International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 5, Number 2, pages 272–289, July 2015

Hypercube Fault Tolerant Routing with Bit Constraint

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

Keiichi Kaneko
Graduate School of Engineering
Tokyo University of Agriculture and Technology
Nakacho 2-24-16, Koganei, Tokyo, Japan 184-8588

Received: February 12, 2015
Revised: May 1, 2015
Accepted: June 1, 2015
Communicated by Susumu Matsumae

Abstract

Thanks to its simple definition, the hypercube topology is very popular as interconnection network of parallel systems. There have been several routing algorithms described for the hypercube topology, yet in this paper we focus on hypercube routing extended with an additional restriction: bit constraint. Concretely, path selection is performed on a particular subset of nodes: the nodes are required to satisfy a condition regarding their bit weights (a.k.a. Hamming weights). There are several applications to such restricted routing, including simplification of disjoint paths routing. We propose in this paper two hypercube routing algorithms enforcing such node restriction: first, a shortest path routing algorithm, second a fault tolerant point-to-point routing algorithm. Formal proof of correctness and complexity analysis for the described algorithms are conducted. We show that the shortest path routing algorithm proposed is time optimal. Finally, we perform an empirical evaluation of the proposed fault tolerant point-to-point routing algorithm so as to inspect its practical behaviour. Along with this experimentation, we analyse further the average performance of the proposed algorithm by discussing the average Hamming distance in a hypercube when satisfying a bit constraint.

Keywords: supercomputer, parallel system, network, cube, disjoint path, dependable system

1 Introduction

Due to its simplicity, the hypercube topology is very popular as interconnection network of parallel systems: hardware and software implementations are effectively greatly simplified compared for instance to permutation-based networks [1]. Hypercubes have been in use from the early days of supercomputing with the Cosmic Cube [2], and up to now with very recent examples of massively parallel machines such as the NASA Pleiades and the NOAA Zeus supercomputers [3]. Not only are hypercubes popular as interconnection network on their own, but they are also very popular as seed for advanced network topologies such as those employed by hierarchical interconnection networks (HINs). Hierarchical hypercubes [4, 5, 6, 7], hierarchical cubic networks [8, 9, 10], metacubes [11, 12], dual-cubes [13, 14] are some examples.

The literature includes several hypercube routing algorithms, with a few variants such as disjoint paths routing, fault tolerant routing and cluster fault tolerant routing [16, 17, 18, 19]. Yet, these previous algorithms do not allow for enforcing any kind of restriction on the nodes selected by the routing process. Extending routing algorithms to enable node restriction (a.k.a. constraint) has however useful applications. This is for example a simple way to select mutually node disjoint paths between distinct pairs of nodes: by generating multiple paths that enforce distinct node constraints, we are ensured that no node will be used by more than one path. For instance, if we apply the algorithm proposed in this paper, we can solve the k-pairwise disjoint paths routing problem in n-dimensional hypercubes in order $O(kn)$ time ($k \leq \lceil n/2 \rceil$). The previous work by Gu and Peng [15] solves this problem in $O(n^2 \log^* n)$ time with $k = \lceil n/2 \rceil$ and $\log^* n$ the smallest j such that $\log^j n \leq 2$. Disjoint paths routing is actually a critical topic for parallel and distributed systems as such routing algorithms guarantee that the notorious resource allocation problems such as deadlocks, livelocks and starvations will never occur. In addition, because of the huge number of computing nodes present in modern parallel systems, there is a high probability that some faults (i.e. broken nodes) will be encountered, thus making fault tolerance an essential characteristic for routing algorithms. Additionally, routing inside specific embedded networks of hypercubes becomes easy when using routing algorithms that satisfy some node constraint. The hypercube sub-network consisting of the nodes with one or two bits set to 1 is an example of such embedded network. The decomposition of a hypercube into several such sub-networks and their manipulation (e.g. data communication) as disjoint entities of a same network is thus greatly facilitated. A hyper-star graph [20, 21] is an example of previous work dealing with such hypercube nodes subsets.

We focus in this paper on a node restriction of type "bit constraint". In practice, and as detailed later in Section 2, such constraint is defined as a tuple of the form $(i, i+1, \ldots, i+\beta)$ with i a positive integer. We described in Section 3 a shortest path routing algorithm satisfying a bit constraint $\gamma_i = (i, i+1)$ in an n-dimensional hypercube, that is the selection of a shortest path whose all nodes satisfy γ_i. Then, we propose in Section 4 a fault tolerant routing algorithm satisfying a bit constraint $\gamma_i = (i, i+1)$ in an n-dimensional hypercube. The maximum number of faulty nodes tolerated is $\min(n-i, i+1) - 1$. We formally prove in Sections 3.3 and 4.3 the correctness of these two algorithms, and formally establish their complexities as well (i.e. time complexity, maximum path length). In addition, we show in Section 3.3 that the shortest path routing algorithm of Section 3 is time optimal. Next, in Section 5, an empirical evaluation of the proposed fault tolerant routing algorithm is conducted. Lastly, Section 6 concludes this paper.

2 Preliminaries

We introduce in this section several definitions and notations used in this paper.

Definition 1. *An n-dimensional hypercube Q_n consists of 2^n nodes, each having a unique n-bit address. Two nodes u and v of a hypercube are adjacent if and only if their Hamming distance $H(u, v)$ is equal to one.*

A Q_n is symmetric and of connectivity, degree and diameter n [22]. The average distance (i.e. the average length of a shortest path) between two nodes in a hypercube is discussed in appendix. Also, a Q_n is recursive: for any dimension δ ($0 \leq \delta \leq n-1$), a Q_n consists of two $(n-1)$-dimensional hypercubes Q^0_{n-1} and Q^1_{n-1}. The subcube Q^0_{n-1} (resp. Q^1_{n-1}) is induced by the set of nodes of Q_n whose δ-th bits are set to 0 (resp. 1). As illustration, a 4-dimensional hypercube Q_4 is given in Figure 1.

It is assumed that for a hypercube, each node address can be stored in a fixed number of machine words, thus allowing constant time complexity for the operations such as node comparison, most significant bit (MSB) detection and the calculation of the Hamming distance and the bit weight (see Definition 2). Also, as usual, logarithms mentioned in this paper are all in base two.

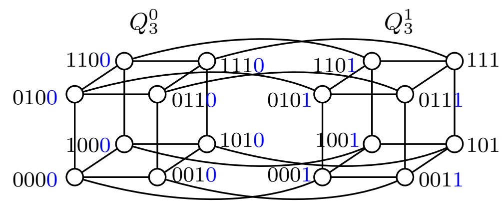

Figure 1: An example of a 4-dimensional hypercube Q_4 with $\delta = 0$.

Definition 2. *For a binary n-bit sequence* $b = b_{n-1} \dots b_1 b_0$, $b_i \in \{0, 1\}$, $0 \le i \le n-1$, *the* bit weight *of* b, *denoted by* $w(b)$, *corresponds to the number of bits of* b *that are set to '1'.*

We recall that the bit weight of a node is also called the Hamming weight. In this paper, the MSB of a bit sequence is the leftmost bit. The bit flip operation is defined and denoted as follows: $u^{(i)} = u \text{ XOR } 2^i$ with XOR denoting the exclusive-or bitwise operation. Also, we define $u^{(i,j)} = (u^{(i)})^{(j)}$. Besides XOR, the AND and NOT bitwise operations (bitwise conjunction and bitwise negation, respectively) are also used hereinafter.

Definition 3. *A* k-constraint *is a* k*-tuple of distinct natural numbers* $(i_1, i_2, \dots, i_k)$.

We focus in this work on 2-constraints applied to the bit weight of a node; hereinafter we simply speak of "bit constraint", denoted by a pair (i, j). Moreover, since in a hypercube adjacent nodes have one single bit different, bit constraints considered here all have the form $(i, i+1)$. One should note that in the case k-constraints on hypercubes would be considered, bit constraints would have the form $(i, i+1, \dots, i+\beta)$ with $i + \beta \le n$.

Definition 4. *In a hypercube* Q_n, *for* $i \in \mathbb{N}$ *and* $0 \le i \le n-1$, *a node* u *satisfies the constraint* $(i, i+1)$ *if and only if* $w(u) = i$ *or* $w(u) = i+1$ *holds.*

Let γ_i denote the $(i, i+1)$ bit constraint. For instance, in a Q_3, the three nodes 010, 110 and 100 all satisfy the constraint $\gamma_1 = (1, 2)$ whereas the node 111 does not.

