International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 7, Number 2, pages 173–186, July 2017

A Set-to-Set Disjoint Paths Routing Algorithm in Tori

Keiichi Kaneko
Graduate School of Engineering
Tokyo University of Agriculture and Technology
Koganei-shi, Tokyo 184-8588, JAPAN

Antoine Bossard
Graduate School of Science
Kanagawa University
Tsuchiya 2946, Hiratsuka, Kanagawa 259-1293, JAPAN

Received: February 14, 2017
Revised: May 5, 2017
Accepted: June 5, 2017
Communicated by Yoshiaki Kakuda

Abstract

Numerous TOP500 supercomputers are based on a torus interconnection network. The torus topology is effectively one of the most popular interconnection networks for massively parallel systems due to its interesting topological properties such as symmetry and simplicity. For instance, the world-famous supercomputers Fujitsu K, IBM Blue Gene/L, IBM Blue Gene/P and Cray XT3 are all torus-based. In this paper, we propose an algorithm that constructs $2n$ mutually node-disjoint paths from a set S of $2n$ source nodes to a set D of $2n$ destination nodes in an n-dimensional k-ary torus $T_{n,k}$ ($n \geq 1, k \geq 3$). This algorithm is then formally evaluated. We have proved that the paths selected by the proposed algorithm have lengths at most $2(k+1)n$ and can be obtained with a time complexity of $O(kn^3 + n^3 \log n)$.

Keywords: Multicomputer, Interconnect, Parallel processing, Hypercube, Fault tolerance, Performance evaluation

1 Introduction

Modern supercomputers are massively parallel systems: they include hundreds of thousands of computing nodes (CPUs). The interconnection of these nodes is critical: inefficient data transmission would annihilate any performance hopes no matter the number of cores available. Effectively, in such case, performance bottleneck would inevitably arise, with most CPUs simply waiting to be fed data. In practice, the torus topology has been proving very popular as interconnection network of massively parallel systems. Effectively, many TOP500 supercomputers are now torus-based: Fujitsu K, IBM Blue Gene/L, IBM Blue Gene/P and Cray XT3 are famous examples [1].

We propose in this paper a solution to the set-to-set disjoint paths routing problem in a torus network. This problem consists of selecting mutually node-disjoint paths between two sets of nodes. This problem is critical and has obvious applications: first, it enables simultaneous usage of the

Table 1: Comparing topological properties of tori with those of other popular interconnection networks.

Network	**Order**	**Degree**	**Diameter**	**Cost**	**Edges**
(n,k)-torus	k^n	$2n$	$n\lfloor k/2 \rfloor$	$2n^2\lfloor k/2 \rfloor$	nk^n
n-hypercube	2^n	n	n	n^2	$2^n n$
n-dual-cube	2^{2n-1}	n	$2n$	$2n^2$	$2^{2n-2}n$
n-star graph	$n!$	$n-1$	$\lfloor 3(n-1)/2 \rfloor$	$(n-1)\lfloor 3(n-1)/2 \rfloor$	$n!(n-1)/2$
(k,m)-metacube	$2^{2^k m+k}$	$m+k$	$2^k(m+1)$	$2^k(m+1)(m+k)$	$2^{2^k m+k-1}(m+k)$

network to transmit data along several, different paths. For a same amount of data to transfer, the network is thus used a shorter period of time, thus inducing a reduced power consumption, making it environmental friendly (see Green IT [2, 3]). In addition, because the paths selected are mutually node-disjoint, the infamous blocking situations of shared resource systems (deadlocks, livelocks and starvations) are guaranteed not to occur. Disjoint paths routing is thus a very desirable algorithm property [4, 5, 6].

Furthermore, by addressing the aforementioned issues, system dependability is subsequently greatly enhanced. When such a huge amount of cores is handled, the probability that a few of these nodes are broken (*faulty nodes*) is high. In the case of disjoint paths routing, as the paths selected share no common node, one faulty node can jeopardise *at most one* of the selected paths, unlike in the case of non-disjoint paths where one faulty node can neutralise several paths at once.

For these reasons, the set-to-set disjoint paths routing problem has been addressed in various interconnection networks: in dual-cubes [7], recursive dual-nets [8], pancake graphs [9], star graphs [10] and hypercubes [11, 12] to only cite a few of them. A similar approach has been discussed in various previous works [13, 14]. For reference, a comparative summary of the topological properties of tori and other networks [15] is given in Table 1 (the network cost is the degree by diameter product). The approach followed in this paper is a divide-and-conquer one, relying on the recursive property of tori. Relying on the recursive property of the network topology is a method that has been proved very efficient when solving disjoint-path routing problems [9, 11, 13]. The same problem could be solved by using for instance a solution to the maximum flow algorithm, such as the Ford-Fulkerson algorithm, or even newer solutions, yet the solutions to this problem are polynomial in the number of edges (or vertices, depending on the algorithm), which thus makes them impractical solutions. The proposed routing algorithm selects $2n$ mutually node-disjoint paths (abbreviated *disjoint* hereinafter) between a set S of $2n$ source nodes and a set D of $2n$ destination nodes in an n-dimensional k-ary torus $T_{n,k}$ ($n \geq 1, k \geq 3$). The paths selected have lengths at most $2(k+1)n$ and can be obtained with a time complexity of $O(kn^3 + n^3 \log n)$.

The rest of this paper is organised as follows. Definitions, notations and intermediary results are established in Section 2. The set-to-set disjoint paths routing algorithm in a torus is described in Section 3. This algorithm is formally evaluated in Section 4. This paper is concluded in Section 5.

2 Preliminaries

In this section, we first give the definitions and notations that will be used throughout this paper. In addition, several important results and properties on which is based the proposed torus set-to-set disjoint paths routing algorithm are established.

Definition 1. *For two n-dimensional vectors $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ and a natural number k, we define the operation $\boldsymbol{a} \oplus \boldsymbol{b} = ((a_1 + b_1) \mod k, (a_2 + b_2) \mod k, \ldots, (a_n + b_n) \mod k)$. Similarly, we define the operation $\boldsymbol{a} \ominus \boldsymbol{b} = ((a_1 - b_1) \mod k, (a_2 - b_2) \mod k, \ldots, (a_n - b_n) \mod k)$.*

Definition 2. *Among n-dimensional vectors $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$, we denote the nodes that satisfy $a_j = 0$ $(1 \leq j \neq i \leq n)$ and $a_i = 1$ by $\boldsymbol{e}_i$.*

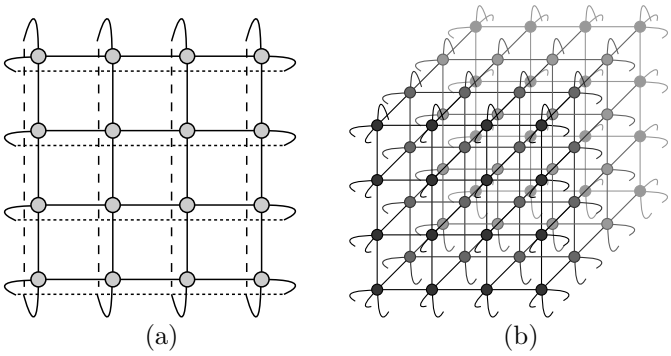

Figure 1: A 2-dimensional 4-ary (a) and a 3-dimensional 4-ary (b) torus network.

We then recall the definition of the torus topology.

Definition 3. *[16] An n-dimensional k-ary torus $T_{n,k}$ ($n \geq 1$, $k \geq 3$) is an undirected graph consisting of k^n nodes where the set of the nodes is given by $\{0, 1, \ldots, k-1\}^n$. For two nodes $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ in a $T_{n,k}$, $\boldsymbol{a}$ and $\boldsymbol{b}$ are adjacent if and only if there exists $\boldsymbol{e}_i$ ($1 \leq i \leq n$) such that $\boldsymbol{b} = \boldsymbol{a} \oplus \boldsymbol{e}_i$ or $\boldsymbol{b} = \boldsymbol{a} \ominus \boldsymbol{e}_i$.*

A 2-dimensional 4-ary torus network is given in Figure 1a, and a 3-dimensional 4-ary torus network is given in Figure 1b.

Next, several topological properties of torus networks are recalled or proven.

