Unsafety Vectors: A New Fault-Tolerant Routing for the Binary $n$-Cube

J. Al-Sadi, K. Day*, M. Ould-Khaoua

Computer Science Department
Strathclyde University
Glasgow
U.K.

*Computer Science Department
Sultan Qaboos University
Muscat
Sultanate of Oman
Unsafety Vectors: A New Fault-Tolerant Routing for the Binary $n$-Cube

Abstract
This paper presents a new fault-tolerant routing algorithm for the binary $n$-cube which overcomes the limitations of the recently-proposed safety vectors algorithm [22]. The algorithm is based on the concept of “unsafety vectors”. Each node $A$ starts by computing a first level unsafety set, $S_1^A$, composed of the set of unreachable neighbours. It then performs $(m-1)$ exchanges with its neighbours to determine the $k$-level unsafety set, $S_k^A$, for all $1 \leq k \leq m$, where $m$ is an adjustable parameter between 1 and $n$. $S_k^A$ represents the set of all nodes at Hamming distance $k$ from node $A$ which are faulty or unreachable from $A$ due to faulty nodes (or links). Equipped with these unsafety sets, each node calculates unsafety vectors, which are then used to achieve an efficient fault-tolerant routing in the binary $n$-cube. The $k$-th element of the unsafety vector of node $A$ represents a measure of the routing unsafety at distance $k$ from $A$. We present an analytical study proving some properties of the proposed algorithm. We also conduct a comparative analysis through extensive simulation experiments that reveal the superiority of the proposed algorithm over the safety vectors algorithm [22] in terms of different performance measures, e.g. routing distances and percentage of reachability.

Keywords: Multicomputers, Interconnection Networks, Hypercube, Fault-Tolerant Routing, Safety Vectors, Performance Evaluation.
1. Introduction

The binary $n$-cube (or the cube for short) has been one of the most popular networks for multicomputers due to its attractive topological properties, e.g. regular structure, low diameter, and ability to exploit communication locality found in many parallel applications. Several commercial and experimental systems have been used this network including the NCUBE-2 [12], iPSC/2 [14], Cosmic Cube [15], and SGI Origin 2000 multiprocessor [16].

A routing algorithm specifies how a message selects a path to cross from source to destination, and has great impact on network performance. Routing in fault-free cubes has been extensively studied in the past [7, 13, 18, 19]. As the network size scales up the probability of processor and link failure also increases. It is therefore essential to design fault-tolerant routing algorithms that allow messages to reach their destinations even in the presence of faulty components (links and nodes). Several fault-tolerant routing algorithms for the cube have been suggested in the literature [1, 3, 10, 8, 11, 17]. Most of these algorithms have assumed that a node knows either only the status of its neighbours (such a model is called local-information-based) [1, 3, 8] or the status of all nodes (global-information-based) [2,20]. Local-information-based routing yields sub-optimal routes (if not routing failure) due to the insufficient information upon which the routing decisions are made. Global-information-based routing can achieve optimal or near optimal routing, but often at the expense of high communication overhead to maintain up-to-date network-wide fault information. The main challenge is therefore to devise a simple and efficient way of representing limited global fault information that allows optimal or near-optimal fault-tolerant routing. There has been recently a number of attempts to design limited-global-information-based algorithms, and the section below briefly reviews some of the important algorithms that have been proposed for the cube [6, 22, 11, 23].

2. Related Work

Gordon and Stout [8] have proposed a fault-tolerant routing based on “Sidetracking”, where a message is de-routed to a randomly chosen fault-free neighbouring node when there exists no fault-free neighbour along any of the existing optimal paths leading to the destination. With this approach a routing failure may occur (although with low probability for large $n$), and excessive delay may arise even in the presence of few faulty components [4]. Chen and shin [3] have proposed a routing strategy based on
depth-first search in which backtracking is required if all the required forward links cannot be used due to faulty components. The traversed path is recorded and attached to the message. A simplified version of this approach that tolerates fewer faults was presented in [1], where routing is progressive without backtracking, and where a message is routed to its destination on an optimal path with high probability. Lan [9] has presented a fault-tolerant routing algorithm based on local information, and which guarantees an optimal or near-optimal routing. However, the algorithm is based on a restricted model of fault distribution as it can tolerate only \((n-1)\) faulty nodes (and/or links) in an \(n\)-dimensional cube.

