Node-to-set disjoint-path routing in perfect hierarchical hypercubes

Antoine Bossard\textsuperscript{a,1,∗}, Keiichi Kanekoa, Shietung Peng\textsuperscript{b}

\textsuperscript{a}Tokyo University of Agriculture and Technology
\textsuperscript{b}Hosei University

Abstract

The perfect hierarchical hypercube structure was proposed in the literature as a topology for interconnection networks of massively parallel systems. It has the useful ability that it can connect many nodes while retaining a low degree as well as a small diameter. In this paper, we introduce an algorithm solving the node-to-set disjoint-path routing problem in a perfect hierarchical hypercube. Inside a \((2^m + m)\)-perfect hierarchical hypercube, given one source node and a set of \(k (\leq m + 1)\) destination nodes, this algorithm finds \(k\) disjoint paths between the source node and all destination nodes of lengths at most \(m2^m + 2^m + 2m + 4\) in \(O(km2^m)\) time complexity.

Keywords: cube-connected cubes, parallel processing, routing algorithm, interconnection network, disjoint paths

1. Introduction

Parallel processing is nowadays a critical topic, and a large amount of research has been conducted in this field. Massively parallel systems have been attracting more and more attention and are studied with enthusiasm. Interconnection networks are introduced when dealing with the connection of a significant number of processors in such massively parallel systems. Simple topologies, like hypercubes, meshes and tori, were first introduced for interconnection networks. Today, more complicated but of high performance topologies have been introduced for these massively parallel systems to replace the more conventional simple topologies cited previously [1, 2, 3, 4].

Perfect hierarchical hypercubes (HHC) have been proposed by Malluhi and Bayoumi [5]. They have the useful ability that they can connect a significant number of nodes while retaining a low degree and a small diameter. Because of these advantages, Wu and Sun proposed independently an identical topology known as cube-connected cubes [6]. Subsequently Wu et al. [7] introduced a routing algorithm that solves the node-to-node disjoint-path routing problem in HHC. In this paper we describe a routing algorithm to find node-disjoint paths from one common source node to a set of destination nodes. This problem is known as node-to-set disjoint-path routing. As the number of nodes in the network increases, faulty nodes are likely to appear. Hence finding disjoint paths is a critical issue to establish communication routes under a faulty environment. A similar routing problem has been solved in star graphs by Gu and Peng [8] and in dual-cubes by Kaneko and Peng [9].
The rest of this paper is structured as follows. Section 2 reviews important definitions and lemmas. In Section 3, an algorithm HHC-N2S solving the node-to-set disjoint-path routing problem in HHC is proposed. The correctness and complexities of HHC-N2S are then studied in Section 4, followed by an example. An empirical evaluation through a computer experiment is given in Section 5. Finally, Section 6 concludes this paper.

2. Preliminaries

We introduce in this section several definitions, notations and lemmas.

Definition 1. An $m$-dimensional hypercube, $Q_m$, consists of $2^m$ nodes, and each node has an $m$-bit unique address. There is an edge between a pair of nodes $a$ and $b$ if and only if the Hamming distance between their addresses $H(a, b)$ is equal to 1.

Definition 2. A $(2^m + m)$-perfect hierarchical hypercube, $HHC_{2^m+m}$, consists of $2^{2^m+m}$ nodes, and each node has a unique address of a pair of $2^m$-bit and $m$-bit sequences, in this order. For two nodes $a = (\sigma_a, \pi_a)$ and $b = (\sigma_b, \pi_b)$, there is an edge $(a, b)$ between them if and only if either one of the following two conditions holds:

- $\sigma_a = \sigma_b$ and $H(\pi_a, \pi_b) = 1$
- $\sigma_a = \sigma_b \oplus 2^e$ and $\pi_a = \pi_b$

where $\oplus$ represents the bitwise exclusive-or operation and the edge $(a, b)$ is denoted by $a \rightarrow b$.

Edges induced by the first and the second conditions are called internal and external edges, respectively. In addition, for any node $a = (\sigma_a, \pi_a)$ in an HHC, we say that $\sigma_a$ and $\pi_a$ denote the subcube ID and the processor ID of $a$, respectively. Also, nodes with the same subcube ID $\sigma$ induce an $m$-dimensional hypercube; we denote this $m$-cube by subcube $Q_m(\sigma)$. Figure 1 shows a 6-perfect hierarchical hypercube, $HHC_6$.

![Figure 1: HHC_6 (m = 2).](image)

An $HHC_{2^m+m}$ is symmetric, and the number of nodes, the number of edges, the degree, and the connectivity are $2^{2^m+m}$, $(m + 1)2^{2^m+m-1}$, $m + 1$, and $m + 1$, respectively.

Let $N(a)$ represent the set of neighbour nodes of $a$. A path $P$ is an alternate sequence of nodes and edges $a_1, (a_1, a_2), a_2, \ldots, a_{l-1}, (a_{l-1}, a_l), a_l$. The length of a path $P$ is the number of edges included in $P$; it is denoted by $L(P)$. We use the notation $a_1 \rightarrow a_2 \rightarrow \ldots \rightarrow a_l$ or simply $a_1 \sim a_l$ to represent the path $P$.

