Hypercube Message Routing in the Presence of Faults

Jesse M. Gordon* and Quentin F. Stout†

Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122 USA

Abstract

We discuss the problem of routing messages on hypercubes which have faulty processors and/or communication links. We are motivated by the belief that simple algorithms, operating under simple assumptions, can ensure high probabilities of successful message routing. In this paper, we consider the basic problem of routing a single message from an arbitrary source to an arbitrary destination. In our study, a fault is assumed to render the processor or link non-functional for purposes of communicating messages. As such, we may also consider communications hot spots as node faults, and our results also apply to routing in congested hypercubes.

A framework for the analysis of fault tolerant routing schemes on a hypercube is presented. This framework includes differing routing schemes, routing information models and fault distribution models. The a priori probabilities of successful routing of a single, indivisible message under each of our possible sets of assumptions are calculated. Using random routing, under the one-step local information routing model, we show that the a priori probability of successful message routing is high even for an exceedingly large number of faults. We also analyze the behavior of *sidetracking*, a routing method which combines the concepts of *local information* and *randomization*. Using sidetracking, and in the one-step local information routing model, a messsage will be routed forward using random routing. If the message reaches a blocked processor (no non-faulty neighbors along a minimal path to the destination) it will be sent to a non-faulty neighbor, chosen uniformly at random from the set of non-faulty neighbors. We use simulation experiments to determine the performance of this routing scheme, analyzing the probability of successful routing and the expected path length of a routed message. The empirical performance of the sidetracking algorithms indicates strongly that, in the limit as the cube dimension grows larger and for a fixed probability of node failure, the probability of successful message routing is 100%.

*Supported in part by NSF Grant No. DCR-85077851 and a Fellowship from the Unisys Corporation.
†Supported in part by NSF Grant No. DCR-85077851 and a DEC Incentives in Excellence Award.

1 Introduction and Definitions

Hypercubes have recently emerged as a viable and practical parallel computer architecture ([4], [7], [8], [9]). A fundamental problem in implementing a hypercube machine is that of interprocessor communications. We discuss the problem of finding high performance fault tolerant routing schemes on hypercubes. We present a notion of faults which is strong enough to model permanent processor failures as well as processors which become communications hot spots. This is in keeping with the idea of *local information*, that is, the idea that effective routing algorithms can be developed which require only knowledge of the status of a processor's immediate neighbors. We show that the combination of local information with *randomization* is powerful enough to provide us with the desired routing algorithm. In contrast to the idea of local information is the idea of *global information*, where routing would require knowledge of the status of the entire hypercube. Global information is the less attractive alternative for two reasons. First, the space overhead per processor is far greater for global information than for local information. Second, it would be considerably more difficult and time intensive to maintain the global state information at each processor. In fact, the problem of maintaining global state information locally is as difficult as the general problem considered in this paper.

The remainder of this section presents a framework for our subsequent analysis of some proposed simple fault tolerant routing schemes. The final section introduces the routing method which we call *sidetracking* and contains the results which demonstrate its extremely high performance.

1.1 Routing

The general problem of routing is that of routing messages from sources to destinations. A *partial permutation* is a routing problem where there is at most one packet initially at a node, each with distinct destination addresses. We consider routing schemes which are *oblivious*, or non-adaptive, i.e. where the path taken by a message depends only upon its source (s) and destination (d) nodes. We discuss two basic oblivious routing schemes: deterministic routing and random routing. In **deterministic** routing, for each (s,d) pair, there is a unique path which any message with s as source and d as destination must take. In **random** routing, for each (s,d) pair, there is a fixed probability distribution (independent of the rest of the permutation) that specifies for each path from s to d the probability that the path will be taken.

On an n-dimensional hypercube or n-cube (which contains $N = 2^n$ nodes), standard deterministic routing is accomplished by correcting differing bits (between the addresses of s and d) in some fixed order, such as highest dimension bit first. By correcting a differing, or wrong, bit, we mean routing the message one step along the edge whose dimension is the same as the differing bit position. This is the same as routing the message in the *forward direction*, as we are reducing the remaining routing distance by one step. In contrast, routing in the *backward direction* moves a message one routing step farther away from its destination. A *minimal length path* is a path on which the message is routed only in the forward direction. Standard deterministic routing is the simplest approach to routing, and one which is used in practice on existing hypercube machines. The problem is that bad bottlenecking may occur; in fact, for a full permutation the standard deterministic routing algorithm has worst case time $\Theta(N^{1/2})$ ([10]). In fact, Borodin and Hopcroft [2] show that in any N node network with in-degree d, the time required in the worst case by *any* oblivious routing strategy is $\Omega(\sqrt{N}/d^{3/2})$. It is an open question whether there is any routing algorithm on an n-cube that is $o(\log^2 N)$.

