The CSI Journal on
Computer Science and Engineering
Vol. 1, No. 4 (b), Winter 2003
Pages 11-18
Regular Paper

A Routing Algorithm for the Star Interconnection Network in the Presence of Faults*

Mostafa Rezazad Hamid Sarbazi-Azad

IPM School of Computer Science and Sharif University of Technology
Tehran, Iran

Abstract

This paper presents a fault tolerant routing algorithm for the star graph. The suggested algorithm is based on the concept of *unsafety vectors,* originally proposed for hypercubes. In this algorithm, each node starts by computing a first level unsafety set, composed of the set of unreachable neighbors. It then performs some exchanges with its neighbors to determine the unsafety nodes. After that, all nodes have the addresses of all faulty nodes. Based on the information gathered in each node, fault-tolerant routing between a source node and a destination node is realized. We conducted a performance comparison between the star and hypercube graph using the unsafety vectors routing algorithm under different working conditions. The results obtained through simulation experiments reveal that the hypercube is of superior performance compared to the star graph in the presence of low fault rates. But its performance under high fault rates falls short of that of the star graph.

Keywords: multicomputers, interconnection networks, stars, hypercube, fault-tolerant routing, performance comparison

1. Introduction

The topological properties of the star graph are very similar to those of the hypercube, e.g. regularity and hierarchical structure. The star graph, however, exhibits superior topological properties in several aspects, e.g. lower diameter and node degree. A routing algorithm specifies the way in which a message selects a path to cross from source to destination, and has great impact on network performance. As network size increases, so does the probability of processor and link failure. 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). Fault-tolerant routing in stars has been studied in the past [13, 19].

Several fault-tolerant routing algorithms for the star graph have been suggested in the literature [2, 5, 6, 7, 8, 10, 11, 12, 14, 15, 16, 17, 18, 20, 21, 24]. Based on the accessibility of information about system faults in the network, fault-tolerant routing can be classified as *global information based* (GIB), *local information based* (LIB), and *limited global information based* (LGIB) [22]. In GIB routing algorithms, information that determines which nodes or links of the network are faulty is accessible from every node of the network, and can thus be utilized to find the best possible path to the destination at the source node. In this method, we can apply backtracking-based methods, such as *depth-first search*, to find a possible path between any two nodes. In LGIB algorithms, fault information is gathered and processed from neighboring nodes so that a guide of the fault-distribution in the system can be obtained. This guide is helpful to the routing, and then the approximate length of a path (not necessarily the best one) can be found. Information gathering steps, in this method, is not as heavy as that in the

GIB method. Thus, there is not much traffic overhead for information exchange between network nodes.

A LIB routing algorithm yields sub-optimal routes due to the insufficient information upon which the routing decisions are based. A GIB routing algorithm can achieve optimal or near optimal routing, but often at the expense of higher communication overhead to maintain up-to-date network-wide fault information. The main challenge is, therefore, to devise a simple and efficient method of representing limited-global fault information that allows optimal or near optimal fault-tolerant routing. The LGIB method is a trade-off solution between the GIB and LIB methods, making a compromise between traffic overhead and routing optimality. The unsafety vectors algorithm [1] is one of the routing algorithms based on the utilization of limited-global fault information.

Two kinds of GIB routing algorithms have been introduced in the literature: safety level [23] and safety vector [22] algorithms. The safety level has also been proposed for the star graph [9]. However, algorithms based on the safety level idea do not perform well. The safety vectors approach (initially introduced in [23] for hypercubes but later applied to the star graph [25]) requires each processor to maintain a bit vectors (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* in the hypercube is set. A major drawback of this algorithm is related to the degree of conservatism of the 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*.

However, the unsafety vector routing algorithm can handle this problem and, moreover, is not a GIB algorithm. This algorithm is capable of routing a large percentage of messages for which the safety vectors algorithm [22] fails to provide a path. It is in fact a form of the LGIB method with a parameter to define information exchange degree. Setting this parameter to its maximum amount, will result in a GIB algorithm.

In this paper, we use the idea of unsafety vectors and introduce a fault-tolerant routing scheme for the star graph. The main goal of introducing this algorithm for the star graph is to provide a new means to compare these two well-known networks. The results obtained through simulation experiments reveal that the hypercube is of better performance than the star graph in the presence of faults.

The rest of the paper is organized as follows. Section 2 gives some preliminaries and notation used in the next sections. In section 3, the proposed routing algorithm is described. The performance of the proposed algorithm is discussed in Section 4. Section 5 reports the results from comparing the performance of star graphs and hypercubes in the presence of faults. Finally, in Section 5, we conclude this work.