We assume that a node address in an $n$-dimensional hypercube $Q_n$ can be stored in a fixed number of machine words. Therefore, for two nodes $a$ and $b$ in a $Q_n$, the comparison of $a$ and $b$, the calculation of their Hamming distance $H(a, b)$, and the detection of the most significant bit position can be performed in constant time complexity.

The HHC routing algorithm we introduce in this paper strongly relies on hypercube node-to-set disjoint-path routing. We describe in Algorithm 1 an algorithm Cube-N2S solving the node-to-set disjoint-path routing in a $Q_n$. 
The main idea of Cube-N2S is to follow a divide-and-conquer strategy using the recursive property of a $Q_n$: for any dimension $\delta$, $0 \leq \delta \leq n - 1$, a $Q_n$ consists of two subcubes $Q_{n-1}^0$ and $Q_{n-1}^1$ induced by the set of nodes of $Q_n$ whose $\delta$-th bit is set to 0 and 1, respectively. Let $\text{SPR}(a, b)$ denote a shortest-path routing (SPR) between any two nodes $a, b \in Q_n$.

**Lemma 1.** For $s \in Q_n$, $D = \{d_1, d_2, \ldots, d_k\} \subset Q_n$ and $M \subset N(s)$ with $|M| \leq n - k$, $k \leq n$, we can find $k$ disjoint paths $s \sim d_i$, $1 \leq i \leq k$ of lengths at most $n + 1$, including neither an edge $s \rightarrow M$ nor a node in $M \setminus D$ in $O(kn)$ time complexity.

**Proof:** Paths disjointness is trivial. Let $D_0 = D \cap Q_{n-1}^0$ and $D_1 = D \cap Q_{n-1}^1$. Let $L(k, n)$ be the maximum length of a path generated by Cube-N2S in a $Q_n$ with $|D| = k$. We have $L(1, n) = n + 1$ and $L(k, n) = \max(L(|D_0|, n - 1), L(|D_1|, n - 1) + 1)$. Similarly let $T(k, n)$ be the time complexity of Cube-N2S. We have $T(1, n) = O(n)$ and $T(k, n) = T(|D_0|, n - 1) + T(|D_1|, n - 1) + O(n)$ with $1 \leq |D_0|, |D_1| \leq k - 1$ and $|D_0| + |D_1| = k$. Consequently $L(k, n) = n + 1$ and $T(k, n) = O(kn)$.

**Algorithm 1 Cube-N2S($Q_n$, $s$, $D$, $M$)**

**Input:** $Q_n \in Q_n$, $D = \{d_1, d_2, \ldots, d_k\} \subset Q_n$ and $M \subset N(s)$ where $|M| \leq n - k$, $k \leq n$.

**Output:** $k$ disjoint paths $s \sim d_i$, $1 \leq i \leq k$ including neither an edge $s \rightarrow M$ nor a node in $M \setminus D$.

if $s \in D$ then $C := \{s\}$; $D := D \setminus \{s\}$ else $C := \emptyset$ end if

if $|D| = 1$ then

Divide $Q_n$ along $\delta$, $0 \leq \delta \leq n - 1$ such that $s \oplus 2^\delta \not\in M$; /* Assume $s \in Q_{n-1}^0 */$

if $d \in Q_{n-1}^1$ then $C := C \cup \{s \rightarrow \text{SPR}(s \oplus 2^\delta, d)\}$ else $C := C \cup \{s \rightarrow \text{SPR}(s \oplus 2^\delta, d \oplus 2^\delta) \rightarrow d\}$ end if

else

Divide $Q_n$ along $\delta$, $0 \leq \delta \leq n - 1$ such that $D \cap Q_{n-1}^0 \not\subset \emptyset$ and $D \cap Q_{n-1}^1 \not\subset \emptyset$; /* Assume $s \in Q_{n-1}^0 */$

$s' := s \oplus 2^\delta$; /* Let $s' \in Q_{n-1}^1$ be the neighbour of $s_i \in Q_{n-1}^0 */$

$M_1 := \{s_i \oplus 2^\delta \mid s_i \in M \cap Q_{n-1}^0\}$;

if $s' \in D \cap M$ then

Find $s'_n \in N(s') \setminus (D \cup M_1 \cup \{s\})$;

$D' := D \cap Q_{n-1}^1 \setminus \{s'\} \cup \{s'_n\}$

else

$D' := D \cap Q_{n-1}^1$

end if

$C_1 := \text{Cube-N2S}(Q_{n-1}^1, s', D', M_1)$;

if $s' \in D \cap M$ then $C_1 := \{C_1 \setminus \{s' \rightarrow s'_n\}\} \cup \{s' \rightarrow s'_n \rightarrow s'\}$ end if

if $s' \in M$ then

Select $P \in C_1$ arbitrarily;

$C_1 := \{s \rightarrow P\} \cup \{s \rightarrow s_i \rightarrow s'_j \sim d_i \mid s' \rightarrow s'_j \sim d_j \in C_1 \setminus \{P\}\}$

else

$C_1 := \{s \rightarrow s_i \rightarrow s'_j \sim d_j \mid s' \rightarrow s'_j \sim d_j \in C_1\}$

end if

$M_0 := (M \cap Q_{n-1}^0) \cup \{s_i \mid s \rightarrow s_i \rightarrow s'_j \sim d_j \in C_1\}$;

$C_0 := \text{Cube-N2S}(Q_{n-1}^0, s, D \cap Q_{n-1}^0, M_0)$;

$C := C \cup C_0 \cup C_1$

end if

return $C$

Finally Lemma 2 shows that inside an $HHC_{2^n+m}$ we can connect $m + 1$ destination nodes to distinct subcubes using disjoint paths of lengths at most two. This process is referred to as destination node distribution.