We recall that a path p in a network is an alternate sequence of nodes and edges: $p = u_1, (u_1, u_2), u_2, \dots, u_k$, with (u_1, u_2) denoting the edge between the nodes u_1 and u_2. This path p can be similarly denoted by $u_1 \to u_2 \to \dots \to u_k$ and simplified to $u_1 \rightsquigarrow u_k$. The length of a path corresponds to its number of edges; it is denoted by $L(p)$ for any path p. Two paths are node disjoint (or simply disjoint) if and only if they have no node in common.

Definition 5. *A path* p *connecting a node* u *to a node* v *satisfies the constraint* $\gamma_i = (i, i+1)$ *if and only if each of all nodes of* p *satisfies* γ_i. *We write* $u \overset{\gamma_i}{\rightsquigarrow} v$.

It can thus be deduced that in a hypercube, as the Hamming distance between any two adjacent nodes is equal to one, a path cannot satisfy a constraint other than that of the form $(i, i+1)$ (or $(i, i-1)$, which is equivalent).

Lastly, for a hypercube reduced into two subcubes according to a bit position, we distinguish a set of paths of lengths at most two that connect a node of one subcube to a node of the other.

Definition 6. *In a* Q_n *reduced into two subcubes* Q_{n-1}^0 *and* Q_{n-1}^1 *according to a bit position* δ $(0 \le \delta \le n-1)$, *given a node* u, *the set of* n *paths* Π_u^δ *is defined as:*

$$\Pi_u^\delta = \left\{ \begin{array}{ll} u \to u^{(\delta)}, & \\ u \to u^{(i)} \to u^{(i,\delta)} & 0 \le i \le n-1, i \ne \delta \end{array} \right\}$$

In other words, the paths of Π_u^δ have lengths two except for $u \to u^{(\delta)}$ which is of length one, and they are all node disjoint except for u. In addition, the two nodes u and $u^{(i)}$ are located inside the same subcube, whereas the two extremal nodes of these paths are located inside distinct subcubes.

3 Hypercube shortest path routing with γ_i constraint

In a Q_n, for two nodes s and d that both satisfy the constraint γ_i, we describe a routing algorithm that selects a shortest path $p : s \rightsquigarrow d$ such that p satisfies γ_i (a.k.a. $p : s \overset{\gamma}{\rightsquigarrow} d$), i.e. exclusively made of nodes of Q_n that satisfy γ_i. We obviously have $L(p) = H(s, d)$.

3.1 Algorithm description

Given any two nodes u, v, we first define the sets $\Delta_{u,v}$ and Θ_u: $\Delta_{u,v}$ is the set of the bit positions whose values differ between u and v; Θ_u is the set of the bit positions of u that can be flipped while still satisfying the considered constraint. For example in a Q_4 with $\gamma_2 = (2, 3)$, given $u = 1011$ and $v = 1100$, we have $\Delta_{u,v} = \{2, 1, 0\}$ and $\Theta_u = \{3, 1, 0\}$. The main and natural idea of this algorithm is to flip bits according to these two sets Δ and Θ.

Step 1. Compute the set $\Delta_{s,d} \cap \Theta_s$ corresponding to the bit positions of s that can be flipped so as to obtain a node satisfying γ_i and that is on a shortest path towards d.

Step 2. Pick an arbitrary bit position from the previously calculated set $\Delta_{s,d} \cap \Theta_s$ and flip the corresponding bit of s; we obtain a new node, say s'.

Step 3. Redefine s as s' and go to Step 1. Repeat until reaching d.

We give a pseudo-code of this shortest path routing algorithm in Algorithm 1. This algorithm uses a subsidiary procedure which is called by the statement "**return** SUBROUTINE(s, s)".

Algorithm 1 HC-SPR-CONS(n, i, s, d)

Input: A Q_n, a bit constraint $\gamma_i = (i, i + 1)$, a source node s and a destination node d.
Output: A shortest path $s \overset{\gamma}{\rightsquigarrow} d$ in Q_n satisfying γ_i.

procedure SUBROUTINE(s, p)
    if $s = d$ **then**
        return p
    else
        $U := \Delta_{s,d} \cap \Theta_s$;
        $\{u_1, u_2, \ldots, u_{|U|}\} := U$;
        $s' := s$ XOR 2^{u_1};
        return SUBROUTINE(s', $p \to s'$)
    end if
end procedure
return SUBROUTINE(s, s)

3.2 Routing example

In a 4-dimensional hypercube Q_4, for a source node $s : 1011$, a destination node $d : 1100$, and a bit constraint $\gamma_2 = (2, 3)$, we give an execution trace of the algorithm of Section 3.1 in Table 1. As a result, the shortest path $s : 1011 \to 1010 \to 1110 \to d : 1100$ satisfying the bit constraint $\gamma_2 = (2, 3)$ is selected.

3.3 Correctness and complexities

The algorithm of Section 3.1 is rather straightforward, the main issue being proving its correctness. This is the objective of this section.

Lemma 1. *The algorithm of Section 3.1 is correct and always terminates.*

Table 1: Shortest path routing example in a Q_4 with $\gamma_2 = (2, 3)$ bit constraint.

s	d	$\Delta_{s,d}$	Θ_s	$\Delta_{s,d} \cap \Theta_s$
1011	1100	$\{2, 1, 0\}$	$\{3, 1, 0\}$	$\{1, \mathbf{0}\}$
selected: $s : 1011 \to s' : 1010$				
1010	1100	$\{2, 1\}$	$\{2, 0\}$	$\{\mathbf{2}\}$
selected: $1011 \to 1010 \to 1110$				
1110	1100	$\{1\}$	$\{3, 2, 1\}$	$\{\mathbf{1}\}$
selected: $1011 \to 1010 \to 1110 \to d : 1100$				

Proof. First, it is trivial to show that the selected path is shortest and satisfies the constraint: each node selected in Step 2 is taken from the set of nodes that reduce the distance to d by one bit flip (precisely the set of differing bit positions $\Delta_{s,d}$), thus making the path shortest, and additionally, this set of nodes is further restricted to the nodes satisfying the constraint (precisely the set of "flippable" bit positions Θ_s). Now, we either show that $s = d$ or that the set $\Delta_{s,d} \cap \Theta_s$ contains at least one node. In other words, in either case the algorithm terminates.

We recall that both s and d satisfy $\gamma_i = (i, i+1)$. Assume without loss of generality that $\Delta_{s,d} \neq \emptyset$. Effectively, if $\Delta_{s,d} = \emptyset$, we would have $s = d$ and a shortest path $s \overset{\gamma}{\rightsquigarrow} d$ satisfying the constraint γ_i would be found.

Assume $w(s) = i$. Now, assume $\Delta_{s,d}$ includes no bit position where s is set to '0'. In other words, we assume that $\Delta_{s,d} \cap \Theta_s = \emptyset$. Then it means that all bits of s set to '0' are set to '0' on d as well. Now, since d satisfies γ_i, it means that the $n - (n - i) = i$ remaining bits of d are set to '1'. Thus $s = d$. So, we have shown that $\Delta_{s,d} \cap \Theta_s = \emptyset \Rightarrow s = d$. In other words, $s \neq d \Rightarrow \Delta_{s,d} \cap \Theta_s \neq \emptyset$.

Assume $w(s) = i + 1$. Now, assume $\Delta_{s,d}$ includes no bit position where s is set to '1'. In other words, we assume that $\Delta_{s,d} \cap \Theta_s = \emptyset$. Then it means that all bits of s set to '1' are set to '1' on d as well. Now, since d satisfies γ_i, it means that the $n - (n - (i + 1)) = i + 1$ remaining bits of d are set to '0'. Thus $s = d$. So, we have shown that $\Delta_{s,d} \cap \Theta_s = \emptyset \Rightarrow s = d$. In other words, $s \neq d \Rightarrow \Delta_{s,d} \cap \Theta_s \neq \emptyset$.

Therefore, we have shown the correctness of the algorithm of Section 3.1. □

Lemma 2. *The algorithm of Section 3.1 is $O(H(s, d))$ optimal time.*

Proof. The sets Δ and Θ are expressed as bit patterns, each included in one machine word. Hence, set calculation in Step 1 can be done in $O(1)$ constant time (Δ directly obtained with one XOR operation; Θ with one bit weight operation: if $w(u) = i + 1$ then $\Theta_u = u$, otherwise $\Theta_u = \text{NOT } u$, which is also $O(1)$ time). The intersection of Δ and Θ can be calculated by one AND operation, which is $O(1)$ time. Step 2 is obviously $O(1)$ time as well (it corresponds to one bit flip operation). Step 3 triggers the repetition of the algorithm until reaching d, that is $O(H(s, d))$ times since each node selected in Step 2 is on a shortest path towards d. Hence, the total time complexity of this algorithm is $O(H(s, d))$, which is obviously optimal. □

Thus, we can summarise this discussion in the theorem below.

