliver a message from processor $\vec{p}$ to $\vec{q}$, the value of $\vec{p}$ in each dimension is "corrected" toward the corresponding value in $\vec{q}$ by the amount and direction ($\pm$) dictated by the shortest cyclic distance between the values in that dimension. The exact way of correcting the dimensions to route the packets is specified by the routing algorithm.

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 Ordered Dimensional Routing (ODR)

The algorithm is simple. Given a placement P on T_k^d to route a packet from $\vec{p} = (p_1,\ p_2, \ldots, p_d)$ to $\vec{q} = (q_1,\ q_2, \ldots,\ q_d)$, both in P:

for $i := 1$ **to** d **do**
Correct p_i in the direction of shortest cyclic distance

That is, the routing path will include the following nodes:

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

Note that if k is odd, $\mid \mathcal{C}^{ODR}_{\vec{p}\rightarrow\vec{q}} \mid = 1$, i.e., there is only one 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 will 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 that corrects p_i in the (+) direction (mod 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 $p_i + 1 \ (mod\ k)$, $p_i + 2 \ (mod\ k)$, $\ldots$, q_i. The shortcoming of having only one path between a pair of processors is the lack of fault tolerance in the network. Specifically, if an edge over which a pair of processors communicate fails, then the pair will no longer be able to exchange messages. In Section 7, we look at another routing algorithm which does not suffer from this limitation.

6.1 Load Analysis for Linear Placements with ODR

Theorem 2. *Given a linear placement $P = \{\vec{p} \mid p_1 + p_2 + \cdots + p_d \equiv 0 \ (mod\ k)\}$ in $T_k^d = (V, E)$, ODR Algorithm results in linear load on edges.*

Proof . Since ODR algorithm ensures one path between each pair of processors, each denominator in (5) for $\mathcal{E}(l)$ is 1. Thus, in order to compute $\mathcal{E}_{max}$, we need only count the (maximum) number of pairs of processors that communicate through a specific edge. Without loss of generality, consider an edge $l \in E$, where

$$l = < (i_1, \ldots, i_s, \ldots, i_d), (i_1, \ldots, i_s + 1, \ldots, i_d) > .$$

We will count pairs of processors which communicate using l. Let $\vec{p}$ and $\vec{q} \in P$ be two processors, where $\vec{p}$ sends messages to $\vec{q}$ through l. Since ODR algorithm is used, we must have

$$\vec{p} = (p_1, \ldots, p_s,\ i_{s+1},\ \ldots,\ i_d)$$

and

$$\vec{q} = (i_1, \ldots,\ i_{s-1},\ q_s,\ q_{s+1}, \ldots,\ q_d)$$

with $p_s \le i_s \ (mod\ k)$ and $q_s \ge i_s + 1 \ (mod\ k)$. Since $\vec{p}$ and $\vec{q}$ are both in P and P is linear, the coordinates satisfy

$$p_1 + \cdots + p_s + i_{s+1} + \cdots + i_d \equiv 0 \ (mod\ k)$$

and

$$i_1 + \cdots + i_{s-1} + q_s + \cdots + q_d \equiv 0 \ (mod\ k).$$

Therefore,

$$p_1 + \cdots + p_s \equiv c_1 \ (mod\ k) \tag{11}$$

$$q_s + \cdots + q_d \equiv c_2 \ (mod\ k). \tag{12}$$

Note that if $\vec{p}$ and $\vec{q}$ are to use the edge l, then, in order to ensure that messages follow shortest paths, we must also have,

$$q_s - p_s \ (mod\ k) \le p_s - q_s \ (mod\ k)$$

by the property of cyclic distance. The number of processors satisfying (11) is less than or equal to k^{s-1}, while those satisfying (12) is less than or equal to k^{d-s}. Thus, the total number of processor pairs cannot be more than $k^{s-1}k^{d-s} = k^{d-1}$. Therefore, $\mathcal{E}_{max} \le k^{d-1}$ and the maximum load is linear in $\mid P \mid = k^{d-1}$.

