sensors

MDPI

Article

Cluster-Fault Tolerant Routing in a Torus

Antoine Bossard [1,*] and Keiichi Kaneko [2]

[1] Graduate School of Science, Kanagawa University, Kanagawa 259-1293, Japan
[2] Institute of Engineering, Tokyo University of Agriculture and Technology, Tokyo 184-8588, Japan; k1kaneko@cc.tuat.ac.jp
* Correspondence: abossard@kanagawa-u.ac.jp

Received: 14 May 2020; Accepted: 3 June 2020; Published: 9 June 2020

Abstract: The number of Internet-connected devices grows very rapidly, with even fears of running out of available IP addresses. It is clear that the number of sensors follows this trend, thus inducing large sensor networks. It is insightful to make the comparison with the huge number of processors of modern supercomputers. In such large networks, the problem of node faults necessarily arises, with faults often happening in clusters. The tolerance to faults, and especially cluster faults, is thus critical. Furthermore, thanks to its advantageous topological properties, the torus interconnection network has been adopted by the major supercomputer manufacturers of the recent years, thus proving its applicability. Acknowledging and embracing these two technological and industrial aspects, we propose in this paper a node-to-node routing algorithm in an n-dimensional k-ary torus that is tolerant to faults. Not only is this algorithm tolerant to faulty nodes, it also tolerates faulty node clusters. The described algorithm selects a fault-free path of length at most $n(2k + \lfloor k/2 \rfloor - 2)$ with an $O(n^2k^2|F|)$ worst-case time complexity with F the set of faulty nodes induced by the faulty clusters.

Keywords: fault tolerance; sensor; IoT; algorithm; information dissemination; interconnection network

1. Introduction

As mentioned, for instance, in [1,2], the number of Internet-connected devices is seeing a very rapid growth, with even fears of running out of available IP addresses. It is clear that the number of sensors follows this trend, thus inducing large sensor networks. The interconnection issue of sensor networks is thus critical. Given the large number of sensors involved, it is critical that these interconnection networks come with efficient data communication algorithms to maximise the performance of the sensor network. Because hardware failure is highly probable given the scale of the network, the tolerance to faults by such routing algorithms is key to data communication efficiency and robustness.

Considering the number of network nodes involved, it is insightful to make the comparison with the number of processors of modern supercomputers. In such large networks of processors or sensors, the problem of faulty nodes necessarily arises, with faults often happening in clusters. The tolerance to faults, and especially cluster faults, is thus critical as detailed below. Featuring advantageous topological properties, the torus network [3] has become very popular as interconnection network of supercomputers (e.g., IBM Blue Gene/L and Blue Gene/P, Cray Titan (Gemini interconnect [4]) and Fujitsu K (Tofu interconnect [5])) [6], thus proving the applicability of the torus topology for large network interconnection. Such topological properties of an n-dimensional k-ary torus include the network degree $2n$, the diameter $n\lfloor k/2 \rfloor$ and the network order (i.e., number of nodes) k^n. This is to be compared with, for instance, the hypercube network [7] (n-cube) that was favoured for earlier supercomputers (e.g., the Cosmic cube [8]), of degree and diameter n and network order

2^n. So, given the physical restrictions on the number of links per node, that is, on the network degree, a torus is able to connect far more nodes than a hypercube: a small dimension n keeps the degree low, while its arity k can be adjusted to fit the number of nodes. The formal definition of a torus is given in the next section. Tori are also used for hierarchical interconnection networks [9].

From Menger's theorem [10], node-to-node routing can tolerate at most $d - 1$ faulty nodes, with d the network degree. By considering a special class of network, here tori, we show that more faults can be tolerated for such a routing scenario. More precisely, in this paper we describe a node-to-node torus routing algorithm under the cluster-fault tolerant model: in addition to faulty nodes, faulty clusters of diameter at most one are considered. This furthermore induces new conditions on the number of tolerable faults in such a network, thus refining the conditions stated by Menger's theorem. And thanks to this cluster-fault tolerant approach, the proposed algorithm is also tolerant to edge faults: given a faulty edge (u, v), it suffices to declare the two nodes u, v as one faulty cluster.

Given the number of nodes k^n and edges nk^n involved in an n-dimensional k-ary torus, solving this cluster-fault tolerant routing problem with a conventional routing algorithm such as Dijkstra's is clearly impractical: its worst-case time complexity is of polynomial order in the number of nodes or edges of the network, and the same discussion holds with a breadth-first search algorithm [11]. It is thus an effective, and as shown in Section 2 a commonly relied upon approach to describe a routing algorithm for a specific class of graph, here a torus, to provide a practical solution to this routing problem.

As recalled earlier, large sensor networks are nowadays common, and they include thousands of network nodes (i.e., sensor nodes). Fault-tolerance (resilience) and scalability are two key requirements of such networks [12]. This research on the torus interconnection network brings a concrete solution to these issues as detailed below. Indeed, we can rely on the torus network topology to interconnect the sensors, which are themselves attached to "things" of the Internet of things (IoT). In this situation, we assume that these things are static, that is, not moving. In order to report the data collected across the sensor network by each sensor on its thing to the requesting device (the user) which is connected via the Internet to one network node (the sink), the data are delivered through the torus to the requester by forwarding the data according to the paths calculated by the proposed algorithm, paths which connect each sensor node to the (current) network sink.

First, from its definition, and precisely the node degree that does not depend on the network arity, the torus topology provides a high scalability, for instance when compared to other network topologies such as hypercubes [7] and star graphs [13]. Second, considering the large number of sensors involved, faulty sensors are unavoidable. Besides, it is usually expected that the sensor faults take place in a cluster rather than occurring independently. By describing a routing algorithm that is tolerant to faults, the resilience and data transmission performance of the sensor network are increased, which thus offers a higher quality-of-service. Third, with such a connection approach, a distributed sensor network can be easily realised: as mentioned above, a master device accesses the sensor network from the Internet, which further increases the network resilience as none of its nodes has special powers.

In addition to increasing the fault tolerance of data transfers and thus of the whole network, it is important to consider the cluster-fault tolerant model given hardware technical properties of the machinery: it is indeed not rare that faults in such a system occur in clusters. For example, one same electrical unit supplies power to a few nodes. When this power unit fails—battery lifetime is a notorious issue of sensor nodes [14]—the corresponding nodes induce a faulty cluster. As another example, several nodes can be managed by one same hosting or controlling device, like two IoT things which depend on one controlling unit, or a blade inside a cabinet of a supercomputer. Failure of the hosting device would similarly result in a faulty cluster.

Smart agriculture is one application field of our research results. For example, in south-east Asian countries paddy fields are popular for the culture of rice. Such paddy fields are wide areas immersed in water (see Figure 1a). The implementation of smart agriculture without interfering with the automated farm operations, a wireless sensor network is suitable to collect information about the

paddy field. Sensors in the field are positioned according to the points of a two-dimensional lattice and each sensor has a transmission range so that it can at least communicate with its four neighbour sensors. In addition, the wrap-around edges that connect the sensors placed on the periphery of the paddy field are implemented with wires. Consequently, the sensor network forms a two-dimensional torus structure (see Figure 1b).

Another example with respect to smart agriculture is a plant factory [15,16]. In such a facility, plant pods are organised in a three-dimensional manner and cultivated under optimal growth conditions. In this situation as well, a wireless sensor network is suitable to not interfere with the factory automated operations. Precisely, in the factory the sensors are positioned according to the points of a three-dimensional lattice and each sensor has a transmission range so that it can at least communicate with its six neighbour sensors. In addition, on the floor, walls and ceiling, the wrap-around edges are implemented with wires. As a result, the sensor network forms a three-dimensional torus structure.

(**a**) (**b**)

Figure 1. (**a**) A paddy field with young crops. (**b**) A sensor network for a paddy field where the small circles represent sensors. The orange disc represents the transmission range of the sensor at its centre. The wireless and wired interconnection network forms a two-dimensional torus structure.

The rest of this paper is organised as follows. Previous and related works are discussed in Section 2. Graph notations and definitions together with lemmas and propositions are recalled and established in Section 3. The proposed algorithm is described and exemplified in Section 4. The algorithm correctness is formally established in Section 5; this is a major part of the paper. Complexity analysis, precisely the maximum path length and worst-case time complexity, is formally conducted in Section 6 to evaluate the theoretical performance of the algorithm, and from which the main theorem of this research is induced. Then, in Section 7, the algorithm performance in average is empirically evaluated with computer experimentations and compared to the theoretical values. Finally, this paper is concluded in Section 8.