Lee and Hayes [11] have used the concept of unsafe nodes to design a fault-tolerant routing strategy. Message routing is achieved by avoiding unsafe nodes, which could possibly lead to communication difficulties and excessive delays. Chiu and Wu [4] have used the concept of unsafe nodes [11] and its extensions to show that a feasible path of length not more than the Hamming distance plus four can be guaranteed provided that the number of faulty nodes does not exceed \((n-1)\). The concept of unsafe nodes has also been discussed in [23].

The author in [21] has presented the concept of safety levels, based on limited global information, as an enhancement of the unsafe node concept. The safety level is an approximate measure of the number as well as the distribution of faulty nodes. Optimal routing is guaranteed if the safety level of the source node is less than the Hamming distance between the source and destination. Chiu and Chen [5] have proposed the concept of routing capability, which further enhances the safety levels concept. Using the routing capability concept, \(n\) binary bits are attached to every node. A node is called \(k\)-capable if the \(k\)th bit is 0; otherwise it is called not \(k\)-capable.

The safety vectors algorithm, proposed by Wu [22], is a similar concept to the routing capability with some extensions related to dynamic routing adaptivity and application to the generalized hypercube. The safety vectors approach require each processor to maintain a bit vector (safety vector) computed through a number of fault information exchanges between adjacent processors. The algorithm guarantees optimal routing to all destinations which are at a Hamming distance \(k\) from node \(A\), if and only if, the \(k\)th bit of the safety vector at node \(A\) is set. A major drawback of this algorithm is related to its conservative approach. Indeed, when the \(k\)th bit of the safety vector of node \(A\) is not set, node \(A\) is not considered for forwarding messages to any destination at distance \(k\). Even when all destinations at distance \(k\) are reach-
able from node $A$ via fault-free optimal paths except for one such destination, node $A$ is excluded as a forwarding node for all these destinations. This causes high percentages of sub-optimal routing and routing failure. Furthermore, Wu’s algorithm fails to route messages in some network partitioning cases causing infinite looping.

This paper proposes a new limited-global information-based routing algorithm for the cube. Each node $A$ starts by determining the set of faulty or unreachable neighbours. Then, each node $A$ performs $(n-1)$ exchanges with its neighbours to determine its faulty set containing all faulty or unreachable nodes at different distances from node $A$. The unsafety sets are derived from the faulty sets according to the Hamming distance between the node and the elements of its faulty set. The $k$-level unsafety set $S_k^A$ for all $1 \leq k \leq m$, where $m$ is an adjustable parameter between 1 and $n$, represents the set of all nodes at Hamming distance $k$ from $A$ which are faulty or unreachable from $A$. Equipped with these unsafety sets, we show how each node calculates numeric unsafety vectors and uses them to achieve efficient fault-tolerant routing. The larger the value of $m$ is, the better the routing decisions are, but at the expense of more computation and communication overhead. The paper includes an analytical study proving some properties of the proposed algorithm. These analytical results are supported by a performance comparison between the proposed routing and the safety vectors algorithm through extensive simulation experiments. The results reveal that the unsafety sets algorithm outperforms the safety vectors algorithm in terms of routing distances and percentage of reachability even when the parameter $m$ is at its lowest value of 1 corresponding to minimum overhead.

The rest of the paper is organised as follows. Section 3 reviews some background information (preliminaries and notation) that will be useful for the subsequent sections. Section 4 presents the proposed fault-tolerant algorithm for the cube, and discusses some of its properties. Section 5 presents a comparative evaluation of the new algorithm and safety vectors algorithm through simulation experiments. Section 6 concludes this study.

3. Preliminaries and Notation

The $n$-dimensional binary $n$-cube $Q_n$ is an undirected graph with $2^n$ vertices, representing nodes, which are labeled by the $2^n$ binary strings of length $n$. Two nodes are joined by an edge if, and only if,
their labels differ in exactly one bit position. The label of node $A$ is written $a_na_{n-1}\ldots a_1$, where $a_i \in \{0, 1\}$ is the $i$-th bit (or bit at $i$-th dimension). The neighbour of a node $A$ along the $i$-th dimension is denoted $A^{(i)}$. A faulty $n$-cube contains faulty nodes and/or links. The Hamming distance between a node $A$ and a node $B$, denoted $H(A, B)$, is the number of bits at which their labels differ. In other words, $H(A, B) = |A \oplus B|$ where $\oplus$ denotes the "exclusive or" binary operation. A path between two nodes $A$ and $B$ is an optimal path if its length is equal to $H(A, B)$.