Lemma 1. *The length of an arbitrary cycle in a torus is at least 4.*

Proof. Omitted. □

Definition 4. *In an n-dimensional k-ary torus $T_{n,k}$, the sub graph derived by the set of the nodes obtained by fixing their rightmost elements to an integer h ($0 \leq h \leq k-1$) is isomorphic to an $(n-1)$-dimensional k-ary torus $T_{n-1,k}$. This sub graph is called a sub torus, and it is denoted by $T_{n-1,k}(h)$.*

Definition 5. *For a node $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ inside a $T_{n,k}$, the sub graph derived from the set of the nodes obtained by fixing their elements other than the rightmost one to $a_1, a_2, \ldots, a_{n-1}$ is isomorphic to a ring of k nodes. This sub graph is called a class to which the node $\boldsymbol{a}$ belongs, and it is denoted by $C(\boldsymbol{a})$.*

Definition 6. *A path p in a graph is an alternate sequence of nodes and edges: $p = u_1, (u_1, u_2), u_2, \ldots, u_k$, with (u_1, u_2) denoting the edge between the nodes u_1 and u_2. This path p can be also denoted by $u_1 \to u_2 \to \ldots \to u_k$ and simplified to $u_1 \rightsquigarrow u_k$. The length of a path corresponds to its number of edges.*

Definition 7. *Two paths are node disjoint (or simply disjoint) if and only if they have no node in common.*

Lemma 2. *For a node $\boldsymbol{a}$ in a $T_{n,k}$ and a sub torus $T_{n-1,k}(h)$ to which $\boldsymbol{a}$ does not belong, we can construct $(2n-1)$ disjoint paths $\vec{P}_l(\boldsymbol{a})$: $\boldsymbol{a} \rightsquigarrow \boldsymbol{b}_l$ ($1 \leq l \leq 2n-1$) of lengths at most k from $\boldsymbol{a}$ to $(2n-1)$ distinct nodes $\boldsymbol{b}_l$ in $T_{n-1,k}(h)$ by traversing a class for each path by the $\oplus$ operation in $O(kn)$ time.*

Proof. For $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$, by traversing classes from $\boldsymbol{a}$ and its neighbour nodes in the sub

torus $T_{n-1,k}(a_n)$, we can construct $(2n-1)$ disjoint paths as follows:

$$\vec{P}_l(\boldsymbol{a}): \begin{cases} \boldsymbol{a} \to \boldsymbol{a} \oplus \boldsymbol{e}_l \to \boldsymbol{a} \oplus \boldsymbol{e}_l \oplus \boldsymbol{e}_n \to \boldsymbol{a} \oplus \boldsymbol{e}_l \oplus \\ \boldsymbol{e}_n \oplus \boldsymbol{e}_n \to \cdots \to \boldsymbol{b}_l \qquad (1 \le l \le n) \\ \boldsymbol{a} \to \boldsymbol{a} \ominus \boldsymbol{e}_{l-n} \to \boldsymbol{a} \ominus \boldsymbol{e}_{l-n} \oplus \boldsymbol{e}_n \to \boldsymbol{a} \ominus \\ \boldsymbol{e}_{l-n} \oplus \boldsymbol{e}_n \oplus \boldsymbol{e}_n \to \cdots \to \boldsymbol{b}_l \\ \qquad\qquad (n+1 \le l \le 2n-1) \end{cases}$$

The maximum length of these paths is k. It takes $O(k)$ time to construct a path. Hence, it takes $O(kn)$ time in total. □

Lemma 3. *For a node $\boldsymbol{a}$ in a $T_{n,k}$ and a sub torus $T_{n-1,k}(h)$ to which $\boldsymbol{a}$ does not belong, we can construct $(2n-1)$ disjoint paths $\vec{P}_l(\boldsymbol{a})$: $\boldsymbol{a} \rightsquigarrow \boldsymbol{b}_l$ $(1 \le l \le 2n-1)$ of lengths at most k from $\boldsymbol{a}$ to $(2n-1)$ distinct nodes $\boldsymbol{b}_l$ in $T_{n-1,k}(h)$ by traversing a class for each path by the $\ominus$ operation with a time complexity of $O(kn)$.*

Proof. For $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$, by traversing classes from $\boldsymbol{a}$ and its neighbour nodes in the sub torus $T_{n-1,k}(a_n)$, we can construct $(2n-1)$ disjoint paths as follows:

$$\vec{P}_l(\boldsymbol{a}): \begin{cases} \boldsymbol{a} \to \boldsymbol{a} \oplus \boldsymbol{e}_l \to \boldsymbol{a} \oplus \boldsymbol{e}_l \ominus \boldsymbol{e}_n \to \boldsymbol{a} \oplus \boldsymbol{e}_l \ominus \\ \boldsymbol{e}_n \ominus \boldsymbol{e}_n \to \cdots \to \boldsymbol{b}_l \qquad (1 \le l \le n-1) \\ \boldsymbol{a} \to \boldsymbol{a} \ominus \boldsymbol{e}_{l-n+1} \to \boldsymbol{a} \ominus \boldsymbol{e}_{l-n+1} \ominus \boldsymbol{e}_n \to \\ \boldsymbol{a} \ominus \boldsymbol{e}_{l-n+1} \ominus \boldsymbol{e}_n \ominus \boldsymbol{e}_n \to \cdots \to \boldsymbol{b}_l \\ \qquad\qquad (n \le l \le 2n-1) \end{cases}$$

The maximum length of these paths is k. It takes $O(k)$ time to construct one path. Hence, it takes $O(kn)$ time in total. □

Lemma 4. *[17] For a source node set S $(|S| = 2)$ and a destination node set D $(|D| = 2)$ in a $T_{n,k}$, we can find two disjoint paths between S and D in the optimal time complexity $O(kn)$. The maximum length of the selected paths is kn.*

Lemma 5. *For three nodes $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ in a torus, if an arbitrary path Q: $\boldsymbol{a} \rightsquigarrow \boldsymbol{b}$ in the torus contains $\boldsymbol{c}$ or multiple neighbour nodes of $\boldsymbol{c}$ in the torus, there exists a path from $\boldsymbol{c}$ to $\boldsymbol{b}$ whose length is shorter than Q.*

Proof. If the path Q contains $\boldsymbol{c}$, there exists a sub path $\boldsymbol{c} \rightsquigarrow \boldsymbol{b}$ of Q that is shorter than Q. If the path Q contains multiple neighbour nodes of $\boldsymbol{c}$ in the torus, let $\hat{\boldsymbol{c}}$ represent the neighbour node of $\boldsymbol{c}$ that is closest to the node $\boldsymbol{b}$. Then, from Lemma 1, the length of the sub path $\boldsymbol{a} \rightsquigarrow \hat{\boldsymbol{c}}$ of Q is more than 1. So, the path $\boldsymbol{c} \to \hat{\boldsymbol{c}} \rightsquigarrow \boldsymbol{b}$ obtained by adding the sub path $\hat{\boldsymbol{c}} \rightsquigarrow \boldsymbol{b}$ of Q to the edge $\boldsymbol{c} \to \hat{\boldsymbol{c}}$ is shorter than Q. □

Lemma 6. *For an arbitrary set of $2n$ nodes $A = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{2n}\}$ in a $T_{n,k}$, we can construct two paths from two nodes in A to $T_{n-1,k}(0)$, and $(2n-2)$ paths from the remaining $(2n-2)$ nodes in A to $T_{n-1,k}(k-1)$ whose lengths are at most k and such that all of these paths are disjoint with a time complexity of $O(n^2 \log n)$.*

Proof. First, let us assume that there is no class that contains multiple nodes in A. It takes $O(n \log n)$ time to check this assertion. Then, from two nodes in A that have the smallest and second smallest rightmost elements, say $\boldsymbol{a}_i$ $(i = 1, 2)$, we can construct the paths $\boldsymbol{a}_i \to \boldsymbol{a}_i \ominus \boldsymbol{e}_n \to \boldsymbol{a}_i \ominus \boldsymbol{e}_n \ominus \boldsymbol{e}_n \to \cdots \to \boldsymbol{a}'_i (\in T_{n-1,k}(0))$. The lengths of these paths are at most $k-1$, and it takes $O(k)$ time to construct them. From the remaining $(2n-2)$ nodes $\boldsymbol{a}_i$ $(3 \le i \le 2n)$, we can construct paths $\boldsymbol{a}_i \to \boldsymbol{a}_i \oplus \boldsymbol{e}_n \to \boldsymbol{a}_i \oplus \boldsymbol{e}_n \oplus \boldsymbol{e}_n \to \cdots \to \boldsymbol{a}'_i (\in T_{n-1,k}(k-1))$. The lengths of these paths are at most $k-1$, too, and it takes $O(kn)$ time to construct these paths.