2. Previous and Related Works

The torus topology is often presented as an extension of the mesh network [3] to which "wrap-around edges" have been added and it has been itself further extended [17], for instance to design a hierarchical interconnection network [9].

There exist a few routing algorithms that are tolerant to faults which have been described for a torus network. Torus fault-tolerant routing algorithms based on the simple node-fault tolerant model (i.e., not considering faulty clusters) have been described in [18] with a node-to-node and a node-to-set disjoint paths routing algorithm. In an n-dimensional k-ary torus ($n \geq 2$, $k \geq 4$), the former algorithm selects a fault-free path of length at most $n\lfloor k/2 \rfloor + 1$ in $O(n^2)$ time with a fault tolerance of at most $2n - 1$ faulty nodes while the latter algorithm selects f ($f \leq 2n$) fault-free paths of lengths at most $n\lfloor k/2 \rfloor + 1$ in $O(n^3)$ time with a fault tolerance of at most $2n - f$ faulty nodes. Still based on the simple node-fault tolerant model, an adaptive node-to-node routing algorithm for a 2-dimensional torus has been given in [19].

Other routing algorithms that apply to a torus network have been proposed. A torus set-to-set disjoint paths routing algorithm has been described in [20]. In an n-dimensional k-ary torus ($n \geq 1$, $k \geq 3$), this algorithm selects $2n$ mutually node-disjoint paths between $2n$ source nodes and $2n$ destination nodes, without imposing a particular pairing. The paths are selected in $O(kn^3 + n^3 \log n)$ time and have lengths that are at most $2(k+1)n$. Then, a torus pairwise disjoint paths routing algorithm has been presented in [21]. In an n-dimensional k-ary torus ($n < k$, $k \geq 5$), given c ($c \leq n$) source–destination node pairs, this algorithm finds c mutually node-disjoint paths that connect the nodes of the c pairs. The paths are selected in $O(nc^4)$ time and have lengths at most $2k(c-1) + n\lfloor k/2 \rfloor$.

In other networks, fault-tolerance under the simple node-fault tolerant model has been treated, for instance, in hypercubes with a unicast algorithm [22] and a set-to-set disjoint paths routing algorithm [23], in star graphs with a set-to-set disjoint paths routing algorithm [24] and in burnt pancake graphs with a unicast algorithm [25]. Furthermore, fault-tolerant routing with an additional constraint regarding the nodes that can be selected has been discussed, for instance, in hypercubes [26]. Finally, routing algorithms based on the cluster-fault tolerant model have been proposed for burnt pancake graphs with a unicast algorithm [27], hypercubes with a unicast algorithm [28], a node-to-set and set-to-set disjoint paths routing algorithm [29] and a pairwise disjoint paths routing algorithm [30], and star graphs with a unicast algorithm [31] and a node-to-set and pairwise disjoint paths routing algorithm [32], amongst others.

3. Preliminaries

First, general graph theory notations and definitions are recalled—the notations and definitions that are not mentioned here are in accordance with [33].

Graphs herein are undirected. For a node u in a graph G, let $N_G(u)$ be the set of the nodes adjacent to u in G. A path in a graph G is a connected acyclic sub-graph of G of maximum degree 2. From this definition, a path necessarily has either one or two nodes of degree at most 1; they are called the end nodes of the path. For the sake of readability, a path is simply denoted by a sequence of nodes and edges as follows: $u_1 \to u_2 \to \ldots \to u_l$, and is further abbreviated to $u_1 \rightsquigarrow u_l$ when non-ambiguous. The node set $V(p)$ that includes the nodes of a path p is simply denoted by p when non-ambiguous. Two paths p, q are mutually node-disjoint (or simply "disjoint") if and only if $p \cap q = \emptyset$. When the path intersection $p \cap q$ consists solely of end nodes, the two paths are said to be internally disjoint. A path is fault-free if and only if it does not include a faulty node. A path is blocked if it includes at least one faulty node. The length of a path is its number of edges.

Definition 1. *An n-dimensional k-ary torus $T(n,k)$, $n \geq 1$, $k \geq 1$ consists of the k^n nodes induced by the set $\{0, 1, \ldots, k-1\}^n$. A node u of a $T(n,k)$ is thus an n-tuple $(u_1, u_2, \ldots, u_n)$ with u_i ($0 \leq u_i \leq k-1$) the coordinate of u for the dimension i ($1 \leq i \leq n$). There is an edge between two nodes $u = (u_1, u_2, \ldots, u_n)$ and $v = (v_1, v_2, \ldots, v_n)$ of a $T(n,k)$ if and only if $\exists j$ ($1 \leq j \leq n$) such that $\forall i$ ($1 \leq i \leq n$, $i \neq j$) $u_i = v_i$ and $u_j = v_j \pm 1 \pmod{k}$.*

A 2-dimensional 3-ary torus $T(2,3)$ and a 3-dimensional 3-ary torus $T(3,3)$ are illustrated in Figure 2a,c, respectively.

Definition 2. *For a node $u = (u_1, u_2, \ldots, u_n) \in T(n,k)$ and a dimension δ ($1 \leq \delta \leq n$), define the two paths*

$$\begin{aligned}
p^+_{u,\delta} = u &\to (u_1, u_2, \ldots, u_{\delta-1}, (u_\delta + 1) \bmod k, u_{\delta+1}, \ldots, u_n) \\
&\to (u_1, u_2, \ldots, u_{\delta-1}, (u_\delta + 2) \bmod k, u_{\delta+1}, \ldots, u_n) \to \ldots \\
&\to (u_1, u_2, \ldots, u_{\delta-1}, (u_\delta - 2) \bmod k, u_{\delta+1}, \ldots, u_n) \\
&\to (u_1, u_2, \ldots, u_{\delta-1}, (u_\delta - 1) \bmod k, u_{\delta+1}, \ldots, u_n) \\
p^-_{u,\delta} = u &\to (u_1, u_2, \ldots, u_{\delta-1}, (u_\delta - 1) \bmod k, u_{\delta+1}, \ldots, u_n) \\
&\to (u_1, u_2, \ldots, u_{\delta-1}, (u_\delta - 2) \bmod k, u_{\delta+1}, \ldots, u_n) \to \ldots
\end{aligned}$$

$$\rightarrow (u_1, u_2, \ldots, u_{\delta-1}, (u_\delta + 2) \bmod k, u_{\delta+1}, \ldots, u_n)$$
$$\rightarrow (u_1, u_2, \ldots, u_{\delta-1}, (u_\delta + 1) \bmod k, u_{\delta+1}, \ldots, u_n)$$

In this research, we consider faulty clusters that include at most two nodes. So, a cluster is formally defined as follows.

Definition 3. *A cluster c of a graph G is a connected subgraph of G that is isomorphic either to a K^1 or to a K^2, where K^n denotes the complete graph of order n. So, a cluster can be denoted simply as a node set.*

Definition 4. *For a graph G and a cluster set C, the set $I(G, C) \subseteq C$ consists of the clusters of C that have at least one node in G. That is, $I(G, C) = \{c \mid c \in C, G \cap c \neq \emptyset\}$.*

Next, several torus properties are recalled. First, a torus has a recursive structure; this is Proposition 1 below.

Proposition 1. *Considering a dimension δ ($1 \leq \delta \leq n$), a $T(n, k)$ consists of k sub-tori $T^{i,\delta}(n-1, k)$ ($1 \leq i \leq k$). Each sub-torus $T^{i,\delta}(n-1, k)$ is induced by the k^{n-1} nodes $(u_1, u_2, \ldots, u_{\delta-1}, i, u_{\delta+1}, \ldots, u_n)$ of $T(n, k)$ with u_j ($1 \leq j \leq n, j \neq \delta$) the node coordinate for the dimension j and i that for the dimension δ.*

The three sub-tori $T^{1,1}(1,3)$, $T^{2,1}(1,3)$ and $T^{3,1}(1,3)$ of a $T(2,3)$ induced by $\delta = 1$ are shown in Figure 2b.