Note that we are actually overcounting since we have not taken the restrictions $p_s \le i_s \ (mod\ k)$, $q_s \ge i_s + 1 \ (mod\ k)$, and $q_s - p_s \ (mod\ k) \le p_s - q_s \ (mod\ k)$ on the sth dimension into account when we count the solutions of (11) and (12). These conditions affect the choices of p_s and q_s. A more accurate expression (though of the same order) can be obtained by paying closer attention to these parameters: To determine the number of different ways p_s and q_s may be chosen, consider the *one-dimensional k-subtorus* (ring) on which the edge l lies. Assume first that k is even. Without loss of generality, also assume that the nodes in the ring are enumerated from 0 to $k-1$ such that $i_s = \frac{1}{2}k - 1$. Then, the ODR algorithm will use l to deliver messages from node 0 to only node $\frac{1}{2}k$ on this ring.

Similarly, it will use edge l for messages from node 1 to node $\frac{1}{2}k$ and from node 1 to node $\frac{1}{2}k+1$ and so on. Messages from node $\frac{1}{2}k-1\ (=i_s)$ can be sent using l to any node indexed $\frac{1}{2}k$ to $k-1$. The total number of choices for p_s and q_s will therefore be:

$$\frac{\frac{k}{2}\left(\frac{k}{2}+1\right)}{2}=\frac{k^2}{8}+\frac{k}{4}.$$

Now, assume that k is odd and, also, that $i_s=\frac{1}{2}(k-1)$. In this case, messages from node 1 can be delivered to only node $\frac{1}{2}(k+1)$ through l, while messages from node 2 can be routed to nodes $\frac{1}{2}(k+1)$ and $\frac{1}{2}(k+1)+1$, and so on. Thus there are a total of

$$\frac{\frac{k-1}{2}\left(\frac{k-1}{2}+1\right)}{2}$$

choices for p_s and q_s when k is odd. Therefore, the number of solutions to (11) and (12) which satisfy the conditions $p_s \le i_s\ (mod\ k)$, $q_s \ge i_s+1\ (mod\ k)$, and $q_s-p_s\ (mod\ k) \le p_s-q_s\ (mod\ k)$ is

$$\mathcal{E}_{max}=\frac{k^{d-1}}{8}+\frac{k^{d-2}}{4}$$

when k is even and

$$\mathcal{E}_{max}=\frac{k^{d-1}}{8}-\frac{k^{d-3}}{8}$$

when k is odd. □

To summarize, regardless of the parity of k, for a linear placement P with ODR, $\mid P \mid = k^{d-1}$ and

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

6.2 Multiple Linear Placements with ODR

Theorem 3. *Multiple linear placements along with* ODR *algorithm on T_k^d results in linear load on edges.*

Proof. The analysis is conceptually similar to that of the previous section. Consider a multiple linear placement $P=P_1\cup P_2\cup\ldots\cup P_t$ in $T_k^d=(V,E)$ with ODR where, for some fixed constant t,

$$\begin{aligned}P_1 &= \{\vec{p}\mid p_1+p_2+\cdots+p_d\equiv 0\ (mod\ k)\}\\ P_2 &= \{\vec{p}\mid p_1+p_2+\cdots+p_d\equiv 1\ (mod\ k)\}\\ &\vdots\\ P_t &= \{\vec{p}\mid p_1+p_2+\cdots+p_d\equiv t-1\ (mod\ k)\}.\end{aligned}$$

Note that $\mid P \mid = tk^{d-1}$. As before, consider an edge $l\in E$ of the form

$$l=<(i_1,\ldots,i_s,\ldots,i_d),(i_1,\ldots,i_s+1,\ldots,i_d)>$$

and a pair of processors $\vec{p}$, $\vec{q}\in P$, which communicate using l. Since ODR algorithm is used, we must have

$$\vec{p}=(p_1,\ldots,\ p_s,\ i_{s+1},\ \ldots,\ i_d)$$

and

$$\vec{q}=(i_1,\ldots,\ i_{s-1},\ q_s,\ q_{s+1},\ldots,\ q_d)$$

with $p_s\le i_s\ (mod\ k)$ and $q_s\ge i_s+1\ (mod\ k)$, as well as $q_s-p_s\ (mod\ k)\le p_s-q_s\ (mod\ k)$. Since $\vec{p}$ and $\vec{q}$ are both in P, each must satisfy an equation among $P_1,P_2,\ldots,P_t$. Thus,

$$p_1+\cdots+p_s+i_{s+1}+\cdots+i_d\equiv 0\ or\ 1\ or\ \ldots\ or\ t-1\ (mod\ k) \quad (13)$$

and

$$i_1+\cdots+i_{s-1}+q_s+\cdots+q_d\equiv 0\ or\ 1\ or\ \ldots\ or\ t-1\ (mod\ k). \quad (14)$$