Our version of random routing on a hypercube is based upon Phase B of Valiant's probabilistic routing scheme (see [10], and also [11], [12]). With high probability, Valiant's method requires only

$O(\log N)$ time for a full permutation. We assign a distribution so that all minimal length paths are equally likely; this translates into correcting differing bits in a uniformly random order.

In this paper, we consider the case of routing a *single* (short and indivisible) *message*. Further, we shall initially restrict our attention to *minimal path routes.* Since we are dealing primarily with minimal path routes, we assume the source and destination nodes are *antipodal*, i.e. at opposite corners of the cube.

1.2 Faults

We consider faults to be *permanent*, as opposed to, say, transient, intermittent or even malicious faults. As such, we are dealing with issues of fault tolerance. We wish to assess system behavior in the presence of a set of faults which is fixed for the duration of any routing attempt. In this paper, we only consider *node faults.* A node fault can be thought of as implying that all of that node's communication links are faulty. In fact, since our notion of fault is intended to capture the idea of a node which is unavailable to be used in any communications scheme, a faulty node need not be damaged in the hardware sense but could be a communications or computational hot spot. Lastly, we assume that the source and destination nodes are both *live* (not faulty).

1.2.1 Routing Information Models

We model the amount of information available to individual nodes in two ways:

1. **Model 1** - No local information. Individual nodes *do not know* which, if any, of their neighbors are faulty.

2. **Model 2** - Local information. Individual nodes *know* which, if any, of their neighbors are faulty. •This model is also called the one-step local information model. Issues pertaining to fault diagnosis, however, are not covered in this paper.

1.2.2 Routing Algorithms

Using these routing information models, we have four basic, simple routing algorithms, each of which can be viewed as making *local* decisions. That is to say, at each step of the routing, the next step is determined from the current position and the destination. This is in contrast to determining the entire message path at the source node (and carrying this routing ticket along with the message). Table 1 summarizes the local decisions made by each of the four algorithms.

	Deterministic	Random
Model 1	Correct highest dimension wrong bit	Correct a bit at random from the set of all wrong bits
Model 2	Correct highest dimension *non-faulty* wrong bit	Correct a bit at random from the set of all *non-faulty* wrong bits

Table 1: **Four Basic Routing Algorithms**

1.2.3 Fault Distribution Models

To model the distribution of faults, we use ideas based on the theory of *random graphs* ([1], [6]). The two differing models are:

1. **Model A** - Fix p, the probability that a single node is faulty, $0 \leq p \leq 1$, and let the set of node faults be distributed binomially with probability p.
2. **Model B** - Fix f, the number of faults, $0 \leq f \leq (2^n - 2)$, and let all possible distributions of f faulty nodes be equally likely. For comparisons with Model A, we set $f = \lfloor p(2^n - 2) \rfloor$.

As mentioned above, the fault distribution is fixed for the duration of any routing attempt.

1.3 The Problem to be Analyzed

The goal of this study is as follows:

```
   Find a fault tolerant routing scheme which can ensure a high
probability of successful message routing on a hypercube.
```

In attempting to fulfill this goal, we are motivated by the belief that that the performance of simple algorithms, operating under simple assumptions, is (already) *very good.* Our approach is to determine the performance of these simple algorithms *analytically* and, if need be, *empirically.*

2 Analysis

In this section, we analyze the probability of successfully routing a single message. Routing succeeds if the message reaches the destination. With no local information (Model 1), routing fails if we attempt to route the message to a faulty node. With one-step local information (Model 2), routing fails if we reach a *blocked* node, one which has no non-faulty neighbors along any minimal path to the destination. Since there are two routing methods, two fault information models and two fault distribution models, there are eight separate analyses to perform. As is shown is the following section, by exploiting the relationship between deterministic and random routing in the one message case, we can reduce the number of analyses to be performed to four. We then derive closed form answers for the success probabilities in each of the four remaining cases.

2.1 Relating Success Probabilities using Deterministic versus Random Routing

Let us presume we working with Model B, where for fixed cardinality, k, all subsets of size k are equally likely to occur. Then, for single message routing on a hypercube, the probabilities of successful routing are *the same* using deterministic and random routing.