With respect to a given destination node, $D$, a neighbour $A^{(i)}$ of node $A$ is called a preferred neighbour for the routing from $A$ to $D$ if the $i$-th bit of $A \oplus D$ is 1. We say in this case that $i$ is a preferred dimension. Neighbours other than preferred neighbours are called spare neighbours. Routing through a spare neighbour increases the routing distance by two over the minimum distance. An optimal path can be obtained by routing through all preferred dimensions in some order. A node $T$ is called an $(A, D)$-preferred transit node if any preferred dimension for the routing from $A$ to $T$ is also a preferred dimension for the routing from $A$ to $D$.

**Example:** Suppose that $A = 1101$ and $D = 1010$. We have $A \oplus D = 0111$. Therefore, among the neighbours of $A$, nodes 1100, 1111, and 1001 are preferred neighbours and node 0101 is a spare neighbour. Nodes 1000, 1001, 1011, 1100, 1110 and 1111 are preferred transit nodes for the routing from $A$ to $D$.

We make the following assumptions for the proposed algorithm and comparative study (presented later). Similar assumptions have been assumed in earlier related works [22].

1. The fault pattern remains fixed for the duration of routing.
2. Each node can determine the status of its own communication links and the status of its neighbouring nodes.
3. If there is a faulty link between two nodes then each of the two nodes considers the node at the other end as faulty.

**4. The Proposed Fault-Tolerant Routing Algorithm**

Our proposed fault-tolerant routing algorithm, based on the concept of unsafety sets (defined below), presents a remedy for the major limitations of the safety vectors algorithm. These limitations are related to its conservative (pessimistic) routing approach and inapplicability to the link partitioning fault configu-
rations. Before presenting the new algorithm let us discuss first how a node calculates its unsafety sets.

### 4.1. Calculating Faulty and Unsafety Sets

**Definition 1:** The first-level unsafety set $S_{1}^{A}$ of a node $A$ is defined as

$$S_{1}^{A} = \bigcup_{1<i<n} f_{i}^{A},$$

where $f_{i}^{A}$ is given by

$$f_{i}^{A} = \begin{cases} \{A^{(i)}\} & \text{if } A^{(i)} \text{ is faulty} \\ \emptyset & \text{Otherwise} \end{cases}$$

**Definition 2:** An isolated node is associated with first-level unsafety set containing $n$ addresses of faulty nodes, i.e., $|S_{1}^{A}| = n$.

**Definition 3:** If for some node $A$, $|S_{1}^{A}| = n - 1$, then node $A$ is called a dead-end node.

Each node then uses the unsafety set to determine the faulty set $F_{A}$, which comprises those nodes which are either faulty or unreachable from $A$ due to faulty nodes or links. This is achieved by performing $(n-1)$ exchanges with the reachable neighbours. After determining $F_{A}$, node $A$ calculates $m$ unsafety sets denoted $S_{1}^{A}, S_{2}^{A}, ..., S_{m}^{A}$ (defined below), where $m$ is an adjustable parameter between 1 and $n$.

**Definition 4:** The $k$-level unsafety set $S_{k}^{A}, 1 \leq k \leq m$, for node $A$ is given by

$$S_{k}^{A} = \{B \in F_{A} | H(A,B) = k\}$$

The $k$-level unsafety set $S_{k}^{A}$ represents node $A$’s view of the set of nodes at Hamming distance $k$ from $A$ which are faulty or unreachable from $A$ due to faulty nodes and links. Notice that if the network is disconnected due to faulty nodes and links, $A$’s view about unreachable nodes may not be accurate. In this case massage looping will occur. We present later (see Section 4.4.) a method for detecting and handling such looping. Fig 1. gives an outline of the $\text{Find\_Unsafety\_Sets}$ algorithm that node $A$ uses it to determine its faulty and unsafety sets.