Next, let us assume that there is a class C that contains multiple nodes in A. Then, among the nodes in $A \cap C$, we select the node that has the smallest rightmost element, say $\boldsymbol{a}_1$, and the node

that has the largest rightmost element, say $\boldsymbol{a}_2$, and construct paths $\boldsymbol{a}_1 \to \boldsymbol{a}_1 \ominus \boldsymbol{e}_n \to \boldsymbol{a}_1 \ominus \boldsymbol{e}_n \ominus \boldsymbol{e}_n \to \cdots \to \boldsymbol{a}'_1 (\in T_{n-1,k}(0))$ and $\boldsymbol{a}_2 \to \boldsymbol{a}_2 \oplus \boldsymbol{e}_n \to \boldsymbol{a}_2 \oplus \boldsymbol{e}_n \oplus \boldsymbol{e}_n \to \cdots \to \boldsymbol{a}'_2 (\in T_{n-1,k}(k-1))$. The lengths of these paths are at most $k-2$, and it takes $O(k)$ time to construct them. Now, we have constructed two paths by using only one class. Therefore, among the remaining $(2n-2)$ nodes, we select the node that has the smallest rightmost element, say $\boldsymbol{a}_3$, and consider the $(2n-1)$ paths $\overleftarrow{P}_l(\boldsymbol{a}_3)$ $(1 \le l \le 2n-1)$ of lengths at most k by Lemma 3. Then, there is at least one path $\boldsymbol{a}_3 \rightsquigarrow \boldsymbol{a}'_3 (\in T_{n-1,k}(0))$ that is disjoint from other nodes in A and the paths from them. It is sufficient to find an unoccupied class to find the disjoint path. Hence, it takes $O(n \log n)$ time. The disjoint path construction takes $O(k)$ time. For each of the remaining $(2n-3)$ nodes $\boldsymbol{a}_i$ $(4 \le i \le 2n)$, we consider the $(2n-1)$ paths $\vec{P}_l(\boldsymbol{a}_i)$ $(1 \le l \le 2n-1)$ of lengths at most k by Lemma 2. Then, there is at least one path $\boldsymbol{a}_i \rightsquigarrow \boldsymbol{a}'_i (\in T_{n-1,k}(k-1))$ that is disjoint from other nodes in A and the paths from them. It is sufficient to find an unoccupied class to find the disjoint path. So, it takes $O(n \log n)$ time. The disjoint path construction takes $O(k)$. Therefore, it takes $O(n^2 \log n)$ time to construct paths from $(2n-3)$ nodes. To summarize, the maximum length of the paths is k, and the path selection is $O(n^2 \log n)$ time. □

3 Set-to-Set Disjoint Paths Routing Algorithm

Because a 1-dimensional k-ary torus ($k \ge 3$) is just a ring, and because any source node can be connected to any destination node, we can easily construct 2 disjoint paths that connect 2 source nodes and 2 destination nodes in $O(k)$ time. In a 2-dimensional k-ary torus ($k \ge 3$), we can construct 4 disjoint paths that connect 4 source nodes and 4 destination nodes in $O(k)$ time. So in this section, we propose an algorithm S2S that constructs set-to-set disjoint paths in an n-dimensional k-ary torus $T_{n,k}$ $(n \ge 3, k \ge 3)$.

The algorithm S2S is divided into three cases depending on the numbers of the source and destination nodes inside the sub torus $T_{n-1,k}(0)$. The three cases distinguished hereinafter cover the source-destination node repartition possibilities. Hence, it suffices to prove the correctness and complexities of these three cases in order to solve the set-to-set disjoint paths routing problem in a torus network.

3.1 Case 1

In Case 1, we assume that a sub torus contains at least two pairs of the source and destination nodes ($\exists h$ $(0 \le h \le k-1)$ such that $|T_{n-1,k}(h) \cap S| \ge 2$, $|T_{n-1,k}(h) \cap D| \ge 2$). Without loss of generality, we can assume that $T_{n-1,k}(0)$ is the sub torus with at least two source-destination pairs. An illustration of Case 1 is given in Figure 2.

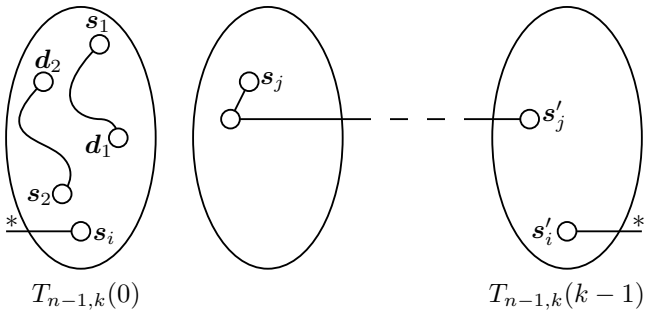

Figure 2: Case 1: the sub torus $T_{n-1,k}(0)$ contains two source-destination pairs.

Step 1 Select two source nodes and two destination nodes in $T_{n-1,k}(0)$, and find two disjoint paths between the source and destination nodes by Lemma 4. While at least one of these paths contains other source or destination nodes in the $T_{n-1,k}(0)$ or multiple neighbour nodes in the

$T_{n-1,k}(0)$ of one of them, apply Lemma 5 to adopt the shorter path. We assume that the paths $\boldsymbol{s}_1 \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{s}_2 \rightsquigarrow \boldsymbol{d}_2$ are obtained finally.

Step 2 For each node $\boldsymbol{s}_i$ of other source nodes in $T_{n-1,k}(0)$, from Lemma 3, consider $(2n-1)$ disjoint paths $\vec{P}_l(\boldsymbol{s}_i)$ $(1 \leq l \leq 2n-1)$ of lengths at most 2 from $\boldsymbol{s}_i$ to $T_{n-1,k}(k-1)$. Among these $(2n-1)$ paths, try to select one that is disjoint from $\boldsymbol{s}_1 \rightsquigarrow \boldsymbol{d}_1$, $\boldsymbol{s}_2 \rightsquigarrow \boldsymbol{d}_2$, other source nodes $\boldsymbol{s}_j$ $(3 \leq j \neq i \leq 2n)$, and the paths originating from them to $T_{n-1,k}(k-1)$. If all the paths are blocked, let $\hat{\boldsymbol{s}}_i$ represent the neighbour node of $\boldsymbol{s}_i$ that $\boldsymbol{s}_1 \rightsquigarrow \boldsymbol{d}_1$ contains. Then, construct the path $\boldsymbol{s}_i \to \boldsymbol{s}_i \oplus \boldsymbol{e}_n \to \hat{\boldsymbol{s}}_i \oplus \boldsymbol{e}_n \to \cdots \to \boldsymbol{s}'_i (\in T_{n-1,k}(k-1))$.

Step 3 For each node $\boldsymbol{d}_i$ of the other destination nodes in the $T_{n-1,k}(0)$, execute the process similar to Step 2 to obtain disjoint paths $\boldsymbol{d}_i \rightsquigarrow \boldsymbol{d}'_i (\in T_{n-1,k}(k-1))$.

Step 4 For each source node $\boldsymbol{s}_i$ in the sub tori other than $T_{n-1,k}(0)$, from Lemma 2, consider $(2n-1)$ paths $\vec{P}_l(\boldsymbol{s}_i)$: $(1 \leq l \leq 2n-1)$ from the node $\boldsymbol{s}_i$ to $T_{n-1,k}(k-1)$. Among them, select one path $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i (\in T_{n-1,k}(k-1))$ which is disjoint from other source nodes $\boldsymbol{s}_j$ $(3 \leq j \neq i \leq 2n)$ as well as the paths originating from them to $T_{n-1,k}(k-1)$.

Step 5 For each node $\boldsymbol{d}_i$ of the other destination nodes in the sub tori other than $T_{n-1,k}(0)$, execute the process similar to Step 4 to obtain disjoint paths $\boldsymbol{d}_i \rightsquigarrow \boldsymbol{d}'_i (\in T_{n-1,k}(k-1))$.

Step 6 For the source and destination nodes such that the paths from them encounter each other, discard the redundant sub paths. For the remaining source and destination nodes, connect the terminal nodes of the paths from them in the $T_{n-1,k}(k-1)$ with disjoint paths by applying the algorithm S2S recursively.

3.2 Case 2

In Case 2, we first assume that there is not any sub torus that contains at least two pairs of the source and destination nodes ($\forall h$ $(0 \leq h \leq k-1)$, $|T_{n-1,k}(h) \cap S| < 2$ or $|T_{n-1,k}(h) \cap D| < 2$). Since $k \geq 3$, we can assume that $|T_{n-1,k}(h) \cap S| \leq 2n-2$, $|T_{n-1,k}(h) \cap D| \leq 2n-2$, and $h = k-1$ without loss of generality. In Case 2, we also assume that $|T_{n-1,k}(0) \cap S| < 2$ and $|T_{n-1,k}(0) \cap D| < 2$. An illustration of Case 2 is given in Figure 3.