**Lemma 2.** Given a set of $m + 1$ nodes $D = \{d_1, d_2, \ldots, d_{m+1}\}$ in an $HHC_{2^n+m}$, we can find $m + 1$ disjoint paths $d_i \sim d'_i$, $1 \leq i \leq m + 1$ of lengths at most two in $O(m^3)$ time complexity such that the subcube of $d'_i$ does not include
any node in $D \cup (D' \setminus \{d'_i\})$ where $D' = \{d'_1, d'_2, \ldots, d'_{m+1}\}$.

**Proof:** For any $d_i = (s, p_i) \in D$, there are $m + 1$ disjoint paths $P_1^{(i)}, \ldots, P_{m+1}^{(i)}$ of length at most two connecting $d_i$ to $m + 1$ distinct subcubes:

$$
\begin{align*}
\forall d_i &= (s, p_i) \rightarrow (s \oplus 2^h, p_i) \in \mathcal{Q}_m(s \oplus 2^h) \\
\forall d_i &= (s, p_i) \rightarrow (s, p_i \oplus 2^h) \rightarrow (s \oplus 2^{h+2^2}, p_i \oplus 2^h) \in \mathcal{Q}_m(s \oplus 2^{h+2^2}) \quad (0 \leq h \leq m - 1)
\end{align*}
$$

These paths are represented in Figure 2.

We show that for any $d_j, 1 \leq j \leq m + 1, i \neq j$, the path $P_w^{(j)} : d_j \rightarrow d'_j \rightarrow d''_j, 1 \leq w_j \leq m + 1$ can block at most one of the $m + 1$ paths $P_1^{(j)}, \ldots, P_{m+1}^{(j)}$.

Consider two paths $P_u^{(i)} : d_i \rightarrow u_1 \rightarrow u_2 \in \mathcal{Q}_m(s_u)$ and $P_v^{(i)} : d_i \rightarrow v_1 \rightarrow v_2 \in \mathcal{Q}_m(s_v)$ with $1 \leq u, v \leq m + 1, u \neq v$. As $u_1$ and $v_1$ are two distinct neighbours of the same node $d_i$, then $H(u_1, v_1) = 2$.

Assume that $d_j \in \mathcal{Q}_m(s)$. Then $d''_j \in \mathcal{Q}_m(s)$ and $d''_j \notin \mathcal{Q}_m(s)$ hold. As $H(u_1, v_1) = 2$, then $u_1$ and $v_1$ cannot be both on $P_u^{(i)}$. Also, if $d_j = u_1$ then $d''_j \notin \mathcal{Q}_m(s_v)$ holds since there is only one external edge $v_1 \rightarrow v_2$ between $\mathcal{Q}_m(s)$ and $\mathcal{Q}_m(s_v)$ and since $d''_j \neq v_1$. Hence $P_v^{(j)}$ cannot block both $P_u^{(i)}$ and $P_v^{(i)}$ if $d_j \in \mathcal{Q}_m(s)$ holds.

Assume that $d_j \notin \mathcal{Q}_m(s)$ and $d_j \in \mathcal{Q}_m(s_v)$. We recall $H(s_u, s_u) = H(s_v, s_v) = 1$ holds. Then, as $u_1 \neq v_1$, we have $H(s_u, s_v) = 2$. Thus there is no external edge between $\mathcal{Q}_m(s_u)$ and $\mathcal{Q}_m(s_v)$, and $d''_j \notin \mathcal{Q}_m(s_v)$ holds because $P_v^{(j)}$ includes only one external edge. Hence $P_v^{(j)}$ cannot block both $P_u^{(i)}$ and $P_v^{(i)}$ if $d_j \in \mathcal{Q}_m(s_v)$ holds.

As a result, we deduce each path $P_w^{(j)}, 1 \leq w_j \leq m + 1$ can block at most one of the $m + 1$ paths $P_1^{(j)}, \ldots, P_{m+1}^{(j)}$. Then at least $(m + 1) - m = 1$ path $P_w^{(j)}, 1 \leq w_j \leq m + 1$ remains to connect $d_i$ to a node $d'_j$. $P_w^{(j)}$ can be found in $O(m^2)$ time complexity by checking the $m + 1$ paths $P_1^{(j)}, \ldots, P_{m+1}^{(j)}$ of lengths at most two for $d_i$. Therefore we can connect all $d_i \in D$ to nodes $d'_j$ using disjoint paths of lengths at most two in $O(m^3)$ time complexity. □