2. Preliminaries and Notation

Let V_n be the set of all $n!$ permutations of symbols 1, 2, 3, …, *n*. For any permutation $v \in V_n$, if we denote the *i*th symbol of *v* by *v*(*i*), then *v* can be written as *v*(1)*v*(2)…*v*(*n*).

A *star graph* on *n* symbols, $S_n = (V_n, E_n)$ is a undirected graph with *n*! nodes where each node *v* is connected to *n*–1 nodes which can be obtained by interchanging the first and *i*th symbols of *v*, i.e. $[v(1)v(2)...v(i)v(i+1)...v(n), v(i)v(2)...v(i-1)v(1)v(i+1)...v(n)] \in E_n$ for $2 \le i \le n$. We call these $n-1$ connections dimensions. Thus each node is connected to n – 1 nodes through dimensions 2, 3, … ,n. S_n is also called an n-star or an n-dimensional star. Fig. 1 shows S_4.

The star graph is an attractive alternative to the hypercube, and compares favorably with it in several aspects [3, 4]. For example, the degree of S_n is $n-1$, i.e. sub-logarithmic in the number of nodes of S_n while a hypercube with $\Theta(n!)$ nodes has degree $Q(\log n!) = Q(n \log n)$, i.e. logarithmic in the number of nodes. The same can be said about the diameter of S_n. Much work has been done to study both the topological properties and parallel algorithms of the star graph lately.

In [4], a greedy algorithm is given which finds a shortest path from any node p to *e* (we will call that $H(p, e)$).The algorithm is given by the following two rules: 1) If $p(1) = 1$, move to any position not occupied by the correct symbol, 2) If $p(1) = x \neq 1$, move to its correct position.

In the star graph, finding the preferred neighbor (the neighbor of node *A* in shortest path from node *A* to node *D*) is not easy as that in the hypercube. In the star graph, we must first determine the distance, *l*, between a source and a destination and then find the distance between neighbors of the source and the destination (l'). If $l = l' + 1$ then this node is the preferred neighbor for node *A* in the path from *A* to *D*. Otherwise, it is a spare neighbor (neighbor other than preferred) for node *A*.

Routing through a spare neighbor increases the routing distance. An optimal path can be obtained by routing through all preferred dimensions in some order.

3. The Proposed Fault-tolerant Routing Algorithm

The proposed fault-tolerant routing algorithm is based on the concept of unsafety sets (defined below). Before presenting the algorithm let us first discuss how a node calculates its unsafety sets. We make the following assumptions for the proposed algorithm:

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

3.1 Calculating faulty and unsafety sets

Definition 1. The *first level unsafety set* S_1^A of a node *A* is defined as

$$S_1^A = \bigcup_{1 \le i \le n} f_A^i \quad \textit{Where } f_A^i \textit{ is given by}$$

$$f_A^i = \begin{cases} \{A^{(i)}\}, & \textit{if } A^{(i)} \textit{ is faulty} \\ f & \textit{Otherwise} \end{cases}$$

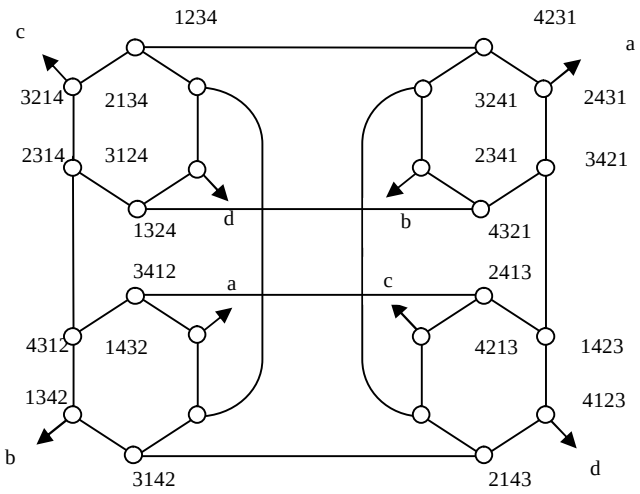

Figure 1. A 4-Star

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 ($\frac{3(n-1)}{2} - 1$) exchanges with reachable neighbors. When F_A is built, node *A* calculates *m* unsafety sets denoted as 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 , $1 \le k \le m$, 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 shortest 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 of unreachable nodes may not be accurate. In this case message looping will occur. We present in the following a method for detecting and handling such looping. Fig. 2 gives an outline of the *find_unsafety_sets* algorithm where node *A* uses it to determine its faulty and unsafety sets.

