On Optimal Placements of Processors in Fault-Tolerant Tori Networks

Mario Blaum[1] Jehoshua Bruck[2] Gustavo D. Pifarré[3] Jorge L. C. Sanz[3]

Abstract

Two and three dimensional k-tori are among the most used topologies in the design of new parallel computers. Traditionally (with the exception of the Tera parallel computer), these networks have been used as *fully-populated* networks, in the sense that *every* routing node in the topology is subjected to message injection. However, fully-populated tori and meshes exhibit a theoretical throughput which degrades as the network size increases. In addition, the performance of those networks is sensitive to link faults. In contrast, multistage networks (that are partially populated) scale well with the network size. We propose to add slackness in fully-populated tori by reducing the number of processors and we study optimal fault-tolerant routing strategies for the resulting interconnections.

The key concept that we study is the average link load in an interconnection network with a given *placement* and a *routing algorithm*, where a placement is the *subset* of the nodes in the interconnection network that are attached to processors. Reducing the load on the links by the choice of a placement and a routing algorithm leads to improvements in both the performance and the fault tolerance of the communication system.

Our main contribution is the construction of optimal placements for 2 and 3-dimensional k-tori networks and their corresponding routing algorithms. Those placements yield a linear (in the number of processors) link load and are of optimal size.

Key Words: parallel computing, fault-tolerant routing, tori networks, partially populated networks, link load.

[1]IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, `blaum@almaden.ibm.com`.

[2]California Institute of Technology, MC 136-93, Pasadena, CA 91125, `bruck@paradise.caltech.edu`. Supported in part by the NSF Young Investigator Award CCR-9457811, by the Sloan Research Fellowship and by a grant from the IBM Almaden Research Center, San Jose, California.

[3]Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Buenos Aires, Argentina, `{pifarre,sanz}@lasvm1.vnet.ibm.com`. Supported in part by the Universidad de Buenos Aires through the research project EX-155 and through the scholarships program Rene Hugo Thalman.

1 Introduction

Parallel computers have received great research attention in recent years. A topic of continuing studies is interconnection networks [8][15] and routing techniques [9][13]. It is well known that enhancements in interprocessor communication technology are central to the improved performance of parallel computers. Topology of the network, oblivious and adaptive routers, processor-network interfaces, and tolerance to faults are some of the key concerns of parallel computer architects.

In particular, hypercubes, meshes, tori, cube-connected cycles, fat-trees, shuffle-exchanges, and the wide class of multistage interconnection networks have been extensively used among many others [11]. Although the hypercube has been a very popular design, in fact 2 and 3-dimensional k-tori are two of the most used topologies in the designs of new parallel computers. The renewed interest in these topologies stems both from their low degree and their scalability. Also, these networks have topologies amenable to natural 2 and 3 dimensional layouts and seem to adapt well to the presence of faults [3][4][5].

Throughput is a widely used measure of network performance, namely, the ability of the network to manage the maximum possible message traffic. A well-known bound on the throughput of a network is based on the bisection of the network and its message-traffic characteristics [2][6][10]. The bisection bandwidth of a network is the minimum number of links that must be cut in order to divide the network into about two equal halves. Multistage networks accomplish their throughput by providing a significant surplus of routing nodes in comparison to points where processors inject messages into the network. A multistage network with $k \times k$ switches (routing nodes) and $\log_k(n)$ stages serves n injection points, has bisection bandwidth of $2n$ (when directed links are considered) and is utilizing $n \log_k(n)$ routing nodes. In this respect, multistage networks can be regarded as *partially-populated* networks, in the sense that only a subset of routing nodes are targets of message injection by processors.

On the other hand, networks such as tori, meshes, and hypercubes have been designed and/or built where the number of routing nodes is equal to the number of processors [7]. Hence, these networks have been used as *fully-populated* networks, in the sense that *every* routing node in the topology is subjected to message injection. Fully-populated tori and meshes exhibit a theoretical throughput which degrades as the network size increases. Note that the bisection bandwidth of a d-dimensional k-torus is $4k^{d-1}$ when directed links are considered. In a fully-populated torus, there would be $k^{2d}/2$ messages passing through the bisection assuming that every pair of processors is communicating simultaneously. This means that there is an edge in the bisection with load at least $k^{d+1}/8$. This is in comparison to a load of n in an n processor multistage network. In fact, the increased load in the case of tori networks has led researchers in [1] and [14] to consider a smaller number of processors when compared to routing nodes in a 3-dimensional mesh type network.

The concept of the load on a link in an interconnection network is crucial also with regard to the reliability of the routing algorithm. By minimizing the maximum load per link we also reduce the sensitivity of the routing algorithm to link faults. In addition, in the case of a link fault, we can adapt the routing algorithms by using alternative minimal paths. Namely, reducing the load on the links by the choice of a placement and a routing algorithm leads to improvements in both the performance and the reliability of the communication system. The main question we address is this paper is: can we achieve a load in a torus similar to the one in a multistage interconnection network? Namely, can we achieve a linear load in partially populated tori networks?