**Example 1:** Consider a four-dimensional cube with five faulty nodes (faulty nodes are represented as black nodes), as shown in Fig.2. Table 1 shows the corresponding first-level unsafety set, $S_{1}^{A}$, associated with each node $A$. The $\text{Find\_Unsafety\_Sets}$ algorithm calculates the sets $S_{k}^{A}$ for all $1 \leq k \leq m$ after calculating $F_{A}$. To achieve this, $(m-1)$ exchanges of fault information are performed among neighbouring
Let $m=n$ and for the sake of specific illustrations let us compute the unsafety sets associated with node $A=0000$. First, the node assigns the addresses of its immediate faulty neighbours to its faulty set $F_A$. Then each node performs $n-1$ exchanges of the new elements of its faulty set $F_A$ with the immediate non-faulty neighbours. After determining $F_A$, node $A$ calculates $m$ unsafety sets denoted $S^A_1$, $S^A_2$, ..., $S^A_m$ according to the Hamming distance between node $A$ and each element of $F_A$. So, the faulty set for node $A$ in our example, given in decimal representation, $F_A = \{1, 8, 10, 12, 15\}$, and the unsafety sets are $S^A_1 = \{1, 8\}$, $S^A_2 = \{10, 12\}$, $S^A_3 = \{\}$, and $S^A_4 = \{15\}$.

\begin{algorithm}
\textbf{Algorithm} \text{Find Unsafety Sets} \ (A: \text{node}) \ \\
/* called by node $A$ to determine its faulty set $F_A$ */ \ \\
$S^A_1 = \text{set of faulty or unreachable immediate neighbours};$ \ \\
$F_A = S^A_1;$ \ \\
\text{for } k := 2 \text{ to } n \text{ do}$ \ \\
\{ \ \\
\text{for } i := 1 \text{ to } n \text{ do}$ \ \\
\quad \text{if } A^{(i)} \not\in F_A \text{ then}$ \ \\
\quad \{ \ \\
\quad \text{send } F_A \text{ to } A^{(i)};$ \ \\
\quad \text{receive } F_A^{(i)} \text{ from } A^{(i)};$ \ \\
\quad F_A = F_A \cup F_A^{(i)}; \ \\
\quad \} \ \\
\} \ \\
\text{for } k := 1 \text{ to } m \text{ do}$ \ \\
$S^A_k = \{B \in F_A \mid H(A, B) = k\}$ \ \\
\text{End.}
\end{algorithm}
4.2. The Unsafety Vectors Routing Algorithm

**Definition 5:** For a given source-destination pair of nodes \((A, D)\), we define the \((A, D)\)-unsafety vector 

\[ U^{A,D} = (u_1^{A,D}, u_2^{A,D}, \ldots, u_m^{A,D}) \]

where its \(k^{th}\) element is given by 

\[ u_k^{A,D} = |\{ T \in S_k^A, \text{ such that } T \text{ is an } (A,D)\text{-preferred transit node}\}|. \]

In other words, \(u_k^{A,D}\) is the number of faulty or unreachable \((A, D)\)-preferred transit nodes at distance \(k\) from \(A\). \(u_k^{A,D}\) can be viewed as a measure of routing unsafety at distance \(k\) from \(A\), hence the name unsafety vectors for \(U^{A,D}\). We also define an ordering relation ‘<’ for numeric vectors as follows. For any two numeric vectors \(U = (u_1, u_2, \ldots, u_m)\) and \(V = (v_1, v_2, \ldots, v_m)\), \(U < V\) iff \(\exists\ i, 1 \leq i \leq m\), such that \(u_i < v_i\),
and $u_j = v_j$ for all $j < i$. Fig. 3 shows the Unsafety Vectors algorithm that each node in the network applies to route a message towards its destination node $D$.

```
Algorithm Unsafety_Vectors (M: message; A,D: node)
/* called by node A to route the message M to its destination node D */
if A is source node then M.Route_distance = 0
if Route_distance <= H(A,D) + 2 * No-FaultyNodes then
    M.Route_distance := M.Route_distance + 1
if A = D then exit; /* destination reached */
if $\exists$ a preferred non-faulty neighbour $A^{(i)}$ such that $u_i^{A^{(i)},D} \leq i$ and $i \leq H(A,D) - 2$ then send to $A^{(i)}$ /* (1)*/
Let $A^{(i)}$ be the reachable preferred neighbour with least $(A^{(i)},D)$-unsafety vector $U^{A^{(i)},D}$ And $A^{(i)}$ is not dead-end
if $A^{(i)}$ exists then
    send M to $A^{(i)}$
else {
    Let $A^{(j)}$ be the reachable spare neighbour with least $(A^{(j)},D)$-unsafety vector $U^{A^{(j)},D}$ And $A^{(j)}$ is not dead-end;
    if $A^{(j)}$ exists then
        send M to $A^{(j)}$
    else failure /* destination unreachable */
}
else Handle looping /* will be discussed in section 4.3. */
End.
```