Example 1. Consider a 4-star with 10 faulty nodes (faulty nodes are represented as black nodes), as shown in Fig. 3. Table 1 shows the corresponding first-level unsafety set, S_1^A, associated with each node *A*. The *find_unsafety_sets* algorithm calculates the sets S_k^A for all $1 \le k \le m$ after calculating F_A. To achieve this, (*m*-1) exchanges of fault information are performed among neighboring nodes. Let $m = 3(n-1)/2$ and, for the sake of specific illustration, let us compute the unsafety sets associated with node $A = 1234$. First, the node assigns the addresses of its immediate faulty neighbors to its faulty set F_A. Then each node performs $m - 1$ exchanges of the new elements of its faulty set F_A, with the immediate non-faulty neighbors. After determining F_A, node *A* calculates *m* unsafety sets denoted S_1^A , S_2^A , … , S_m^A according to the shortest distance between node *A* and each element of F_A. So, the faulty set for node *A* in our example is F_A={3214, 1324, 2431, 2341, 4321, 2413, 2143, 1432, 3142}, and the unsafety sets are S_1^A ={3214}, S_2^A ={2431}, S_3^A ={1432, 2341, 1324, 3142, 2413}, and S_4^A ={4321, 2143}.

3.2 The unsafety vectors routing algorithm

Definition 5. For a given source-destination pair of nodes (*A,D*), we define the (*A,D*)-*unsafety vector*, $U^{A,D} = (u_1^{A,D},...,u_k^{A,D},..u_m^{A,D})$, where its k^{th} element is given by $u_k^{A,D} = |\{T \in S_k^A$, such that *T* is an (*A,D*)-preferred transit node}|.

In other words, $u_k^{A,D}$ is the number of faulty or unreacable (*A,D*)-preferred transit nodes at distance *k* from *A*. Determining the preferred neighbors at a specific distance from an arbitrary node in the star graph is not as trivial as it is in the hypercube. We must therefore find all of the minimum paths from *A* to *D*, and then reach the preferred neighbors at distance *k* by passing *k* hops through the paths. Fig. 4 represents an algorithm to find all of the minimum paths for any arbitrary node pair (*A, D*). $u_k^{A,D}$ 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 $\exists i$, $1 \le i \le m$, such that $u_i < v_i$, and $u_j = v_j$ for all $j < i$.

Fig. 6 shows the *unsafety_vectors* algorithm that the current node *A* in the network applies in order to route a message towards its destination node *D*. At the start out, the current node is the source node. Thus, the the number of hops the message has taken, shown by *M.Route_distance*, is initialized to zero. Then the node checks if the message has been trapped in a loop. This is checked by comparing the number of hops the message has taken so far with the minimum source-destination distance (function *H* returns the minimum distance between two nodes *A* and *D* in the star graph) plus a threshold $\lceil 3(n-1)/2 \rceil$. This threshold shows the maximum number of hops a message may be misrouted from its destination. This number is equal to the diameter of the network. The reason behind choosing such a threshold is that we can consider a message trapped in looping if it misroutes a number of times greater than the network diameter. If the message is not in a looping, the number of hops the message has taken is increased by one to account for the next hop it is taking. If the current node is the destination node, the message is delivered to the current node. Otherwise, it is sent to the preferred neighbor with the minimum unsafety vector which is not a dead-end node. If such a node does not exit, the message is sent to a spare neighbor with the minimum unsafety vector which is not dead-end. If such a node does not exits once again, a failure to send the message towards its destination is reported and the message will not be able to reach its destination.

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 an specified limit since looping occurs when a destination is not reachable from the source node. We may add the destination node to the faulty set of the node that has detected the looping. As explained above, the threshold used in our simulator was set to $\lceil 3(n-1)/2 \rceil$, the smallest integer greater than the network diameter. Our experiments show that further increasing this threshold, will not result in a noticeable change in routing performance. Thus, we have used this threshold in all simulation experiments.

In [1], according to Theorem 1 it was guaranteed that if there is a minimal path between the current node and the destination node with no faults, this path would be taken. If such a safe path towards the destination does not exist, the algorithm can use a neighbor with the smallest unsafety vector to make the next hop in the route. Unfortunately, this property (given by Theorem 1 in [1]) does not exist in the star graph. The routing algorithm, given in Fig. 6, for the star graph can only choose the neighbor with the smallest unsafety vector, as there is mathematically no a guarantee to find a path with no faults in the star.