The number of solutions to (13) is no more than tk^{s-1}. Similarly, the number of solutions to (14) is no more than tk^{d-s}. Therefore, the total number of processor pairs communicating through l is bounded by t^2k^{d-1}, which is linear in $\mid P \mid$ for any constant t. □

7 Unordered Dimensional Routing (UDR)

We mentioned in Section 6 that ODR algorithm suffers from lack of fault tolerance since there is only one path between each pair of processors. In this section, we introduce *Unordered Dimensional Routing (UDR)*, which eliminates this problem. The algorithm is as follows: To route a packet from $\vec{p}=(p_1,\ p_2,\ldots,\ p_d)$ to $\vec{q}=(q_1,\ q_2,\ldots,\ q_d)$, both in P

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

As was the case 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., $\mid \mathcal{C}_{\vec{p}\to\vec{q}}^{UDR} \mid = s!$. Next, we show that UDR algorithm also results in linear load in edges.

7.1 Load Analysis for Linear Placements with UDR

For a linear placement P which uses UDR algorithm, the load on an edge l is

$$\mathcal{E}(l)=\sum_{\vec{p}\in P,\vec{q}\in P}\frac{\mid \mathcal{C}_{\vec{p}\to l\to\vec{q}}^{UDR}\mid}{\mid \mathcal{C}_{\vec{p}\to\vec{q}}^{UDR}\mid}.$$

Since there exist some pairs of processors for which $\mid \mathcal{C}_{\vec{p}\to\vec{q}}^{UDR}\mid >1$, we have,

$$\mathcal{E}(l)<\sum_{\vec{p}\in P,\vec{q}\in P}\mid \mathcal{C}_{\vec{p}\to l\to\vec{q}}^{UDR}\mid .$$

The upper bound on the righthand side of this inequality specifies the number of messages sent between pairs of processors which could "potentially" route their messages through l. Such processors can also use other paths that do not include l, since UDR algorithm provides multiple routing paths.

Theorem 4. *Given a linear placement*

$$P=\{\vec{p}\mid p_1+p_2+\cdots+p_d\equiv 0\ (mod\ k)\}$$

in $T_k^d=(V,E)$, Unordered Dimensional Routing Algorithm *results in linear load on edges.*

Proof. Without loss of generality, $l\in E$ is of the form

$$l=<(i_1,\ldots,i_s,\ldots,i_d),(i_1,\ldots,i_s+1,\ldots,i_d)>\ .$$

Suppose $\vec{p}$ and $\vec{q}\in P$ are two processors communicating through l. Our aim is to find an upper bound on the number of pairs of processors communicating through l. A moment of thought reveals that $\vec{p}$ and $\vec{q}$ must have either $p_j=i_j$ and q_j arbitrary or $q_j=i_j$ and p_j arbitrary, for $j\ne s$. This means the number of possible choices in dimension j is less than $2k$.

Hence, the total number of choices for all of the coordinates of $\vec{p}$ and $\vec{q}$ excluding s is less than $2^{d-1}k^{d-1}$. Since $\vec{p}$ and $\vec{q}$ are both in P, they satisfy

$$p_1 + \cdots + p_d \equiv 0 \ (mod\ k)$$

and

$$q_1 + \cdots + q_d \equiv 0 \ (mod\ k).$$

There are at most one solution pair for each one of $2^{d-1}k^{d-1}$ choices (as before, the coordinates in the sth dimension are restricted by the conditions $p_s \leq i_s \ (mod\ k)$, $q_s \geq i_s + 1 \ (mod\ k)$, and $q_s - p_s \ (mod\ k) \leq p_s - q_s \ (mod\ k)$). Therefore, the total number of processor pairs communicating through l is bounded by $2^{d-1}k^{d-1}$, i.e.,

$$\mathcal{E}_{max} < \max_{l \in E} \left\{ \sum_{\vec{p} \in P, \vec{q} \in P} | \ \mathcal{C}^{UDR}_{\vec{p} \to l \to \vec{q}} \ | \right\} < \ 2^{d-1}k^{d-1},$$