More precisely, let $s = 0, d = 2^n - 1$ and let $\mathcal{F} \subseteq \{1, \ldots, 2^n - 2\}$ be a subset of faults. If $Det(\mathcal{F}) =$ the probability of successful routing (from s to d) using deterministic routing on a hypercube with all of the nodes in $\mathcal{F}$ faulty, and if $Ran(\mathcal{F}) =$ the probability of successful routing (from s to d) using random routing on a hypercube with all of the nodes in $\mathcal{F}$ faulty, then the following is true.

Theorem: For both Model 1 and Model 2, in Model B, $\sum_{|\mathcal{F}|=k} Det(\mathcal{F}) = \sum_{|\mathcal{F}|=k} Ran(\mathcal{F})$, for $0 \leq k \leq 2^n - 2$.

In other words, the probabilities are the same for a random set of faults of fixed size k. This result holds in both the no-local information case and in the one-step local information case. The main idea behind both proofs is that the probabilities of success using each possible (random) path balance out so that their average is the same as the probability for the deterministic path. In addition, even though this result was defined in the context of Model B, it is easy to see to that an analogous result holds for Model A. This is true because all fault sets of fixed cardinality k are also equiprobable when faults are distributed binomially.

2.2 Deriving Closed Form Success Probabilities

We now have four cases to analyze based only on our fault information and fault distribution models. Let us call the desired probabilities $p_A^1(n,p)$, $p_A^2(n,f)$, $p_B^1(n,p)$ and $p_B^2(n,f)$.

2.2.1 Model 1 Analysis

Under the assumptions of Model 1, individual nodes have no local information about the fault status of their neighbors. Hence, the message can be successfully routed only if there are *no faults* on the fixed, deterministic path from the source, s, to the destination, d. Therefore,

$$p_A^1(n,p) \;=\; (1-p)^{n-1} \tag{1}$$

and

$$p_B^1(n,f) \;=\; \binom{2^n-2-(n-1)}{f} \Big/ \binom{2^n-2}{f}. \tag{2}$$

Theorem: For any $0 \le p \le 1$, $\lim_{n\to\infty} p_B^1(n, \lfloor p(2^n-2)\rfloor) / p_A^1(n,p) \;=\; 1$.

Results of this type are common in random graph theory. In this context, the theorem says that we can use whichever closed form probability is more convenient for our purposes. In fact, the proof shows that the ratio converges to 1 quite quickly; specifically, the ratio converges on the order of $\exp(n^2/2^n)$. Unfortunately, since $\lim_{n\to\infty} p_A^1 = 0$ for fixed $p > 0$, we see that routing using no local information is not good at all.

2.2.2 Model 2 Analysis

The case of one-step local information is the more difficult one to analyze. We are able to arrive at closed form solutions by making some observations. First, we note that the probability of success using deterministic routing equals the sum over all deterministic paths r of the probability of successful routing using r times the probability that r is the deterministic path. The Model 1 analysis above tells us what the probability of successful routing using a given deterministic path, r, is. Second, there are $n!$ possible deterministic paths/routes, each corresponding to the order in which the n dimensions are traversed. Hence, routes correspond to permutations on n elements. The third and key insight is that given a specific route, i.e. σ a permutation on $1\ldots n$, *the number of faults required for σ to be the deterministic path is equal to the number of inversions of σ*, which is $|\{(i,j) : 1 \le i < j \le n, \sigma(i) > \sigma(j)\}|$.

Following Knuth [5], let $I_n(k)$ be the number of permutations on n elements with exactly k inversions. Note that $\binom{n}{2}$ is the maximum number of inversions for any permutation on n elements.

Then, we have shown that:

$$p_A^2(n,p) = (1-p)^{n-1} \sum_{k=0}^{\binom{n}{2}} I_n(k) \cdot p^k . \tag{3}$$

Since the generating function for inversions is well-known, we can simplify this result as follows:

$$\begin{aligned} p_A^2(n,p) &= (1-p)^{n-1} \sum_{k=0}^{\binom{n}{2}} I_n(k) \cdot p^k && (4) \\ &= (1-p)^{n-1} \prod_{k=0}^{n-1} (1 + p + p^2 + \cdots + p^k) && (5) \\ &= \prod_{k=2}^{n} (1 - p^k) . && (6) \end{aligned}$$

(Alternate) Proof: Consider the local routing decision of a node k steps away from the destination. The probability that all k neighbors in the forward direction are faulty is p^k, so the probability of successfully routing the message this one step is $(1-p^k)$. But the overall probability of successfully routing the message is the probability that we can route the message each step from $k = n$ steps away down to $k = 2$ steps away (since the destination is assumed to be live). Hence, $p_A^2(n,p) = \prod_{k=2}^{n}(1-p^k)$. □

But now, from real analysis, we know that $\lim_{n\to\infty} p_A^2(n,p)$ converges. Of course, so did $p_A^1(n,p)$ but it converged to 0. Unfortunately, we cannot derive a simpler closed form for this value; but we can compute and graph its values quite easily. Figure 1 shows the values for $p_A^2(n,p)$ as a function of p.