To improve the performance of the algorithm and to make it more comparable to the algorithm given in [1] for the hypercube, we have chosen the path to be taken by the message in the current node more carefully. In order to select the best shortest path from the current node towards the destination node, we divide the *i*–th element of the unsafety vector of each neighboring node (of the current node) by the total number of nodes at distance *i* from that node (on all available shortest paths to the destination node). For example, assume that node *A* has 2 preferred neighbors *B* and *C* to destination node *D*. Assume that node *B* has 3 preferred neighbors to destination node *D* and 2 of them are faulty, i.e. $u_1^{B,D} = 2$. As node *B* has 3 paths to node *D*, we set $u_1^{B,D}$ to 2/3. In addition, we assume that node *C* has one preferred faulty neighbor, i.e. $u_1^{C,D} = 1$. As node *C* has a single path to *D*, $u_1^{C,D}$ remains equal to one. In this manner, node *A* may choose node *B* for its next movement as it has a lower unsafety vector than node *C*. Figure 5 illustrates this scenario.

```
Algorithm Find_Unsafety_Sets (A: node)  /* called by node A to
determine its faulty set F_A */
  S_1^A = set of faulty immediate neighbors;
  F_A = S_1^A ;
 for k:=2 to n do
 {
       for i:=1 to n do
       if A^(i) ∉ F_A then
         {        send F_A to A^(i);
                  receive F_A^(i) from A^(i);
                  F_A = F_A ∪ F_A^(i);
         }
 }
for k:= 1 to m do S_k^A = {B ∈ F_A | H(A,B) = k}
End.
```

Figure 2. A description of the Find_Unsafety_Sets algorithm

Table 1. The unsafety sets of nodes in a 4-star with 10 faulty nodes

Node(A)	S_1^A	Node	S_1^A
1234	{3214}	3412	{14312,4312,2413}
2134	{}	1432	Faulty
3124	{1324}	4132	{1432,3142}
1324	Faulty	3142	Faulty
2314	{1324,3214,4312}	1342	{3142,4312,2341}
3214	Faulty	4312	Faulty
4231	{2431}	2413	Faulty
2431	Faulty	1423	{2413}
3421	{2431,4321}	4123	{2143}
4321	Faulty	2143	Faulty
2341	Faulty	1243	{2143}
3241	{2341	4213	{2413,3214}

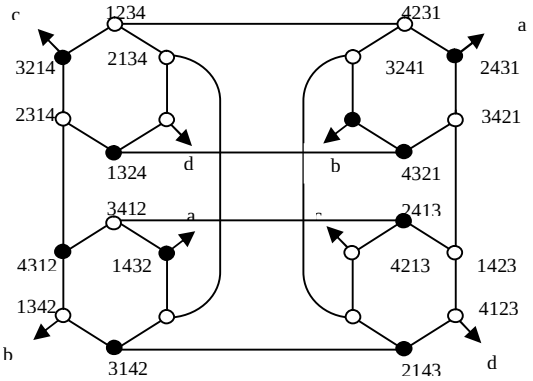

Figure 3. A 4-star garph with 10 faulty nodes (represented in dark colour)

```
Algorithm Find_All_of_the_ Minimum_Paths (A,D : node)
/*  Calld by node A to find the all of the minimum paths between A
and D  */
  Root, Flag = A;
 While (Flag != D)
 {
        for i:=1 to n-1 do
        {   if ( ith neighbor of A is preferred neighbor ) then
              Flag.Child(i) = neighbor( i, A);
        }
        Flag = Flag .Child(i);
 }
End.
```