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

7.2 Multiple Linear Placements with UDR

Theorem 5. *Multiple linear placements along with UDR algorithm on T^d_k results in linear load on edges.*

Proof. We have $|P| = tk^{d-1}$. As before, consider a processor pair, $\vec{p}$, $\vec{q} \in P$, which communicate using l where

$$l = < (i_1, \ldots, i_s, \ldots, i_d), (i_1, \ldots, i_s + 1, \ldots, i_d) > .$$

The number of choices for processor pairs using l is strictly less than $2^{d-1}k^{d-1}$, as in the case of linear placements with UDR. Since there are t equations for each of $\vec{p}$ and $\vec{q}$, there are t^2 solutions for every one of $2^{d-1}k^{d-1}$ choices of pairs. Therefore, the number of pairs of processors communicating through l is less than $t^2 2^{d-1}k^{d-1}$, which is linear in $| P |$ for any fixed d and any constant t. □

8 Conclusion

We have considered communication in partially populated torus networks in terms of placements of processors and associated routing algorithms introduced by Blaum et al. [4], [5]. 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 placements with enough symmetries 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 tk^{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 combinatorial 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. A related question is on how much the uniformity condition on the placements can be relaxed and still have enough structure to be able to eliminate the dependence of the constant on d in Theorem 1.

Appendix

In the following proposition, we treat T^d_k as an undirected graph, rather than a directed one. Thus, each edge below translates into a pair of edges of the directed case.

Proposition 1. *Any subset P of nodes of the d-dimensional k-torus can be bisected by removing $O(k^{d-1})$ edges of T^d_k. In particular, any subgraph of T^d_k has bisection width $O(k^{d-1})$.*

Proof. Since T^d_k has only $O(k^{d-1})(= dk^{d-1})$ more edges than the d-dimensional k-ary array A^d_k, we can instead work with A^d_k.

Let $\mathbb{Z}^d$ denote the collection of all lattice points in the d-dimensional Euclidean space $\mathbb{R}^d$. The length of $\vec{x} \in \mathbb{R}^d$ is denoted by $\| \vec{x} \|$. A *hyperplane* $\mathcal{H}_t = \mathcal{H}_t(\vec{\eta})$ in $\mathbb{R}^d$ is defined by a unit vector $\vec{\eta} = \vec{\gamma}/\| \vec{\gamma} \|$ and a real number t, as the set of points $\vec{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ satisfying

$$\vec{\eta} \cdot \vec{x} = \eta_1 x_1 + \eta_2 x_2 + \cdots + \eta_d x_d = t \ .$$

As t ranges over $\mathbb{R}$, $\mathcal{H}_t$ sweeps $\mathbb{R}^d$ in the direction $(\pm)$ of $\vec{\eta}$. Suppose we pick γ to be a transcendental number (such as π or e) and define $\mathcal{H}_t$ by the unit vector $\vec{\eta}$ in the direction of $(1, \gamma, \gamma^2, \ldots, \gamma^{d-1})$. Then, there can be no two distinct points $\vec{a}, \vec{b} \in \mathbb{Z}^d$ which lie on $\mathcal{H}_t$ for a fixed t. For, otherwise,

$$(a_1 - b_1) + (a_2 - b_2)\gamma + \cdots + (a_d - b_d)\gamma^{d-1} = 0,$$

with not all $a_i - b_i = 0$, and this would contradict the fact that γ is transcendental.

The standard embedding of A^d_k in $\mathbb{R}^d$ is obtained by mapping each vertex $(a_1, a_2, \ldots, a_d)$ of A^d_k, where $0 \leq a_i < k$ for each i, to the point $\vec{a} = (a_1, a_2, \ldots, a_d) \in \mathbb{Z}^d$. An edge in A^d_k between two vertices $\vec{a} = (a_1, \ldots, a_i, \ldots, a_d)$ and $\vec{b} = (a_1, \ldots, a_i + 1, \ldots, a_d)$ is represented by the unit line segment in $\mathbb{R}^d$ parallel to the ith axis, having $\vec{a}$ and $\vec{b}$ as its endpoints. We identify P (nodes + edges) with its geometric embedding in $\mathbb{R}^d$ as a subgraph of A^d_k.