Introducing slackness in fully-populated tori, i.e., reducing the number of processors, and studying

optimal fault-tolerant routing strategies for the resulting interconnections are the central subjects of this paper. The key concept that we study is a *placement* of the processors in a network. Namely, a placement is the *subset* of the nodes in the interconnection network that is attached to processors. In addition, given a placement P we need to define a routing method between arbitrary pairs of processors in P. We assume that the routing consists of only minimal length paths. Our goal is to find P, a placement of processors, in a d-dimensional k-torus, such that $|P| = k^i$ with $i \leq d$ is as large as possible while the maximal load on the links is linear in $|P|$ (like in the case of multistage networks). This linear link load makes the routing algorithms more robust to faults. Our main contribution is the construction of optimal placements of size k and k^2 and their corresponding routing algorithms for the cases $d = 2$ and $d = 3$, respectively.
In Section 2, we give some formal definitions and notations, introduce a simple lower bound and present an optimal placement together with a routing algorithm for the case of a 2-dimensional tori. In Section 3 we present an optimal placement and routing algorithm for 3-dimensional tori networks.

2 Preliminaries and the 2-Dimensional Case

In this section we formalize the processor placement problem and present an optimal solution for the 2-dimensional case as well as a lower bound. The optimal placement for the three-dimensional case is presented in the next section.

2.1 Problem Definition

We model an interconnection network of a parallel computer as a directed graph where the nodes represent switches and the directed edges represent links connecting the switches. In particular, we are interested in tori networks.

Definition 2.1 A *d-dimensional k-torus* is a graph with k^d nodes. The nodes are identified with the d-tuples $\vec{a} = (a_{d-1}, a_{d-2}, \ldots, a_0)$, where each element a_i in the d-tuple is in Z_k, the integers modulo k. Two nodes $\vec{a} = (a_{d-1}, a_{d-2}, \ldots, a_0)$ and $\vec{b} = (b_{d-1}, b_{d-2}, \ldots, b_0)$ are connected by a link, if and only if there is an i, $0 \leq i \leq d-1$, such that $a_j = b_j$ for $j \neq i$ and $a_i \equiv b_i \pm 1 \pmod{k}$. Notice that the links connect each node of the torus with its neighboring $2d$ nodes.

We assume that a *subset* of the nodes in the interconnection network has processors attached to them. We call this subset a *placement*. Formally,

Definition 2.2 A *placement* of processors in a *d-dimensional k-torus* is a subset of nodes, $P = \{\vec{a} \mid \vec{a} \in \{0, 1, \ldots, k-1\}^d\}$.

Given a placement P, we need to define the routing algorithm between arbitrary pairs of processors in P. In this paper we will assume that the routing occurs only through minimal length paths.

Definition 2.3 Let P be a placement. Let $\vec{p}$ and $\vec{q}$ be a pair of processors in P. The *routing algorithm* to transmit packets from $\vec{p}$ to $\vec{q}$ will use a set of minimal length (shortest) paths, by choosing them uniformly at random from this set. Let $\mathcal{C}_{\vec{p} \to \vec{q}}$ be the set of minimal length paths from $\vec{p}$ to $\vec{q}$ given by the routing algorithm. Let l be a link in the network, then denote by $\mathcal{C}_{\vec{p} \to l \to \vec{q}} \subseteq \mathcal{C}_{\vec{p} \to \vec{q}}$ the set of paths from $\vec{p}$ to $\vec{q}$ through link l .

The main goal of this paper is to find a "good" method for placing processors in tori interconnection networks. The key concept for defining the quality of a placement is the *load* of a link.

Definition 2.4 Given a placement P with a routing algorithm, the *load* of link l is:

$$\mathcal{E}(l) = \sum_{\vec{p} \in P, \vec{q} \in P} \frac{|\, \mathcal{C}_{\vec{p} \to l \to \vec{q}} \,|}{|\, \mathcal{C}_{\vec{p} \to \vec{q}} \,|} \tag{1}$$

We denote by $C(N, P)$ the maximum value of $\mathcal{E}(l)$ for a network N with a placement P. Namely,

$$C(N, P) = \max_{l \in N} \mathcal{E}(l).$$

Our goal is to find P, a placement of processors, in a d-dimensional k-torus N, such that $|P| = k^i$ with $i \leq d$ being as large as possible, while $C(N, P)$ is linear in $|P|$. This will provide a way to scale-up the size of the network. Our main contribution is the construction of optimal placements for the cases $d = 2$ and $d = 3$. In particular, we present placements that achieve the following lower bounds: for $d = 2$, $C(N, P) \geq |P|/4$ and for $d = 3$, $C(N, P) \geq |P|/6$.

2.2 A Lower Bound

In this section, we present a lower bound for $C(N, P)$, N being a d-dimensional k-torus.

Lemma 2.1 Let P be a placement in a d-dimensional k-torus N together with a routing algorithm. Then $C(N, P) \geq (|P| - 1)/(2d)$.