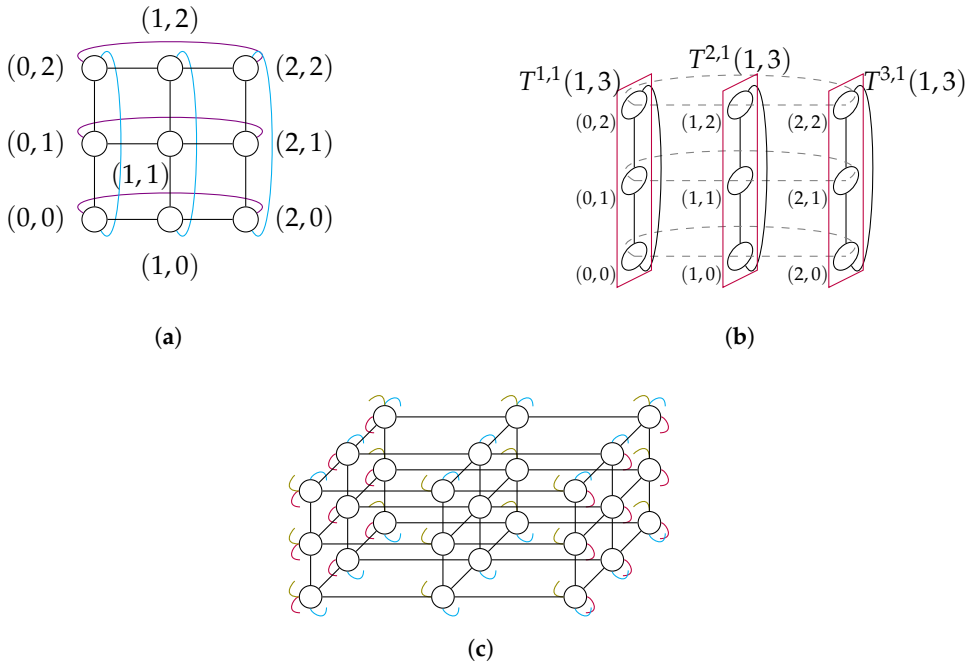

Figure 2. (**a**) A 2-dimensional 3-ary torus $T(2,3)$. (**b**) Illustration of the recursive structure of a torus: a $T(2,3)$ consists in three 1-dimensional 3-ary sub-tori $T(1,3)$. (**c**) A 3-dimensional 3-ary torus $T(3,3)$.

Definition 5. *For a node $u \in T(n, k)$ and a dimension δ ($1 \leq \delta \leq n$), T_u^δ is the sub-torus of u and $t_u^\delta \in \mathbb{N}$ is such that $u \in T^{t_u^\delta,\delta}(n-1, k)$.*

Definition 6. *For a node $u = (u_1, u_2, \ldots, u_n) \in T(n, k)$, a dimension δ ($1 \leq \delta \leq n$) and a sub-torus $T^{i,\delta}(n-1, k)$, $\rho_u^{i,\delta} = (u_1, u_2, \ldots, u_{\delta-1}, i, u_{\delta+1}, \ldots, u_n)$ corresponds to the node u "projected" into $T^{i,\delta}(n-1, k)$.*

Second, there exist disjoint paths between sub-tori.

Lemma 1. *For a node $u \in T(n,k)$ and a dimension δ ($1 \leq \delta \leq n$), there exists a set $P_u^{i,\delta}$ of $2n-1$ internally disjoint paths of lengths at most $k-1$ between u and the nodes of a sub-torus $T^{i,\delta}(n-1,k)$ with $1 \leq i \leq k$.*

Proof. A constructive proof is given. For the node set $\{u_1, u_2, \ldots, u_{2n-2}\} = N_{T_u^\delta}(u)$, define the path set $P_u^{i,\delta} = \{\{u \rightsquigarrow v \in T^{i,\delta}(n-1,k)\} \subseteq p_{u,\delta}^+\} \cup \bigcup_{j=1}^{2n-2}\{u \rightarrow \{u_j \rightsquigarrow v \in T^{i,\delta}(n-1,k)\} \subseteq p_{u_j,\delta}^+\}$. The paths in $P_u^{i,\delta}$ are internally disjoint by Definition 2. □

Lemma 2. *For a node $u \in T(n,k)$ and a dimension δ ($1 \leq \delta \leq n$), each path $p \in P_u^{i,\delta}$ has a variant p' of same end nodes such that the paths of the set $(P_u^{i,\delta} \setminus P) \cup P'$ are internally disjoint for any subset $P \subseteq P_u^{i,\delta}$ with $P' = \{p' \mid p \in P\}$.*

Proof. A constructive proof is given. For the node set $\{u_1, u_2, \ldots, u_{2n-2}\} = N_{T_u^\delta}(u)$, define $\tilde{P}_u^{i,\delta}$ a set of path variants with respect to the path set $P_u^{i,\delta}$ as $\tilde{P}_u^{i,\delta} = \{\{u \rightsquigarrow v \in T^{i,\delta}(n-1,k)\} \subseteq p_{u,\delta}^-\} \cup \bigcup_{j=1}^{2n-2}\{u \rightarrow \{u_j \rightsquigarrow v \in T^{i,\delta}(n-1,k)\} \subseteq p_{u_j,\delta}^-\}$. Further define the bijection $r : P_u^{i,\delta} \rightarrow \tilde{P}_u^{i,\delta}$ that associates each of the $2n-2$ paths $p = u \rightarrow u_j \rightarrow \ldots$ of $P_u^{i,\delta}$ to the path $p' = u \rightarrow u_j \rightarrow \ldots$ of $\tilde{P}_u^{i,\delta}$ ($1 \leq j \leq 2n-2$) and the unique path $p = u \rightarrow u' \notin N_{T_u^\delta}(u) \rightarrow \ldots$ of $P_u^{i,\delta}$ to the path $p' = u \rightarrow u' \notin N_{T_u^\delta}(u) \rightarrow \ldots$ of $\tilde{P}_u^{i,\delta}$. The paths in $(P_u^{i,\delta} \setminus P) \cup P'$ are internally disjoint for any subset $P \subseteq P_u^{i,\delta}$ with $P' = \{r(p) \mid p \in P\}$ by Definition 2. □

Lemma 3. *For a node $u = (u_1, u_2, \ldots, u_n) \in T(n,k)$ and a dimension δ ($1 \leq \delta \leq n$), there exists a set $Q_u^{i,\delta}$ of $4n-3$ paths of lengths at most k between u and the nodes of a sub-torus $T^{i,\delta}(n-1,k)$ with $1 \leq i \leq k$.*

Proof. A constructive proof is given. For the node set $N = \bigcup_{j=1,j\neq\delta}^{n}\{(u_1, u_2, \ldots, (u_j+2) \bmod k, \ldots, u_n), (u_1, u_2, \ldots, (u_j-2) \bmod k, \ldots, u_n)\}$ and the path set $P = \{u \rightarrow v \rightarrow \{w \rightsquigarrow x \in T^{i,\delta}(n-1,k)\} \subseteq p_{w,\delta}^+ \mid v \in N_{T_u^\delta}(u), w \in N \cap N_{T_u^\delta}(v)\}$, define the path set $Q_u^{i,\delta} = P_u^{i,\delta} \cup P$. □

Definition 7. *For a node $u \in T(n,k)$, a dimension δ ($1 \leq \delta \leq n$), the node set $N = \bigcup_{j=1,j\neq\delta}^{n}\{(u_1, u_2, \ldots, (u_j+2) \bmod k, \ldots, u_n), (u_1, u_2, \ldots, (u_j-2) \bmod k, \ldots, u_n)\}$ and the path set $P = \{u \rightarrow v \rightarrow \{w \rightsquigarrow x \in T^{i,\delta}(n-1,k)\} \subseteq p_{w,\delta}^- \mid v \in N_{T_u^\delta}(u), w \in N \cap N_{T_u^\delta}(v)\}$, define the set of $4n-3$ paths $\tilde{Q}_u^{i,\delta} = \tilde{P}_u^{i,\delta} \cup P$.*

The paths of $P_u^{i,\delta}$, $\tilde{P}_u^{i,\delta}$, $Q_u^{i,\delta}$ and $\tilde{Q}_u^{i,\delta}$ are illustrated in the case of a $T(2,5)$ in Figure 3.

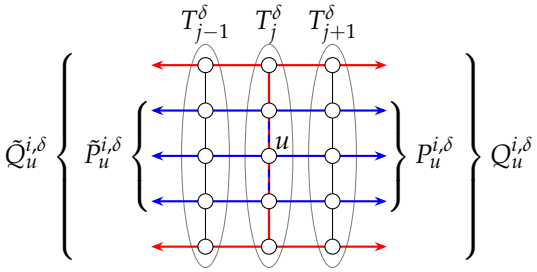

Figure 3. Illustrating the paths of $P_u^{i,\delta}$, $\tilde{P}_u^{i,\delta}$, $Q_u^{i,\delta}$ and $\tilde{Q}_u^{i,\delta}$ in the case of a $T(2,5)$.