### 3. Node-to-set disjoint-path routing algorithm in an HHC

We propose in this section an algorithm HHC-N2S which finds $k \leq m + 1$ disjoint paths from a source node $s = (s_0, p_0)$ to $k$ destination nodes $d_i = (s_i, p_i), 1 \leq i \leq k$ in an HHC$_{2^n+m}$. The main idea of the algorithm is to reduce the node-to-set disjoint-path routing problem in an HHC to the node-to-set disjoint-path routing problem in a hypercube via a $2^n$-to-$1$ mapping of an HHC$_{2^n+m}$ onto a $2^n$-cube. Concretely, for each node $(s, p)$ of the HHC, we map its subcube $\mathcal{Q}_m(s)$ to the single node $s$ of a $2^n$-cube. From there, we distinguish two types of nodes: HHC-level nodes are nodes of the HHC$_{2^n+m}$, and cube-level nodes are nodes of the $2^n$-cube $\mathcal{Q}_m$ obtained after the mapping operation. An HHC-level path is a path made of HHC-level nodes, and similarly a cube-level path is a path made of cube-level nodes. A first case handles the situation at least $k - 1$ destination nodes are inside $\mathcal{Q}_m(s_0)$, and otherwise, a second case...
proceeds as follows. In Step 1 destination nodes are distributed into distinct subcubes. These subcubes are treated as destination nodes when applying Cube-N2S onto $Q_m$ in Step 2. In Step 3 unnecessary cube-level paths returned by Step 2 are discarded. Finally, in Step 4 the cube-level paths not discarded are converted back to HHC-level paths by performing routing inside each subcube corresponding to one node of the cube-level paths. It is important to note that if $k = m + 1$ one path must use the edge $s \to (s_0 \oplus 2^{p_0}, p_0)$ so that we can disjointly connect inside $Q_m(s_0)$ the other $m$ paths to $s$.

**Case I.** $|D \cap Q_m(s_0)| \geq k - 1$

If there are at least $k - 1$ destination nodes inside $Q_m(s_0)$, we solve the node-to-set disjoint-path routing problem in an HHC as follows. We connect $s$ to at most $m$ of these destination nodes by applying Cube-N2S inside $Q_m(s_0)$. If either of $|D \cap Q_m(s_0)| = m + 1$ or $|D \cap Q_m(s_0)| = k - 1$ holds, there must be one destination node, say $d_j$, that has not been connected to $s$. If $d_k \in Q_m(s_0)$ then we connect $s$ to $d_k$ with the path $s \to (s_0 \oplus 2^{p_0}, p_0) \rightsquigarrow (s_0 \oplus 2^{p_0}, p_k) \to (s_k \oplus 2^{p_0} \oplus 2^{p_1}, p_k) \rightsquigarrow (s_k \oplus 2^{p_0} \oplus 2^{p_1}, p_0) \to (s_k \oplus 2^{p_0}, p_k) \rightsquigarrow (s_0, p_k) = d_k$ where $\rightsquigarrow$ represents here a SPR in a $Q_m$. Let $d_k$ be included on one of the paths returned by Cube-N2S, say $s \rightsquigarrow d_j$, we discard the subpath from $d_k$ to $d_j$ and exchange the indices of the nodes $d_j$ and $d_k$. Otherwise, that is if $d_k \notin Q_m(s_0)$, then $d_k$ is connected to $s$ by the path $s = (s_0, p_0) \to (s_0 \oplus 2^{p_0}, p_0) \rightsquigarrow (s_0, p_k) = d_k$ such that this path does not include a node of $Q_m(s_0)$ except $s$. All the paths are found, the algorithm is thus terminated. Figure 3 shows the $m + 1$ disjoint paths constructed in the case $D \subset Q_m(s_0), k = m + 1$.

**Case II.** $|D \cap Q_m(s_0)| \leq k - 2$

**Step 1 - Preprocessing** Assume without loss of generality that $Q_m(s_0) \cap D = \{d_1, \ldots, d_r\}, r < k - 1$. Let $Z_1$ be the set of subcube IDs such that corresponding subcubes contain at least two destination nodes $(s_0 \in Z_1)$, formally $Z_1 = \{\sigma \mid |Q_m(\sigma) \cap D| \geq 2\} \setminus \{s_0\}$. Let $Z_2$ be the set of subcube IDs not in $Z_1$ such that each of corresponding subcubes is linked to $Q_m(s_0)$ with an external edge whose end node in $Q_m(s_0)$ is a destination node, formally $Z_2 = \{s_0 \otimes 2^{p_i}, \ldots, s_0 \otimes 2^{p_j}\} \setminus Z_1$. We connect with at most two edges each destination node $d_i = (s_i, p_i)$ with $s_i \in Z_1 \cup Z_2$ to a node $d'_i = (s'_i, p'_i)$ (we call it distributed destination node for $d_i$) by calling Algorithm 2 with distrib($s_i, D, Z_1 \cup Z_2$).

Concretely, for each $d_i$ with $|Q_m(s_i) \cap D| = 1$ and $s_i \notin Z_2$, we do not distribute $d_i$. For each $d_i \in Q_m(s_0)$, we consider as if $d_i$ is distributed to $(s_0 \otimes 2^{p_i}, p_i)$ so as to forbid distribution into subcubes whose corresponding subcube IDs are in $Z_2$. Then we distribute any destination node in $Z_1 \cup Z_2$ according to the proof of Lemma 2. Formally, $\forall d_i: 1 \leq i \leq k$ with $s_i \in Z_1 \cup Z_2$, three statements hold with respect to its distribution path $d_i \to d''_i \to d'_i: s'_i \notin Z_2$, $Q_m(s'_i) \cap (D \cup D' \setminus \{d'_i\}) = \emptyset$ and $d''_i \notin D$, where $D' = \{d'_j \mid s_j \in Z_1 \cup Z_2\}$. Note that if $Q_m(s_0) \cap D = \emptyset$, one destination node can be distributed to a node in $Q_m(s_0)$. 