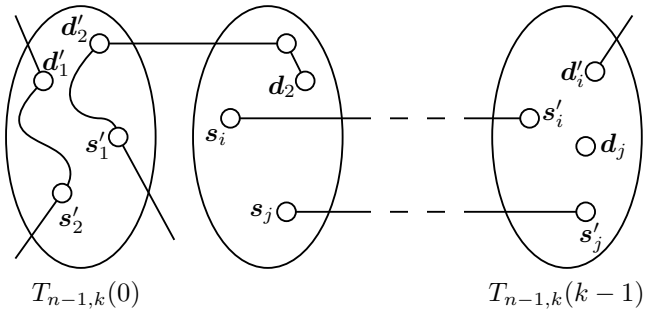

Figure 3: Case 2: any sub torus includes at most one source-destination pair, and the condition $|T_{n-1,k}(0) \cap S| < 2$ and $|T_{n-1,k}(0) \cap D| < 2$ holds.

Step 1 For the set of source nodes S, apply Lemma 6 to find $2n$ disjoint paths. Without loss of generality, assume that $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i (\in T_{n-1,k}(0))$ $(i = 1, 2)$ and $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i (\in T_{n-1,k}(k-1))$ $(3 \leq i \leq 2n)$ are constructed.

Step 2 For the set of destination nodes D, apply Lemma 6 to find $2n$ disjoint paths. Without loss of generality, assume that $\boldsymbol{d}_i \rightsquigarrow \boldsymbol{d}'_i (\in T_{n-1,k}(0))$ $(i = 1, 2)$ and $\boldsymbol{d}_i \rightsquigarrow \boldsymbol{d}'_i (\in T_{n-1,k}(k-1))$ $(3 \leq i \leq 2n)$ are found.

Step 3 If a path $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i(\in T_{n-1,k}(0))$ and a path $\boldsymbol{d}_j \rightsquigarrow \boldsymbol{d}'_j(\in T_{n-1,k}(0))$ encounter, discard the redundant sub path. Also, if a path $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i(\in T_{n-1,k}(k-1))$ and a path $\boldsymbol{d}_j \rightsquigarrow \boldsymbol{d}'_j(\in T_{n-1,k}(k-1))$ encounter, discard the redundant sub path.

Step 4 If a path $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i(\in T_{n-1,k}(0))$ and a path $\boldsymbol{d}_j \rightsquigarrow \boldsymbol{d}'_j(\in T_{n-1,k}(k-1))$ encounter, discard the redundant sub paths and execute the following Steps 5 and 6. Otherwise go to Step 7.

Step 5 If both of the paths $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i(\in T_{n-1,k}(0))$ $(i = 1, 2)$ encounter other paths, execute Step 6 for both of $\boldsymbol{d}_i$ $(i = 1, 2)$. Otherwise, without loss of generality, we assume that $i = 2$. Construct a path $\boldsymbol{s}'_1 \rightsquigarrow \boldsymbol{d}'_1$ and while the path contains $\boldsymbol{d}'_2$ or its multiple neighbour nodes, apply Lemma 5 to adopt the shorter path. We assume that the path $\boldsymbol{s}'_1 \rightsquigarrow \boldsymbol{d}'_1$ is eventually obtained.

Step 6 From Lemma 3, consider $(2n-1)$ disjoint paths $\overleftarrow{P}_l(\boldsymbol{d}'_i)$ $(1 \leq l \leq 2n-1)$ of lengths at most 2 from $\boldsymbol{d}'_i$ to $T_{n-1,k}(k-1)$. Then, select one of the paths that does not contain any other destination node and that is disjoint from the paths from the other destination nodes. Go to Step 8.

Step 7 In the $T_{n-1,k}(0)$, for the source nodes $\boldsymbol{s}'_1$, $\boldsymbol{s}'_2$, and the destination nodes $\boldsymbol{d}'_1$, $\boldsymbol{d}'_2$, from Lemma 4, construct two disjoint paths.

Step 8 For the remaining source and destination nodes, connect the terminal nodes of the paths from them in the $T_{n-1,k}(k-1)$ with disjoint paths by applying the algorithm S2S recursively.

3.3 Case 3

In this case, we assume that there is not any sub torus that contains at least two pairs of the source and destination nodes ($\forall h$ $(0 \leq h \leq k-1)$, $|T_{n-1,k}(h) \cap S| < 2$ or $|T_{n-1,k}(h) \cap D| < 2$). Since $k \geq 3$, we can assume that $|T_{n-1,k}(k-1) \cap S| \leq 2n-2$ and $|T_{n-1,k}(k-1) \cap D| \leq 2n-2$ without loss of generality. In Case 3, we assume that $|T_{n-1,k}(0) \cap S| < 2$ and $|T_{n-1,k}(0) \cap D| \geq 2$. For the case that $|T_{n-1,k}(0) \cap S| \geq 2$ and $|T_{n-1,k}(0) \cap D| < 2$, it can be treated similarly. An illustration of Case 3 is given in Figure 4.

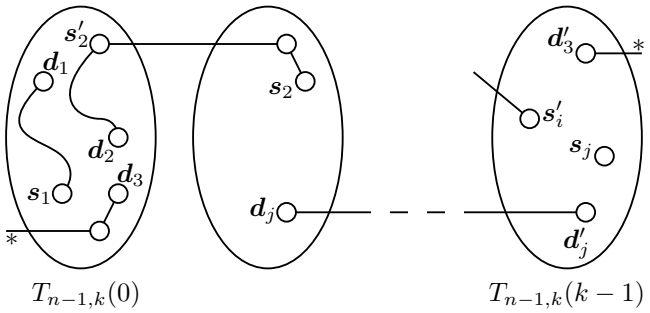

Figure 4: Case 3: any sub torus includes at most one source-destination pair, and the condition $|T_{n-1,k}(0) \cap S| < 2$ and $|T_{n-1,k}(0) \cap D| \geq 2$ (or similarly $|T_{n-1,k}(0) \cap S| \geq 2$ and $|T_{n-1,k}(0) \cap D| < 2$) holds.

Step 1 For the set of source nodes S, apply Lemma 6 to find $2n$ disjoint paths. Without loss of generality, assume that $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i(\in T_{n-1,k}(0))$ $(i = 1, 2)$ and $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i(\in T_{n-1,k}(k-1))$ $(3 \leq i \leq 2n)$ are constructed.

Step 2 Select two destination nodes, say $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$, in $T_{n-1,k}(0)$, and construct two disjoint paths between the source nodes $\boldsymbol{s}'_1$, $\boldsymbol{s}'_2$ and the destination nodes $\boldsymbol{d}_1$, $\boldsymbol{d}_2$ by applying Lemma 4. While at least one of these paths contains other destination nodes in the $T_{n-1,k}(0)$ or multiple neighbour nodes in the $T_{n-1,k}(0)$ of one of them, apply Lemma 5 to adopt the shorter path. We assume that the paths $\boldsymbol{s}'_1 \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{s}'_2 \rightsquigarrow \boldsymbol{d}_2$ are obtained finally.

Step 3 For each node $\boldsymbol{d}_i$ of other destination nodes in $T_{n-1,k}(0)$, from Lemma 3, consider $(2n-1)$ disjoint paths $\overleftarrow{P}_l(\boldsymbol{d}_i)$ $(1 \leq l \leq 2n-1)$ of lengths at most 2 from $\boldsymbol{d}_i$ to $T_{n-1,k}(k-1)$. Among these $(2n-1)$ paths, try to select one that is disjoint from $\boldsymbol{s}_1 \rightsquigarrow \boldsymbol{d}_1$, $\boldsymbol{s}_2 \rightsquigarrow \boldsymbol{d}_2$, other destination nodes $\boldsymbol{d}_j$ $(3 \leq j \neq i \leq 2n)$, and the paths from them to $T_{n-1,k}(k-1)$. If there is no such path, let $\hat{\boldsymbol{d}}_i$ represent the neighbour node of $\boldsymbol{d}_i$ that $\boldsymbol{s}_1 \rightsquigarrow \boldsymbol{d}_1$ contains. Then, construct the path $\boldsymbol{d}_i \to \boldsymbol{d}_i \oplus \boldsymbol{e}_n \to \hat{\boldsymbol{d}}_i \oplus \boldsymbol{e}_n \to \cdots \to \boldsymbol{d}'_i (\in T_{n-1,k}(k-1))$.