4. Cluster-Fault Tolerant Routing Algorithm

Inside an n-dimensional k-ary torus $T(n,k)$ that includes a set C of at most $2n-1$ faulty clusters (which induce the set of faulty nodes F), we describe a routing algorithm that selects a fault-free path between any two nodes $s, d \in T(n,k)$ with $s, d \notin F$.

4.1. Algorithm Description

First, we present the assumptions made and the main idea of the proposed routing algorithm. A $T(1,k)$ is isomorphic to a ring, and it is thus trivial to find a fault-free path between any two non-faulty nodes given that there is at most $2n-1=1$ faulty cluster. So, we can assume that $n \geq 2$.

A $T(n,1)$ has one single node and thus it is trivial to find a fault-free $s \rightsquigarrow d$ path ($s = d$). It is easy to show that this problem is not solvable when $k \in \{2,3,4\}$: see Figure 4. Hence, a torus arity $k \geq 5$ is considered hereinafter.

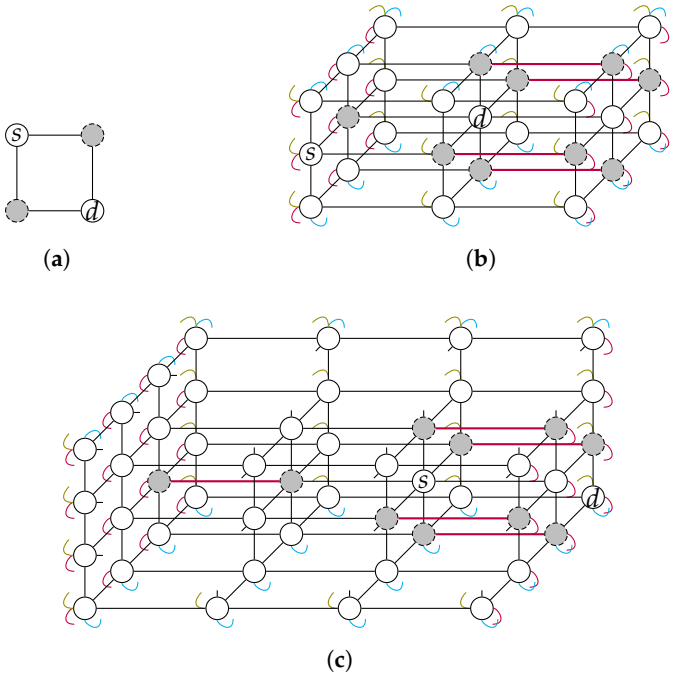

Figure 4. Unsolvable problem instances in the case $k = 2$ (**a**), $k = 3$ (**b**) and $k = 4$ (**c**), within a $T(2,2)$ with two clusters, a $T(3,3)$ with five clusters and a $T(3,4)$ (some nodes are omitted for clarity) with five clusters, respectively. Faulty nodes are greyed and the clusters that include two nodes are materialised with thicker lines.

The main idea of this algorithm is to follow a divide-and-conquer approach by routing s to a node of T_d^δ and to apply this algorithm recursively in T_d^δ. Consider an arbitrary dimension δ ($1 \leq \delta \leq n$). We distinguish the following mutually exclusive cases.

Case 0 (*base case*) $T(n,k)$ is fault-free (i.e., $C = F = \emptyset$):

This is simple point-to-point routing. A path between s and d is selected with a dimension-order routing algorithm [3].

Case 1 (*special case*) T_d^δ unavailable (i.e., $|I(T_d^\delta, C)| > 2n-3$):

Case 1.1 T_s^δ unavailable (i.e., $|I(T_s^\delta, C)| > 2n-3$):

We can apply the algorithm recursively in neither T_s^δ nor T_d^δ, so we use another sub-torus. Route s and d to an available sub-torus $T^{i,\delta}(n-1,k)$, that is satisfying $|I(T^{i,\delta}(n-1,k), C)| \leq 2n-3$, with a fault-free path of $P_s^{i,\delta} \cup \tilde{P}_s^{i,\delta}$ and a fault-free path of $P_d^{i,\delta} \cup \tilde{P}_d^{i,\delta}$, respectively. Let $p_s : s \rightsquigarrow s' \in T^{i,\delta}(n-1,k)$ (resp. $p_d : d \rightsquigarrow d' \in T^{i,\delta}(n-1,k)$) be the selected path that connects s (resp. d) to a node of $T^{i,\delta}(n-1,k)$. If these paths are not disjoint, consider the node $u \in p_s \cap p_d$ that is the closest to s, discard the sub-paths $(u \rightsquigarrow s') \subset p_s$ and $(u \rightsquigarrow d') \subset p_d$ and terminate. Otherwise, apply this algorithm recursively in $T^{i,\delta}(n-1,k)$ with s' as source node, d' as destination node and $\{c \cap T^{i,\delta}(n-1,k) \mid c \in I(T^{i,\delta}(n-1,k), C)\}$ as cluster set.

Case 1.2 T_s^δ available (i.e., $|I(T_s^\delta, C)| \leq 2n-3$):
Exchange the roles of s and d; this is Case 2.

Case 2 T_d^δ available (i.e., $|I(T_d^\delta, C)| \leq 2n-3$):
Define $Q = Q_s^{t_d^\delta,\delta} \cup \tilde{Q}_s^{t_d^\delta,\delta}$ the set of $8n-6$ paths from s to a node of T_d^δ.

Case 2.1 (*special case*) s not routable to T_d^δ (i.e., $\forall p \in Q, p \cap F \neq \emptyset$):
Route s and d to an available sub-torus $T^{i,\delta}(n-1,k)$, that is satisfying $|I(T^{i,\delta}(n-1,k), C)| \leq 2n-3$, other than T_d^δ with a fault-free path of $P_s^{i,\delta} \cup \tilde{P}_s^{i,\delta}$ and a fault-free path of $P_d^{i,\delta} \cup \tilde{P}_d^{i,\delta}$, respectively.
Let $p_s : s \rightsquigarrow s' \in T^{i,\delta}(n-1,k)$ (resp. $p_d : d \rightsquigarrow d' \in T^{i,\delta}(n-1,k)$) be the selected path that connects s (resp. d) to a node of $T^{i,\delta}(n-1,k)$. If these paths are not disjoint, consider the node $u \in p_s \cap p_d$ that is the closest to s, discard the sub-paths $(u \rightsquigarrow s') \subset p_s$ and $(u \rightsquigarrow d') \subset p_d$ and terminate. Otherwise, apply this algorithm recursively in $T^{i,\delta}(n-1,k)$ with s' as source node, d' as destination node and $\{c \cap T^{i,\delta}(n-1,k) \mid c \in I(T^{i,\delta}(n-1,k), C)\}$ as cluster set.

Case 2.2 (*general case*) s routable to T_d^δ (i.e., $\exists p \in Q, p \cap F = \emptyset$):
Route s to T_d^δ with a fault-free path of Q. Let $p_s : s \rightsquigarrow s' \in T_d^\delta$ be the selected path of Q that connects s to a node of T_d^δ. If $d \in p_s$ (i.e., $s' = d$), terminate. Otherwise, apply this algorithm recursively in T_d^δ with s' as source node, d as destination node and $\{c \cap T_d^\delta \mid c \in I(T_d^\delta, C)\}$ as cluster set.

4.2. Routing Example in a $T(3,5)$

We give a non-trivial example of execution trace for the proposed routing algorithm. Let $s = (1,1,0)$, $d = (1,0,1)$ and $C = \{\{(1,1,1),(0,1,1)\},\{(0,1,0)\},\{(1,0,0)\},\{(2,1,0)\},\{(1,2,0),(1,2,4)\}\}$ be the source node, destination node and cluster set in a $T(3,5)$, respectively. Furthermore, δ is arbitrarily initialised to 3. The execution trace of the algorithm is given in Table 1. The leftmost column indicates the iteration step, and the "Sub-Torus" column the sub-torus selected for recursion. It is recalled that the torus arity (here, $k = 5$) is constant throughout the execution of the algorithm; it is thus not repeated in the table. The first iteration step falls in Case 2.1, the second in Case 2.2 and the third in Case 0 of the algorithm.

Table 1. A sample execution trace of the algorithm for a non-trivial routing example in a $T(3,5)$.