Theorem 1. *In a Q_n, given any two distinct nodes s and d and a bit constraint $\gamma_i = (i, i + 1)$, we can select a shortest path $s \overset{\gamma}{\rightsquigarrow} d$ (i.e. of length $H(s, d)$) satisfying γ_i in $O(H(s, d))$ optimal time.*

Proof. This can be directly deduced from Lemmas 1 and 2. □

4 Hypercube fault tolerant point-to-point routing with γ_i constraint

First, given a node $u \in Q_n$ satisfying $\gamma_i = (i, i+1)$, let us discuss the number of its neighbours that satisfy γ_i. If $w(u) = i+1$, then u has $i+1$ neighbours satisfying the constraint. If $w(u) = i$, then u has $n-i$ neighbours satisfying the constraint. Therefore, in a Q_n, given two non-faulty nodes s, d satisfying γ_i, we can select a fault-free path $s \overset{\gamma}{\rightsquigarrow} d$ that satisfies $\gamma_i = (i, i+1)$ with a set F of at most $\min(n-i, i+1) - 1$ faulty nodes (this is an application of Menger's theorem [23]). In addition, one can note that in the case $i = 0$, the maximum number of faulty nodes becomes $\min(n-i, i+1) - 1 = 0$ and thus Q_n is fault-free and it is more efficient to apply the shortest path routing algorithm of Section 3.1. So, let us assume that $i \geq 1$.

4.1 Algorithm description

If $n - i = 0$, the constraint $\gamma_i = (i, i+1)$ cannot be satisfied as $i+1 > n$. If $n-i = 1$, the maximum number of faulty nodes becomes $\min(n-i, i+1) - 1 \leq 0$ and thus Q_n fault free and we directly apply the routing algorithm of Section 3.1. So, we can assume that $n - i \geq 2$.

First, if Q_n is fault free, we directly apply the algorithm of Section 3.1. So, we can assume that Q_n includes at least one faulty node. The main idea of this algorithm is to reduce Q_n into two subcubes Q^0_{n-1} and Q^1_{n-1} as explained in Section 2 and apply the algorithm recursively in one of these two subcubes.

Step 1. If $|F| = 1$, pick an arbitrary bit position δ $(0 \leq \delta \leq n-1)$. If $|F| \geq 2$, find a bit position δ $(0 \leq \delta \leq n-1)$ to reduce Q_n such that Q^0_{n-1} and Q^1_{n-1} each include at least one fault (i.e. both $F \cap Q^0_{n-1} \neq \emptyset$ and $F \cap Q^1_{n-1} \neq \emptyset$ hold).

Step 2. We distinguish several cases below.

Assume s, d are in distinct subcubes, say $s \in Q_{n-1}(s)$ and $d \in Q_{n-1}(d)$. Assume further that $Q_{n-1}(d)$ is the most faulty, that is $|F \cap Q_{n-1}(d)| \geq |F \cap Q_{n-1}(s)|$. If $|F \cap Q_{n-1}(d)| < |F \cap Q_{n-1}(s)|$, the same discussion holds by exchanging the roles of s and d.

Case $d \in Q^1_{n-1}$ and $w(d) = i$. Select a fault-free path of length two of Π^δ_d that satisfies γ_i and connecting d to a non-faulty node d' of Q^0_{n-1}. Apply the algorithm recursively in Q^0_{n-1} with $\gamma_i = (i, i+1)$ the constraint to find a fault-free path $s \overset{\gamma}{\rightsquigarrow} d'$. See Figure 2.

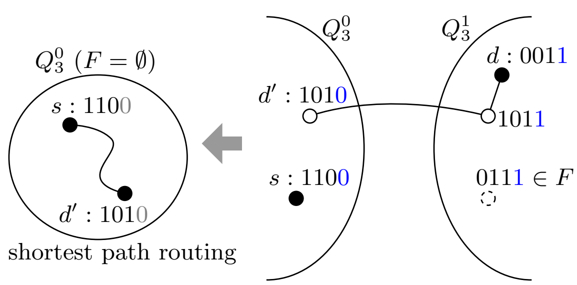

Figure 2: Illustration of Step 2, Case $s \in Q^0_{n-1}$, $d \in Q^1_{n-1}$ and $w(d) = i$. In a Q_4 with reduction bit $\delta = 0$ and constraint $\gamma_2 = (2, 3)$, d is connected to a node $d' \in Q^0_3$ and the algorithm solved recursively in Q^0_3.

Case $d \in Q^1_{n-1}$ and $w(d) = i+1$. If the unique path of length one of Π^δ_d is fault-free, it is selected, say $d \to d'$. Otherwise, consider the fault-free paths of lengths two of Π^δ_d, say $d \to d''_j \to \tilde{d}_j$, with d''_j satisfying γ_i. For each of them, replace the edge $d''_j \to \tilde{d}_j$ by the path of length two $d''_j \to d'''_j \to d'_j$ where $d'''_j = (d''_j)^{(\alpha)}$, $d'_j = (d'''_j)^{(\delta)}$ and α is a bit position such that the α-th bit of d is set to 0. Select one of these paths of lengths three that is fault-free, say $d \to d'' \to d''' \to d'$. Apply the algorithm recursively in Q^0_{n-1} with $\gamma_i = (i, i+1)$ the constraint to find a fault-free path $s \overset{\gamma}{\rightsquigarrow} d'$. See Figure 3.

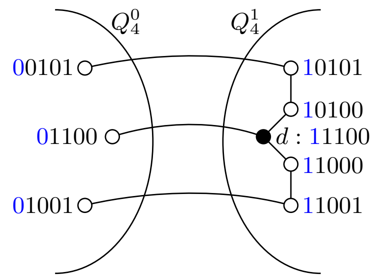

Figure 3: Illustration of Step 2, Case $d \in Q^1_{n-1}$ and $w(d) = i+1$. In a Q_5 with reduction bit $\delta = 4$ and constraint $\gamma_2 = (2,3)$, three candidate disjoint paths that satisfy γ_2 connecting $d \in Q^1_4$ to a node of Q^0_4.

Case $d \in Q^0_{n-1}$ and $w(d) = i+1$. Select a fault-free path of length two of Π^δ_d that satisfies γ_i and connecting d to a non-faulty node d' of Q^1_{n-1}. Apply the algorithm recursively in Q^1_{n-1} with $\gamma_{i-1} = (i-1, i)$ the constraint to find a fault-free path $s \overset{\gamma}{\rightsquigarrow} d'$.

Case $d \in Q^0_{n-1}$ and $w(d) = i$. If the unique path of length one of Π^δ_d is fault-free, it is selected, say $d \to d'$. Otherwise, consider the fault-free paths of lengths two of Π^δ_d, say $d \to d''_j \to \tilde{d}_j$, with d''_j satisfying γ_i. For each of them, replace the edge $d''_j \to \tilde{d}_j$ by the path of length two $d''_j \to d'''_j \to d'_j$ where $d'''_j = (d''_j)^{(\alpha)}$, $d'_j = (d'''_j)^{(\delta)}$ and α is a bit position such that the α-th bit of d is set to 1. Select one of these paths of lengths three that is fault-free, say $d \to d'' \to d''' \to d'$. Apply the algorithm recursively in Q^1_{n-1} with $\gamma_{i-1} = (i-1, i)$ the constraint to find a fault-free path $s \overset{\gamma}{\rightsquigarrow} d'$.

Assume s, d are inside the same subcube.

Case $s, d \in Q^0_{n-1}$. If $|F \cap Q^0_{n-1}| \le |F \cap Q^1_{n-1}|$, apply this algorithm recursively in Q^0_{n-1} with $\gamma_i = (i, i+1)$ the constraint. Assume $|F \cap Q^0_{n-1}| > |F \cap Q^1_{n-1}|$. Depending on whether $w(d) = i$ or $w(d) = i+1$ holds, select as described previously a fault-free path of length at most three that satisfies γ_i connecting d to a node d' of Q^1_{n-1}. Similarly, depending on whether $w(s) = i$ or $w(s) = i+1$ holds, select as described previously a fault-free path of length at most three that satisfies γ_i connecting s to a node s' of Q^1_{n-1}. If these two paths $s \overset{\gamma}{\rightsquigarrow} s'$ and $d \overset{\gamma}{\rightsquigarrow} d'$ are not disjoint, then a path $s \overset{\gamma}{\rightsquigarrow} d$ is found. Otherwise, apply the algorithm recursively in Q^1_{n-1} with $\gamma_{i-1} = (i-1, i)$ the constraint to find a fault-free path $s' \overset{\gamma}{\rightsquigarrow} d'$.