Fig. 3: A description of the proposed Unsafety Vectors routing algorithm
**Example 2:** Consider the cube depicted in Fig 2. If the source node $A=0010$ and the destination node $D=1101$. Let $m=1$. According to the unsafety vectors algorithm, the source node $A$ will route a message to a preferred neighbour associated with the least number of preferred faulty nodes in its unsafety sets, which is node 0110. By performing the same operations the message will be routed to node 0100 then 0101 and finally to its destination 1101.

4.3. Handling Message Looping

The unsafety vectors algorithm can be improved to minimize the effect of looping. Notice from the above algorithm that looping is detected if the routing distance exceeds the specified limit. Since looping occurs when a destination is not reachable from the source we can add the destination node to the faulty set of the node that detected the looping. When this occurs $(n-1)$ exchanges of information between all neighbours is then initiated to propagate the new information among reachable nodes in the whole cube. We show later in Section 5 that the percentage of looping decreases significantly to less than 1% regardless of the number of faulty nodes in the cube when we include this simple mechanism to handle message looping.

4.4. Properties of the Unsafety Vectors Algorithm

The proposed unsafety vectors Algorithm is capable to route messages in a faulty $n$-cube via fault-free minimal paths if they exist, otherwise the algorithm routes massages via fault-free near-minimal paths if they exist. The unsafety vectors algorithm guarantees to route messages by minimum distance between source node and destination node.

**Theorem 1:** Given a non-faulty pair of source node and destination node $(A, D)$, if $u_i^{A,D} \leq i$, for all $1 \leq i \leq H(A,D)-1$, then there exists at least one minimal fault free path between the source $A$ and the destination $D$.

**Proof:** (By induction on $H(A, D)$). For $H(A, D)=2$, assume that $u_1^{A,D} \leq 1$. Let $A^{(i)}$ and $A^{(j)}$ be the two $(A,D)$-preferred transit nodes. Since $u_1^{A,D} \leq 1$ either $A^{(i)}$ or $A^{(j)}$ is non faulty (assume it is $A^{(i)}$). Therefore, the path $A \rightarrow A^{(i)} \rightarrow D$ is minimal and fault-free.

Let us assume that the property satisfied for $H(A, D) \leq k$ for some $2 \leq k \leq n$. Now, consider the case
\( H(A, D) = k+1. \) Since \( U_k^{A,D} \leq k \) at least one of the \( k+1 \) \((A,D)\)-preferred transit nodes at distance \( k \) from \( A \) is non faulty. Let \( B \) be such a node. Notice that the set of \((A,B)\)-preferred transit nodes at any distance \( i \) from \( A, 1 \leq i \leq k \), is a subset of the set of \((A,D)\)-preferred transit nodes at the same distance \( i \) from \( A \). Therefore \( U_i^{A,B} \leq U_i^{A,D} \leq i \).

By induction hypothesis, there exists a minimal faulty free path from \( A \) to \( B \). Hence the existence of a minimal fault-free path from \( A \) to \( D \) going through \( B \).

**Corollary 1:** The unsafety vectors algorithm selects in case (1) a preferred neighbour positioned on a minimal fault-free path.

**Theorem 2:** Let \( A^{(i)} \) and \( A^{(j)} \) be two non faulty \((A,D)\)-preferred neighbours of \( A \). If all preferred neighbours of \( A^{(j)} \) are faulty and at least one preferred neighbour of \( A^{(i)} \) is non faulty then the Unsafety Vectors algorithm does not route messages of destination \( D \) via \( A^{(j)} \).

**Proof:** Since \( u_i^{A^{(i)},D} < u_i^{A^{(j)},D} \) then \( U^{A^{(i)},D} < U^{A^{(j)},D} \). Therefore, \( U^{A^{(j)},D} \) is not the minimal such vector (for the preferred neighbours).

