Set-to-Set Disjoint Paths Routing in Torus-Connected Cycles

Antoine BOSSARD†, Nonmember and Keiichi KANEKO††, Member

SUMMARY   Extending the very popular tori interconnection networks [1]–[3], Torus-Connected Cycles (TCC) have been proposed as a novel network topology for massively parallel systems [5]. Here, the set-to-set disjoint paths routing problem in a TCC is solved. In a TCC, it is proved that paths of lengths at most \( kn^2 + 2n \) can be selected in \( O(kn^2) \) time.

1. Introduction

We describe here an algorithm solving the set-to-set disjoint paths routing problem in Torus-Connected Cycles (TCC). This problem is about selecting mutually node-disjoint paths (disjoint hereinafter) between a set of source nodes and a set of destination nodes [4].

Definition 1: A k-ary n-dimensional torus-connected cycles network \( TCC(k,n) \) has \( 2nk^m \) nodes. Each node \( a \) has a cluster ID \( c(a) = (a_0, a_1, \ldots, a_{n-1}) \) and a processor ID \( p(a) = p_a \), and the node consists of the pair \( (c(a), p(a)) \) where \( 0 \leq a_i \leq k - 1 \) and \( 0 \leq p_a \leq 2n - 1 \). Each node \( a \) has three adjacent nodes \( n_1(a), n_2(a) \) and \( n_3(a) \), which are defined as follows:

\[
\begin{align*}
 n_1(a) &= (c(a), (p_a + (1)^{a_n}) \mod 2n) \\
 n_2(a) &= (c(a), (p_a - (1)^{a_n}) \mod 2n) \\
 n_3(a) &= ((a_0, a_1, \ldots, (a_{p_a/2} + (1)^{a_n})) \mod k, \ldots, a_{n-1}), \\
 p_a + (1)^{a_n} 
\end{align*}
\]

Note that a cluster forms a cycle, with a cluster being a subnetwork induced by the set of nodes having the same cluster ID. For a node \( a \) in a \( TCC(k,n) \), let \( C(a) \) be the cycle to which the node \( a \) belongs. Furthermore, let \( \mathcal{C} \) be the set of all the cycles of the network.

Finally, \( a \rightarrow b \) and \( a \leadsto b \) stand for the edge \((a, b)\) and a path from \( a \) to \( b \), respectively, and \( a \Rightarrow b \) denotes in-cycle shortest-path routing from \( a \) to \( b \).

2. Set-to-Set Disjoint Paths Routing in a \((k,n)\)-Torus

First, the routing algorithm 3-S2S selecting three disjoint paths between three distinct source nodes and three distinct destination nodes in a torus is described. In a \((k,n)\)-torus, for a set \( S = \{s_1, s_2, s_3\} \) of three distinct source nodes, and for a set \( D = \{d_1, d_2, d_3\} \) of three distinct destination nodes, with \( S \cap D \) not necessarily empty, the described routing algorithm selects three disjoint paths \( s_i \leadsto d_j \) with \( s_i \in S \) and \( d_j \in D \).

Step 1 Find three dimensions \( x, y, z \) with the three \( d_i[xyz] \) \((1 \leq i \leq 3)\) distinct: \( d_1[xyz] \neq d_2[xyz], d_2[xyz] \neq d_3[xyz], d_3[xyz] \neq d_1[xyz] \).

Step 2 Generate all the six source-destination pairings: \([ (s_1, d_1), (s_2, d_2), (s_3, d_3) \], \([ (s_1, d_1), (s_2, d_3), (s_3, d_2) \], \([ (s_1, d_2), (s_2, d_3), (s_3, d_1) \]

Step 3 For each pairing, generate the six connection orders possible: first connect \( s_1 \), then \( s_2 \), then \( s_3 \); first connect \( s_2 \), then \( s_1 \), then \( s_3 \); etc. by permuting source nodes.

Step 4 For each source-destination pair of the current connection order of the current pairing, apply dimension-order routing to all six possible dimension orders (i.e. permutations of the previously selected dimensions \( x, y, z \) combined with the eight possible traversals for each dimension order \((asc, asc, asc), (asc, dsc, asc), etc.) until three disjoint paths are found. Nodes of \( S \cap D \) are paired together, thus connected by a 0-length path.