Case $s, d \in Q^1_{n-1}$. If $|F \cap Q^1_{n-1}| \le |F \cap Q^0_{n-1}|$, apply this algorithm recursively in Q^1_{n-1} with $\gamma_{i-1} = (i-1, i)$ the constraint. Assume $|F \cap Q^1_{n-1}| > |F \cap Q^0_{n-1}|$. Depending on whether $w(d) = i$ or $w(d) = i+1$ holds, select as described previously a fault-free path of length at most three that satisfies γ_i connecting d to a node d' of Q^0_{n-1}. Similarly, depending on whether $w(s) = i$ or $w(s) = i+1$ holds, select as described previously a fault-free path of length at most three that satisfies γ_i connecting s to a node s' of Q^0_{n-1}. If these two paths $s \overset{\gamma}{\rightsquigarrow} s'$ and $d \overset{\gamma}{\rightsquigarrow} d'$ are not disjoint, then a path $s \overset{\gamma}{\rightsquigarrow} d$ is found. Otherwise, apply the algorithm recursively in Q^0_{n-1} with $\gamma_i = (i, i+1)$ the constraint to find a fault-free path $s' \overset{\gamma}{\rightsquigarrow} d'$. See Figure 4.

Pseudo-code is given in Algorithm 2.

Algorithm 2 HC-FT-CONS(n, i, s, d, F)

Input: A Q_n, a bit constraint $\gamma_i = (i, i+1)$, a source node s, a destination node d and a set of faulty nodes F.

Output: A fault free path $s \overset{\gamma}{\rightsquigarrow} d$ in Q_n satisfying γ_i.

```
if $F = \emptyset$ then
    HC-SPR-CONS($n$, $i$, $s$, $d$)
else if $|F| = 1$ then                                  ▷ Selection of the reduction bit $\delta$.
    $\delta = 1$
else
    $\{f_1, f_2, \ldots, f_{|F|}\} := F$;
    $\delta = \text{MSB } (f_1 \text{ XOR } f_2)$
end if
if $s$ AND $2^\delta = d$ AND $2^\delta$ then           ▷ $s$ and $d$ are in the same subcube.
    if $s$ AND $2^\delta = 0$ then
        if $|Q^0_{n-1} \cap F| \leq |Q^1_{n-1} \cap F|$ then
            HC-FT-CONS($n-1$, $i$, $s$, $d$, $Q^0_{n-1} \cap F$)
        else
            Select fault-free paths $s \overset{\gamma}{\rightsquigarrow} s' \in Q^1_{n-1}$ and $d \overset{\gamma}{\rightsquigarrow} d' \in Q^1_{n-1}$ of lengths at most three;
            if $(s \overset{\gamma}{\rightsquigarrow} s') \cap (d \overset{\gamma}{\rightsquigarrow} d') = \emptyset$ then
                HC-FT-CONS($n-1$, $i-1$, $s'$, $d'$, $Q^1_{n-1} \cap F$)
            else
                Select $s \overset{\gamma}{\rightsquigarrow} u \overset{\gamma}{\rightsquigarrow} d$ with $u \in (s \overset{\gamma}{\rightsquigarrow} s') \cup (d \overset{\gamma}{\rightsquigarrow} d')$
            end if
        end if
    else
        if $|Q^1_{n-1} \cap F| \leq |Q^0_{n-1} \cap F|$ then
            HC-FT-CONS($n-1$, $i-1$, $s$, $d$, $Q^1_{n-1} \cap F$)
        else
            Select fault-free paths $s \overset{\gamma}{\rightsquigarrow} s' \in Q^0_{n-1}$ and $d \overset{\gamma}{\rightsquigarrow} d' \in Q^0_{n-1}$ of lengths at most three;
            if $(s \overset{\gamma}{\rightsquigarrow} s') \cap (d \overset{\gamma}{\rightsquigarrow} d') = \emptyset$ then
                HC-FT-CONS($n-1$, $i$, $s'$, $d'$, $Q^0_{n-1} \cap F$)
            else
                Select $s \overset{\gamma}{\rightsquigarrow} u \overset{\gamma}{\rightsquigarrow} d$ with $u \in (s \overset{\gamma}{\rightsquigarrow} s') \cup (d \overset{\gamma}{\rightsquigarrow} d')$
            end if
        end if
    end if
else                                                    ▷ $s$ and $d$ are in distinct subcubes.
    if $s$ AND $2^\delta = 0$ then
        if $|Q^0_{n-1} \cap F| \leq |Q^1_{n-1} \cap F|$ then
            Select a fault-free path $d \overset{\gamma}{\rightsquigarrow} d' \in Q^0_{n-1}$ of length at most three;
            HC-FT-CONS($n-1$, $i$, $s$, $d'$, $Q^0_{n-1} \cap F$)
        else
            Reverse HC-FT-CONS($n$, $i$, $d$, $s$, $F$)
        end if
    else
        if $|Q^0_{n-1} \cap F| \geq |Q^1_{n-1} \cap F|$ then
            Select a path $d \overset{\gamma}{\rightsquigarrow} d' \in Q^1_{n-1}$;
            HC-FT-CONS($n-1$, $i-1$, $s$, $d'$, $Q^1_{n-1} \cap F$)
        else
            Reverse HC-FT-CONS($n$, $i$, $d$, $s$, $F$)
        end if
    end if
end if
```

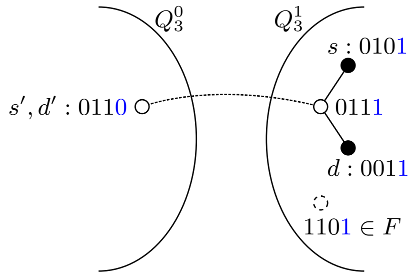

Figure 4: Illustration of Step 2, Case $s, d \in Q^1_{n-1}$. In a Q_4 with reduction bit $\delta = 0$ and constraint $\gamma_2 = (2, 3)$, s, d are connected to the nodes $s', d' \in Q^0_3$, respectively. The paths $s \overset{\gamma}{\leadsto} s'$ and $d \overset{\gamma}{\leadsto} d'$ are not disjoint, and thus a path $s \overset{\gamma}{\leadsto} d$ is found in Q^1_3.

4.2 Routing example

We give in this section an example of the execution trace of the algorithm proposed in Section 4.1. In a Q_5, given a source node s : 01010, a destination node d : 11100, a set of faulty nodes $F = \{11000, 01011\}$ and a bit constraint $\gamma_2 = (2, 3)$, an execution trace of the algorithm of Section 4.1 is given in Table 2. As a result, the fault-free path selected: $s : 01010 \rightarrow 01110 \rightarrow 00110 \rightarrow 00111 \rightarrow 00101 \rightarrow 01101 \rightarrow 01100 \rightarrow d : 11100$ satisfying the constraint γ_2 is selected. An illustration is given in Figure 5.

Table 2: Fault tolerant routing example in a Q_5 with $\gamma_2 = (2, 3)$ bit constraint.

n	2^δ	s	d	F	$\in Q^0_{n-1}$	$\in Q^1_{n-1}$
5	16	01010	11100	$\{f_1 : 11000, f_2 : 01011\}$	s, f_2	d, f_1
selected: $d : 11100 \rightarrow d' : 01100$ *induction on* Q^0_4						
4	8	01010	01100	$\{f_2 : 01011\}$	-	s, d, f_2
selected: $s : 01010 \rightarrow 01110 \rightarrow s' : 00110$ selected: $d : 01100 \rightarrow 01101 \rightarrow d' : 00101$ *induction on* Q^0_3						
3	-	00110	00101	$\emptyset$	-	-
selected: $s : 00110 \rightarrow 00111 \rightarrow d : 00101$ (shortest path routing)						

4.3 Correctness and complexities

In this section, we formally show the correctness of the algorithm described in Section 4.1, and we establish its complexities: maximum path length and worst-case time complexity.

Lemma 3. *The algorithm of Section 4.1 is correct and always terminates.*

Proof. If $|F| \geq 2$, we need to select a bit position δ $(0 \leq \delta \leq n - 1)$ such that Q^0_{n-1} and Q^1_{n-1} each include at least one fault. Such δ can be found simply by computing the exclusive-or bitwise operation of two faulty nodes, and then taking the position of the MSB of the result.