↻	δ	n	s	d	C	**Sub-Torus**	**Selected Paths**
1	3	3	$(1,1,0)$	$(1,0,1)$	$\{\{(1,1,1),(0,1,1)\},$ $\{(0,1,0)\},$ $\{(1,0,0)\},$ $\{(2,1,0)\},$ $\{(1,2,0),(1,2,4)\}\}$	$T^{4,3}(2,5)$	$s = (1,1,0) \to (1,1,4) = s'$ $d = (1,0,1) \to (1,0,2) \to (1,0,3) \to (1,0,4) = d'$
2	2	2	$(1,1,4)$	$(1,0,4)$	$\{\{(1,2,4)\}\}$	$T^{0,2}(1,5)$	$s = (1,1,4) \to (1,0,4) = s'$
3	-	1	$(1,0,4)$	$(1,0,4)$	$\emptyset$	-	$s = (1,0,4) = d$
Output: $s = (1,1,0) \to (1,1,4) \to (1,0,4) \to (1,0,3) \to (1,0,2) \to (1,0,1) = d$							

5. Proof of Correctness

In this section, the correctness of the proposed algorithm is established. Each of all the distinguished cases is treated separately.

5.1. Case 0

There is no faulty node inside $T(n,k)$, so a dimension-order routing algorithm can be applied.

5.2. Case 1.1

The sub-tori T_s^δ and T_d^δ each include at least $2n-2$ clusters (possibly partially, i.e., $|I(T_d^\delta, C)| > 2n-3$) and are thus unavailable in order to solve the problem recursively. Here are two necessary conditions for this situation to occur: 1) T_s^δ is adjacent to T_d^δ (i.e., $t_d^\delta = t_s^\delta \pm 1 \pmod k$) and 2) out of the at most $2n-1$ clusters, at least $2n-3$ of them have two nodes, with one node in T_s^δ and the other in T_d^δ.

First, we show that there exist at least three sub-tori that are available for recursion, that is, that include at most $2n-3$ clusters. It is recalled that both T_s^δ and T_d^δ are not. At least $2n-3$ clusters are included (completely contained) in $T_s^\delta \cup T_d^\delta$. Hence, for both T_s^δ and T_d^δ to satisfy $|I(T_d^\delta, C)| > 2n-3$, there remains at most two faulty nodes that are included in other sub-tori (i.e., neither in T_s^δ nor T_d^δ). These two faulty nodes are either part of the same cluster, or, they are part of two distinct 2-node clusters that have one node in T_s^δ and one node in T_d^δ, respectively. So, importantly, if these at most two faulty nodes are part of two distinct 2-node clusters, these two faulty nodes are necessarily located in different sub-tori. See Figure 5.

Figure 5. The two possible cluster repartitions in Case 1.1 with respect to T_s^δ and T_d^δ when $n = 3$ and thus at most five clusters. Ellipses separate sub-tori.

To apply the algorithm recursively inside a sub-torus, it can include at most $2n-3$ clusters. So, one sub-torus can be made unavailable with at least $2n-2$ clusters. Therefore, either the at most one 2-node cluster or the at most two faulty nodes located in distinct sub-tori suffice not to make another sub-torus unavailable for recursion since they would induce at most one cluster inside a sub-torus, and $1 < 2n-2$ given that $n \geq 2$. Therefore, since there exist at least $k \geq 5$ sub-tori and at most two of them are unavailable (T_s^δ and T_d^δ), at least three sub-tori always remain available for recursion.

Next, we show that both s and d are routable to at least one of these three available sub-tori. If s and d are adjacent, select $s \to d$ and there is nothing else to prove. So, we can assume that s and d are not adjacent. We distinguish the following mutually exclusive sub-cases which are exhaustive.

Sub-case $N_{T_s^\delta}(s) \cup N_{T_d^\delta}(d) \subseteq F$. This case can occur only when $n = 2$ and it implies that either (a) there are two 2-node clusters each included in $N_{T_s^\delta}(s) \cup N_{T_d^\delta}(d)$, or (b) there is only one such cluster and the other two clusters respectively have at least one node in $N_{T_s^\delta}(s)$ and one node in $N_{T_d^\delta}(d)$. As shown in Figure 6, the former case (a) occurs only when $k = 4$ and the latter case (b) only when $k = 3$. Hence, given that $k \geq 5$ is assumed, these two cases shall never occur and there is thus nothing to prove.

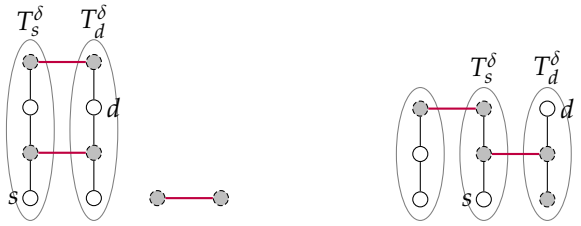

Figure 6. The two situations for Case 1.1's sub-case $N_{T_s^\delta}(s) \cup N_{T_d^\delta}(d) \subseteq F$. (**a**) $k = 4$ required; (**b**) $k = 3$ required. Ellipses separate sub-tori, faulty nodes are greyed and the clusters that include two nodes are materialised with thicker lines.

Sub-case $N_{T_s^\delta}(s) \subset F$ and $N_{T_d^\delta}(d) \not\subset F$ (the case $N_{T_s^\delta}(s) \not\subset F$ and $N_{T_d^\delta}(d) \subset F$ is discussed similarly). Since both T_s^δ and T_d^δ unavailable, at least $2n-3$ clusters each have one node in T_s^δ and the other in T_d^δ, and either (a) at least one other cluster also has, or (b) the two other clusters each have at least one node in $T_s^\delta \cup T_d^\delta$. In the former case (a), at least $2n-2$ clusters each have one node in T_s^δ and the other in T_d^δ, and those clusters thus can block at most one path of $\{p_{s,\delta}^+, p_{s,\delta}^-\}$ and at most one path of $\{p_{d,\delta}^+, p_{d,\delta}^-\}$. The remaining cluster, if any (there is at most 1), can either block at most two paths of $\{p_{s,\delta}^+, p_{s,\delta}^-\}$ for at most two available sub-tori, or at most two paths of $\{p_{d,\delta}^+, p_{d,\delta}^-\}$ for at most two available sub-tori. Since both situations cannot occur at the same time, there always remains at least one available sub-torus to which both s and d are routable. In the latter case (b), $2n-3$ clusters cannot block any path of $\{p_{s,\delta}^+, p_{s,\delta}^-, p_{d,\delta}^+, p_{d,\delta}^-\}$ for the same reason, and the other two clusters each have at least one node in $T_s^\delta \cup T_d^\delta$, hence they can block at most two paths of $\{p_{s,\delta}^+, p_{s,\delta}^-\}$ for at most one available sub-torus, and at most two paths of $\{p_{d,\delta}^+, p_{d,\delta}^-\}$ for at most one available sub-torus (the blocked at most two sub-tori are necessarily distinct). Therefore, there always remains at least one available sub-torus to which both s and d are routable.

Sub-case $N_{T_s^\delta}(s) \not\subset F$ and $N_{T_d^\delta}(d) \not\subset F$. This is the same proof as for the previous case (i.e., $N_{T_s^\delta}(s) \subset F$ and $N_{T_d^\delta}(d) \not\subset F$).

5.3. Case 1.2

Case 2 is applied, so refer to the proofs of Cases 2.1 and 2.2.

5.4. Case 2.1

We indeed need to consider the paths of $Q_s^{t_d^\delta,\delta} \cup \tilde{Q}_s^{t_d^\delta,\delta}$ (they are each of length at most k) given that the paths of $P_s^{t_d^\delta,\delta} \cup \tilde{P}_s^{t_d^\delta,\delta}$ (they are each of length at most $k-1$) may not suffice to route s to another sub-torus as shown in Figure 7a.

So, s not routable to T_d^δ implies that the $8n-6$ paths of $Q = Q_s^{t_d^\delta,\delta} \cup \tilde{Q}_s^{t_d^\delta,\delta}$ are all blocked. This situation occurs only when both 1) at least $2n-2$ clusters each have at least one node in $N_{T_s^\delta}(s) \cup N_{T_d^\delta}(\rho_s^{t_d^\delta,\delta})$, and 2) one of these clusters includes $\rho_s^{t_d^\delta,\delta}$, or there is one other cluster with no node in $T_s^\delta \cup N_{T_d^\delta}(\rho_s^{t_d^\delta,\delta})$ that does. See Figure 7b.