Proof: Assume that a given processor is sending packets to all the other $|P| - 1$ processors. Those packets are distributed over the $2d$ directed links adjacent to that processor. Namely there exists an edge l such that,

$$\mathcal{E}(l) \geq \frac{|P| - 1}{2d} \tag{2}$$

Hence $C(N, P) \geq (|P| - 1)/(2d)$. □

Notice that an optimal placement in a d-dimensional k-torus cannot contain more than k^{d-1} nodes. The reason is that if we have $k^{d-1} + 1$ nodes in the placement then at least two nodes will be on the same cycle (they will have $d - 1$ identical coordinates). Namely, by a similar argument to the the one in Lemma 2.1 there will be a link with load higher than the average load. Hence, in the rest of the paper we will focus on constructing optimal load placements with maximum size (i.e. k^{d-1}) for the cases $d = 2$ and $d = 3$.

2.3 The Two-Dimensional Case

In this section we describe an optimal placement for the 2-dimensional k-torus. The placement is optimal in the sense that it achieves the lower bound for $C(N, P)$ given by Lemma 2.1.
The placement that we use is $P = \{(i, i) \mid 0 \leq i \leq k - 1\}$, so, $|P| = k$. Figure 1 shows this placement in a 2-dimensional 5-torus. The routing algorithm associated with P is very simple. The

dimensions are corrected in any order, but if the message begins correcting one dimension, it must finish that dimension before correcting the other one. Figure 2 shows the paths allowed by the algorithm between the processors (3,3) and (0,0) and between processors (2,2) and (1,1).
Let us be more specific. Given two integers m and n, the cyclic distance between m and n modulo k is given by

$$\min\{m - n \pmod k \; ; \; n - m \pmod k\}.$$

The Lee distance between two nodes $\vec{a}$ and $\vec{b}$ in the d-dimensional k-torus, denoted $d_L(\vec{a}, \vec{b})$, is the sum of the cyclic distances between the coordinates [12]. The Lee distance represents the length of the shortest path in the d-dimensional k-torus between two nodes.
Consider now the 2-dimensional k-torus. Assume that we have two processors in P, for instance $\vec{a} = (a, a)$ and $\vec{b} = (b, b)$, and we want to send a packet from $\vec{a}$ to $\vec{b}$. We choose a dimension at random, say dimension 0, and we correct that dimension first. This means, we move from (a, a) to (a, b) along a path with minimal cyclic distance. We then move from (a, b) to (b, b), also along a path with minimal cyclic distance.
We observe that if k is odd, the algorithm above gives only two minimal paths between two different processors. If k is even, we may have more than two minimal paths since, when the cyclic distance in one dimension is exactly $k/2$, we have more than one choice to correct that distance. Stating formally the discussion above, we have the following algorithm:

Algorithm 2.1 Consider the placement P in a 2-dimensional k-torus. Assume that we want to send a packet from processor $\vec{a}$ to processor $\vec{b}$. Then proceed as follows:

1. Correct either dimension 0 or dimension 1 through the shortest cyclic distance.
2. Correct the dimension 0 or 1 not corrected in the previous step, also through the shortest cyclic distance.

We want to show next that Algorithm 2.1 is optimal in the sense that $C(N, P) = (|P| - 1)/4$, i.e., the lower bound on $C(N, P)$ given by Lemma 2.1 is achieved. Without loss of generality, consider a link l in dimension 0. As stated before, l is contained in a unique cycle and there is exactly one processor in the cycle to which the link belongs. Let $\vec{r}$ be such a processor. We denote by s the cyclic distance between one of the two nodes that are the endpoints of l and $\vec{r}$, whichever is smaller. The next lemma gives the value of $\mathcal{E}(l)$ for such a link.

Lemma 2.2 Consider placement P in the 2-dimensional k-torus. Let l be a link in, say, dimension 0, $\vec{r} \in P$ is in the cycle to which l belongs, and l is at (cyclic) distance s from $\vec{r}$. Then, the load on l when Routing Algorithm 2.1 is implemented is given by

$$\mathcal{E}(l) \quad = \quad \frac{k-1}{4} - \frac{s}{2} \tag{3}$$

Proof: Routing Algorithm 2.1 uses link l only for packets in which either the source or the destination are $\vec{r}$. So, Equation (1) becomes:

$$\mathcal{E}(l) = \sum_{\vec{q}\in P} \frac{|\ \mathcal{C}_{\vec{r}\to l\to\vec{q}}\ |}{|\ \mathcal{C}_{\vec{r}\to\vec{q}}\ |} \tag{4}$$

Assume that k is odd. Then, between any pair of nodes, Algorithm 2.1 allows exactly two paths, and Equation (4) becomes

$$\mathcal{E}(l) = \frac{1}{2}\sum_{\vec{q}\in P} |\ \mathcal{C}_{\vec{r}\to l\to\vec{q}}\ | \tag{5}$$

Thus, we have to determine the number of paths containing l with source $\vec{r}$. Without loss of generality, assume that $\vec{r} = (0,0)$ and l has end-points $(s,0)$ and $(s+1,0)$, where $s \leq (k-1)/2$. There cannot be a path between (0,0) and (j,j) containing l with $(k+1)/2 \leq j \leq k-1$, since such a path would not be minimal. Thus, we are left with $(k-1)/2$ possible paths. We also have to exclude the s processors (j,j) with $1 \leq j \leq s$: there cannot be a minimal path between $(0,0)$ and (j,j) containing l. We are left with $((k-1)/2) - s$ processors, which is the value of $\sum_{\vec{q}\in P} |\mathcal{C}_{\vec{r}\to l\to\vec{q}}|$. Replacing this value in Equation (5), we obtain Equation (3).
If k is even, a similar and slightly more complex argument gives also Equation (3). $\square$

Corollary 2.1 Consider the placement P in the 2-dimensional k-torus with Routing Algorithm 2.1. Then $C(N,P) = (k-1)/4$.