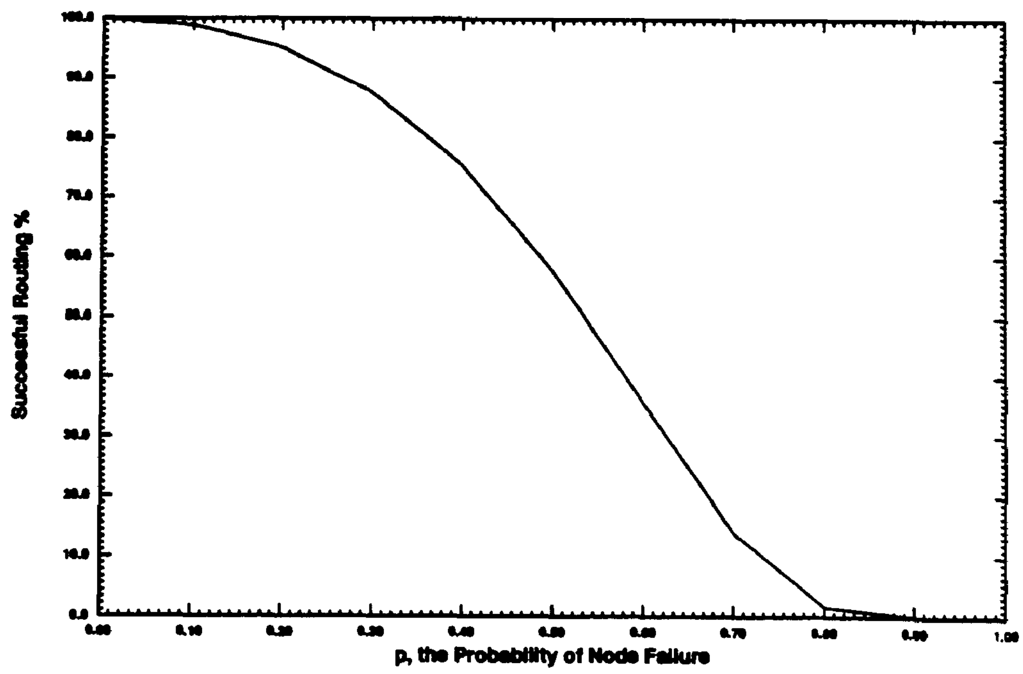

Figure 1: $\lim_{n\to\infty} p_A^2$: **Random Routing with Local Information**

Using the same insights that gave us:

$$p_A^2(n,p) = (1-p)^{n-1} \sum_{k=0}^{\binom{n}{2}} I_n(k) \cdot p^k$$

we see that:

$$p_B^2(n,f) = \binom{2^n-2}{f}^{-1} \sum_{k=0}^{\binom{n}{2}} I_n(k) \cdot \binom{2^n-2-(n-1)-k}{f-k} . \tag{7}$$

Theorem: For any $0 \leq p \leq 1$, $\lim_{n \to \infty} p_B^2(n, \lfloor p(2^n-2) \rfloor) / p_A^2(n,p) = 1$.

As in the Model 1 case, the proof gives an estimate of the rate of convergence, which here is on the order of $\exp(n^5/2^{2n})$. We see that the ratio converges more quickly in the Model 2 case than it does in the Model 1 case. Since we have now derived closed form solutions for each of the four possible model combinations, we conclude that the one message case is fully analyzed.

3 Sidetracking and Empirical Results

The analysis of random routing using one-step local information showed that the probability of successful message routing is high even for an exceedingly large number of faults. However, since random routing attempts to route via minimal length paths, it uses only about one half of the local information presumed available, i.e. only that in the forward direction. We have devised a method called *sidetracking* which takes advantage of all of the information available at each node. Sidetracking, considered as a local decision, first attempts to route the message in the forward direction using random routing. If the message is *blocked*, i.e. no non-faulty neighbors along a minimal path to the destination, then it routes randomly to a live neighbor in the backward direction. Except at the source node, we are guaranteed to have at least the node which routed the message to the blocked node as a live backward neighbor.