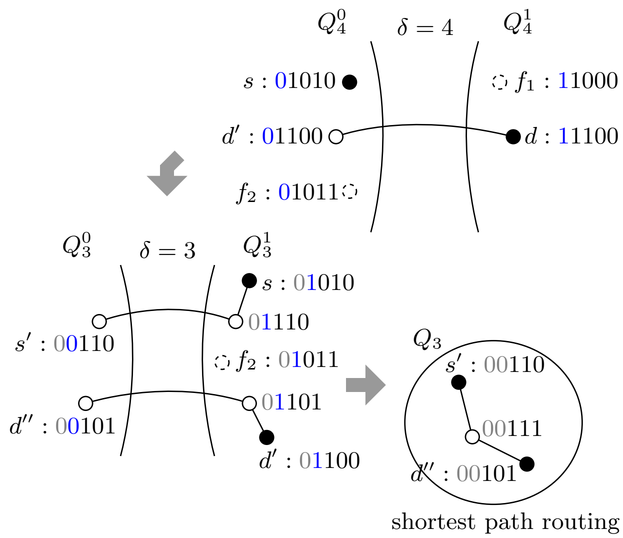

Figure 5: Complete fault tolerant routing example from $s : 01010$ to $d : 11100$ in a Q_5 with $\gamma_2 = (2, 3)$ bit constraint and $f_1 : 11000, f_2 : 01011$ faults.

We now show the existence of at least one fault-free path of length at most three satisfying γ_i. Assume s, d are in distinct subcubes, say $s \in Q_{n-1}(s)$ and $d \in Q_{n-1}(d)$. Assume further that $|F \cap Q_{n-1}(d)| \geq |F \cap Q_{n-1}(s)|$. If $|F \cap Q_{n-1}(d)| < |F \cap Q_{n-1}(s)|$, the same discussion holds by exchanging the roles of s and d.

Assume $d \in Q_{n-1}^1$ and $w(d) = i$. In Q_{n-1}^1, d has $n-1$ neighbours, out of which $n-i$ satisfy γ_i. Because the corresponding $n-i$ paths of lengths two of Π_d^δ, say $d \to d_j'' \to d_j'$, are disjoint except for d, and because d non-faulty, one fault can be included in at most one of these $n-i$ candidate paths. Since $w(d) = i$ and $d \in Q_{n-1}^1$, we have $w(d_j'') = i+1$ and $w(d_j') = i$, thus these $n-i$ paths of lengths two of Π_d^δ all satisfy γ_i. Note that for the same reason, the unique path of length one of Π_d^δ is not a candidate as the unique neighbour of d in Q_{n-1}^0 does not satisfy γ_i. Assume $n-i > i+1$. Then $|F| \leq i \leq n-i-1$, and there always remain at least $(n-i) - |F| \geq (n-i) - (n-i-1) = 1$ fault-free candidate paths connecting d to Q_{n-1}^0. Assume $n-i \leq i+1$. Then $|F| \leq n-i-1$, and there always remain at least $(n-i) - |F| \geq (n-i) - (n-i-1) = 1$ fault-free candidate paths connecting d to Q_{n-1}^0.

Assume $d \in Q_{n-1}^1$ and $w(d) = i+1$. In Q_{n-1}^1, the node d has $n-1$ neighbours, out of which i satisfy γ_i. Consider the corresponding i paths of lengths two of Π_d^δ, say $d \to d_j'' \to \tilde{d}_j$ $(1 \leq j \leq i)$, that is $w(d_j'') = i$ and $w(\tilde{d}_j) = i-1$. Since $n-i \geq 2$, there always exists at least one bit position α such that the α-th bit of d is set to 0. Then, since the i paths considered are mutually disjoint except for d, the paths of lengths three obtained bit flipping the same α-th bit of d_j'', say $d \to d_j'' \to d_j''' \to d_j'$, remain mutually disjoint except for d. Since the α-th bit of d is set to 0 and $w(d_j'') = i$, we have $w(d_j''') = i+1$ and $d_j''' \neq d$. Also, because the paths of Π_d^δ are mutually disjoint except for d and because $d_j''' \neq d$, these paths of lengths three are disjoint (except for d) with the unique path of length one of Π_d^δ, say $d \to d_0'$. So, because d non-faulty, one fault can be included in at most one of these $i+1$ candidate paths. Since $d \in Q_{n-1}^1$ and $w(d) = i+1$, $w(d_0') = i$. Also, since $w(d_j''') = i+1$, $w(d_j') = i$. Thus these $i+1$ candidate paths all satisfy γ_i. Assume $n-i > i+1$. Then $|F| \leq i$, and there always remain at least $(i+1) - |F| \geq 1$ fault-free candidate paths connecting d to Q_{n-1}^0. Assume $n-i \leq i+1$. Then $|F| \leq n-i-1 \leq i$, and there always remain at least $(i+1) - |F| \geq 1$ fault-free candidate paths connecting d to Q_{n-1}^0.

Assume $d \in Q_{n-1}^0$ and $w(d) = i+1$. In Q_{n-1}^0, d has $n-1$ neighbours, out of which $i+1$ satisfy γ_i. Because the corresponding $i+1$ paths of lengths two of Π_d^δ, say $d \to d_j'' \to d_j'$, are disjoint except for d, and because d non-faulty, one fault can be included in at most one of these $i+1$ candidate paths. Since $w(d) = i+1$ and $d \in Q_{n-1}^0$, we have $w(d_j'') = i$ and $w(d_j') = i+1$, thus these $i+1$

paths of lengths two of Π_d^δ all satisfy γ_i. Note that for the same reason, the unique path of length one of Π_d^δ is not a candidate as the unique neighbour of d in Q_{n-1}^1 does not satisfy γ_i. By using the same arguments as in the case $d \in Q_{n-1}^1$ and $w(d) = i+1$, we can deduce that there always remain at least one fault-free candidate path connecting d to Q_{n-1}^1.

Assume $d \in Q_{n-1}^0$ and $w(d) = i$. In Q_{n-1}^0, the node d has $n-1$ neighbours, out of which $(n-1)-i$ satisfy γ_i. Consider the corresponding $(n-1)-i$ paths of lengths two of Π_d^δ, say $d \to d_j'' \to \tilde{d}_j$ $(1 \le j \le i)$, that is $w(d_j'') = i+1$ and $w(d_j') = i+2$. Since $i \ge 1$, there always exists at least one bit position α such that the α-th bit of d is set to 1. Then, since the $(n-1)-i$ paths considered are mutually disjoint except for d, the paths of lengths three obtained bit flipping the same α-th bit of d_j'', say $d \to d_j'' \to d_j''' \to d_j'$, remain mutually disjoint except for d. Since the α-th bit of d is set to 1 and $w(d_j'') = i+1$, we have $w(d_j''') = i$ and $d_j''' \neq d$. Also, because the paths of Π_d^δ are mutually disjoint except for d and because $d_j''' \neq d$, these paths of lengths three are disjoint (except for d) with the unique path of length one of Π_d^δ, say $d \to d_0'$. So, because d non-faulty, one fault can be included in at most one of these $(n-1)-i+1 = n-i$ candidate paths. Since $d \in Q_{n-1}^0$ and $w(d) = i$, $w(d_0') = i+1$. Also, since $w(d_j''') = i$, $w(d_j') = i+1$. Thus these $n-i$ candidate paths all satisfy γ_i. By using the same arguments as in the case $d \in Q_{n-1}^1$ and $w(d) = i$, we can deduce that there always remain at least one fault-free candidate path connecting d to Q_{n-1}^1.

Now assume that s and d are included in the same subcube, say without loss of generality $s, d \in Q_{n-1}^0$. The same discussion holds for $s, d \in Q_{n-1}^1$. If $|F \cap Q_{n-1}^0| \le |F \cap Q_{n-1}^1|$, no path is selected. So assume $|F \cap Q_{n-1}^0| > |F \cap Q_{n-1}^1|$. We have shown previously that there exists at least one fault-free path, say ρ_d, of length at most three satisfying γ_i connecting d to a node of Q_{n-1}^1, and thus similarly a fault-free path, say ρ_s, of length at most three satisfying γ_i connecting s to a node of Q_{n-1}^1. If ρ_s and ρ_d are not disjoint, say they both include the node $u \in Q_{n-1}^0$ with $s \overset{\gamma}{\rightsquigarrow} u \overset{\gamma}{\rightsquigarrow} s'$ and $d \overset{\gamma}{\rightsquigarrow} u \overset{\gamma}{\rightsquigarrow} d'$, then discard the two sub-paths $u \overset{\gamma}{\rightsquigarrow} s'$ and $u \overset{\gamma}{\rightsquigarrow} d'$ so that the path $s \overset{\gamma}{\rightsquigarrow} u \overset{\gamma}{\rightsquigarrow} d$ is selected; the algorithm is terminated.

Now we show that the problem can be solved recursively in one of the two subcubes Q_{n-1}^0 and Q_{n-1}^1. In other words, we show that at each recursive call, the number of faulty nodes in Q_n is at most $\min(n-i, i+1) - 1$. Because the problem is solved recursively inside the subcube containing the least number of faulty nodes, the number f of faulty nodes inside the subcube for induction satisfies $f \le \lfloor |F|/2 \rfloor$.