Proof: Simply take $s = 0$ in Lemma 2.2. $\square$

3 The Three Dimensional Case

In this section, we will present an optimal routing algorithm for a particular regular placement in the 3-dimensional k-torus which is an extension of the placement considered before (Figure 3 shows this placement for the 3-dimensional 5-torus). The first option to consider for the routing algorithm in the 3-dimensional case is the straightforward extension of Routing Algorithm 2.1. However, it turns out that this approach results in a link load that is not optimal. The key observation is that a path between two processors that goes through a third processor, say p, increases the load on the links adjacent to p beyond the optimal link load (this follows from the lower bound argument in Lemma 2.1). Now notice that this might happen in the 3-dimensional case since any two processors that are not in the same plane determine a cube, six minimal paths can be used (see the paths between processors (0,0,0) and (2,1,1) in Figure 4). However, some of those paths may pass through processors (see the paths between processors (3,3,4) and (4,4,2) in Figure 4). We will overcome this difficulty by presenting a routing algorithm for the shifted diagonal placement (defined below) that avoids going through processors and results in an optimal link load.

Definition 3.1 We call a placement P in a 3-dimensional k-torus a *shifted diagonal* placement, if

$$P \;=\; \{\vec{a} = (a_2, a_1, a_0) \;:\; a_0 + a_1 + a_2 \equiv 0 \ (\mathrm{mod}\ k)\}. \tag{6}$$

Example 3.1 Figure 3 depicts a shifted-diagonal placement for the 3-dimensional 5-torus.

Next, we will present an optimal minimal routing algorithm for the shifted diagonal placement. Essentially, the algorithm starts by correcting one of possible three dimensions in the direction of minimal cyclic distance. After it corrects this first dimension, it corrects a second one (also in the direction of minimal cyclic distance) as long as by doing so, it does not encounter a processor. The key in the algorithm is that it has the property that it will not encounter a processor in at least one of the remaining two dimensions. Finally, the algorithm corrects the remaining dimension. Namely, the following algorithm has the property that any pair of processors in the network can communicate without passing over another processor.

Algorithm 3.1 Assume that we have a 3-dimensional k-torus with a shifted diagonal placement, and we want to route a message from processor $\vec{a} = (a_2, a_1, a_0)$ to processor $\vec{b} = (b_2, b_1, b_0)$. Let I be the set $I = \{i \in \{0,1,2\} : a_i \neq b_i\}$ (notice that, either $|I| = 2$ or $|I| = 3$). Then proceed as follows (whenever we correct a dimension, we do it through the path of minimal cyclic distance):

Choose $j \in I$ and correct dimension j.
If $|I| = 2$ then correct dimension $l \in I - \{j\}$ and stop.
Else, choose $i \in I - \{j\}$.
    If, when correcting dimension i, the path does not pass over a processor,
        then correct dimension i.
        Correct dimension $t \in I - \{j, i\}$ and stop.
    Else, correct dimension $t \in I - \{j, i\}$.
        Correct dimension i and stop.

Figure 4 depicts the paths allowed by the algorithm in two different cases. Between processors (0,0,0) and (2,1,2) the messages can take any of the six minimal paths between them. In contrast, between processors (3,3,4) and (4,4,2), only four paths are allowed. If the message corrects dimension 2 first and then dimension 0, it will pass through a processor. The same will happen if dimension 1 is corrected first followed by dimension 0. Thus, these two paths are discarded. Our goal is now to prove that Routing Algorithm 3.1 allows for any pair of processors in the network to communicate without passing through another processor.

Theorem 3.1 A message between any pair of processors in a 3-dimensional k-torus with shifted diagonal placement routed using Algorithm 3.1, never passes through another processor.

Proof: Assume that we want to route a message from $\vec{r}$ to $\vec{a}$, $\vec{r}, \vec{a} \in P$, P defined by (6). Without loss of generality, assume that $\vec{r} = (0,0,0)$ and $\vec{a} = (a_2, a_1, a_0)$, with $0 \leq a_0, a_1, a_2 \leq k-1$. If $a_i = 0$ for some $i \in \{0,1,2\}$, then the message cannot pass over another processor, since $\vec{r}$ and $\vec{a}$ are in the same plane, and the algorithm behaves exactly as Algorithm 2.1.