Figure 4. Description of the find all of the minimum paths (*A* to *D)*

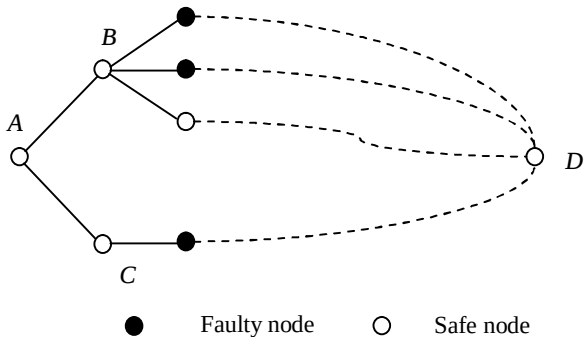

Figure 5. An example scenario

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) + $\lceil 3(n-1)/2 \rceil$ *then*
{ *M.Route_distance:=M.Route_distance+1;*
If A = D then exit; / destination reached */*
Let $A^{(i)}$ *be the reachable preferred neighbor with least* $(A^{(i)}, D)$*-unsafety vector* $U^{A(i),D}$ *and* $A^{(i)}$ *is not dead-end;*
if $A^{(i)}$ *exist then send M to* $A^{(i)}$
else { Let $A^{(j)}$ *be the reachable spare neighbor with least* $(A^{(j)}, D)$*-unsafety vector* $U^{A(j),D}$ *and* $A^{(j)}$ *is not dead-end;*
if $A^{(j)}$ *exist then send M to* $A^{(j)}$
*else report failure; /*destination unreachable */*
}
}
else Handl looping
End.

Figure 6. A description of the proposed unsafety vectors routing algorithm

4. Simulation Results

In this section, we report the results of applying the unsafety vectors algorithm in the star graph. We report below the results only for a 720-node star (6-star), as the general conclusions were found not to change for other network sizes we simulated. We considered, in our experiments, a uniform random distribution for the faulty nodes in the network. We started with a non-faulty star network in the first experiment and then increased, in each subsequent experiment, the number of faulty nodes gradually up to 83% of the network size. In each step, we selected each non-faulty node as a source node and all other non-faulty nodes as the destination nodes. For example, in a 6-star a total of 620*619 source-destination pairs were selected for a non-faulty network. For the *n*-star network and with f percent faults, $0 \le f \le 1$, the number of different source-destination pairs will be $n!(1-f)[n!(1-f)-1]$.

Before presenting the results, we define the following variables that have been used to compute the performance measure.

- *Total*: total number of generated messages.
- *Routing Distance*: number of links crossed by a message.
- *Distance*: distance between a source node and a destination node (using a minimal path).
- *Fail Count*: number of routing failure cases.
- *Lopping Count*: number of messages that cross a number of links beyond a maximum threshold before being discarded.
- *Total Reach*: total number of generated messages that reached their destinations.

The following four performance measures are calculated and reported.

- Percentage of unreachability as (*Fail Count*/*Total*)*100.
- Average deviation from optimality as (∑(*Routing Distance* – *Distance*)/*Minimum Distance*)/*Total*)*100.
- Percentage of looping as (*Loop Count* / *Total*)*100.
- Average Diameter as (∑(*Routing Distance*))/*Total Reach*.

In all reported results, the parameter *m* has been set to its highest value in the proposed algorithm, i.e. *m* = (3(*n*-1)/2. The percentage of unreachability measures the percentage of messages that the algorithm fails to deliver to destination due to faulty components. Fig. 7 shows that when the number of faulty nodes is less than about 50% the reachability is still good. However, when it increases beyond 50% the unreachability sharply increases, as the network may be split in to several disjoint regions.

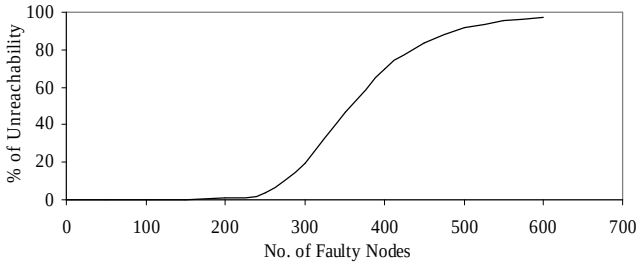

Figure 7. Percentage of unreachability of unsafety vectors

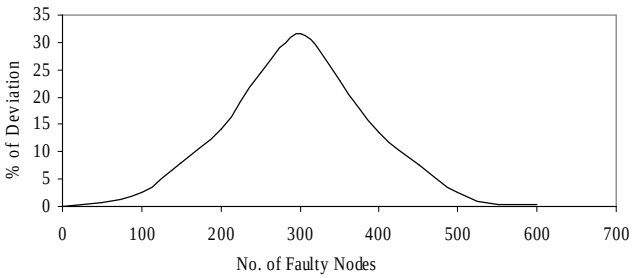

Figure 8. Percentage of deviation of unsafety vectors

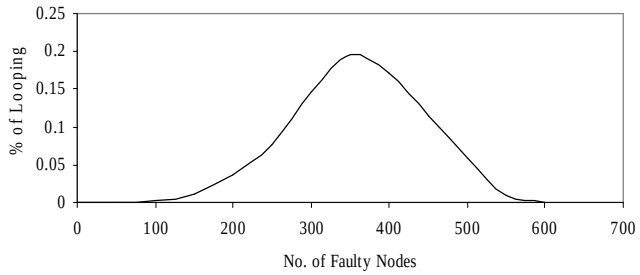

Figure 9. Percentage of looping of unsafety vectors

The average deviation from optimality indicates how close the routing behaves to a minimal distance routing. As shown in Fig. 8, with an increase in the number of faulty nodes, the deviation grows accordingly. However, after about 50% fault rate, as the reachability rate decreases so does the average and average deviation of the number of messages that may reach their destination.