**Example 3:** Considering a four-dimensional cube with seven faulty nodes, as depicted in Fig. 4 (faulty nodes are represented by dark nodes). Table 2 shows the unsafety set and the safety vector associated with each node. Consider a source \( A=1110 \) and a destination \( D=1001 \). The unsafety vectors algorithm fails to route a message between the pair \((A, D)\) since the third bit \( (h=3) \) of the safety vector of the source node is not one and none of the preferred neighbours has one at the second bit \( (h-1) \) of their safety vectors. Also none of the spare neighbours has one at the fourth bit \( (h+1) \) of their safety vectors. On the other hand, the unsafety vectors algorithm is capable of achieving an optimal route between the source and destination \((A, D)\) in this case. The unsafety vectors algorithm will route the message to the intermediate node 1010 since it has the least number of preferred faulty nodes in its unsafety set, then to node 1000, and finally to the destination node 1001. The following paths were achieved using the unsafety vectors approach, while the safety vectors approach was not capable to route at all (unreachable destinations).

- \((0011 \rightarrow 0001 \rightarrow 1001 \rightarrow 1101)\)
- \((1110 \rightarrow 1010 \rightarrow 1000 \rightarrow 1001 \rightarrow 0001)\)
- \((1010 \rightarrow 1110 \rightarrow 1111 \rightarrow 1101)\)
• \((1111 \rightarrow 1101 \rightarrow 1001 \rightarrow 0001 \rightarrow 0011)\)

![Image of a 4-dimensional cube with seven faulty nodes](image)

Fig. 4: A 4-dimensional cube with seven faulty nodes (represented in dark colour).

<table>
<thead>
<tr>
<th>Node</th>
<th>0000</th>
<th>0001</th>
<th>0010</th>
<th>0011</th>
<th>0100</th>
<th>0101</th>
<th>0110</th>
<th>0111</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S_1^A)</td>
<td>Faulty</td>
<td>{0,5}</td>
<td>Faulty</td>
<td>{2,7,11}</td>
<td>{0,5,6,12}</td>
<td>Faulty</td>
<td>Faulty</td>
<td>Faulty</td>
</tr>
<tr>
<td>S.V</td>
<td>0,0,0,0</td>
<td>1,0,0,0</td>
<td>0,0,0,0</td>
<td>1,0,0,0</td>
<td>1,0,0,0</td>
<td>0,0,0,0</td>
<td>0,0,0,0</td>
<td>0,0,0,0</td>
</tr>
</tbody>
</table>

Table 2. The first level unsafety sets of a 4-dimensional cube with 5 faulty nodes.

<table>
<thead>
<tr>
<th>Node</th>
<th>1000</th>
<th>1001</th>
<th>1010</th>
<th>1011</th>
<th>1100</th>
<th>1101</th>
<th>1110</th>
<th>1111</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S_1^A)</td>
<td>{0,12}</td>
<td>{11}</td>
<td>{2,11}</td>
<td>Faulty</td>
<td>Faulty</td>
<td>{12,15}</td>
<td>{6,12}</td>
<td>{7,11}</td>
</tr>
<tr>
<td>S.V</td>
<td>1,0,0,0</td>
<td>1,1,0,0</td>
<td>1,0,0,0</td>
<td>0,0,0,0</td>
<td>0,0,0,0</td>
<td>1,0,0,0</td>
<td>1,0,0,0</td>
<td>1,0,0,0</td>
</tr>
</tbody>
</table>

5. Performance Comparison

This section conducts a performance comparison between the proposed unsafety vectors approach and a recently proposed algorithm, namely the safety vectors [22]. The reason we have selected the safety vectors algorithm in our study is because besides being the most recent algorithm proposed in the literature, it has been shown to possess superior characteristics than existing similar algorithms [22]. A simulation study of both algorithms has been carried out over a cube of different sizes. However, we report be-
low the results for a 1024 node cube only as the general conclusions have been found not to change much for other network sizes. We have considered in our experiments different random distributions of faulty nodes in the network; we started with a non-faulty cube in the first experiment, and then increased in each subsequent experiment the number of faulty nodes gradually up to 75% of the cube size with random fault distribution. A total of 10,000 source-destination pairs where selected randomly in each simulation run. Before presenting the results, we define the following variables that have been used to compute the performance measures considered in the comparative analysis.