So, assume that $a_i \neq 0$ for $0 \leq i \leq 2$. Also without loss of generality, assume that we first correct the x-dimension, i.e., $(0,0,0)$ is first routed to $(0,0,a_0)$ through minimal cyclic distance. By the definition of P, there are processors in $(k-a_0,0,a_0)$ and in $(0,k-a_0,a_0)$. We have to show that the message does not pass over any of these two processors, so assume that it does. There are three cases, each one of them leading to a contradiction:

$0 < a_1, a_2 \leq k/2$: Therefore, $a_0 + a_1 + a_2 = k$. Also, the y dimension is corrected through the path

$$(0,0,a_0) \rightarrow (0,1,a_0) \rightarrow \ldots \rightarrow (0,a_1,a_0),$$

and, since, the message goes through $(0,k-a_0,a_0)$, in particular, $k-a_0 < a_1$. Thus, $k-a_0 = a_1+a_2 > a_1 > k-a_0$, a contradiction.

$0 < a_1 \leq k/2$ and $k/2 < a_2 \leq k-1$ or $0 < a_2 \leq k/2$ and $k/2 < a_1 \leq k-1$: Without loss of generality, assume that $0 < a_1 \leq k/2$ and $k/2 < a_2 \leq k-1$. Since $0 < a_1 \leq k/2$, like in the previous case, we obtain that $k-a_0 < a_1$. On the other hand, since $k/2 < a_2 \leq k-1$, the z dimension is corrected through the path

$$(0,0,a_0) \rightarrow (k-1,0,a_0) \rightarrow \ldots \rightarrow (a_2,0,a_0),$$

and, since, the message goes through $(k-a_0,0,a_0)$, in particular, $k-a_0 > a_2$. Thus, $k/2 < a_2 < k-a_0 < a_1 \leq k/2$, a contradiction.

$k/2 < a_1, a_2 \leq k-1$: As in the previous case, this implies that $a_1, a_2 < k-a_0$. Also, $a_0+a_1+a_2 = 2k$. Since $a_1 < k-a_0$, $a_2 < k-a_0$ and $-k < -(k-a_0)$, adding these three inequalities, we obtain $a_1+a_2-k < k-a_0$. But $a_1+a_2-k = k-a_0$, a contradiction.

This completes the proof. □

Our next goal is finding the load per link with Algorithm 3.1, similarly to the 2-dimensional case. Given a link l, there is a unique processor $\vec{r}$ in the cycle to which l belongs with the diagonal shifted placement. Then, we call s the distance between l and $\vec{r}$, and by this, again, we mean the closest cyclic distance from one of the endpoints of l to $\vec{r}$. The next theorem is the main result in this section.

Theorem 3.2 Let l be a link at distance s from a processor in a 3-dimensional k-torus with a shifted diagonal placement. Then the load on l when Routing Algorithm 3.1 is implemented is given by

$$\mathcal{E}(l) = \frac{k^2-1}{6} - \frac{s(s+1)}{2}.$$

Proof: We will only consider the case in which k is odd, the case k even is similar and will be omitted.
Without loss of generality, assume that the link l has endpoints $(0,0,s)$ and $(0,0,s+1)$ and that $0 \leq s \leq (k-1)/2$. We will prove the theorem by finding the right hand side of (1). We will consider ten cases:

1.

$$S_1 = \sum_{s+1 \le i \le (k-1)/2} \frac{|\mathcal{C}_{(0,0,0)\to l\to(0,k-i,i)}|}{|\mathcal{C}_{(0,0,0)\to(0,k-i,i)}|}$$

In this case, there is only one path from (0,0,0) to $(0,k-i,i)$, $s+1 \le i \le (k-1)/2$, through l: the message is routed from (0,0,0) to $(0,0,i)$, and from $(0,0,i)$ to $(0,k-i,i)$. Thus, $|\mathcal{C}_{(0,0,0)\to l\to(0,k-i,i)}| = 1$. A second path from (0,0,0) to $(0,k-i,i)$ is obtained by routing the message from (0,0,0) to $(0,k-i,0)$ first, and then from $(0,k-i,0)$ to $(0,k-i,i)$. Thus, $|\mathcal{C}_{(0,0,0)\to(0,k-i,i)}| = 2$. Counting the number of terms added in S_1, we obtain

$$S_1 = \frac{1}{2}\left(\frac{k-1}{2} - s\right) = \frac{1}{4}(k-2s-1).$$

2.

$$S_2 = \sum_{s+1 \le i \le (k-1)/2} \frac{|\mathcal{C}_{(0,0,0)\to l\to(k-i,0,i)}|}{|\mathcal{C}_{(0,0,0)\to(k-i,0,i)}|}$$

This case is analogous to the previous one, thus,

$$S_2 = \frac{1}{4}(k-2s-1).$$

3.

$$S_3 = \sum_{\substack{s+1 \le i \le ((k-1)/2)-1 \\ 1 \le j \le ((k-1)/2)-i}} \frac{|\mathcal{C}_{(0,0,0)\to l\to(k-i-j,j,i)}|}{|\mathcal{C}_{(0,0,0)\to(k-i-j,j,i)}|} \tag{7}$$