The percentage of loops indicates the ratio of messages that fail to reach their destinations due to the partitioning of the

network. As shown in Fig. 9, looping and deviation grows when the number of faulty nodes increases until unreachability reaches to the range of 50%. After that, the number of loops reduces as the percentage of unreachability increases.

The average diameter indicates the average distance between any pair of source and destination nodes in the presence of faulty nodes. As shown in Fig. 10, average increases (due to faulty nodes) until unreachability reaches to about 50%. After that, the average diameter reduces since each node can only send messages to its near neighbors (because most of the destination nodes reside in unreachable partitions).

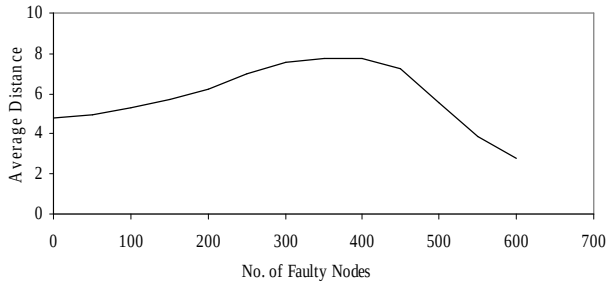

Figure10. Average Distance of unsafety vectors.

5. Performance Comparison: Star vs. Hypercube

In this section, we conduct a comparison between the star and hypercube networks in the presence of faults. The measure used in our comparison is the average message latency. This measure is obtained by the simulator, in which we have calculated the transmission time of a message over network channels in its route. Assuming a unit channel cycle time for both graphs, the average message latency will simply change into the average routing distance in these networks.

Fig. 11 illustrates the results obtained by simulation experiments for the two networks, the star and hypercube with the same size, for different fault rates f = 5% and 80%. In both graphs, the horizontal axis denotes the network size and the vertical axis shows the calculated average message latency.

Simulation results reveal, as can be seen in these graphs, that in the absence of faults (or when the fault rate is small), the hypercube is of slightly superior performance compared to an equivalent star graph. Note that the performance of the hypercube increases when fault rate increases. Since the node degree of the hypercube is greater than that of an equivalent star graph, the number of channels in the hypercube is much greater than the star graph. However, when a large number of nodes or links become faulty (e.g. 80% of nodes), messages take more time to reach to their destinations in the hypercube. This is because the hypercube has more channels than the star graph and can resist being separated due to large fault rates. Thus, the total number of messages that can reach their destinations in the hypercube is much greater than that in an equivalent star graph, and each message travels a longer route in the network, resulting in a larger average message communication-time. This condition does not occur in lower fault rates and the hypercube thus performs better than the star graph.