If the problem is solved recursively in Q_{n-1}^1, the constraint becomes $\gamma_{i-1} = (i-1, i)$ and thus at most $\min((n-1)-(i-1), (i-1)+1) - 1 = \min(n-i, i) - 1$ faulty nodes can be tolerated in Q_{n-1}^1. Assume $n - i > i+1$. Then in Q_n, $|F| \le i$, and at most $\min(n-i, i) - 1 \le i-1$ faulty nodes are tolerated in Q_{n-1}^1. Thus, since in Q_{n-1}^1 there are at most $\lfloor |F|/2 \rfloor$ faulty nodes $(= f)$, we have $f \le \lfloor i/2 \rfloor \le i-1$ as $i \ge 1$, and the problem can be solved recursively in Q_{n-1}^1. Assume $n - i \le i+1$. Then in Q_n, $|F| \le n-i-1$, and at most $\min(n-i, i) - 1 \le n-i-1$ faulty nodes are tolerated in Q_{n-1}^1. Thus, since in Q_{n-1}^1 there are at most $\lfloor |F|/2 \rfloor$ faulty nodes $(= f)$, we have $f \le \lfloor (n-i-1)/2 \rfloor \le n-i-1$ as $i \ge 1$ and $n > i$, and the problem can be solved recursively in Q_{n-1}^1.

If the problem is solved recursively in Q_{n-1}^0, the constraint stays $\gamma_i = (i, i+1)$ and thus at most $\min((n-1)-i, i+1) - 1$ faulty nodes can be tolerated in Q_{n-1}^0. Assume $n-i > i+1$. Then in Q_n, $|F| \le i$, and at most $\min(n-i-1, i+1) - 1 \le i$ faulty nodes are tolerated in Q_{n-1}^0. Thus, since in Q_{n-1}^0 there are at most $\lfloor |F|/2 \rfloor$ faulty nodes $(= f)$, we have $f \le \lfloor i/2 \rfloor \le i$ as $i \ge 1$, and the problem can be solved recursively in Q_{n-1}^0. Assume $n-i \le i+1$. Then in Q_n, $|F| \le n-i-1$, and at most $\min(n-i-1, i+1) - 1 \le n-i-2$ faulty nodes are tolerated in Q_{n-1}^0. Thus, since in Q_{n-1}^0 there are at most $\lfloor |F|/2 \rfloor$ faulty nodes $(= f)$, we have $f \le \lfloor (n-i-1)/2 \rfloor \le n-i-2$ as $i \ge 1$ and $n > i$, and the problem can be solved recursively in Q_{n-1}^0. □

Lemma 4. *The algorithm of Section 4.1 selects a fault-free path* $s \overset{\gamma}{\rightsquigarrow} d$ *of length at most* $n + 5\lceil \log |F| \rceil + 5$ *in* $O(n \log |F|)$ *time.*

Proof. The algorithm is applied recursively in the subcube containing the least number of faulty nodes, until obtaining a fault-free subcube. Hence, at most $r \le 1 + \lceil \log |F| \rceil$ recursive calls are made. Now, at each step of the reduction, at most three edges are selected for the fault-free path connecting s to the opposite subcube, and similarly for d. Once a fault-free subcube is reached, that is after $n-r$ reductions, the shortest path routing algorithm of Section 3.1 is applied in Q_{n-r}, thus selecting a path of length at most $n-r$. Therefore, in total, a fault-free path of length at most $6r+(n-r)=n+5\lceil\log|F|\rceil+5$ is selected. Note that if s and d in the same, most faulty, subcube, we could directly apply recursion on that subcube (proof to be slightly updated), and thus have at most three edges selected at each recursion step, at most $|F|$ recursive calls, therefore resulting in a maximum path length of $3|F|+(n-|F|)=n+2|F|$.

If Q_n is fault-free, the shortest path routing algorithm of Section 3.1 is applied, thus requiring $O(n)$ time. Otherwise, in Step 1, a bit position δ can be obtained with a single XOR operation and MSB detection, thus being constant time. In Step 2, selection of a fault-free path connecting s and d to the opposite subcube requires $O(n)$ time as the n paths of Π_s^δ and Π_d^δ, possibly extended by one edge, may be iterated; including the 1-edge possible extension, these paths are of lengths at most three and node comparison is constant time. Finally, the algorithm is applied recursively on either Q_{n-1}^0 or Q_{n-1}^1, with at most $r\leq 1+\lceil\log|F|\rceil$ recursive calls, thus requiring in total $O(n\log|F|)$ time. □

We can summarise the previous discussion in the following theorem.

Theorem 2. *In a Q_n, given two distinct non-faulty nodes s and d, a set F of at most* $\min(n-i,i+1)-1$ *faulty nodes, and a bit constraint $\gamma_i=(i,i+1)$, we can select a fault-free path $s \overset{\gamma}{\rightsquigarrow} d$ satisfying γ_i of length at most $n+5\lceil\log|F|\rceil+5$ in $O(n\log|F|)$ time.*

Proof. This can be directly deduced from Lemmas 3 and 4. □

5 Empirical evaluation

In this section, we conduct several experiments in order to make an empirical evaluation of the proposed algorithm (Section 4), and thus inspect its practical behaviour. To realise these experiments, the proposed algorithm has been implemented using the Scheme functional programming language [24]. We have then run the program to solve 10,000 random instances of the fault tolerant node-to-node routing problem with bit constraint $\gamma_i=(i,i+1)$ inside an n-dimensional hypercube for each value of n with $i=2$ and $i+2=4\leq n\leq 16$. In total, we have thus solved $13\times 10,000=130,000$ instances of the considered routing problem. First, we conduct such an experiment to measure the average execution time as well as the average and maximum maximal path length. Then, in order to further refine our practical assessment of the proposed algorithm, we conduct a second experiment regarding paths lengths, this time discussing average distances in a hypercube.

5.1 Average execution time and maximum path length

In these experiment, the source node, destination node and faulty nodes were all randomly selected from the set of nodes satisfying the bit constraint γ_i. The source node and the destination node are non-faulty and not necessarily distinct. Also, the number of faulty nodes was always maximised (i.e. $\min(n-i,i+1)-1\leq 2$ faulty nodes). Even though the number of faulty nodes remains low since it depends on the value of i used to define the bit constraint, it is interesting to observe the variations of the paths lengths and how these lengths compare to the theoretical estimations.

The first experimentation consisted in the measurement of the average execution time of the proposed algorithm in the conditions described above, and for each value of n the hypercube dimension. In practice, for each n, we have calculated the average execution time to solve one instance of the routing problem. The results are given in Figure 6, and we also give on this figure for reference the estimated theoretical worst-case time complexity of the algorithm as established in Section 4.3.

We notice a few data bumps which can be easily explained: measured times are very small, of order 0.01ms, and thus time measurement is subject to such small bumps for instance due to the computer core activity. The main observation regarding execution time is that the estimation for the theoretical worst-case time complexity of the proposed algorithm has not been significantly overestimated.

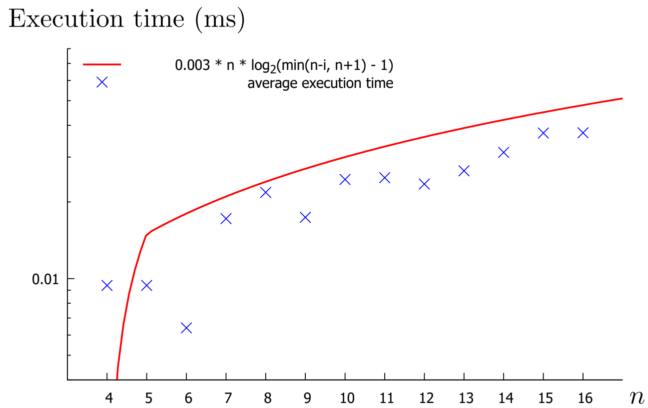

Figure 6: Average execution time for each value of n ($i = 2$ and $4 \leq n \leq 16$).

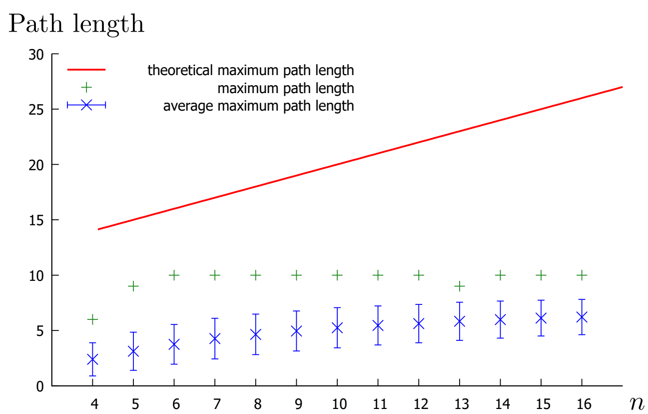

Figure 7: Maximum path length and average maximum path length with standard deviation for each value of n ($i = 2$ and $4 \leq n \leq 16$).