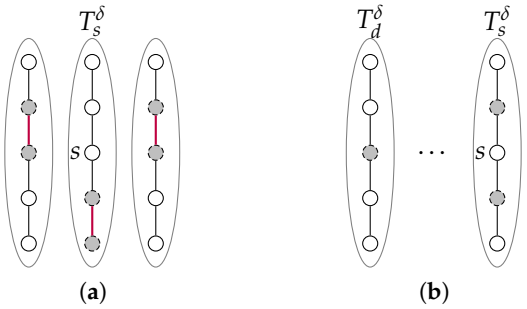

Figure 7. Situations of Case 2 in a $T(2,5)$: (**a**) the situation in Case 2.2 where the paths of $P_s^{t_d^\delta,\delta} \cup \tilde{P}_s^{t_d^\delta,\delta}$ do not suffice to route s to another sub-torus; (**b**) the situation in Case 2.1 where s is not routable to T_d^δ.

We show the existence of a sub-torus available for recursion other than T_d^δ. Since at least $2n-2$ clusters each including at least one node in $N_{T_s^\delta}(s) \cup N_{T_d^\delta}(\rho_s^{t_d^\delta,\delta})$, they can induce at most two unavailable sub-tori. If two unavailable sub-tori are so induced, they are T_s^δ and one sub-torus adjacent to T_s^δ (that is not T_d^δ since T_d^δ available by assumption), and the remaining cluster includes $\rho_s^{t_d^\delta,\delta}$ and thus suffices not to make one additional sub-torus unavailable. If only one unavailable sub-torus is so induced, it is necessarily T_s^δ since T_d^δ available by assumption, and thus exactly $2n-2$ clusters each have at least one node in $N_{T_s^\delta}(s)$ and the remaining cluster which has no node in T_s includes $\rho_s^{t_d^\delta,\delta}$. Therefore, since there exist at least $k \geq 5$ sub-tori and at most two of them are unavailable (one of which necessarily being T_s^δ), at least three sub-tori always remain available for recursion.

Next, we show that both s and d are routable to at least one of these three available sub-tori. If s and d are adjacent, select $s \to d$ and there is nothing else to prove. So, we can assume that s and d are not adjacent. We distinguish the following mutually exclusive sub-cases which are exhaustive.

Sub-case $N_{T_s}(s) \subset F$. Exactly $2n-2$ clusters thus have at least one node in T_s^δ (T_s^δ is thus unavailable). These clusters cannot block a path of $\{p_{s,\delta}^+, p_{s,\delta}^-\}$ and can block at most two paths of $\{p_{d,\delta}^+, p_{d,\delta}^-\}$ for at most one available sub-torus. The remaining cluster necessarily includes $\rho_s^{t_d^\delta,\delta}$ and can thus block at most two paths of $\{p_{s,\delta}^+, p_{s,\delta}^-\}$ for at most two available sub-tori, one being T_d^δ; it can block no path of $\{p_{d,\delta}^+, p_{d,\delta}^-\}$. Therefore, there always remains at least one available sub-torus other than T_d^δ to which both s and d are routable.

Sub-case $N_{T_s}(s) \not\subset F$. Since T_d^δ available, $N_{T_d}(d) \not\subset F$. Thus, there exist two non-faulty nodes $u \in N_{T_s^\delta}(s)$ and $v \in N_{T_d}(d)$. If the remaining cluster (i.e., that includes no node of $T_s^\delta \cup N_{T_d^\delta}(\rho_s^{t_d^\delta,\delta})$) includes $\rho_s^{t_d^\delta,\delta}$, it can block at most two paths of $\{p_{s,\delta}^+, p_{s,\delta}^-\}$ for at most one available sub-torus. If it does not, it can block either: at most two paths of $\{p_{s,\delta}^+, p_{s,\delta}^-\}$ for at most two available sub-tori, or at most two paths of $\{p_{d,\delta}^+, p_{d,\delta}^-\}$ for at most two available sub-tori, or at most two paths of $\{p_{s,\delta}^+, p_{s,\delta}^-\}$ and at most two paths of $\{p_{d,\delta}^+, p_{d,\delta}^-\}$ for at most one available sub-torus. Therefore, there always remains at least one available sub-torus other than T_d^δ to which both s and d are routable.

5.5. Case 2.2

There is nothing to show given that T_d^δ is available for recursion and s is routable to T_d^δ.

6. Complexity Analysis

In this section, we establish the length of a longest path as selected by the proposed algorithm, as well as the worst-case time complexity of this algorithm. It is assumed that the value of each dimension of a node address can be accessed in constant time $O(1)$. Let $\tau(c,n)$ be the worst-cast time complexity of the algorithm when applied in a $T(n,k)$ with $c \leq 2n-1$ faulty clusters, and let $\lambda(c,n)$ be the maximum length of the generated path.

Case 0 The path is obtained with a dimension-order routing algorithm. Hence, in a $T(n,k)$ it is of length at most $n\lfloor k/2 \rfloor$. This algorithm takes $O(nk)$ time.

Case 1 It takes $O(n)$ time to check if $|I(T_d^\delta, C)| > 2n-3$ holds.

Case 1.1 It takes $O(n)$ time to check if $|I(T_s^\delta, C)| > 2n-3$ holds. An available sub-torus $T^{i,\delta}(n-1,k)$ is found when $|I(T^{i,\delta}(n-1,k), C)| \leq 2n-3$ holds and both s,d are routable to it with the defined paths. Checking for one sub-torus candidate for $T^{i,\delta}(n-1,k)$ whether $|I(T^{i,\delta}(n-1,k), C)| \leq 2n-3$ holds takes $O(n)$ time. The path p_s for one sub-torus candidate for $T^{i,\delta}(n-1,k)$ can be found in $O(nk|F|)$ time since by Lemma 2 $|P_s^{i,\delta} \cup \tilde{P}_s^{i,\delta}| = 4n-2$ and with each path from those tried for p_s being of length at most $k-1$ by Lemma 1. A similar discussion holds for p_d. So, checking if one sub-torus is suitable as $T^{i,\delta}(n-1,k)$ takes $O(nk|F|)$ time. Hence, an available sub-torus $T^{i,\delta}(n-1,k)$ can be found in $O(nk^2|F|)$ by enumerating the k sub-tori. Checking the intersection of p_s and p_d takes $O(k^2)$ time. If it is not empty, sub-paths are discarded in constant time and the algorithm is terminated. Otherwise, the algorithm is applied recursively in $T^{i,\delta}(n-1,k)$, thus inducing a $\tau(c',n-1)$ time complexity and a $\lambda(c',n-1)$ maximum path length, with $c' \leq 2(n-1)-1$ (and obviously $c' \leq c$). So, in total, this case is $O(nk^2|F| + \tau(c',n-1))$ time and induces a $2k-2+\lambda(c',n-1)$ maximum path length.

Case 1.2 The time complexity and maximum path length induced by Case 2 apply.

Case 2 It is not needed to check again whether T_d^δ is available.

Case 2.1 It takes $O(n)$ time to check whether s is routable to T_d^δ. An available sub-torus $T^{i,\delta}(n-1,k)$ is found when $|I(T^{i,\delta}(n-1,k), C)| \leq 2n-3$ holds and both s,d are routable to it with the defined paths. Checking for one sub-torus candidate for $T^{i,\delta}(n-1,k)$ whether $|I(T^{i,\delta}(n-1,k), C)| \leq 2n-3$ holds takes $O(n)$ time. The path p_s for one sub-torus candidate for $T^{i,\delta}(n-1,k)$ can be found in $O(nk|F|)$ time since by Lemma 2 $|P_s^{i,\delta} \cup \tilde{P}_s^{i,\delta}| = 4n-2$ and with each path from those tried for p_s being of length at most $k-1$ by Lemma 1. A similar discussion holds for p_d. So, checking if one sub-torus is suitable as $T^{i,\delta}(n-1,k)$ takes $O(nk|F|)$ time. Hence, an available sub-torus $T^{i,\delta}(n-1,k)$ can be found in $O(nk^2|F|)$ by enumerating the $k-1$ sub-tori (T_d^δ excluded). Checking the intersection of p_s and p_d takes $O(k^2)$ time. If it is not empty, sub-paths are discarded in constant time and the algorithm is terminated. Otherwise, the algorithm is applied recursively in $T^{i,\delta}(n-1,k)$, thus inducing a $\tau(c',n-1)$ time complexity and a $\lambda(c',n-1)$ maximum path length, with once again $c' \leq 2(n-1)-1$. So, in total, this case is $O(nk^2|F| + \tau(c',n-1))$ time and induces a $2k-2+\lambda(c',n-1)$ maximum path length.