- **Total**: total number of generated messages
- **Routing_Distance**: number of links crossed by a message.
- **Hamming_Distance**: Hamming distance between the message source and destination.
- **FailCount**: number of routing failure cases.
- **LoopingCount**: number of messages which cross a number of links beyond a maximum threshold before being discarded.

We propose the following three performance measures as the basis for comparing the unsafety vectors and safety vectors algorithms.

- **Percentage of unreachability** = \( \frac{\text{Fail Count}}{\text{Total}} \times 100 \)

- **Average deviation from optimality** = \( \frac{1}{100} \sum \left( \frac{\text{Routing Distance} - \text{Hamming Distance}}{\text{Hamming Distance}} \right) \)

- **Percentage of looping** = \( \frac{\text{Looping Count}}{\text{Total}} \times 100 \)

The percentage of unreachability measures the percentage of messages that the algorithm fails to deliver to destination due to faulty components. The average deviation from optimality indicates how close the achieved routing is to the minimal distance routing. The percentage of loops indicates the fraction of messages that fail to reach destinations due to network partitioning. We believe that these measures can serve the purpose of our present study as they give realistic indications on the performance of the fault-tolerance routing algorithms.

Figs. 5, 6, and 7 show results from a performance comparison between the unsafety vectors and safety vectors algorithms with respect to the above measures. In all the reported results, the parameter \( m \)
has been set to its lowest value ($m=1$) in the proposed algorithm. As expected, our simulation experiments have confirmed that larger values of $m$ greatly improve performance, but, of course, at the expenses of increased communication overhead. Fig. 5 reveals that even with the modest value of $m=1$ the unsafety vectors algorithm achieves much higher reachability than the safety vectors algorithm with low to moderate deviation from optimality, as depicted by Fig. 6. That figure (i.e., Fig. 6) also shows that the deviation from optimality becomes noticeable for the unsafety vectors algorithm only when the percentage of faulty nodes exceeds 50% of the total number of nodes in the network. Fig. 7 reveal that message looping in the unsafety vectors remains very low (practically zero) when the percentage of faulty nodes is less than 30%. From the results of the three figures we can conclude that the new algorithm exhibits superior performance characteristics over the safety vectors algorithm under realistic network working conditions.

![Fig. 5: Percentage of Unreachability of Unsafety Vectors and Safety Vectors.](image)
The unsafety vectors algorithm is more capable to route messages using optimal distance paths especially for a large number of faulty components. Under high fault rates our algorithm is capable to route a large percentage of messages for which the safety vectors algorithm announces a routing failure. This is due to the fact that the unsafety vectors algorithm repeatedly chooses to route through areas of the cube with the least number of faults in the neighbourhood applying a greedy approach giving more weight to the nearest neighbourhood. The safety vectors algorithm, on the other hand, routes via a neighbour only if that neighbour guarantees optimal routing to all destinations at the desired distance not just to the desired destination. Furthermore, owing to its inherent properties the unsafety vectors algorithm tends to select paths that diverge from areas with high counts of faulty components.
6. Conclusions

This paper has proposed a new fault-tolerant routing based on the concept of unsafety vectors for the binary $n$-cube. As a first step in this algorithm, each node $A$ determines its view of the faulty set $F_A$ of nodes which are either faulty or unreachable from $A$. This is achieved by performing $(n-1)$ exchanges with the reachable neighbours. Node $A$ then calculates $m$ unsafety sets denoted $S^A_1, S^A_2, \ldots, S^A_k$, where $m$ is an adjustable parameter between 1 and $n$. The $k$-level unsafety set represents the set of all nodes at Hamming distance $k$ from $A$ which are faulty or unreachable from $A$ due to faulty links or nodes. Nodes use these unsafety sets to compute unsafety vectors and use them to achieve a fault-tolerant routing algorithm in the cube. The larger the value of $m$ is the better the routing decisions are, but at the expense of more communication overhead. A comparison between the proposed and safety vectors algorithm have been presented. Results have revealed that the new algorithm outperforms the safety vectors algorithm in terms of the routing distance and percentage of reachability even when the parameter $m$ is at its lowest value of 1, corresponding to minimum communication overhead.

References