Theorem 1: In a \((k,n)\)-torus, the nodes of two sets \( S \) and \( D \) with \( \|S\| = \|D\| = 3 \) can be disjointly connected with paths of lengths at most \( kn \) in \( O(kn) \) optimal time.

3. Set-to-Set Disjoint Paths Routing in a TCC

A routing algorithm selecting three disjoint paths between three distinct source nodes and three distinct destination nodes in a TCC is described. In a \( TCC(k,n) \), for a set \( S = \{s_1, s_2, s_3\} \) of three distinct source nodes, and for a set \( D = \{d_1, d_2, d_3\} \) of three distinct destination nodes (\( S \cap D \) not necessarily empty), the described routing algorithm selects three disjoint paths \( s_i \leadsto d_j \) with \( s_i \in S \) and \( d_j \in D \).

3.1 Special Case 1: \( \exists C \in \mathcal{C}, S \cup D \subseteq C \)

At least two source-destination pairs are connected with in-cycle shortest-path routing in a \( C \) for the closest source-destination node pairs. If all three pairs are connected, terminate, and otherwise, say \( s_i, d_j \) unconnected, select the path \( s_i \leadsto n_3(s_i) \Rightarrow (c(n_3(s_i), p_{d_j}) \leadsto n_3(c(n_3(s_i)), p_{d_j})) \Rightarrow s' \Rightarrow (c(s'), p_{n_3(s_i)}) \leadsto (c(n_3(d_j)), p_{s_i}) \leadsto n_3(d_j) \leadsto d_j \).
3.2 Special Case 2: $3C \in \mathcal{G}, S \subset C, |D \cap C| = 2$

Two source-destination node pairs are connected inside $C$ with in-cycle shortest-path routing. The remaining pair $s_i, d_j$ is connected with the path $s_i \rightarrow n_3(s_i) \Rightarrow (c(n_3(s_i)), p_{d_j}) \rightarrow n_3(c(n_3(s_i)), p_{d_j})) = s' \Rightarrow (c(s'), p_{n_3(s_i)}) \rightarrow (c(n_3(d_j)), p_{n_3(s_i)}) \Rightarrow n_3(d_j) \rightarrow d_j$.

In the general case, the source and the destination nodes are first distributed into distinct cycles, which are then used to apply the torus set-to-set disjoint paths routing algorithm. Finally, the selected torus disjoint paths are converted to TCC disjoint paths with successive in-cycle routings. Four patterns are distinguished.

3.3 Pattern 1: $3C \in \mathcal{G}, |S \cap C| = 2, |D \cap C| \leq 1$

Assume without loss of generality that $S \cap C = \{s_1, s_2\}$. If $n_3(s_1) \in C(s_1)$, select $s_2 \rightarrow n_3(s_2) = s'$, otherwise select $s_1 \rightarrow n_3(s_1) = s'$. Three distinct cycles for all source nodes are found: $C, C(s'), C(s_3)$.

3.4 Pattern 2: $3C \in \mathcal{G}, |S \cap C| = 2, |D \cap C| = 2$

Assume without loss of generality that $s_3, d_3 \notin C$. Select a cycle $C'$ with $C' \not\subset \{C, C(s_3), C(d_3)\}$. The existence of $C'$ is obvious. Three distinct cycles $C, C(s_3), C'$ for source nodes, and three distinct cycles $C, C(d_3), C'$ for destination nodes are found. Then, the 3-S2S torus routing algorithm is applied.