Case 2.2 The path p_s can be found in $O(nk|F|)$ time since by Lemma 3 and Definition 7 $|Q_s^{t_d^\delta,\delta} \cup \tilde{Q}_s^{t_d^\delta,\delta}| = 8n-6$ and with each path from those tried for p_s being of length at most k by Lemma 3. The algorithm is applied recursively in T_d^δ, thus inducing a $\tau(c',n-1)$ time complexity and a $\lambda(c',n-1)$ maximum path length, with once again $c' \leq 2(n-1)-1$. So, in total, this case is $O(nk|F| + \tau(c',n-1))$ time and induces a $k+\lambda(c',n-1)$ maximum path length.

From this discussion, we can derive the following theorem.

Theorem 1. *In a $T(n,k)$ with $k \geq 5$, given two fault-free nodes s,d and a set of at most $2n-1$ faulty clusters, a fault-free path between s and d of length at most $n(2k+\lfloor k/2 \rfloor - 2)$ can be found in $O(n^2k^2|F|)$ time with F the set of faulty nodes induced by the faulty clusters.*

Proof. The existence of a fault-free path $s \rightsquigarrow d$ is shown in Sections 4 and 5. The complexities are derived from Section 6 which induces the following recursive expressions regarding the time complexity and maximum path length:

$$\begin{aligned} \tau(0,n) &= O(nk) \\ \tau(c,n) &= O(nk^2|F| + \tau(c',n-1)) \qquad \text{if } c > 0 \end{aligned}$$

and

$$\begin{aligned} \lambda(0,n) &= n\lfloor k/2 \rfloor \\ \lambda(c,n) &= 2k - 2 + \lambda(c',n-1) \qquad \text{if } c > 0 \end{aligned}$$

with $c' \leq 2(n-1) - 1$ (and obviously $c' \leq c$). The relation $c' \leq 2(n-1) - 1$ is the invariant of the recursion. Since n is decreased by one at each step, c' is guaranteed to reach 0, that is the base case of the recursion. Therefore, the total worst-case time complexity of the proposed algorithm is $O(n^2k^2|F|)$ and the maximum path length is $n(2k-2) + n\lfloor k/2 \rfloor = n(2k + \lfloor k/2 \rfloor - 2)$. □

The described algorithm selects a fault-free path of length at most $n(2k + \lfloor k/2 \rfloor - 2)$ with an $O(n^2k^2|F|)$ worst-case time complexity with F the set of faulty nodes induced by the faulty clusters. The maximum path length is of the same order as the network diameter: $O(nk)$, which is thus on par with previous works on node-to-node routing under the cluster-fault tolerant model [27,28,31].

7. Empirical Evaluation

Now that the worst-case complexities have been established in Section 6, we inspect the average behaviour of the proposed algorithm, implemented to this end. Two experiments were conducted: the first one aims at measuring the maximum length of a path selected by the proposed algorithm, and the second one at measuring the average execution time taken by the algorithm to solve one instance of the torus cluster-fault tolerant routing problem. These experiments were conducted on a computer equipped with an Intel Core i5-1035G7 processor (clocked at 1.20 GHz) and 8 GB RAM, and running Windows 10 Home 64-bit.

The experimental conditions for the first experiment (i.e., maximum path length measurement) were as follows: in a $T(n,5)$, the source node and destination nodes are randomly selected in the set of all the torus nodes. The torus arity k was fixed to 5 in this experiment to maximize the routing difficulty as indeed the number of faults depends on n and not on k. Then, the maximum number of faulty clusters $2n-1$ that can be tolerated were also randomly generated. The faulty clusters are all of diameter one to once again maximize the routing problem difficulty (i.e., a higher number of faulty nodes). Then, the algorithm implementation was used to solve the corresponding routing problem and the length of the selected path output was recorded. This process was repeated 10,000 times for each value of n with $2 \leq n \leq 7$, each time calculating two path length values: the maximum path length and the average path length of the 10,000 selected paths. The results of this first experiment are shown in Figure 8, together with the theoretical maximum path length as established previously in Section 6 for reference.

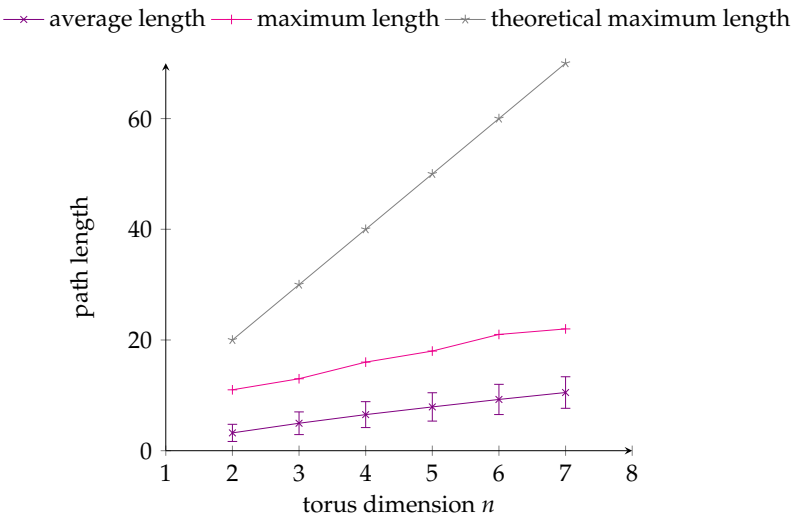

Figure 8. Empirical evaluation: average and maximum (with standard deviation) path lengths of the paths selected by the proposed algorithm in a $T(n, 5)$.

The second experiment (i.e., execution time measurement) was conducted in the same experimental conditions as the first experiment at the exception that the routing problem was solved in a $T(n, \max\{5, n+1\})$: the arity k was set to $\max\{5, n+1\}$ in this time experiment to evaluate the average time complexity as k and n both increase. The path selection algorithm was run 10,000 times for each $(n, \max\{5, n+1\})$ pair with $2 \leq n \leq 7$, each time measuring the real CPU time (i.e., excluding the time for garbage collection) taken to solve the problem instance. The obtained results are given in Figure 9, together with the worst-case time complexity as established previously in Section 6 for reference.

The following observations can be made from the obtained experimental results. First, regarding the maximum path length, one can note that it remains at distance from the theoretical upper bound, which is an indicator of the good performance of the algorithm. Second, regarding the average execution time, one can note that it remains well below the worst-case time complexity, which is yet another indicator of the efficiency of the proposed algorithm.

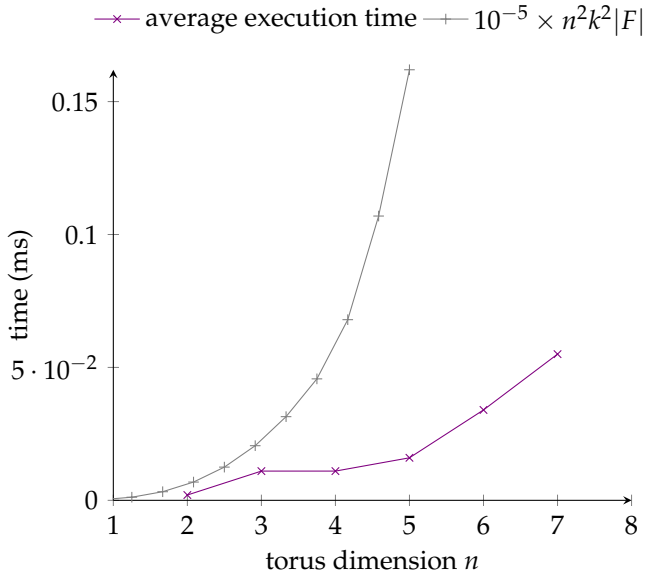

Figure 9. Empirical evaluation: average execution time to solve one problem instance with the proposed algorithm in a $T(n, \max\{5, n+1\})$.

8. Conclusions

The growing number of Internet-connected devices and their sensors, comparable to that of computing nodes included in modern supercomputers, induces large interconnection networks. Hence, the performance of networks on this scale is tied to efficient and robust data routing. For example, major supercomputer makers such as IBM, Cray and Fujitsu have been relying on the torus topology for the interconnection network for its advantageous topological properties. The torus topology is also applicable to interconnect sensor networks, for instance, to report information collected by sensors across the network to the network user. Given the huge number of network nodes involved, faults are very likely to occur. A routing algorithm in a torus that is tolerant to faults is thus key for the future of such networks and has direct implications to the quality-of-service issue by reducing the number of failed data communications. Furthermore, hardware technical properties inducing that faults often happen in clusters (e.g., a same power supply unit applies to a few nodes), it is critical to not only tolerate node faults but also cluster-faults. Improving on Menger's condition on the maximum number of node faults that can be tolerated, and on torus fault-tolerant routing algorithms described in previous works, we have proposed in this paper for the first time a node-to-node routing algorithm in a torus that is tolerant to cluster-faults. In a $T(n,k)$ with at most $2n-1$ faulty clusters of diameter at most 1, the described algorithm selects a fault-free path of length at most $n(2k+\lfloor k/2 \rfloor - 2)$ with an $O(n^2k^2|F|)$ worst-case time complexity with F the set of faulty nodes induced by the faulty clusters.