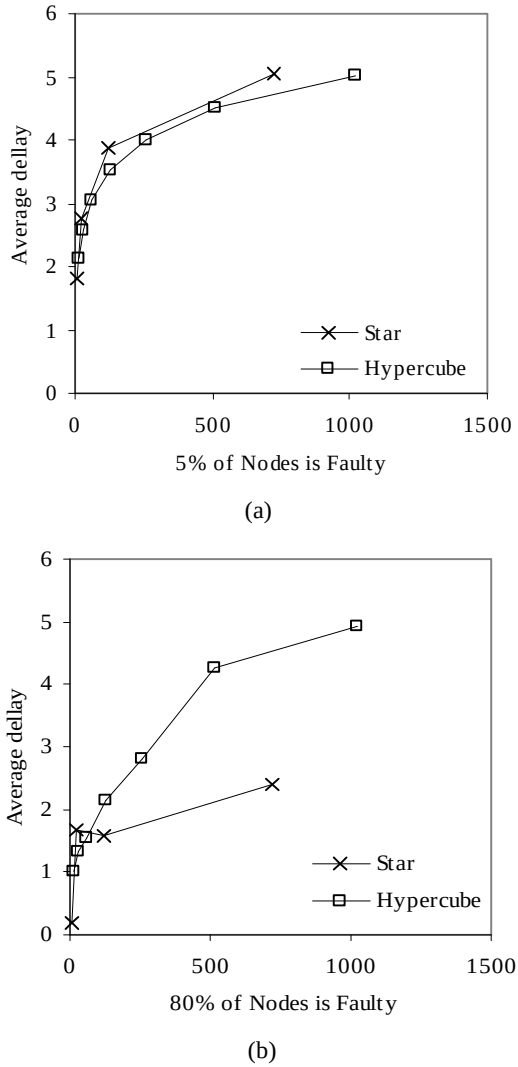

Figure11. The average message latency in the star graph and hypercube for different network sizes when **(a)** f = 5% of nodes, **(b)** f = 80% of nodes

6. Conclusions

The star has been proposed as an attractive alternative to the well-known hypercube for its desirable properties. Despite the many works done on the topological properties and application algorithms in the stars, there are few works reported on the design of fault-tolerant routing in such networks. This paper has proposed a fault-tolearnt routing algorithm in the star graph. It is based on the concept of unsafety vectors proposed in [1] for binary cubes. In the first step of this algorithm, each node determines its view of the faulty nodes of the network which are either faulty or unreachable. Then, nodes use these unsafety sets to compute unsafety vectors and apply it to achieve a fault-tolerant routing algorithm in the star graph.

We compared the overall performance of the star graph with its equivalent hypercube using the proposed routing algorithm. The simulation results revealed that the performance of the hypercube is greater than the star graph for low fault rates. However, when fault rate increases the star graph can perform better.

A more detailed comparison, taking into account technological constraints (such as constant node-size and

constant bisection width, more suitable when building very large systems using cabled clusters or VLSI implementation), can be seen as interesting issues to focus on in future work in this line.

References

[1] J. Al-Sadi, K. Day, and M. Ould-Khaoua, "Unsafety Vectors: a New Fault-Tolerant Routing for the Binary *n*-cube," *Journal of Systems Archtecture* 47, pp. 783-793, 2002.

[2] S. B. Akers and B. Krishnamurthy, "The Fault-Tolerance of Star Graphs," *Proc. 2th Int. Conf. Supercomputing*, pp. 270-276, 1987.

[3] S. B. Akers and B. Krishnamurthy, "A Group Theoretic Model for Symmetric Interconnection Networks," *IEEE TC*, vol. c-38, no. 4, pp. 555-566, 1989.

[4] S. B. Akers, B. Krishnamurthy and D. Harel, "The Star Graph: An Attractive Alternative to the n-cube," *Proc. Int. Conf. on Parallel Processing*, pp. 393-400, 1987.

[5] S. B. Akers and B. Krishnamurthy, "On Group Graphs and their Fault-tolerance," *IEEE Transactions on Computers*, vol. 36, no. 7, pp. 885-888, 1987.

[6] N. Bagherzadeh and M. Nassif, "Computation in Faulty Stars [Hypercube Networks] ," *IEEE Transactions on Reliability*, vol. 44, no. 1, pp. 114-119, 1995.

[7] N. Bagherzadeh, M. Nassif and S. Latifi,"A Routing and Broadcasting Scheme on Faulty Star Graphs," *IEEE Transactions on Computers*, vol. 42, no. 11, pp. 1398-1403, 1993.

[8] L. Q. Bai, H. Ebara and H. Nakano, "An Efficient Adaptive Routing Algorithm for the Faulty Star Graph,"*Proceeding of the International Conference on Parallel and Distributed Systems*, pp. 82-87, 1997.

[9] C. H. Chang, "A Study of Fault-tolerant Routing on Star Graph using Limited-global-information," Master's thesis, National Taiwan University of Science and Technology, 1997.

[10] C. C. Chen and J. Chen, "Vertex-disjoint Routing is Star Graphs," *IEEE First International Conference on Algorithms and Architectures for Parallel Processing*, vol. 1, pp. 460-464, 1995.

[11] Y. S. Chen and J. P. Sheu, "A Fault-Tolerant Reconfiguration Scheme in the Star Graph," *Journal of Information Science and Engineering*, 16(25-40), pp. 221-231,1999.

[12] K. Day and A. AL-Ayyoub, "Adaptive Fault-Tolerant Routing in Star Networks," *Journal of Interconnection Networks*, vol. 2, no. 2, pp. 213-231, 2001.

[13] K. Day and A. Tripathi, "A Comparative Study of Topological Properties of Hypercube and Star Graph," *IEEE TPDS*, vol. 5, no. 1, 1994.