3.5 Pattern 3: $3C \in \mathcal{G}, S \subset C, |D \cap C| = 1$

Assume without loss of generality that $d_3 \in C$ and that $s_1$ is the closest source node to $d_3$ on $C$. If $3C' \in \mathcal{G}, D \cap C' = \{d_1, d_2\}$, select $d_1 \rightarrow n_3(d_1) = d'$ if $n_3(d_1) \notin C$ and $d_2 \rightarrow n_3(d_2) = d'$ otherwise. Three distinct cycles $C, C', C(d')$ for destination nodes are found. If there is no such cycle $C'$, directly three distinct cycles $C, C(d_1), C(d_2)$ for destination nodes are found. Select $s_2 \rightarrow n_3(s_2) = s'$ and $s_3 \rightarrow n_3(s_3) = s'_3$. Three distinct cycles $C, C(s'_3), C(s'_3)$ for source nodes are found. Then, the 3-S2S torus routing algorithm is applied. The trivial path $C$ of length zero will always be selected.

3.6 Pattern 4: $3C \in \mathcal{G}, S \subset C, D \cap C = \emptyset$

The three edges $s_1 \rightarrow n_3(s_1) = s'_1, s_2 \rightarrow n_3(s_2) = s'_2$ and $s_3 \rightarrow n_3(s_3) = s'_3$ are selected. By torus routing, select three disjoint paths $p_{s_1}, p_{s_2}, p_{s_3}$ respectively starting from $C(s'_1), C(s'_2)$ and $C(s'_3)$. Figure 1 (left) is provided for more clarity. If $C$ not included in one of these three torus paths, they are directly converted to TCC paths.

Hence, assume without loss of generality that $C$ is included in one of $p_{s_1}, p_{s_2}, p_{s_3}$. If $C$ in $p_{s_1}$ or $p_{s_3}$, in-cycle routing in $C$ is possible, but not if $C$ in $p_{s_2}$. Hence, assume $C$ included in the path $p_{s_2}$, thus of the form $p_{s_2} : C(s'_2) \rightarrow C$.

$C'' \rightarrow \cdots$, with $C''$ another distinct cycle, adjacent to $C$ and obviously distinct from $C(s'_2)$ and $C(s'_3)$. Depending on the dimension differing between $C(s'_2)$ and $C$, and that between $C$ and $C''$, additional post-processing (rerouting) may be needed. Let $w \in C$ be the unique node satisfying $n_3(w) \in C''$. If there is an in-cycle path $s_2 \sim w$ that includes neither $s_1$ nor $s_3$, then no rerouting is necessary, and paths are directly converted to TCC paths. If there is no such $s_2 \sim w$ path, rerouting is needed. Rerouting situations are detailed below.

3.6.1 Case $C'$ included in $p_{s_2} : C(s'_2) \rightarrow C \rightarrow C'' \rightarrow \cdots$

The path $p_{s_2}$ is modified; discard the sub-path $C(s'_2) \rightarrow C \rightarrow C'' \sim C' \sim p_{s_2}$; select the edge $C(s'_2) \rightarrow C'$.

3.6.2 Case $C'$ included in $p_{s_1} : C(s'_1) \rightarrow \cdots$

First, $p_{s_1}$ is modified: discard the sub-path $C(s'_1) \sim C' \sim p_{s_1}$; select the edge $C(s'_1) \rightarrow C'$. Then, $p_{s_2}$ is modified: discard the edge $C(s'_1) \sim C' \sim p_{s_2}$; select the edge $C(s'_1) \rightarrow C'$. In other words, the torus path for $C(s'_1)$ is rerouted to the torus path initially for $C(s'_1)$, and the torus path for $C(s'_2)$ is rerouted to the torus path initially for $C(s'_1)$.

3.6.3 Case $C' \not\subset p_{s_1} \cup p_{s_2}$

Assume $C'$ included in $p_3$. First, $p_3$ is modified: discard the sub-path $C(s'_3) \sim C' \sim p_3$; select the edge $C(s'_3) \rightarrow C'$. Then, $p_2$ is modified: discard the edge $C(s'_3) \rightarrow C \sim p_2$, and the edge $s_3 \rightarrow s'_3$. Assume $C'$ not included in $p_3$. First, $p_3$ is modified: select the path $C(s'_3) \rightarrow C \sim p_3$; then, $p_2$ is modified: discard the edge $C(s'_3) \rightarrow C \sim p_2$, and the edge $s_3 \rightarrow s'_3$.