The second experiment consisted in the measurement of the length of the generated path for each solution of the problem instances. We have then deduced from these data the maximum length of a generated path for each value of n. In addition, we measured the average maximum length (and deduced the standard deviation) obtained for each value of n. In practice, for each value of n, we stored for each of the 10,000 problem instances the length of the generated path. Then, for each value of n, we calculated from these data the maximum and average values of these path lengths. The results obtained are given in Figure 7, and we also give on this figure for reference the estimated theoretical worst-case maximum path length of the algorithm as established in Section 4.3.

In practice, one can see that the average performance of the proposed algorithm is significantly better than the theoretical worst-case estimations; paths are effectively shorter. In addition, one can observe that maximum length of a generated path is stabilising for dimensions beyond 6. One can think of two reasons for this behaviour, both strongly related to the constraint γ_i. First, because the source and destination nodes both satisfy γ_i, their maximum Hamming distance is limited to $i + (i + 1)$, that is 5 in our experiment. Also, the value of i induces a rather low number of faults, and as the dimension of the hypercube used to perform routings increases, the probability that a path has to make a detour to avoid a faulty node is getting lower given that the number of faulty nodes remains constant in our experiment for $n \geq 5$ due to the experimental parameters.

5.2 Comparison with average distances in a hypercube

So as to better assess the average performance of the proposed routing algorithm, we shall here conduct additional experimentation regarding the lengths of generated paths. We start by discussing the average length of a shortest path in a hypercube between any two nodes, not necessarily distinct. To this aim, we calculate the length of a shortest path in average, ignoring the presence of faulty nodes. In a hypercube Q_n, for a source node s, there are exactly $C_n^1 = n$ nodes distant from 1 edge to s, actually these nodes are the neighbours of s, C_n^2 nodes distant from 2 edges to s, and in general C_n^k nodes distant from k edges to s, with $0 \leq k \leq n$. In order to establish the average distance from s to any node of a Q_n, we weight each number of nodes at distance k to s by the distance itself, that is k. Hence, in total, the sum of the distances from a node s to every other nodes of a Q_n is equal to $0C_n^0 + 1C_n^1 + 2C_n^2 + \ldots + nC_n^n$. This total value is divided by the number of nodes, that is 2^n, and we obtain λ_1 the average distance (i.e. the average length of a shortest path) between any two nodes:

$$\lambda_1 = \frac{1}{2^n} \sum_{i=0}^{n} i \times C_n^i$$

We show in appendix with Theorem 3 that this average distance in a hypercube Q_n can be simply expressed as:

$$\lambda_1 = \frac{n}{2}$$

In order to assess even more accurately the average performance of the proposed algorithm, let us discuss the average length of a shortest path that satisfies a bit constraint $\gamma_i = (i, i+1)$ (i.e. average Hamming distance). For a node $u = 1\ldots 10\ldots 0$ with $w(u) = i$, first consider a node $v = v_1 v_2 \ldots v_i v_{i+1} \ldots v_n$ where $\sum_{k=i+1}^{n} v_k = j$ and $\sum_{k=1}^{i} v_k = i - j$. Then $w(v) = i$ and the Hamming distance between u and v is $2j$. There are $\sum_{j=0}^{i} C_i^{i-j} \cdot C_{n-i}^{j} = C_n^i$ such nodes v. Next, let us consider a node $x = x_1 x_2 \ldots x_i x_{i+1} \ldots x_n$ where $\sum_{k=i+1}^{n} x_k = j+1$ and $\sum_{k=1}^{i} x_k = i - j$. Then $w(x) = i+1$ and the Hamming distance between u and x is $2j+1$. There are $\sum_{j=0}^{i} C_i^{i-j} \cdot C_{n-i}^{j+1} = C_n^{i+1}$ such nodes x. Now, we can calculate λ_2 the average Hamming distance between u and v or x as follows.

$$\begin{aligned}
\lambda_2 &= \left(\sum_{j=0}^{i} C_i^{i-j} \cdot C_{n-i}^{j} \cdot 2j + \sum_{j=0}^{i} C_i^{i-j} \cdot C_{n-i}^{j+1} \cdot (2j+1) \right) \Big/ \left(C_n^i + C_n^{i+1} \right) \\
&= \left(\sum_{j=0}^{i} C_i^{i-j} \cdot 2j \cdot \left(C_{n-i}^{j} + C_{n-i}^{j+1} \right) + \sum_{j=0}^{i} C_i^{i-j} \cdot C_{n-i}^{j+1} \right) \Big/ \left(C_n^i + C_n^{i+1} \right) \\
&= \left(\sum_{j=0}^{i} C_i^{j} \cdot 2j \cdot \left(C_{n-i}^{j} + C_{n-i}^{j+1} \right) + C_n^{i+1} \right) \Big/ C_{n+1}^{i+1} \\
&= \left(2i \sum_{j=0}^{i} C_{i-1}^{j-1} \cdot C_{n-i+1}^{j+1} + C_n^{i+1} \right) \Big/ C_{n+1}^{i+1} \\
&= \left(2i \cdot C_n^{i+1} + C_n^{i+1} \right) / C_{n+1}^{i+1} \\
&= \frac{(2i+1)(n-i)}{n+1}
\end{aligned}$$

A similar discussion can be applied to the case $w(u) = i+1$ and $w(v) \in \{i, i+1\}$, and we obtain in this case the following average distance.

$$\lambda_3 = \frac{(2(n-i)-1)(i+1)}{n+1}$$

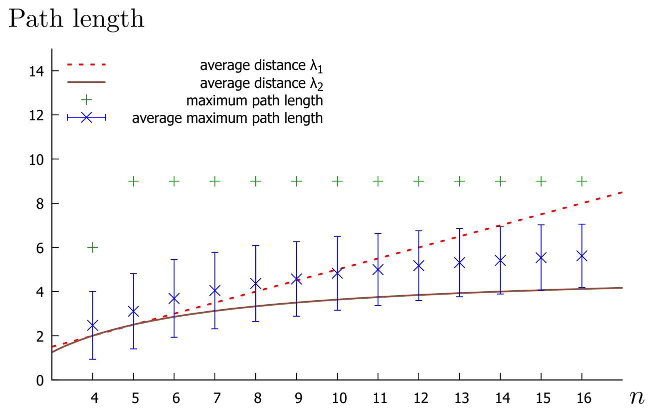

Figure 8: Maximum path length and average maximum path length with standard deviation for each value of n ($i = 2$ and $4 \leq n \leq 16$) when $w(s) = i$. The average distances λ_1 and λ_2 in a Q_n are also represented.

We can thus estimate these average distances λ_1, λ_2 and λ_3 for the values of n used in this experiment. Notably, the average distances λ_2, λ_3 give a lower bound on the path length as obtained by the proposed algorithm. So as to fairly compare the empirical results with these bounds λ_1, λ_2 and λ_3, we conduct a similar experiment as in Section 5.1: we solve 10,000 random instances of the fault tolerant node-to-node routing problem with bit constraint $\gamma_i = (i, i+1)$ inside a Q_n for each n with $i = 2$ and $i + 2 = 4 \leq n \leq 16$. Yet here, we conduct two different measurements, one with the source node randomly selected from the set of nodes whose weights are equal to i, and another measurement with the source node randomly selected from the set of nodes whose weights are equal to $i+1$. We recall that the source node and the destination node are non-faulty and not necessarily distinct, and that the number of faulty nodes was always maximised. The results in the case when $w(s) = i$ are illustrated in Figure 8, and the results in the case when $w(s) = i + 1$ are illustrated in Figure 9.

One can observe that the average maximum path length obtained when running the proposed algorithm is close to the average distance as formally discussed previously (λ_2, λ_3). Let us recall that faulty nodes were not considered in our calculation of the average distance between any two nodes in a hypercube. Taking into consideration faulty nodes may raise the average distance values even closer to the obtain average maximum path lengths for small values of n. As explained previously, the constraint γ_i is inducing an upper bound on the Hamming distance between the source and destination nodes, and thus, as the hypercube dimension n increases, the average maximum path length is getting more distant from the average distance λ_1. So, by considering small values of n, the small gap between the average distance λ_1 and the average maximum path length is yet another strong indication of the good performance of our algorithm.