There is only one path from (0,0,0) to $(k-i-j,j,i)$, $s+1 \le i \le ((k-1)/2)-1$, $1 \le j \le ((k-1)/2)-i$, through l: the message is routed from (0,0,0) to $(0,0,i)$, from $(0,0,i)$ to $(0,j,i)$, and from $(0,j,i)$ to $(k-i-j,j,i)$. Notice that, in the second step, it cannot be routed from $(0,0,i)$ to $(k-i-j,0,i)$, since $i+j \le (k-1)/2$, thus $k-i-j \ge (k+1)/2$ and the message would be routed through the path

$$(0,0,i)\to(k-1,0,i)\to\ldots\to(k-i,0,i)\to\ldots\to(k-i-j,0,i).$$

But $(k-i,0,i) \in P$, contradicting Algorithm 3.1. Thus, $|\mathcal{C}_{(0,0,0)\to l\to(k-i-j,j,i)}| = 1$ in (7). Similarly, we can see that $|\mathcal{C}_{(0,0,0)\to(k-i-j,j,i)}| = 4$ in (7). Counting the number of terms that are added in (7), we obtain

$$\begin{aligned} S_3 &= \frac{1}{4}\,\frac{\left(\frac{k-1}{2} - s - 1\right)\left(\frac{k-1}{2} - s\right)}{2} \\ &= \frac{1}{32}(k^2 - 4ks - 4k + 4s^2 + 8s + 3) \end{aligned}$$

4.

$$S_4 = \sum_{\substack{s+1\le i\le((k-1)/2)-1\\(k+1)/2\le j\le k-i-1}} \frac{|\mathcal{C}_{(0,0,0)\to l\to(k-i-j,j,i)}|}{|\mathcal{C}_{(0,0,0)\to(k-i-j,j,i)}|}$$

This case is analogous to the previous one, thus,

$$S_4 = S_3 = \frac{1}{32}(k^2 - 4ks - 4k + 4s^2 + 8s + 3)$$

5.

$$S_5 = \sum_{\substack{s+1\le i\le((k-1)/2)-1\\((k+1)/2)-i\le j\le(k-1)/2}} \frac{|\mathcal{C}_{(0,0,0)\to l\to(k-i-j,j,i)}|}{|\mathcal{C}_{(0,0,0)\to(k-i-j,j,i)}|} \tag{8}$$

In this case, we can see that there are two possible paths from (0,0,0) to $(k-i-j,j,i)$ through l (we omit the details), thus, $|\mathcal{C}_{(0,0,0)\to l\to(k-i-j,j,i)}| = 2$ in (8). Similarly, $|\mathcal{C}_{(0,0,0)\to(k-i-j,j,i)}| = 6$ in (8). Finally, a counting argument gives

$$\begin{aligned} S_5 &= \frac{1}{3}\left(\left(\frac{k-1}{2}-s\right)\left(\frac{k-1}{2}\right) - \frac{\left(\frac{k-1}{2}-s\right)\left(\frac{k-1}{2}-s-1\right)}{2}\right) \\ &= \frac{1}{24}(k^2 - 4s^2 - 4s - 1). \end{aligned}$$

6.

$$S_6 = \sum_{\substack{s+1\le i\le(k-1)/2\\k+1-i\le j\le k-1}} \frac{|\mathcal{C}_{(0,0,0)\to l\to(k-i-j,j,i)}|}{|\mathcal{C}_{(0,0,0)\to(k-i-j,j,i)}|} \tag{9}$$

In this case, we can verify that $|\mathcal{C}_{(0,0,0)\to l\to(k-i-j,j,i)}| = 2$ and $|\mathcal{C}_{(0,0,0)\to(k-i-j,j,i)}| = 4$ in (9). This gives,

$$\begin{aligned} S_6 &= \frac{1}{2}\left(\left(\frac{k-1}{2}-s\right)s + \frac{\left(\frac{k-1}{2}-s\right)\left(\frac{k-1}{2}-s-1\right)}{2}\right) \\ &= \frac{1}{16}(k^2 - 4k - 4s^2 + 4s + 3). \end{aligned}$$

7.

$$S_7 = \sum_{\substack{1\le i\le s\\s+1\le j\le(k-1)/2}} \frac{|\mathcal{C}_{(0,k-i,i)\to l\to(k-j,0,j)}|}{|\mathcal{C}_{(0,k-i,i)\to(k-j,0,j)}|} \tag{10}$$

In this case, we can verify that $|\mathcal{C}_{(0,k-i,i)\to l\to(k-j,0,j)}| = 1$ and $|\mathcal{C}_{(0,0,0)\to(k-i-j,j,i)}| = 4$ in (10), giving

$$S_7 = \frac{1}{4}\left(\frac{k-1}{2} - s\right)s = \frac{1}{8}(ks - 2s^2 - s).$$

8.

$$S_8 = \sum_{\substack{1\le i\le s \\ s+1\le j\le (k-1)/2}} \frac{|\mathcal{C}_{(k-i,0,i)\to l\to(0,k-j,j)}|}{|\mathcal{C}_{(0,k-i,i)\to(k-j,0,j)}|}$$

This case is analogous to the previous one, thus,

$$S_8 = \frac{1}{8}(ks - 2s^2 - s).$$

9.

$$S_9 = \sum_{\substack{1\le i\le s \\ (k+1)/2\le j\le ((k-1)/2)+i}} \frac{|\mathcal{C}_{(0,k-i,i)\to l\to(k-j,0,j)}|}{|\mathcal{C}_{(0,k-i,i)\to(k-j,0,j)}|} \tag{11}$$