![Figure 3: Disjoint paths constructed if $D \subset Q_m(s_0), k = m + 1$. A dashed circle represents a subcube.](image-url)
Step 2 - Hypercube routing
We apply in this step Cube-N2S in $Q_{2m}$. The source node for Cube-N2S is $s_0$ where $(s_0, p_0) = s$. The set of destination nodes for Cube-N2S is built as follows. We create fake destination nodes so that the cube-level paths returned by Cube-N2S are disjoint and they can be converted to HHC-level paths. For this purpose we introduce four disjoint sets of subcube IDs.

$Z_1$ and $Z_2$ have already been defined in Step 1. $Z_3$ is defined as the set of subcube IDs whose corresponding subcubes contain either one distributed destination node or, for subcubes whose corresponding subcube ID is not in $Z_2$, only one destination node. Formally $Z_3 = \{ s'_i | s_i \in Z_1 \cup Z_2 \cup \{ s_j \in Z_1 \cup Z_2, r + 1 \leq i \leq k \} \}$. $Z_4$ is defined as a set of subcube IDs neighbours of $s_0$ such that $Z_4 \subset N(s_0) \setminus (Z_1 \cup Z_2 \cup Z_3)$, $|Z_4| = 2^m - |Z_1| - |Z_2| - |Z_3 \setminus \{ s_0 \}|$ and $Z_4$ should not include $s_0 + 2^n$ if possible. Note that $Z_4$ is not always unique. The aim of $Z_4$ is to force one cube-level path connecting $s_0$ to a node of $Z_1 \cup (Z_3 \setminus \{ s_0 \})$ to go through the node $s_0 + 2^n$. See Figure 4 for an illustration of these four sets.

We apply the node-to-set disjoint-path routing algorithm of Lemma 1 in $Q_{2m}$ by calling Cube-N2S($Q_{2m}$, $s_0$, $Z_1 \cup (Z_3 \setminus \{ s_0 \})$, $Z_2 \cup Z_4$). If $s_0 \in Z_3$, the cube-level path of length zero $s_0$ is added to the set of paths returned by Cube-N2S.

Let us introduce an additional treatment for the case $Z_4$ inevitably includes $s_0 + 2^n$. If $Z_1 \cup Z_2 \cup (Z_3 \setminus \{ s_0 \}) \subset N(s_0)$, then $|N(s_0) \setminus (Z_1 \cup Z_2 \cup Z_3)| > 2^m - |Z_1| - |Z_2| - |Z_3 \setminus \{ s_0 \}| = |Z_4|$, that is $Z_4$ can be constructed so as not to include $s_0 + 2^n$. It is a contradiction, then $Z_1 \cup Z_2 \cup (Z_3 \setminus \{ s_0 \}) \subset N(s_0)$ holds. Now assuming $Z_1 \cap N(s_0) = \emptyset$, then at least one of the corresponding nodes of $Z_3$ cannot be in $N(s_0) \cup \{ s_0 \}$. Hence $Z_1 \setminus \{ s_0 \} \subset N(s_0)$. It is a contradiction, then $Z_1 \cap N(s_0) = \emptyset$. As a result from $Z_1 \subset N(s_0)$ and $Z_1 \cap N(s_0) = \emptyset$, $Z_1 = \emptyset$ holds. Therefore, because by definition of $Z_3$ we have $s_0 \in Z_3$ only if $Q_m(s_0)$ contains a distributed destination node, $s_0 \notin Z_3$. Hence $Z_1 \cup Z_2 \cup Z_3 \subset N(s_0)$.

If $Z_3 \cap N(s_0) = \emptyset$ then $|D \cap Q_m(s_0)| = k \geq k - 1$, case handled in Step 0. Hence $Z_1 \cap N(s_0) = \emptyset$. We first select an arbitrary subcube ID $s_i = s_0 + 2^n \in Z_3 \cap N(s_0)$. We then create the path $s_0 \rightarrow s_0 + 2^n \rightarrow s_0 + 2^n + 2^n \rightarrow s_0 + 2^n = s_i$, of length three and discard the path $s_0 \rightarrow s_i$ of length one returned by Cube-N2S.

Step 3 - Path discarding
We can now remove all the unnecessary paths created in Step 2. A path $P : s_0 \rightarrow \sigma$ is discarded if $\sigma \in Z_1$ and $s_0 + 2^n \notin P$ hold. There may exist a path $P : s_0 \rightarrow s_0 + 2^n \rightarrow s_i \rightarrow \sigma \in Z_1$. In this case, considering the HHC-level node set $E = \{ d_i | d_i \in Q_m(\sigma) \} \cup \{ d''_i | d_i \rightarrow d''_i \rightarrow d'_i, d_i, d''_i \in Q_m(\sigma) \}$, let $e(= d_j$ or $d''_j) \in E$ be the closest node of $E$ to the node $(\sigma, \log_2(\sigma \oplus \sigma))$. $P$ will be used to connect $s$ to $e$ in Step 4. Thus we discard the cube-level path generated in Step 2 connecting $s_0$ to $s'_i$ where $(s'_i, p'_j) = d'_j$.