Step 4 For each destination node $\boldsymbol{d}_i$ in the sub tori other than $T_{n-1,k}(0)$, from Lemma 2, consider $(2n-1)$ paths $\vec{P}_l(\boldsymbol{d}_i)$: $(1 \leq l \leq 2n-1)$ from the node $\boldsymbol{d}_i$ to $T_{n-1,k}(k-1)$. Among them, select one path $\boldsymbol{d}_i \rightsquigarrow \boldsymbol{d}'_i (\in T_{n-1,k}(k-1))$ that is disjoint from other source nodes $\boldsymbol{d}_j$ $(3 \leq j \neq i \leq 2n)$ and the paths from them to $T_{n-1,k}(k-1)$.

Step 5 If a path $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i (\in T_{n-1,k}(k-1))$ and a path $\boldsymbol{d}_j \rightsquigarrow \boldsymbol{d}'_j (\in T_{n-1,k}(k-1))$ encounter, discard the redundant sub path.

Step 6 If a path $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i (\in T_{n-1,k}(0))$ and a path $\boldsymbol{d}_j \rightsquigarrow \boldsymbol{d}'_j (\in T_{n-1,k}(k-1))$ encounter, discard the redundant sub paths and execute the following Step 7. Otherwise go to Step 8.

Step 7 If both of the paths $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i (\in T_{n-1,k}(0))$ $(i = 1, 2)$ encounter other paths, execute the following for $i = 1, 2$. Otherwise, without loss of generality, we assume that $i = 2$. For $\boldsymbol{d}_i$, from Lemma 3, we consider $(2n-1)$ disjoint paths $\overleftarrow{P}_l(\boldsymbol{d}_i)$ $(1 \leq l \leq 2n-1)$ of lengths at most 2 from $\boldsymbol{d}_i$ to $T_{n-1,k}(k-1)$. Then, select one path that does not contain any other destination node and that is disjoint from the paths from the other destination nodes.

Step 8 For the remaining source and destination nodes, connect the terminal nodes of the paths from them in the $T_{n-1,k}(k-1)$ with disjoint paths by applying the algorithm S2S recursively.

3.4 Routing example

In this section, a concrete routing example in a 3-dimensional 3-ary torus $T_{3,3}$ is given. Let $S = \{\boldsymbol{s}_1 = (0,0,0), \boldsymbol{s}_2 = (0,0,1), \boldsymbol{s}_3 = (0,0,2), \boldsymbol{s}_4 = (2,0,0), \boldsymbol{s}_5 = (2,0,1), \boldsymbol{s}_6 = (2,0,2)\}$ be the set of source nodes, and $D = \{\boldsymbol{d}_1 = (1,0,0), \boldsymbol{d}_2 = (1,0,1), \boldsymbol{d}_3 = (1,0,2), \boldsymbol{d}_4 = (1,1,0), \boldsymbol{d}_5 = (1,1,1), \boldsymbol{d}_6 = (1,1,2)\}$ be the set of destination nodes.

First, $\boldsymbol{s}_2, \boldsymbol{s}_5$ and $\boldsymbol{d}_2, \boldsymbol{d}_5$ are inside the same sub torus $T_{2,3}(1)$, hence this is Case 1, and two disjoint paths connecting $\boldsymbol{s}_2, \boldsymbol{s}_5$ and $\boldsymbol{d}_2, \boldsymbol{d}_5$ are obtained.

Next, route remaining source and destination nodes towards $T_{2,3}(2)$ in at most 2 edges: the paths $\boldsymbol{s}_1 \to (0,1,0) \to (0,1,2)$, $\boldsymbol{s}_4 \to (2,1,0) \to (2,1,2)$ and $\boldsymbol{d}_1 \to (1,2,0) \to (1,2,2)$, $\boldsymbol{d}_4 \to (2,1,0) \to (2,1,2)$ are selected ($\boldsymbol{s}_3, \boldsymbol{s}_6$ and $\boldsymbol{d}_3, \boldsymbol{d}_6$ already in $T_{2,3}(2)$).

Third, the paths from $\boldsymbol{s}_4$ and $\boldsymbol{d}_4$ encounter, hence $\boldsymbol{s}_4$ is connected to $\boldsymbol{d}_4$ with the path $\boldsymbol{s}_4 \to (2,1,0) \to \boldsymbol{d}_4$ and the edge $(2,1,0) \to (2,1,2)$ is discarded.

Fourth, the algorithm is applied recursively in $T_{2,3}(2)$ with $\boldsymbol{s}'_1 = (0,1,2), \boldsymbol{s}'_3 = \boldsymbol{s}_3 = (0,0,2), \boldsymbol{s}'_6 = \boldsymbol{s}_6 = (2,0,2)$ as source nodes and $\boldsymbol{d}'_1 = (1,2,2), \boldsymbol{d}'_3 = \boldsymbol{d}_3 = (1,0,2), \boldsymbol{d}'_6 = \boldsymbol{d}_6 = (1,1,2)$ as destination nodes. The sub torus $T_{1,3}(0)$ includes $\boldsymbol{s}'_3, \boldsymbol{s}'_6, \boldsymbol{d}'_3$, hence this is Case 3.

The node $\boldsymbol{d}'_1$ is routed disjointly to $T_{1,3}(0)$ with the path $\boldsymbol{d}'_1 \to (2,2,2) \to (2,0,2) = \boldsymbol{d}''_1$. Two disjoint paths inside $T_{1,3}(0)$ connecting $\boldsymbol{s}'_3, \boldsymbol{s}'_6, \boldsymbol{d}'_3, \boldsymbol{d}''_1$ are found, and thus the nodes $\boldsymbol{s}_3, \boldsymbol{s}_6, \boldsymbol{d}_3, \boldsymbol{d}_1$ are disjointly connected.

The remaining source node $\boldsymbol{s}'_1$ and destination node $\boldsymbol{d}'_6$ are routed to $T_{1,3}(2)$ with the paths $\boldsymbol{s}'_1 = (0,1,2) \to (0,2,2)$ and $\boldsymbol{d}'_6 = (1,1,2) \to (0,1,2) \to (0,2,2)$. These two paths encounter, and thus $\boldsymbol{s}'_1$ is connected to $\boldsymbol{d}'_6$, and the edge $(0,1,2) \to (0,2,2)$ is discarded. In other words, $\boldsymbol{s}_1$ has been disjointly connected to $\boldsymbol{d}_6$, which solves this routing problem.

4 Evaluation

In this section, we formally establish the complexities of the proposed set-to-set disjoint paths routing algorithm in a torus S2S. We assume that a node address can be stored in a fixed number of machine

words, and that therefore comparison can be done in constant time. Let $\tau(n,k)$ and $\lambda(n,k)$ represent the time complexity of the algorithm S2S when applied in a $T_{n,k}$, and the maximum length of the obtained paths, respectively. We shall rigorously go through each step of the three distinguished cases.

Lemma 7. *In Step 1 of Case 1, we can find 2 disjoint paths not including any of the other source or destination nodes in $T_{n-1,k}(0)$ and not including multiple neighbour nodes of each of them in $O(kn^2)$ time. The maximum path length is $k(n-1)$.*

Proof. From Lemma 4, we can construct two paths of lengths at most $k(n-1)$ in $O(kn)$ time. For each of these paths, we can check if it contains the other source or destination nodes in $T_{n-1,k}(0)$ and if it contains multiple neighbour nodes of each of them in $O(kn) \times O(n) = O(kn^2)$. Selection of the minimal sub path can be done in $O(1)$ time. Then, the total time complexity is $O(kn^2)$. □

Lemma 8. *In Step 2 of Case 1, we can find disjoint paths from the source nodes $\boldsymbol{s}_i$ in $T_{n-1,k}(0)$ other than $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ to $T_{n-1,k}(k-1)$ in $O(kn^2)$ time. The maximum path length is k.*

Proof. For the $(2n-1)$ paths of lengths at most 2 from $\boldsymbol{s}_i$ to $T_{n-1,k}(k-1)$, we can check if there is a disjoint path from other source nodes or the paths from them in $O(kn)$ time. If there is a disjoint path, we can select it. Otherwise, if such a path does not exist, that is, if all the paths are blocked, the paths from the other source nodes $\boldsymbol{s}_j$ $(3 \leq j \neq i \leq 2n)$ or the nodes themselves must block the $(2n-3)$ paths from $\boldsymbol{s}_i$ to $T_{n-1,k}(k-1)$. Note that in this situation all the source nodes are included in either $T_{n-1,k}(0)$ or $T_{n-1,k}(k-1)$. Also, the paths $\boldsymbol{s}_1 \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{s}_2 \rightsquigarrow \boldsymbol{d}_2$ must include two neighbour nodes $\hat{\boldsymbol{s}}_i$ of $\boldsymbol{s}_i$ that correspond to the remaining two paths. We can find $\hat{\boldsymbol{s}}_i$ in $O(kn)$ time. Then, from $k \geq 3$, we can construct a disjoint path $\boldsymbol{s}_i \to \boldsymbol{s}_i \oplus \boldsymbol{e}_n \to \hat{\boldsymbol{s}}_i \oplus \boldsymbol{e}_n \to \cdots \to \boldsymbol{s}'_i$ of length at most k from $\boldsymbol{s}_i$ to $T_{n-1,k}(k-1)$ in $O(k)$ time. Hence, for all the nodes $\boldsymbol{s}_i$, we can construct disjoint paths in $O(kn^2)$ in total, and the maximum path length is k. □

Similarly, we can deduce the following lemma.