Consider the collection of hyperplanes $\mathcal{H}_t$ defined as above using a transcendental number γ in the range $1 < \gamma < \sqrt[d-1]{2}$ and $0 \leq t \leq (k-1)$. For t in this range, $\mathcal{H}_t$ sweeps A^d_k from the point $(0, 0, \ldots, 0)$ to the farthest point $(k-1, k-1, \ldots, k-1)$ in the direction of $\vec{\eta}$. Note that, by our choice of γ,

1. $1 < \gamma < \cdots < \gamma^{d-1} < 2$,
2. $0 < \eta_1 < \eta_2 < \cdots < \eta_d < 1$,
3. For any $r \geq 2$ and $i = 1, 2, \ldots, d$, $r\gamma^{i-1} \geq 2 > \gamma^{d-1}$ and, consequently, $r\eta_i > \eta_d$.

For a fixed value $t = t_0$, $\mathcal{H}_t$ crosses an edge between $\vec{a}$ and $\vec{b}$ of A^d_k if these two endpoints lie on opposite sides of the hyperplane:

$$\vec{a} \cdot \vec{\eta} < t_0 < \vec{b} \cdot \vec{\eta} = \vec{a} \cdot \vec{\eta} + \eta_i \ .$$

Without any loss of generality, assume that there is no vertex of A^d_k on $\mathcal{H}_{t_0}$. Since an edge in the embedding has unit length, the distance from $\vec{a}$ to $\mathcal{H}_{t_0}$ (and also from $\vec{b}$ to $\mathcal{H}_{t_0}$) is strictly less than 1. The distance between $\vec{x}$ and the hyperplane $\mathcal{H}_{t_0}$ is given by

$$| \ \vec{x} \cdot \vec{\eta} - t_0 \ | \ .$$

Therefore, the number S of edges crossed by $\mathcal{H}_{t_0}$ is no larger than the number of solutions $\vec{a} \in A^d_k$ of

$$t_0 - 1 < \vec{a} \cdot \vec{\eta} < t_0 \quad and \quad t_0 < \vec{a} \cdot \vec{\eta} + \eta_i < t_0 + 1 \ , \tag{15}$$

summed over $i = 1, 2, \ldots, d$. The two sets of inequalities in (15) are in turn equivalent to

$$t_0 - \eta_i < \vec{a} \cdot \vec{\eta} < t_0 \ . \tag{16}$$

Let

$$S_i = \{ \vec{a} \in A^d_k \mid t_0 - \eta_i < \vec{a} \cdot \vec{\eta} < t_0 \ \} \ .$$

Then,

1. $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_d$, (since $\eta_1 < \eta_2 < \cdots < \eta_d$);
2. $S = \mid S_1 \mid + \mid S_2 \mid + \cdots + \mid S_d \mid$.

We claim that $\mid S_d \mid \leq 2dk^{d-1}$ and, therefore, $S \leq 2dk^{d-1}$.

Let W be a subset of nodes of A_k^d. Define the *discrepancy* $D(W) = d -$ number of indices j such that the jth coordinate of each element in W is identical. For example, when $W = \{(0,0,0),(0,0,2),(1,0,0),(1,0,2)\}$, $D(W) = 2$. Let $N[d,k]$ be the maximum cardinality collection $\mathcal{N}$ of $\vec{a} \in A_k^d$ such that any subset W of three elements has $D(W) \geq 2$. Grouping the elements in $\mathcal{N}$ according to their first coordinates, we obtain the bound

$$N[d,k] \leq kN[d-1,k]$$

for $d > 2$. Also, by the pigeonhole principle, $N[2,k] \leq 2k$. Therefore, $N[d,k] \leq 2k^{d-1}$. Now, by way of contradiction, assume that $\mid S_d \mid > 2k^{d-1}$. We have just shown with our count of $\mathcal{N}$ that S_d must contain a set W consisting of three vectors such that $D(W) = 1$, i.e., three vectors which differ only in a single coordinate i. In particular, S_d has two solutions $\vec{a} = (a_1, \ldots, a_i, \ldots, a_d)$ and $(a_1, \ldots, a_i + r, \ldots, a_d)$ with $r \geq 2$ (this is guaranteed since we have three distinct numbers in the ith coordinates in W). Using (16), this implies

$$t_0 - \eta_d < \vec{a} \cdot \vec{\eta} < t_0 \quad and \quad t_0 - \eta_d < \vec{a} \cdot \vec{\eta} + r\eta_i < t_0,$$

which combine to give $t_0 - \eta_d < t_0 - r\eta_i$ or, equivalently, $r\eta_i < \eta_d$. However, by our choice of γ, $r\eta_i \geq \eta_d$. This contradiction proves that $\mid S \mid \leq 2dk^{d-1}$. Therefore, for any value of t, $0 < t < k-1$, the hyperplane $\mathcal{H}_t$ crosses no more than $2dk^{d-1}$ edges of A_d^k.