Step 4 - Subcube routing
After Step 3 there remain $k - r$ disjoint cube-level paths. At least $k - r - 1$ of them connect $s_0$ to subcube IDs of $Z_3$. At most one of them connects $s_0$ to a subcube ID of $Z_1$. For each cube-level path $s_0 \rightarrow s'_i$, we extend it with an edge $s'_i \rightarrow s_i$. Now we convert the $k - r$ cube-level paths back to HHC-level paths. Concretely we perform a subcube routing inside each subcube corresponding to a subcube ID (ie. cube-level node) included in these $k - r$ paths $P_i : s_0 \rightarrow s_i, r + 1 \leq i \leq k$ as below. We assume without loss of generality that the path
starting with the edge \( s_0 \to s_0 \oplus 2^p \) is \( P_{r+1} \).

First we need to handle routing inside \( Q_m(s_0) \) particularly. We apply Cube-N2S to connect \( s \) to the \( k - 1 \leq m \) nodes of the set \( (Q_m(s_0) \cap D) \cup \{ a = (s_0, \pi_0) \} \cap s_0 \oplus 2^v \cap P_r, r + 2 \leq i \leq k \).

Now inside the other subcubes, we proceed as follows. For each cube-level path \( P_i, r + 1 \leq i \leq k : s_{i,0}(= s_0) \to s_{i,1} \to \cdots \to s_{i,k-1} \to s_{i,k}(= s_i) \), let \( p_{i,j} \) be an \( m \)-bit sequence that satisfies \( s_{i,j-1} \oplus 2^v = s_{i,j} \). For each path \( P_i \) we construct an HHC-level path \( (s_{i,0}, p_{i,1}) \to (s_{i,1}, p_{i,1}) = (s_{i,0} \oplus 2^v, p_{i,1}) \to (s_{i,1}, p_{i,2}) \to \cdots \to (s_{i,k-1}, p_{i,k-1}) \to (s_{i,k}, p_{i,k}) = d_i \), using a SPR inside each subcube to connect each node \( (s_{i,j}, p_{i,j}) \) to \( (s_{i,j}, p_{i,j+1}) \). Algorithm 3 describes this operation. Figure 5 shows the disjoint paths generated by this step.

If \( P_{r+1} \) connects an element of \( Z_1 \) (ie. \( s_{r+1} \in Z_1 \)), we connect \( (s_{r+1}, P_{r+1,A_{r+1}}) \) and \( e = (d_{r+1} \to d_{r+1}') \), see Step 3) with a SPR instead of connecting \( (s_{r+1}, P_{r+1,A_{r+1}}) \) and \( d_{r+1} \). If \( e \neq d_{r+1} \) then \( e = d_{r+1}' \) holds, hence we connect \( e \) to \( d_{r+1} \) in one edge.

4. Correctness and complexities

In this section we prove the correctness of the algorithm HHC-N2S and estimate its time complexity as well as the maximum path length.

Case I applies Cube-N2S to connect by disjoint paths \( s \) and at most \( m \) destination nodes in \( Q_m(s_0) \). By Lemma 1, these paths are generated in \( O(m^2) \) time complexity, and their lengths are at most \( m + 1 \). In the case there is a path going outside \( Q_m(s_0) \), the length of that additional path is at most \( (2^m + 1) + m(2^m - 1) = 2m^2 + 2^m - m + 1 \) since it...
consists of at most \(2^m + 1\) external edges and at most \(2^m - 1\) subcube routings. The path can be constructed in \(O(m2^m)\) time complexity. Moreover, all the nodes on this path other than its end nodes \(s\) and \(d_k\) are outside \(Q_m(s_0)\). Hence, this path is disjoint from the other paths. If \(d_k \in Q_m(s_0)\), checking if \(d_k\) is included on a path generated inside \(Q_m(s_0)\) takes \(O(m^2)\) time complexity since there are at most \(m\) paths of lengths at most \(m + 1\).

Now in Step 1 of Case II, the sets \(Z_1\) and \(Z_2\) can be created in \(O(k^2)\) and \(O(k)\) time complexity, respectively. Step 1 then distributes every destination node included in a subcube whose corresponding subcube ID is in \(Z_1 \cup Z_2\) to a distinct subcube by disjoint paths of length at most two, requiring \(O(m^2)\) time complexity (Lemma 2). Lemma 2 ensures distribution feasibility of all destination nodes, but for HHC-N2S only those inside subcubes whose corresponding subcube IDs are in \(Z_1 \cup Z_2\) need to be distributed.