Lemma 9. *In Step 3 of Case 1, we can construct disjoint paths from the destination nodes $\boldsymbol{d}_i$ in $T_{n-1,k}(0)$ other than $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ to $T_{n-1,k}(k-1)$ in $O(kn^2)$ time. The maximum path length is k.*

Lemma 10. *In Step 4 of Case 1, we can construct disjoint paths from the source nodes $\boldsymbol{s}_i$ ($\notin T_{n-1,k}(0)$) to $T_{n-1,k}(k-1)$ in $O(n^2 \log n)$ time. The maximum path length is $k-1$.*

Proof. For the $(2n-1)$ paths of lengths at most $k-1$ from $\boldsymbol{s}_i$ to $T_{n-1,k}(k-1)$, we can find a disjoint path from other source nodes or the paths from them by checking the occupied classes by $\boldsymbol{s}_j$ $(3 \leq j \neq i \leq 2n)$ in $O(n \log n)$ time. So in total, we can construct disjoint paths for all of the source nodes not included in $T_{n-1,k}(0)$ in $O(n^2 \log n)$ time. □

Similarly, we can deduce the following lemma.

Lemma 11. *In Step 5 of Case 1, we can find disjoint paths from the destination nodes $\boldsymbol{d}_i$ ($\notin T_{n-1,k}(0)$) to $T_{n-1,k}(k-1)$ in $O(n^2 \log n)$ time. The maximum path length is $k-1$.*

And finally, the last step of Case 1:

Lemma 12. *In Step 6 of Case 1, we can connect the paths from the source and destination nodes in $\tau(n-1,k)+O(n^2)$ time. The maximum length of the connected paths is $\max(k+1, \lambda(n-1,k)+2k)$.*

Proof. Since the paths from the source and destination nodes are in similar structure – i.e. for a path, the nodes on that path are either all included in a class or all included in a class except for one (that is, the source or destination node) –, we can find their overlapping by checking their classes in $O(n^2)$ time. The lengths of the paths obtained by connection are at most $k+1$. The application of the algorithm in $T_{n-1,k}(k-1)$ takes $\tau(n-1,k)$ time and the lengths of the paths generated are at most $\lambda(n-1,k)$. Then, the lengths of the connected paths from the source nodes to the destination nodes are at most $\lambda(n-1,k)+2k$. Hence, Step 6 in Case 1 takes $\tau(n-1,k)+O(n^2)$ time to connect the paths from the source and destination nodes. The maximum path length is $\max(k+1, \lambda(n-1,k)+2k)$. □

This discussion for Case 1 is summarised in the following theorem.

Theorem 1. *In an n-dimensional k-ary torus $T_{n,k}$, for a set of $2n$ source nodes S and a set of $2n$ destination nodes D, assume that at least two pairs of the source and destination nodes are included in a sub torus. We can find $2n$ mutually node-disjoint paths between S and D in $O(kn^3 + n^3 \log n)$ time. The maximum length of the selected paths is $2kn$.*

Proof. From Lemmas 7, 8, 9, 10, 11 and 12, we can find $2n$ disjoint paths between S and D in $\tau(n,k) = \tau(n-1,k) + O(kn^2 + n^2 \log n)$ time and the maximum path length is $\lambda(n,k) = \max(k+1, k(n-1), \lambda(n-1,k) + 2k)$. Then, $\tau(n,k) = O(kn^3 + n^3 \log n)$ and $\lambda(n,k) = 2kn$ hold. □

From Lemma 6, we can derive the following two lemmas.

Lemma 13. *In Step 1 of Case 2, we can construct two paths from two source nodes to $T_{n-1,k}(0)$ and $(2n-2)$ paths from the remaining $(2n-2)$ source nodes to $T_{n-1,k}(k-1)$ so that all of the paths are mutually disjoint in $O(n^2 \log n)$ time. The maximum path length is k.*

Lemma 14. *In Step 2 of Case 2, we can construct two paths from two destination nodes to $T_{n-1,k}(0)$ and $(2n-2)$ paths from the remaining $(2n-2)$ destination nodes to $T_{n-1,k}(k-1)$ so that all of the paths are mutually disjoint in $O(n^2 \log n)$ time. The maximum path length is k.*

Lemma 15. *In Step 3 of Case 2, we can check if two paths $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i (\in T_{n-1,k}(0))$ and $\boldsymbol{d}_j \rightsquigarrow \boldsymbol{d}'_j (\in T_{n-1,k}(0))$ encounter, or if two paths $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i (\in T_{n-1,k}(k-1))$ and $\boldsymbol{d}_j \rightsquigarrow \boldsymbol{d}'_j (\in T_{n-1,k}(k-1))$ encounter in $O(n^2)$ time. The maximum length of the connected paths is $k+1$.*

Proof. Since the paths from the source and destination nodes are in similar structure, we can find their overlapping by checking their classes $C(\boldsymbol{s}_i), C(\boldsymbol{s}'_i), C(\boldsymbol{d}_j), C(\boldsymbol{d}'_j)$ in $O(n^2)$ time. The lengths of the paths obtained by connection are at most $k+1$. □

Lemma 16. *In Step 4 of Case 2, we can check if two paths $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i (\in T_{n-1,k}(0))$ and $\boldsymbol{d}_j \rightsquigarrow \boldsymbol{d}'_j (\in T_{n-1,k}(k-1))$ encounter in $O(n)$ time. The maximum length of the connected paths is k.*

Proof. As there are only 2 paths $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i (\in T_{n-1,k}(0))$, and the paths from the source and destination nodes are in similar structure, we can find their overlapping by checking their classes $C(\boldsymbol{s}_i), C(\boldsymbol{s}'_i), C(\boldsymbol{d}_j), C(\boldsymbol{d}'_j)$ in $O(n)$ time. The lengths of the paths obtained by connection are at most k. □

Lemma 17. *In Step 5 of Case 2, we can construct a path that does not include any of the other source or destination nodes in $T_{n-1,k}(0)$ and does not include multiple neighbour nodes of each of them in $O(kn)$ time. The path length is at most $\lfloor k/2 \rfloor (n-1)$.*

Proof. In an $T_{n-1,k}(0)$, we can construct a shortest path of length at most $\lfloor k/2 \rfloor (n-1)$ in $O(kn)$ time. We can check if the path contains the other source or destination nodes in $T_{n-1,k}(0)$ and if it contains multiple neighbour nodes of each of them in $O(kn) \times O(1) = O(kn)$ time. Selection of the minimal sub path can be done in $O(1)$ time. Then, the total time complexity is $O(kn)$. □

Lemma 18. *In Step 6 of Case 2, we can construct a disjoint path from $\boldsymbol{d}'_i$ to $T_{n-1,k}(k-1)$ of length at most 2 in $O(n \log n)$.*

Proof. Without loss of generality, we can assume that $i = 2$. Since $|T_{n-1,k}(0) \cap D| < 2$, there is no destination node in $T_{n-1,k}(0)$ other than $\boldsymbol{d}_1$. Also, if $\boldsymbol{d}_j \in T_{n-1,k}(k-1)$, it means that $\boldsymbol{s}_2 \in T_{n-1,k}(k-1)$, too. However, from the proof of Lemma 6, $\boldsymbol{s}_2, \boldsymbol{s}_3, \ldots, \boldsymbol{s}_{2n} \in T_{n-1,k}(k-1)$ holds, and this is a contradiction that $|T_{n-1,k}(k-1) \cap S| \leq 2n-2$. Therefore, $\boldsymbol{d}_j \notin T_{n-1,k}(k-1)$. Hence, if we consider $(2n-1)$ disjoint paths $\tilde{P}_l(\boldsymbol{d}'_2)$ $(1 \leq l \leq 2n-1)$ of lengths at most 2 from $\boldsymbol{d}'_2$ to $T_{n-1,k}(k-1)$ from Lemma 3, there is at least one path that does not contain any other destination node and that is disjoint from the paths from the other destination nodes. We can find this path by checking the occupied classes by the other destination nodes in $O(n \log n)$ time. □

From Lemma 4, the following lemma is derived.