In other words, and to conclude Case 3.6.3, the torus path for $C(s'_2)$ is rerouted to the torus path initially for $C(s'_1)$, and the torus path for $C(s'_3)$ (being now simply $C$) is rerouted to the torus path initially for $C(s'_1)$.

Returning to our general discussion of Pattern 4, note that if $|D \cap C''| \geq 2$, say $d_1 \in C''$, and $d'_1 = w$, the same solution is applied (instead of $C(s'_2) \rightarrow C \rightarrow C''$, it is simply $C(s'_2) \rightarrow C$).

Pattern 4 is concluded by showing that the case $D \subset C'$ is solved similarly. Select the three edges $d_1 \rightarrow n_3(d_1) = d'_1, d_2 \rightarrow n_3(d_2) = d'_2$ and $d_3 \rightarrow n_3(d_3) = d'_3$. The 3-S2S torus routing algorithm is applied, thus three disjoint paths $q_1, q_2, q_3$ respectively ending at $C(d'_1), C(d'_2)$ and $C(d'_3)$ are obtained. Again, if $C'$ is not included in one of these three
torus paths, or if $C'$ is included in either $q_1$ or $q_3$, no additional post-processing is required.

Hence, assume $C' \in q_2$. Therefore, $q_2$ is of the form $q_2 : C(d_2') \rightarrow C' \rightarrow C''' \rightarrow \cdots$, with $C'''$ another distinct cycle, adjacent to $C'$ and obviously distinct from $C(d_i')$ and $C(d_j')$. Note that $C$ is still assumed to be included in the path $p_2$ so as to discuss the trickiest situation, thus excluding the easier case $\exists d_i' \in C(s_2')$. The case $\exists d_i' \notin C(s_2')$ is detailed in Fig. 1 (right); the case $\exists d_i' \in C(s_2')$ is similar and thus omitted. The problem is solved as in Pattern 4 with $C'''$ getting the role of $C'$, and $C''$ that of $C'''$. The case $D \subset C'''$ is identical to this case $D \subset C'$, with the sub-case $C'''$ included in $C(d_i') \rightarrow C' \rightarrow \cdots$ never occurring.

From the above discussion, and that of [5], we have:

**Theorem 2:** In a $TCC(k,n)$, for two sets $S$, $D$ each of three distinct nodes, three disjoint paths between $S$ and $D$ of lengths at most $kn^2 + 2n$ can be selected in $O(kn^3)$ time.

4. Conclusions

In this paper, we have proposed an algorithm solving the set-to-set disjoint paths routing problem in a TCC. Precisely, in a $TCC(k,n)$, given a set of three source nodes and a set of three destination nodes, the source and destination nodes can be connected by disjoint paths of lengths at most $kn^2 + 2n$ with a time complexity of $O(kn^3)$. Future works include solving in a TCC the pairwise disjoint paths routing problem.

**References**

[1] J. Moroo, M. Yamada, and T. Kato, “Operation system for the K computer,” Fujitsu Sci. Tech. J., vol.48, no.3, pp.295–301, 2012.

[2] A. Singh, W.J. Dally, A.K. Gupta, and B. Towles, “Goal: A load-balanced adaptive routing algorithm for torus networks,” SIGARCH Computer Architecture News, vol.31, no.2, pp.194–205, 2003.

[3] J.M. Camara, M. Moreto, E. Vallejo, R. Beivide, J. Miguel-Alonso, C. Martinez, and J. Navaridas, “Twisted torus topologies for enhanced interconnection networks,” IEEE Trans. Parallel Distrib. Syst., vol. 21, no.12, pp.1765–1778, 2010.

[4] Q.-P. Gu and S. Peng, “Node-to-set and set-to-set cluster fault tolerant routing in hypercubes,” Parallel Computing, vol.24, no.8, pp.1245–1261, 1998.

[5] A. Bossard and K. Kaneko, “A set-to-set disjoint paths routing algorithm in a torus-connected cycles network,” Proc. 31st Int. Conf. on Comp. and Their App., pp.81–88, Las Vegas, NV, USA, April 2016.