6 Conclusion

Enforcing a bit constraint when routing in a hypercube has several interesting applications. In this paper, we have first described a shortest path routing algorithm in a hypercube Q_n that selects a path $s \overset{\gamma}{\rightsquigarrow} d$ of length $H(s, d)$ and satisfies a bit constraint $\gamma_i = (i, i+1)$. We have formally shown the correctness of this algorithm and that it is time optimal $O(H(s, d))$. Next, we have proposed a fault tolerant node-to-node routing algorithm in a Q_n that selects a fault-free path $s \overset{\gamma}{\rightsquigarrow} d$ of length at most $n + 5\lceil \log |F| \rceil + 5$ in $O(n \log |F|)$ time. Given a set of faulty nodes F, the maximum number of faults tolerated is $|F| \leq \min(n-i, i+1) - 1$. We have formally proved the correctness and complexities of this algorithm. Lastly, we have conducted several experiments in order to inspect the practical behaviour of the proposed algorithm. We have shown through these experiments that in practice, the algorithm is performing very well compared to the theoretical worst-case estimations.

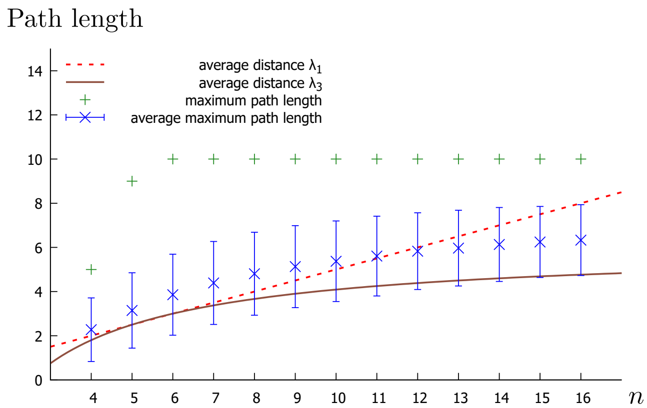

Figure 9: Maximum path length and average maximum path length with standard deviation for each value of n ($i = 2$ and $4 \leq n \leq 16$) when $w(s) = i + 1$. The average distances λ_1 and λ_3 in a Q_n are also represented.

As for future works, it would be interesting to consider faulty clusters rather than just faulty nodes, starting with clusters of small diameters.

Acknowledgements

The authors sincerely thank the reviewers for their insightful comments and suggestions which significantly helped improving this paper. This study was partly supported by a Grant-in-Aid for Scientific Research (C) of the Japan Society for the Promotion of Science under Grant No. 25330079.

References

[1] Sheldon B. Akers and Balakrishnan Krishnamurthy, “A group-theoretic model for symmetric interconnection networks”, IEEE Transactions on Computers, vol. C-38, no. 4, pp. 555–566, 1989.

[2] Charles L. Seitz, “The cosmic cube”, Communications of the ACM, vol. 28, no. 1, pp. 22–33, 1985.

[3] TOP500. List. `http://top500.org/list/2014/06/`, June 2014. Last accessed July 2014.

[4] Qutaibah Marwan Malluhi and Magdy A. Bayoumi, “The hierarchical hypercube: a new interconnection topology for massively parallel systems”, IEEE Transactions on Parallel and Distributed Systems, vol. 5, no. 1, pp. 17–30, 1994.

[5] Antoine Bossard and Keiichi Kaneko, “The set-to-set disjoint-path problem in perfect hierarchical hypercubes”, The Computer Journal, vol. 55, no. 6, pp. 769–775, 2012.

[6] Antoine Bossard and Keiichi Kaneko, “k-pairwise disjoint paths routing in perfect hierarchical hypercubes”, The Journal of Supercomputing, vol. 67, no. 2, pp. 485–495, 2014.

[7] Shuming Zhou, Limei Lin and Jun-Ming Xu, “Conditional fault diagnosis of hierarchical hypercubes”, International Journal of Computer Mathematics, vol. 89, no. 16, pp. 2152–2164, 2012.

[8] Kanad Ghose and Kiran Raghavendra Desai, “The HCN: a versatile interconnection network based on cubes”, In Proceedings of the 1989 ACM/IEEE Conference on Supercomputing, pp. 426–435, Reno, NV, USA, November 12–17, 1989.

[9] Antoine Bossard and Keiichi Kaneko, “Node-to-set disjoint-path routing in hierarchical cubic networks”, The Computer Journal, vol. 55, no. 12, pp. 1440–1446, 2012.

[10] Antoine Bossard and Keiichi Kaneko, “Set-to-set disjoint paths routing in hierarchical cubic networks”, The Computer Journal, vol. 57, no. 2, pp. 332–337, 2014.

[11] Yamin Li, Shietung Peng, and Wanming Chu, “Metacube - a versatile family of interconnection networks for extremely large-scale supercomputers”, The Journal of Supercomputing, vol. 53, no. 2, pp. 329–351, 2010.

[12] Antoine Bossard, Keiichi Kaneko, and Shietung Peng, “Node-to-set disjoint paths routing in a metacube”, International Journal of High Performance Computing and Networking, (in press), 2014.

[13] Yamin Li, Shietung Peng, and Wanming Chu, “Efficient collective communications in dual-cube”, The Journal of Supercomputing, vol. 28, no. 1, pp. 71–90, 2004.

[14] Yuan-Kang Shih, Hui-Chun Chuang, Shin-Shin Kao, and Jimmy J. Tan, “Mutually independent Hamiltonian cycles in dual-cubes”, The Journal of Supercomputing, vol. 54, no. 2, pp. 239–251, 2010.

[15] Qian-Ping Gu and Shietung Peng, “An efficient algorithm for the k-pairwise disjoint paths problem in hypercubes”, Journal Parallel and Distributed Computing, vol. 60, no. 6, pp. 764–774, 2000.

[16] Qian-Ping Gu and Shietung Peng, “Node-to-set and set-to-set cluster fault tolerant routing in hypercubes”, Parallel Computing, vol. 24, pp. 1245–1261, 1998.

[17] Jianer Chen, Iyad A. Kanj, and Guojun Wang, “Hypercube network fault tolerance: a probabilistic approach”, Journal of Interconnection Networks, vol. 6, no. 1, pp. 17–34, 2005.

[18] Ozgur Sinanoglu, Mehmet Hakan Karaata, and Bader AlBdaiwi, “An inherently stabilizing algorithm for node-to-node routing over all shortest node-disjoint paths in hypercube networks”, IEEE Transactions on Computers, vol. 59, no. 7, pp. 995–999, 2010.

[19] Cheng-Nan Lai, “Optimal construction of all shortest node-disjoint paths in hypercubes with applications”, IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 6, pp. 1129–1134, 2012.

[20] Hyeong-Ok Lee, Jong-Seok Kim, Eunseuk Oh, and Hyeong-Seok Lim, “Hyper-Star Graph: a new interconnection network improving the network cost of the hypercube”, In Proceedings of the First EurAsian Conference EurAsia ICT: Information and Communication Technology, pp. 858–865, Shiraz, Iran, October 29–31, 2002.

[21] Antoine Bossard, “A set-to-set disjoint paths routing algorithm in hyper-star graphs”, ISCA International Journal of Computers and Their Applications, vol. 21, no. 1, pp. 76–82, 2014.

[22] Youcef Saad and Martin H. Schultz, “Topological properties of hypercubes”, IEEE Transactions on Computers, vol. 37, no. 7, pp. 867–872, 1988.

[23] Karl Menger, “Zur allgemeinen Kurventheorie”, Fundamenta Mathematicae, vol. 10, pp. 96–115, 1927.

[24] Robert Bruce Findler, John Clements, Cormac Flanagan, Matthew Flatt, Shriram Krishnamurthi, Paul Steckler, Matthias Felleisen, “DrScheme: a programming environment for scheme”, Journal of Functional Programming, vol. 12, no. 2, pp. 159–182, 2002.

A Appendix

In this appendix, we discuss the average distance between any two nodes in a hypercube, in other words the average length of a shortest path.

Theorem 3. *The average distance between any two nodes in a hypercube Q_n is $n/2$.*

Proof. A Q_n consists of 2^n nodes. For any one node s, there is $C_n^0 = 1$ node at distance 0 from s, $C_n^1 = n$ nodes at distance 1 from s, and so on, and in general there are C_n^k nodes at distance k ($0 \leq k \leq n$) from s. Hence, the sum of all the distances to each node of Q_n from s is equal to $\sum_{i=0}^{n} iC_n^i$. From the binomial theorem, we have $(1+x)^n = \sum_{k=0}^{n} C_n^k x^k$ and we can deduce by derivation that $n(1+x)^{n-1} = \sum_{k=0}^{n} kC_n^k x^{k-1}$. By setting $x = 1$, we obtain $n2^{n-1} = \sum_{k=0}^{n} kC_n^k$, with the right-hand side expression thus representing the sum of all the distances from node 0 to each node of Q_n. We divide the left-hand side by 2^n to obtain the average distance $n/2$ between any two nodes of a Q_n. $\square$