Lemma 19. *In Step 7 of Case 2, we can construct two disjoint paths of lengths at most $k(n-1)$ between $\{\boldsymbol{s}'_1, \boldsymbol{s}'_2\}$ and $\{\boldsymbol{d}'_1, \boldsymbol{d}'_2\}$ in $O(kn)$ time.*

And finally, the last step of Case 2:

Lemma 20. *In Step 8 of Case 2, we can connect the paths from the source and destination nodes in $\tau(n-1,k)$ time. The maximum length of the connected paths is $\lambda(n-1,k)+2k+2$.*

Proof. The application of the algorithm S2S in $T_{n-1,k}(k-1)$ takes $\tau(n-1,k)$ time and the lengths of the paths generated are at most $\lambda(n-1,k)$. From Lemma 13, the lengths of the paths from $\boldsymbol{s}_i$ to $T_{n-1,k}(k-1)$ are at most k. From Lemmas 14 and 18, the lengths of the paths from $\boldsymbol{d}_i$ to $T_{n-1,k}(k-1)$ are at most $k+2$. Then, the lengths of the connected paths from the source nodes to the destination nodes are at most $\lambda(n-1,k)+2k+2$. □

This discussion for Case 2 is summarised in the following theorem.

Theorem 2. *In an n-dimensional k-ary torus $T_{n,k}$, for a set of $2n$ source nodes S and a set of $2n$ destination nodes D, assume that there is not any sub torus that contains at least two pairs of the source and destination nodes. We also assume that $|T_{n-1,k}(k-1) \cap S| \leq 2n-2$ and $|T_{n-1,k}(k-1) \cap D| \leq 2n-2$, as well as $|T_{n-1,k}(0) \cap S| < 2$ and $|T_{n-1,k}(k-1) \cap D| < 2$. Then, we can select $2n$ mutually node-disjoint paths between S and D in $O(kn^2 + n^3 \log n)$ time. The maximum length of the selected paths is $2(k+1)n$.*

Proof. From Lemmas 13, 14, 15, 16, 17, 18, 19, and 20, we can construct $2n$ disjoint paths between S and D in $\tau(n,k) = \tau(n-1,k) + O(kn + n^2 \log n)$ time and the maximum path length is $\lambda(n,k) = \max(k+1, k, \lfloor k/2 \rfloor (n-1), k(n-1), \lambda(n-1,k)+2k+2)$. Then, $\tau(n,k) = O(kn^2 + n^3 \log n)$ and $\lambda(n,k) = 2(k+1)n$ hold. □

Next, Case 3 is analysed. From Lemma 6, we can derive the following lemma.

Lemma 21. *In Step 1 of Case 3, we can construct two paths from two source nodes to $T_{n-1,k}(0)$ and $(2n-2)$ paths from the remaining $(2n-2)$ source nodes to $T_{n-1,k}(k-1)$ so that all of the paths are mutually disjoint in $O(n^2 \log n)$ time. The maximum path length is k.*

Lemma 22. *In Step 2 of Case 3, we can construct two disjoint paths that do not include any of the other destination nodes in $T_{n-1,k}(0)$ and do not include multiple neighbour nodes of each of them in $O(kn^2)$ time. The maximum path length is $k(n-1)$.*

Proof. From Lemma 4, we can construct two disjoint paths $\boldsymbol{s}'_i \rightsquigarrow \boldsymbol{d}_i$ $(i = 1, 2)$ of lengths at most $k(n-1)$ in $O(kn)$ time. For each of these paths, we can check if it contains the other destination nodes in $T_{n-1,k}(0)$ and if it contains multiple neighbour nodes of each of them in $O(kn) \times O(n) = O(kn^2)$ time. Selection of the minimal sub path can be done in $O(1)$ time. Then, the total time complexity is $O(kn^2)$. □

Lemma 23. *In Step 3 of Case 3, we can construct disjoint paths from the destination nodes $\boldsymbol{d}_i$ in $T_{n-1,k}(0)$ other than $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ to $T_{n-1,k}(k-1)$ in $O(kn^2)$ time. The maximum path length is k.*

Proof. For the $(2n-1)$ paths of lengths at most 2 from $\boldsymbol{d}_i$ to $T_{n-1,k}(k-1)$, we can check if there is a disjoint path from other destination nodes or the paths from them in $O(kn)$ time. If there is a disjoint path, we can select it. Otherwise, if such a path does not exist, that is, if all the paths are blocked, the paths from the other source nodes $\boldsymbol{d}_j$ $(3 \leq j \neq i \leq 2n)$ or the nodes themselves must block the $(2n-3)$ paths from $\boldsymbol{d}_i$ to $T_{n-1,k}(k-1)$. Note that in this situation all the source nodes are included in either $T_{n-1,k}(0)$ or $T_{n-1,k}(k-1)$. Also, the paths $\boldsymbol{s}_1 \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{s}_2 \rightsquigarrow \boldsymbol{d}_2$ must include two neighbour nodes $\hat{\boldsymbol{d}}_i$ of $\boldsymbol{d}_i$ that correspond to the remaining two paths. We can find such node $\hat{\boldsymbol{d}}_i$ in $O(kn)$ time. Then, from $k \geq 3$, we can construct a disjoint path $\boldsymbol{d}_i \rightarrow \boldsymbol{d}_i \oplus \boldsymbol{e}_n \rightarrow \hat{\boldsymbol{d}}_i \oplus \boldsymbol{e}_n \rightarrow \cdots \rightarrow \boldsymbol{d}'_i$ of length at most k from $\boldsymbol{d}_i$ to $T_{n-1,k}(k-1)$ in $O(k)$ time. Hence, for all the nodes $\boldsymbol{d}_i$, we can construct disjoint paths in $O(kn^2)$ time in total, and the maximum path length is k. □

Lemma 24. *In Step 4 of Case 3, we can construct disjoint paths from the destination nodes* $\boldsymbol{d}_i(\notin T_{n-1,k}(0))$ *to* $T_{n-1,k}(k-1)$ *in* $O(n^2 \log n)$ *time. The maximum path length is* $k-1$.

Proof. For the $(2n-1)$ paths of lengths at most $k-1$ from $\boldsymbol{d}_i$ to $T_{n-1,k}(k-1)$, we can find a path disjoint from other destination nodes or the paths from them by checking the occupied classes by $\boldsymbol{d}_j$ $(3 \leq j \neq i \leq 2n)$ in $O(n \log n)$ time. So in total, we can find disjoint paths for all of the destination nodes not included in $T_{n-1,k}(0)$ in $O(n^2 \log n)$ time. □

Lemma 25. *In Step 5 of Case 3, we can check if two paths* $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i(\in T_{n-1,k}(k-1))$ *and* $\boldsymbol{d}_j \rightsquigarrow \boldsymbol{d}'_j(\in T_{n-1,k}(k-1))$ *encounter in* $O(n^2)$ *time. The maximum length of the connected paths is* $k+1$.

Proof. Since the paths from the source and destination nodes are in similar structure, we can find their overlapping by checking their classes $C(\boldsymbol{s}_i), C(\boldsymbol{s}'_i), C(\boldsymbol{d}_j), C(\boldsymbol{d}'_j)$ in $O(n^2)$ time. The lengths of the paths obtained by connection are at most $k+1$. □

Lemma 26. *In Step 6 of Case 3, we can check if two paths* $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i(\in T_{n-1,k}(0))$ *and* $\boldsymbol{d}_j \rightsquigarrow \boldsymbol{d}'_j(\in T_{n-1,k}(k-1))$ *encounter in* $O(n)$. *The maximum length of the connected paths is* k.

Proof. Since there are only two paths $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i(\in T_{n-1,k}(0))$, and the paths from the source and destination nodes are in similar structure, we can find their overlapping by checking their classes $C(\boldsymbol{s}_i), C(\boldsymbol{s}'_i), C(\boldsymbol{d}_j), C(\boldsymbol{d}'_j)$ in $O(n)$ time. The lengths of the paths obtained by connection are at most k. □

Lemma 27. *In Step 7 of Case 3, we can construct a disjoint path from* $\boldsymbol{d}'_2$ *to* $T_{n-1,k}(k-1)$ *of length at most 2 in* $O(n \log n)$.