Now, we can prove that P (or P minus an element if $\mid P \mid$ is odd) can be divided into two equal size subsets by removing no more than $2dk^{d-1}$ edges of A_d^k. This division is obtained by using the fact that $\mathcal{H}_t$ contains at most one element of A_d^k and, therefore, of P for any t. Thus, the number of elements of P which are on the origin side of $\mathcal{H}_t$ goes up by units of one with increasing t and is equal to $\frac{1}{2} \mid P \mid$ for some $t = t_0$. □

ACKNOWLEDGEMENT

We would like to thank the referee who made valuable suggestions on simplifying our earlier asymptotic notation. The first author was supported in part by a fellowship from Izmir Institute of Technology, Izmir, Turkey.

REFERENCES

[1] R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith, "The Tera Computer System," *ACM Supercomputing*, 1990.

[2] M.C. Azizoglu and Ö. Egecioglu, "Lower Bounds on Communication Loads and Optimal Placements in Torus Networks," *Proc. IEEE 1998 Int'l Parallel Processing Symp./Symp. Parallel and Distributed Processing Symp.*, pp. 460–464, Mar. 1998.

[3] M.M. Bae and B. Bose, "Resource Placement in Torus-Based Networks," *IEEE Trans. Computers*, vol. 46, no. 10, pp. 1,083–1,092, 1997.

[4] M. Blaum, J. Bruck, G.D. Pifarré, and J.L. Sanz., "On Optimal Placements of Processors in Tori Networks," *Proc. Eighth IEEE Symp. Parallel and Distributed Processing*, pp. 552–555, Oct. 1996.

[5] M. Blaum, J. Bruck, G.D. Pifarré, and J.L. Sanz., "On Optimal Placements of Processors in Fault-Tolerant Tori Networks," preprint, 1997.

[6] B. Bose, R. Broeg, Y. Kwon, and Y. Ashir, "Lee Distance and Topological Properties of k-ary n-cubes," *IEEE Trans. on Computers*, vol. 44, no. 8, pp. 1,021–1,030, Aug. 1995.

[7] Y-C. Tseng, T-H. Lin, S.K.S. Gupta, and D.K. Panda, "Bandwidth-optimal Complete Exchange on Wormhole-routed 2D/3D Torus Networks: A Diagonal-Propagation Approach," *IEEE Trans. Parallel and Distributed Systems*, vol. 8, no. 4, pp. 380–396, Apr. 1997.

[8] A. Gerbessiotis and L.G. Valiant, "Direct Bulk-Synchronous Parallel Algorithms," *J. Parallel and Distributed Computing*, vol. 22, pp. 251–267, 1994.

[9] F.T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays· Trees· Hypercubes.* San Mateo, Calif.: Morgan Kaufmann, 1992.

[10] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes.* Amsterdam: North Holland, 1977.

[11] L.M. Ni and P.K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *Computer*, vol. 26, no. 2, pp. 62–76, Feb. 1993.

[12] F. Pitteli and D. Smitley, "Analysis of a 3D Toroidal Network for a Shared Memory architecture," *Proc. Supercomputing '88*, pp. 35–41, Nov. 1988.

[13] S. Rao, T. Suel, T. Tsantilas, and M. Goudreau, "Efficient Communication Using Total Exchange," *Proc. Int'l Parallel Processing Symp.'95*, pp. 544–550, 1995.

[14] J.F. Sibeyn, "Routing on Triangles, Tori and Honeycombs," *Int'l J. Foundations of Computer Science*, vol. 8, no. 3, pp. 269–287, 1997.

[15] L.G. Valiant, "A Bridging Model for Parallel Computation," *Comm. ACM*, vol. 33, no. 8, pp. 103–111, 1990.