[14] L. Gargano, U. Vaccaro and A. Vozella, "Fault-tolerant Routing in the Star and Pancake Interconnection Networks," *Information Processing Letters*, vol. 45, no. 6, pp. 315-320, 1993.

[15] Q. P. Gu and S. Peng, "The Shortest Routing Path in Star Graphs with Faulty Clusters," *Proceeding of Second Aizu International Symposium Parallel Algorithms/Architecture Synthesis*, pp. 91-96, 1997.

[16] S. C. Hu and C. B. Yang, "Fault Tolerance on Star Graphs," *International Journal of Foundations of Computer Science*, vol. 8, no. 2, pp. 127-142, 1997.

[17] C. Liang, S. Bhattacharya and J. Tan, "Performance Evaluation of Fault-tolerant Routing on Star Networks," *Proceeding of the Scalable High-Performance Computing Confrence*, pp. 650-657, 1994.

[18] C. Liang, S. Bhattacharya and J. Tan, "Design and Analysis of Fault-tolerant Star Networks," *Proceeding of the International Confrence on Parallel Processing*, pp. 346-349, 1997.

[19] K. Qiu and S. G. Akl," On Some Properties of the Star Graph," *VLSI design*, vol. 2, no. 4, pp. 389-396, 1995.

[20] C. P. Ravikumar and A. M. Goel, "Deadlock-free Wormhole Routing Algorithms for Star Graph Topology,"*IEEE Proceeding Computers and Digital Techniques*, pp.195-400, 1995.

[21] S. Sur and P. K. Srimani, "A Fault-Tolerant Routing Algorithm in Star Graph Interconnection Networks," *Prod. Int. Conf. Parallel Processing*, vol. 3, pp. 267-270, 1991.

[22] J. Wu, "Adaptive Fault-tolerant Routing in Cube-base Multicomputers using Safety Vectors, "*IEEE TPDS*, vol. 9, no. 4, pp. 321-334, 1998.

[23] J. Wu, "Unicasting in Faulty Hypercube using Safety Levels," *IEEE Transactions on Computers*, vol. 46, no. 2, pp. 241-244, 1997.

[24] C. B. Yang and T. H. Liu, "Wormhole Routing on the Star Graph Interconnection Network," *Proceedings of the 4th Australian Theory Symposium, CATS'98*, pp. 51-65, 1998.

[25] S. Yeh, C. Yang and H. Chen, "Fault-Tolerant Routing on the Star Graph with Safety Vectors," *Proceeding of the international Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'02)*, pp. 301, 2002.

[26] M. Rezazad and H. Sarbazi-Azad, "A Constraint Based Performance Comparison of Hypercube and Star Multicomputer Networks with Failures," Technical Report, IPM School of Computer Science, 2004.

Mostafa Rezazad received his BSc in Computer Engineering from Azad University, Central Tehran, Iran, in 2001, and his MSc degree in Computer Engineering from Sharif University of Technology, Tehran, Iran, in 2004. He is currently a Research Assistant at the IPM School of Computer Science. His research interests include interconnection networks and performance evaluation of parallel/distributed systems.
Email: rezazad@ipm.ir

Hamid Sarbazi-Azad received his BSc degree in Electrical and Computer Engineering from Shahid-Beheshti University, Tehran, Iran, in 1992, his MSc degree in Computer Engineering from Sharif Univ. of Technology, Tehran, Iran, in 1994, and his PhD degree in Computing Science from the University of Glasgow, Glasgow, UK, in 2002. He is currently a faculty member of the Department of Computer Engineering at Sharif University of Technology, and heads the School of Computer Science, in the Institute for Studies in Theoretical Physics and Mathematics (IPM), Tehran, Iran. His research interests include high-performance parallel computer architectures, parallel and distributed systems, performance modeling/evaluation, image processing, computational aspects of atomic physics, graph theory and combinatorics. Dr Sarbazi-Azad has served as a guest co-editor for the special issue "Performance modeling and evaluation of high-performance parallel and distributed systems" in *Performance Evaluation* journal, the special issue "Design and performance of networks for super-, cluster-, and grid-computing" in the *Journal of Parallel and Distributed Computing*, and the special issue "Performance evaluation of networks in parallel, cluster, and grid computing systems" in *Parallel Computing* journal.
Email: azad@sharif.edu

* This research was in part supported by a grant (No. CS1382-1-01) from the IPM School of Mathematics, Iran.