Proof. Note that if this step is executed, the paths $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{s}'_i$ $(\in T_{n-1,k}(0))$ and $\boldsymbol{d}_j \rightsquigarrow \boldsymbol{d}'_j$ $(\in T_{n-1,k}(k-1))$ encounter. If $\boldsymbol{d}_j \notin T_{n-1,k}(0)$, $(2n-1)$ disjoint paths $\overleftarrow{P}_l(\boldsymbol{d}'_i)$ $(1 \leq l \leq 2n-1)$ and $\boldsymbol{s}_i \rightsquigarrow \boldsymbol{d}_j$ are disjoint. Therefore, at least one path exists among them that is disjoint from the other destination nodes and the paths from them. If $\boldsymbol{d}_j \in T_{n-1,k}(0)$, $\boldsymbol{d}_j \rightsquigarrow \boldsymbol{d}'_j$ is the path $\boldsymbol{d}_j \to \boldsymbol{d}_j \oplus \boldsymbol{e}_n \to \hat{\boldsymbol{d}}_j \oplus \boldsymbol{e}_n \to \ldots \to \boldsymbol{d}'_j(\in T_{n-1,k}(k-1))$. Then, $(2n-1)$ disjoint paths $\overleftarrow{P}_l(\boldsymbol{d}_j)$ $(1 \leq l \leq 2n-1)$ are blocked, and it means that the node in $C(\boldsymbol{d}_j) \cap T_{n-1,k}(k-1)$ is the destination node. Therefore, $C(\boldsymbol{d}_j)$ contains multiple destination nodes. Among $(2n-1)$ disjoint paths $\overleftarrow{P}_l(\boldsymbol{d}'_i)$ $(1 \leq l \leq 2n-1)$ of lengths at most 2, there is at least one path that is disjoint from the other destination nodes and the paths from them. In either case, we can find the path in $O(n \log n)$ time by checking the classes occupied by the destination nodes. □

And finally, the last step of Case 3:

Lemma 28. *In Step 8 of Case 3, we can connect the paths from the source and destination nodes in* $\tau(n-1,k)$ *time. The maximum length of the connected paths is* $\lambda(n-1,k)+2k$.

Proof. The application of the algorithm S2S in $T_{n-1,k}(k-1)$ takes $\tau(n-1,k)$ time and the lengths of the paths generated are at most $\lambda(n-1,k)$. From Lemma 21, the lengths of the paths from $\boldsymbol{s}_i$ to $T_{n-1,k}(k-1)$ are at most k. From Lemma 23, the lengths of the paths from $\boldsymbol{d}_i$ to $T_{n-1,k}(k-1)$ are at most k. So, the lengths of the connected paths from the source nodes to the destination nodes are at most $\lambda(n-1,k)+2k$. □

This discussion for Case 3 is summarised in the following theorem.

Theorem 3. *In an* n*-dimensional* k*-ary torus* $T_{n,k}$*, for a set of* $2n$ *source nodes* S *and a set of* $2n$ *destination nodes* D*, assume that there is not any sub torus that contains at least two pairs of the source and destination nodes. We also assume that* $|T_{n-1,k}(k-1) \cap S| \leq 2n-2$ *and* $|T_{n-1,k}(k-1) \cap D| \leq 2n-2$*, as well as* $|T_{n-1,k}(0) \cap S| < 2$ *and* $|T_{n-1,k}(k-1) \cap D| \geq 2$*. Then, we can select* $2n$ *mutually node-disjoint paths between* S *and* D *in* $O(kn^3 + n^3 \log n)$ *time. The maximum length of the selected paths is* $2kn$.

Proof. From Lemmas 21, 22, 23, 24, 25, 26, 27, and 28, we can construct $2n$ disjoint paths between S and D in $\tau(n,k) = \tau(n-1,k) + O(kn^2 + n^2 \log n)$ time and the maximum path length is $\lambda(n,k) = \max(k(n-1), k+1, k, \lambda(n-1,k) + 2k)$. Then, $\tau(n,k) = O(kn^3 + n^3 \log n)$ and $\lambda(n,k) = 2kn$ hold. □

From Theorems 1, 2, and 3, the main theorem is deduced.

Theorem 4. *In an n-dimensional k-ary torus $T_{n,k}$, for a set of $2n$ source nodes S and a set of $2n$ destination nodes D, we can construct $2n$ mutually node-disjoint paths between S and D in $O(kn^3 + n^3 \log n)$ time. The maximum length of the constructed paths is $2(k+1)n$.*

5 Conclusions and Future Works

The torus topology is one of the most popular interconnection network for modern massively parallel systems. Many world-famous supercomputers rely on it: Fujitsu K, IBM Blue Gene/L and Blue Gene/P, Cray XT3, etc. In this paper, we have proposed an algorithm that constructs mutually node-disjoint paths between two sets of nodes in an n-dimensional k-ary torus. This problem is critical for efficient data transmission and system dependability. We have proved that the time complexity of the described routing algorithm is $O(kn^3 + n^3 \log n)$, and that the maximum path length is $2(k+1)n$.

Regarding future works, the authors are aware there might be a possibility to reduce the maximum path length by carefully selecting the sub torus to concentrate the source and destination nodes. It is also included in the future works to implement our algorithm for evaluation.

References

[1] TOP500. China's Tianhe-2 Supercomputer Maintains Top Spot on List of World's TOP500 Supercomputers, 2015. `https://www.top500.org/news/lists/2015/11/press-release/`. Last accessed January 2017.

[2] San Murugesan. Harnessing green IT: Principles and practices. *IT Professional*, 10(1):24–33, February 2008.

[3] Green500. November 2016 List. `https://www.top500.org/green500/`. Last accessed January 2017.

[4] Alok Aggarwal, Jon Kleinbergt, David P. Williamson. Node-Disjoint Paths on the Mesh and a New Trade-Off in VLSI Layout. *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 585–594, Philadelphia, PA, USA, May 22–24, 1996.

[5] Yamin Li, Shietung Peng, Wanming Chu. Disjoint-Paths and Fault-Tolerant Routing on Recursive Dual-Net. *International Journal of Foundations of Computer Science*, 22(5):1001–1018, 2011.

[6] Yamin Li, Shietung Peng, Wanming Chu. Node-to-Set Disjoint-Paths Routing in Recursive Dual-Net. *International Journal of Networking and Computing*, 1(2):178–190, 2011.

[7] Keiichi Kaneko and Shietung Peng. Set-to-set Disjoint Paths Routing in Dual-cubes. *Proceedings of the Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 129–136, Dunedin, Otago, New Zealand, December 1–4, 2008.

[8] Yamin Li, Shietung Peng, Wanming Chu. Set-to-Set Disjoint-Paths Routing in Recursive Dual-Net. *Proceedings of the 11th International Conference on Algorithms and Architectures for Parallel Processing*, volume 1, pages 54–65, Melbourne, Australia, October 24–26, 2011.

[9] Shietung Peng and Keiichi Kaneko. Set-to-set disjoint paths routing in pancake graphs. Proceedings of the 18th International Conference on Parallel and Distributed Computing and Systems, pages 253–258, Dallas, TX, USA, November 13–15, 2006.

[10] Qiang-Ping Gu and Shietung Peng. Set-to-set fault tolerant routing in star graphs. *IEICE Transactions on Information & Systems*, E79-D(4):282–289, 1996.

[11] Qian-Ping Gu, Satoshi Okawa and Shietung Peng. Set-to-set fault tolerant routing in hypercubes. *IEICE Transactions on Fundamentals*, E79-A(4):483–488, 1996.

[12] Qian-Ping Gu and Shietung Peng. Node-to-set and set-to-set cluster fault tolerant routing in hypercubes. *Parallel Computing*, 24(8):1245–1261, 1998.

[13] Antoine Bossard and Keiichi Kaneko. Time Optimal Node-to-Set Disjoint Paths Routing in Hypercubes. *Journal of Information Science and Engineering*, 30(4):1087–1093, 2014.

[14] Keiichi Kaneko and Yasuto Suzuki. An Algorithm for Node-to-Set Disjoint Paths Problem in Rotator Graphs. IEICE Transactions on Information and Systems, E84-D(9):1155–1163, 2001.

[15] Yamin Li, Shietung Peng and Wanming Chu. Metacube—a versatile family of interconnection networks for extremely large-scale supercomputers. The Journal of Supercomputing 53(2):329–351, 2010.

[16] Jose Duato, Sudhakar Yalamanchili, and Lionel Ni. *Interconnection networks: an engineering approach*, Morgan Kaufmann, 2003.

[17] Antoine Bossard and Keiichi Kaneko. A Set-to-Set Disjoint Paths Routing Algorithm in a Torus-Connected Cycles Network. Proceedings of the 31st International Conference on Computers and Their Applications, pages 81–88, Las Vegas, NV, USA, April 4–6, 2016.