In this case, $|\mathcal{C}_{(0,k-i,i)\to l\to(k-j,0,j)}| = 1$ and $|\mathcal{C}_{(0,0,0)\to(k-i-j,j,i)}| = 6$ in (11), giving

$$S_9 = \frac{1}{6}\left(\frac{s(s+1)}{2}\right) = \frac{1}{12}(s^2 + s).$$

10.

$$S_{10} = \sum_{\substack{1\le i\le s \\ (k+1)/2\le j\le ((k-1)/2)+i}} \frac{|\mathcal{C}_{(k-i,0,i)\to l\to(0,k-j,j)}|}{|\mathcal{C}_{(0,k-i,i)\to(k-j,0,j)}|}$$

This case is analogous to the previous one, thus,

$$S_{10} = \frac{1}{12}(s^2 + s).$$

The right hand side of (1) is a sum over all possible paths between pairs of processors through l. We can see that either the source or the destination should be in the plane $z = 0$, otherwise the path from one to the other given by Algorithm 3.1 cannot pass through l. It can be verified that all possible paths containing l are considered in the sums $S_1, S_2, \ldots, S_{10}$. Adding $S_1 + S_2 + \ldots + S_{10}$, we obtain the right hand side of (1), proving the theorem. □

In particular, if we take $s = 0$ in Theorem 3.2, we obtain $\mathcal{E}(l) = (k^2 - 1)/6$, showing the optimality of Algorithm 3.1.

4 Conclusions

We have presented optimal placements and routing algorithms in 2-dimensional and 3-dimensional tori networks. Our main result is a routing algorithm for the 3-dimensional case that avoids going through processors and results in an optimal link load. The optimal link load provides robustness in the case of link failures.
Further work may include extending our results to tori of higher dimensions and finding a non-minimal optimal routing algorithm with uniform load distribution over the links.

References

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

[2] A. Agarwal, "Limits on Interconnection Network Performance," IEEE Trans. on Parallel and Distributed Systems, Vol. 2, No. 4, pp. 396-412, October 1992.

[3] J. Bruck, R. Cypher and C. T. Ho, "Fault-Tolerant Meshes and Hypercubes With Minimal Number of Spares," IEEE Trans. on Computers, Vol. 42, No. 9, pp. 1089–1104, September 1993.

[4] J. Bruck, R. Cypher and C. T. Ho, "Tolerating Faults in a Mesh with a Row of Spare Nodes," Theoretical Computer Science–Special Issue of Dependable Parallel Computing, pp. 241-252, July 1994.

[5] J. Bruck, R. Cypher and C. T. Ho, "Wildcard Dimension, Coding Theory and Fault Tolerant Meshes and Hypercubes," IEEE Trans. on Computers, Vol. 44, No. 1, pp. 150–155, January 1995.

[6] W. Dally, "Performance Analysis of k-ary n-cube Interconnection Networks," IEEE Trans. on Computers, Vol. 39, No. 6, June 1990.

[7] W. Dally, A. Chien, S. Fiske, W. Horwat, J. Keen, M. Larivee, R. Lethin, P. Nuth and S. Wills, "The J-Machine: A Fine Grain Concurrent Computer," IFIP Congress, 1989.

[8] T. Feng, "A Survey of Interconnection Networks," Computer, pp. 12-27, Dec. 1981.

[9] S. A. Felperin, L. Gravano, G. D. Pifarré and J. L. C. Sanz, "Routing Techniques for Massively Parallel Communication," Proceedings of the IEEE, Special Issue on Massively Parallel Computers", Vol. 79, No. 4, April 1991

[10] S. Konstantinidou and L. Snyder, "Chaos Router: Architecture and Performance," 18-th International Symposium on Computer Architecture, pp. 212–221, May 1991.

[11] F. T. Leighton, "Introduction to Parallel Algorithms and Architectures," Morgan Kaufmann Publishers, Inc., California, 1992.

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

[13] L. M. Ni and P. K. McKinley, "A Survey of Routing Techniques in Wormhole Networks," Computer, Vol. 26 (2), pp. 62–76, Feb. 1993.

[14] F. Pitteli and D. Smitley, "Analysis of a 3D Toroidal Network for a Shared Memory Architecture," Proceedings of Supercomputing 88, pp. 35-41, November 1988.

[15] H. J. Siegel, "Interconnection Networks for SIMD Machines," Computer, pp. 57-65, June 1979.

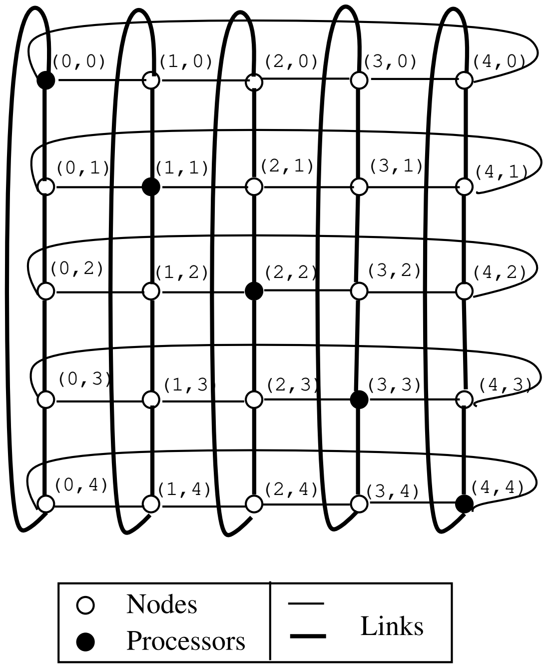

Figure 1: An optimal placement of processors for the 2-dimensional 5-torus.