Let us prove that the construction of \(Z_4\) in Step 2 is possible, that is that \(|Z_4| \geq 0\) holds. Because we have \(|Z_4| = 2^m - |Z_1| - |Z_2| - |Z_3 \setminus \{s_0\}|\), we need that \(|Z_1| + |Z_2| + |Z_3 \setminus \{s_0\}| \leq 2^m\) holds. Let us count how many elements each set \(Z_1\), \(Z_2\) and \(Z_3 \setminus \{s_0\}\) contains. We can assume that \(Q_m(s_0) \cap D = \{d_1, \ldots, d_r\}, r < k - 1\). Hence we have \(|Z_3| \leq r\) (equality does not always hold because a subcube neighbour of \(s_0\) can be an element of \(Z_1\), thus excluded from \(Z_2\)). Since \(k - r\) destination nodes are outside \(Q_m(s_0)\), \(|Z_1| \leq \lfloor (k - r)/2 \rfloor\). Also we trivially have \(|Z_3| = k - r\) and \(|Z_3 \setminus \{s_0\}| \leq k - r\). We solve the following inequality \((|Z_1| + |Z_2| + |Z_3 \setminus \{s_0\}|)\) is maximised for \(r = 0\) and \(k = m + 1\):

\[
|Z_1| + |Z_2| + |Z_3 \setminus \{s_0\}| \leq \left\lfloor \frac{k - r}{2} \right\rfloor + r + (k - r) = \left\lfloor \frac{k - r}{2} \right\rfloor + k \\
\leq \left\lfloor \frac{m + 1}{2} \right\rfloor + (m + 1) \\
\leq 2^m \iff m \geq 2
\]

Therefore \(|Z_4| \geq 0\) holds for \(m \geq 2\). Since \(|Z_1| + |Z_2| + |Z_3 \setminus \{s_0\}| + |Z_4| = 2^m\), we can apply Cube-N2S to solve the node-to-set disjoint-path routing problem in \(Q_m\). If \(m = 1\), the corresponding HHC is a cycle and it is then trivial to disjointly route to at most \(m + 1 = 2\) destination nodes. The sets \(Z_1\) and \(Z_4\) can be created in \(O(k)\) and \(O(2^m)\) time complexity, respectively. By Lemma 1, the paths generated in Step 2 are disjoint, have a maximum length of \(2^m + 1\) and require \(O(k2^m)\) time complexity since \(|Z_1|\) and \(|Z_3 \setminus \{s_0\}|\) are \(O(k)\), which is the dominant time complexity for Step 2.

Step 3 discards paths by testing each of the cube-level paths returned by Step 2 (at most \(|Z_1| + |Z_3|\) paths are returned). \(|Z_1|\) and \(|Z_3|\) are of order \(O(k)\), hence we can check for each path if it should be discarded or not in \(O(k)\) time complexity. In total, the time complexity of Step 3 is \(O(k^3)\).

Step 4 first performs routing inside \(Q_m(s_0)\) with Cube-N2S. The generated paths are proved to be disjoint by Lemma 1. Then Step 4 performs a subcube routing inside each subcube whose corresponding subcube ID is included on the \(k - r\) cube-level disjoint paths not discarded by Step 3. Hence the HHC-level paths produced by Algorithm 3 are disjoint. An HHC-level path generated by Cube-N2S inside \(Q_m(s_0)\) and an HHC-level path generated based on Cube-N2S and Algorithm 3 outside \(Q_m(s_0)\) are disjoint since they do not share any node except for a common end node. Because the cube-level paths returned by Step 2 have a length of order \(O(2^m)\), it means that \(O(2^m)\) subcubes of the HHC will be traversed. In each intermediate subcube, a routing of \(O(m)\) time complexity is required. Hence the time complexity of Step 4 for each path is \(O(m2^m)\). Therefore the total time complexity of Step 4 is \(O(km2^m)\), which is the dominant time complexity of HHC-N2S.

Let us analyze the maximum length of the paths generated by HHC-N2S. First, in the special case \(|D \cap Q_m(s_0)| \geq k - 1\), the paths inside \(Q_m(s_0)\) have a length of at most \(m + 1\) by Lemma 1. Assume that \(|d_1, \ldots, d_{k-1}\) \(\subset Q_m(s_0)\). If \(d_k \in Q_m(s_0)\), the length of the path \(s \leadsto d_k\) is at most \(3m + 4\). If \(d_k \notin Q_m(s_0)\), the length of the corresponding cube-level path \(s_0 \leadsto s_k\) is at most \(2^m + 1\). We have to perform a subcube routing in each subcube excepted the first one \(Q_m(s_0)\) and the last one \(Q_m(s_k)\). Hence the length of this path is at most \((2^m + 1) - 1)(m + 1) + 1\).

The lengths of the paths generated in Step 2 are at most \(2^m + 1\). Some of the paths may be extended by one edge at the beginning of Step 4. Hence the maximum path length of the cube-level paths \(P_{r+1}, \ldots, P_k\) is \(2^m + 2\). For a path of length \(2^m + 2\), Cube-N2S generates a path of length at most \(m + 1\) in \(Q_m(s_0)\). Inside the \(2^m + 1\) intermediate subcubes, subcube routings generate paths of length at most \(m\). In the last subcube, a path of one edge is generated. See Figure 6. In total, the maximum length of the paths generated by HHC-N2S is:

\[
(m + 2) + (m + 1)(2^m + 1) + 1 = m2^m + 2^m + 2m + 4
\]
Antoine Bossard et al. / Procedia Computer Science 4 (2011) 442–451

We can now recapitulate the above discussion to state the following theorem.