Regarding future works, it will be meaningful to first try to consider faulty clusters of diameter 2, possibly reducing the number of tolerated faulty clusters. Then, selecting several fault-free disjoint paths between the source and destination nodes can be considered. Furthermore, measuring the average performance of the proposed algorithm and comparing the results with the formally established worst-case complexities (maximum path length and time complexity) is yet another research route.

Author Contributions: Conceptualization, A.B. and K.K.; methodology, A.B. and K.K.; software, A.B.; validation, A.B. and K.K.; formal analysis, A.B. and K.K.; investigation, A.B. and K.K.; resources, A.B. and K.K.; data curation, A.B.; writing—original draft preparation, A.B. and K.K.; writing—review and editing, A.B. and K.K.; visualization, A.B. and K.K.; supervision, A.B. and K.K.; project administration, A.B. and K.K.; funding acquisition, A.B. and K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partly supported by a Grant-in-Aid for Scientific Research (C) of the Japan Society for the Promotion of Science under grant Nos. 19K11887 and 20K11729. The APC was funded by the above grant No. 19K11887.

Acknowledgments: The authors sincerely thank the reviewers for their insightful comments and suggestions.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Hsu, C.L.; Lin, J.C.C. An empirical examination of consumer adoption of Internet of things services: Network externalities and concern for information privacy perspectives. *Comput. Hum. Behav.* **2016**, *62*, 516–527. [CrossRef]
2. Nordrum, A. Popular Internet of things forecast of 50 billion devices by 2020 is outdated. *IEEE Spectrum* **2016**. Available online: https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated (accessed on 8 June 2020).
3. Duato, J.; Yalamanchili, S.; Ni, L. *Interconnection Networks: An Engineering Approach*; Morgan Kaufmann: Burlington, MA, USA, 2003.
4. Cray Inc. Cray XE6 Brochure. 2010. Available online: https://www.cray.com/sites/default/files/resources/CrayXE6Brochure.pdf (accessed on 8 June 2020).

5. Ajima, Y.; Inoue, T.; Hiramoto, S.; Uno, S.; Sumimoto, S.; Miura, K.; Shida, N.; Kawashima, T.; Okamoto, T.; Moriyama, O.; et al. Tofu interconnect 2: System-on-chip integration of high-performance interconnect. In Proceedings of the 29th International Supercomputing Conference, Leipzig, Germany, 22–26 June 2014; pp. 498–507.
6. TOP500. TOP500 List Refreshed, US Edged out of Third Place. 2017. Available online: https://www.top500.org/news/top500-list-refreshed-us-edged-out-of-third-place/ (accessed on 8 June 2020).
7. Saad, Y.; Schultz, M. Topological properties of hypercubes. *IEEE Trans. Comput.* **1988**, *37*, 867–872. [CrossRef]
8. Seitz, C. The cosmic cube. *Commun. ACM* **1985**, *28*, 22–33. [CrossRef]
9. Bossard, A.; Kaneko, K. Torus-Connected Cycles: A simple and scalable topology for interconnection networks. *Int. J. Appl. Math. Comput. Sci.* **2015**, *25*, 723–735. [CrossRef]
10. Menger, K. Zur allgemeinen Kurventheorie. *Fundam. Math.* **1927**, *10*, 96–115. [CrossRef]
11. Sedgewick, R. *Algorithms in C—Part 5, Graph Algorithms*, 3rd ed.; Addison-Wesley: Boston, MA, USA, 2002.
12. Chakraborty, S.; Chakraborty, S.; Nandi, S.; Karmakar, S. Fault resilience in sensor networks: Distributed node-disjoint multi-path multi-sink forwarding. *J. Netw. Comput. Appl.* **2015**, *57*, 85–101. [CrossRef]
13. Akers, S.; Krishnamurthy, B. A group-theoretic model for symmetric interconnection networks. *IEEE Trans. Comput.* **1989**, *38*, 555–566. [CrossRef]
14. Guerroumi, M.; Pathan, A.S.K. Hybrid data dissemination protocol (HDDP) for wireless sensor networks. *Wirel. Netw.* **2018**, *24*, 1739–1754. [CrossRef]
15. Shi, X.; An, X.; Zhao, Q.; Liu, H.; Xia, L.; Sun, X.; Guo, Y. State-of-the-art Internet of things in protected agriculture. *Sensors* **2019**, *19*, 1833. [CrossRef]
16. Ohishi-Yamazaki, M.; Watanabe, M.; Nakanishi, A.; Che, J.; Horiuchi, N.; Ogiwara, I. Shortening of the juvenile phase of the southern highbush blueberry (*Vaccinium corymbosum* L. interspecific hybrid) grown controlled rooms under artificial light. *Hortic. J.* **2018**, *87*, 329–339. [CrossRef]
17. Yang, Y.; Funahashi, A.; Jouraku, A.; Nishi, H.; Amano, H.; Sueyoshi, T. Recursive diagonal torus: An interconnection network for massively parallel computers. *IEEE Trans. Parallel Distrib. Syst.* **2001**, *12*, 701–715. [CrossRef]
18. Gu, Q.P.; Peng, S. Fault tolerant routing in toroidal networks. *IEICE Trans. Inf. Syst.* **1996**, *79*, 1153–1159.
19. Li, Y.; Peng, S.; Chu, W. Online adaptive fault-tolerant routing in 2D torus. In Proceedings of the Third International Symposium on Parallel and Distributed Processing and Applications, Nanjing, China, 2–5 November 2005; pp. 150–161.
20. Kaneko, K.; Bossard, A. A set-to-set disjoint paths routing algorithm in tori. *Int. J. Netw. Comput.* **2017**, *7*, 173–186. [CrossRef]
21. Bossard, A.; Kaneko, K. Torus pairwise disjoint-path routing. *Sensors* **2018**, *8*, 3912. [CrossRef] [PubMed]
22. Gu, Q.P.; Peng, S. Unicast in hypercubes with large number of faulty nodes. *IEEE Trans. Parallel Distrib. Syst.* **1999**, *10*, 964–975.
23. Gu, Q.P.; Okawa, S.; Peng, S. Set-to-set fault tolerant routing in hypercubes. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **1996**, *79*, 483–488.
24. Gu, Q.P.; Peng, S. Set-to-set fault tolerant routing in star graphs. *IEICE Trans. Inf. Syst.* **1996**, *79*, 282–289.
25. Iwasaki, T.; Kaneko, K. Fault-tolerant routing in burnt pancake graphs. *Inf. Process. Lett.* **2010**, *110*, 535–538. [CrossRef]
26. Bossard, A.; Kaneko, K. Hypercube fault tolerant routing with bit constraint. *Int. J. Netw. Comput.* **2015**, *5*, 272–289. [CrossRef]
27. Iwasawa, N.; Watanabe, T.; Iwasaki, T.; Kaneko, K. Cluster-fault-tolerant routing in burnt pancake graphs. In Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing, Busan, Korea, 21–23 May 2010; pp. 264–274.
28. Gu, Q.P.; Peng, S. An efficient algorithm for node-to-node routing in hypercubes with faulty clusters. *Comput. J.* **1996**, *39*, 14–19. [CrossRef]
29. Gu, Q.P.; Peng, S. Node-to-set and set-to-set cluster fault tolerant routing in hypercubes. *Parallel Comput.* **1998**, *24*, 1245–1261. [CrossRef]
30. Gu, Q.P.; Peng, S. *k*-pairwise cluster fault tolerant routing in hypercubes. *IEEE Trans. Comput.* **1997**, *46*, 1042–1049.
31. Gu, Q.P.; Peng, S. Node-to-node cluster fault tolerant routing in star graphs. *Inf. Process. Lett.* **1995**, *56*, 29–35. [CrossRef]

32. Gu, Q.P.; Peng, S. Cluster fault-tolerant routing in star graphs. *Networks* **2000**, *35*, 83–90. [CrossRef]
33. Diestel, R. *Graph Theory*, 4th ed.; Springer: Heidelberg, NY, USA, 2010.