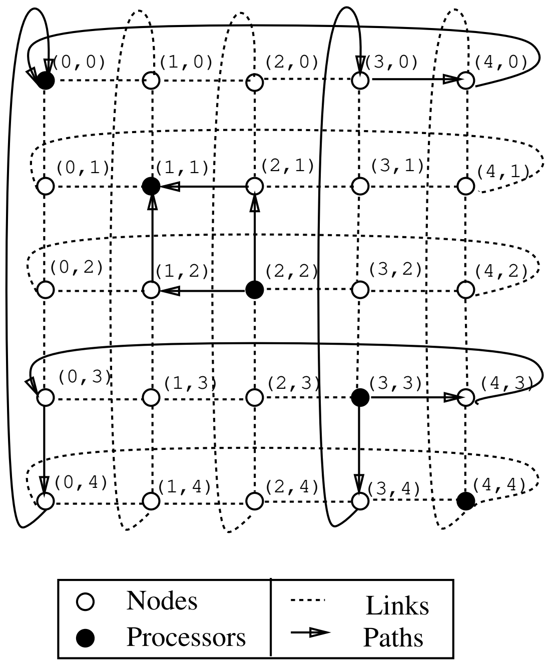

Figure 2: The paths allowed by the algorithm between the processors (3,3) and (0,0) and between processors (2,2) and (1,1) for the 2-dimensional 5-torus.

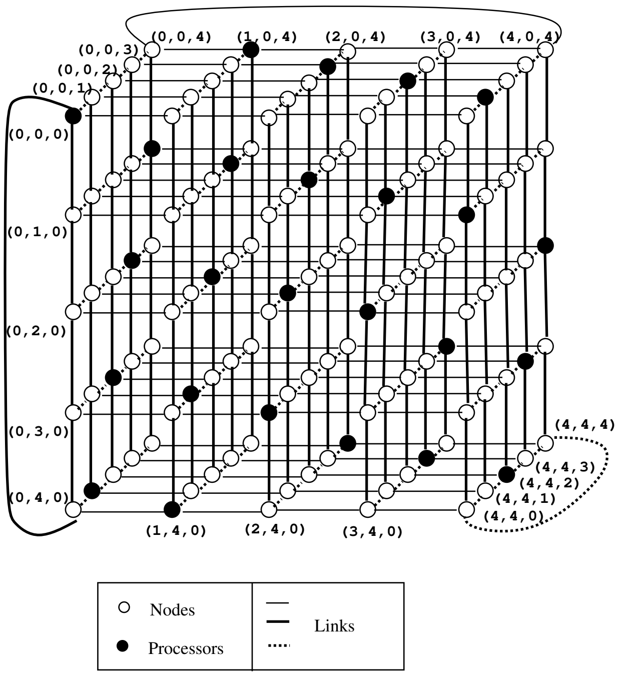

Figure 3: An optimal placement of processors for the 3-dimensional 5-torus.

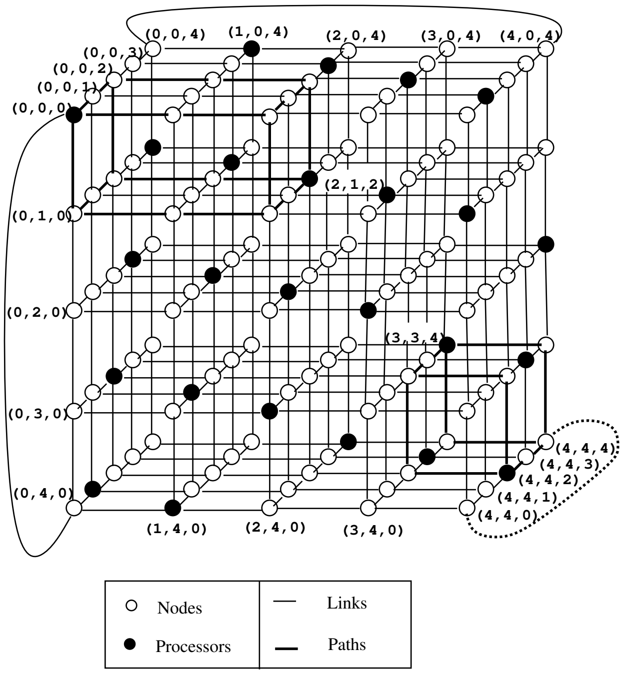

Figure 4: The paths allowed by the algorithm between the processors (0,0,0) and (2,1,2) and between processors (3,3,4) and (4,4,2) for the 3-dimensional 5-torus.