**Theorem 1.** In an \( HHC_{2^{m}+m} \), given a node \( s \) and a set of \( k \) (\( \leq m + 1 \)) nodes \( D = \{d_1, \ldots, d_k\} \), we can find \( k \) disjoint paths \( s \leadsto d_i, 1 \leq i \leq k \) of lengths at most \( 2m^2 + 2m + 2m + 4 \) in \( O(km^2) \) time complexity.

As an example we solve using HHC-N2S a node-to-set disjoint-path routing problem inside an \( HHC_{11} \) (\( m = 3 \)). To increase clarity, we write all numbers in binary format. Let the source node be \( s = (00000000, 000) \) and the set of destination nodes be \( D = \{d_1 = (00001010, 000), d_2 = (00001010, 001), d_3 = (00111000, 100), d_4 = (10000010, 010)\} \). We note that \( d_1 \) and \( d_2 \) are inside the same subcube \( Q_m(00001010) \) and will thus need to be distributed to \( d_1' \) and \( d_2' \), respectively. The disjoint paths returned by HHC-N2S are given in Table 1.

<table>
<thead>
<tr>
<th>( Q_8 ) path</th>
<th>( HH_{11} ) path</th>
<th>( Q_8 ) path</th>
<th>( HH_{11} ) path</th>
<th>( Q_8 ) path</th>
<th>( HH_{11} ) path</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000 (00000000, 000) ( s )</td>
<td>00000000 (00000000, 000) ( s )</td>
<td>00000000 (00000000, 000) ( s )</td>
<td>00000000 (00000000, 000) ( s )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000001 (00000001, 000)</td>
<td>00001000 (00001000, 011)</td>
<td>01000000 (01000000, 111)</td>
<td>10000000 (10000000, 111)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00100001 (00100001, 011)</td>
<td>00010100 (00010100, 001) ( d_2' )</td>
<td>01001000 (01001000, 011)</td>
<td>10000100 (10000100, 001)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00100111 (00100111, 001)</td>
<td>00100110 (00100110, 001) ( d_2' )</td>
<td>01011000 (01011000, 100)</td>
<td>10000101 (10000101, 001)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00101111 (00101111, 001)</td>
<td>01111000 (01111000, 100)</td>
<td>01111000 (01111000, 100)</td>
<td>10000101 (10000101, 001)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01100101 (01100101, 000) ( d_1' )</td>
<td>01110000 (01110000, 100)</td>
<td>01111000 (01111000, 100)</td>
<td>10000101 (10000101, 001)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00010100 (00010100, 000) ( d_1' )</td>
<td>00110000 (00110000, 100)</td>
<td>00111000 (00111000, 100)</td>
<td>10000101 (10000101, 001)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We now show experimentally measure HHC-N2S to inspect its practical behaviour. HHC-N2S has been implemented using the Scheme functional programming language under the development environment DrScheme [10]. We first evaluated the execution time of this algorithm for different values of \( m = 2 \) to \( 9 \). We then, for each value of \( m \), measured the average and the maximum of all the maximal path lengths, each maximal length collected when solving one routing problem for this value of \( m \).

Practically, we solved 10,000 routing problems with HHC-N2S for each value of \( m \). We bounded \( m \) by \( 2 \leq m \leq 9 \), that is using natural integers of up to \( 2^9 = 512 \) bits and routing inside perfect hierarchical hypercubes as large as an \( HH_{521} \). Such big integers are natively handled by Scheme as the integer datatype. The source node and the destination nodes were generated randomly; moreover they are distinct. We imposed that the number of destination nodes \( k \) always be equal to \( m + 1 \).

Figure 7 illustrates the average time in milliseconds required to solve a node-to-set disjoint-path routing problem for each value of \( m \). We see the measured average time converges to \( O(m^22^m) \) time complexity. Figure 8 illustrates the average and maximum maximal path length for each value of \( m \). The theoretical maximum path length of the...
algorithm \( m^2 + 2^m + 2m + 4 \) is also drawn for comparison. As \( m \) increases, the probability to generate a path of maximum length decreases, which explains the divergence between the results and the theoretical maximum path length.

6. Conclusion

We have introduced in this paper a routing algorithm solving the node-to-set disjoint-path problem in perfect hierarchical hypercubes. Inside an \( \text{HHC}_{2^m + m} \), for a common source node and a set of \( k \) \((\leq m + 1)\) destination nodes, the algorithm finds \( k \) disjoint paths between the source node and the destination nodes, whose lengths are at most \( m^2 + 2^m + 2m + 4 \) in \( O(km^2) \) time complexity. Future works include solving other disjoint-path routing problems in perfect hierarchical hypercubes, such as set-to-set or \( k \)-pairwise disjoint-path routing algorithms. Topics related to simple fault-tolerant and cluster fault-tolerant routing algorithms are also to be considered.

Acknowledgments

This study is partly supported by the Fund for Promoting Research on Symbiotic Information Technology of Ministry of Education, Culture, Sports, Science and Technology (MEXT) Japan. It is also partly supported by a Grant-in-Aid for Scientific Research (C) of the Japan Society for the Promotion of Science (JSPS) under Grant No. 22500041.

References