The problem with implementing sidetracking is to decide when to stop and say that the message routing has failed. We would ideally like to carry as little extra information along with the message as possible. An extra $\Theta(n)$ bits are alright since that much is already required to carry along the destination address. We simulate this decision by having each message carry with it a maximum path length (actually a count of length to go) and declaring that routing has failed if the destination has not been reached within that many routing steps. One problem that may occur with sidetracking is *tunneling*. Tunneling arises when we reach a blocked node with only one live backward neighbor. If the backward node's only live forward neighbor is the blocked node, then the message will cycle between these two nodes until the maximum path length limit is reached and the routing attempt will fail.

At present, the analysis of sidetracking has been carried out using empirical results from simulation experiments. The inputs of the simulation program are the cube dimension n, the node fault probability p and the maximum path length multiple mpl* (of n, so that the maximum path length is mpl* times n). Since the program is calculating a probability, it runs a number of Bernoulli trials of routing sufficient to ensure that the final answer is within one per cent of the actual value (with 95% confidence). For each experiment, the outputs of the simulation program are the probability of successful message routing $S(n,p)$ and the histogram of path lengths of successfully routed messages from which we compute the average or expected path length (APL).

We have found that the performance of sidetracking is a substantial improvement over random routing. In fact, the empirical results have led us to the following:

Conjecture: For fixed $p < 1$, $lim_{n \to \infty} S(n,p) = 1$.

Note that the value 1 is attainable only asymptotically since there is a fixed probability, p^n, that the message will be blocked at the source. Not only are we almost assured of routing the message successfully but the penalty of sidetracking, as measured by path length excess (over n, the length of a minimal path route) is small. For example, for $p = 1/2$, mpl* = 3, APL/$n \leq 5/4$. Table 2 summarizes the results of using sidetracking for fault probabilies greater than 1/2. These results show that, despite our concern, tunneling rarely occurs.

n	p	mpl*	p_A^2 %	$S(n,p)$ %	APL	σ/n	$\frac{APL-n}{n}$
20	.50	20	58.0	100.0	23.99	0.43	0.20
20	.55	20	46.9	99.9	26.43	0.62	0.32
20	.60	20	35.8	99.8	30.48	0.97	0.52
20	.65	20	23.9	98.8	38.12	1.55	0.91
20	.70	20	14.0	95.0	51.29	2.45	1.56
20	.75	20	6.2	81.6	67.57	3.23	2.37
20	.80	20	1.8	50.0	82.99	3.62	3.15
20	.85	20	0.3	12.6	86.76	3.43	3.33
20	.90	20	0.01	0.3	86.44	2.92	3.32
20	.95	20	0.0	0.0	0.00	0.00	0.00

Table 2: **Sidetracking Results using Model A**

n	p	mpl*	p_A^2 %	$S(n,p)$ %	APL	σ/n	$\frac{APL-n}{n}$
20	.50	20	58.0	100.0	22.11	0.20	0.11
20	.55	20	46.7	100.0	23.29	0.27	0.16
20	.60	20	35.7	99.9	25.05	0.38	0.25
20	.65	20	23.9	99.8	28.51	0.64	0.43
20	.70	20	14.2	99.0	34.20	0.99	0.71
20	.75	20	6.3	94.1	44.97	1.60	1.24
20	.80	20	1.8	71.0	61.57	2.36	2.07
20	.85	20	0.2	21.4	69.36	2.50	2.47
20	.90	20	0.0	0.6	48.37	1.41	1.41
20	.95	20	0.0	0.0	0.00	0.00	0.00

Table 3: **Backtracking Results using Model A**

In addition to sidetracking, we implemented a version of *randomized backtracking* where when we reach a blocked node we set that node to faulty before routing backwards. This change to sidetracking is sufficient to eliminate all tunneling. Note that this method is not that the same as standard backtracking since the backwards routing is to a random live node and not to the node from which we came previously. Given enough time, randomized backtracking will find a path (for the message) between the source and the destination, if any such path exists. Hence, it is useful for comparisons with sidetracking. On a real machine, however, to implement backtracking would require far more effort and resources than sidetracking. Such an implementation would require

	Sidetracking				**Backtracking**			
	Dimension (mpl* = 5)				Dimension (mpl* = 5)			
p	5	10	15	20	5	10	15	20
0.1	99.9	100.0	100.0	100.0	100.0	100.0	100.0	100.0
0.2	99.8	100.0	100.0	100.0	99.9	100.0	100.0	100.0
0.3	98.3	99.9	100.0	100.0	99.2	99.9	100.0	100.0
0.4	93.4	99.5	99.9	99.9	96.7	99.9	100.0	100.0
0.5	80.3	96.7	99.4	99.8	87.1	99.3	99.9	100.0
0.6	58.5	84.8	94.9	98.4	65.7	95.4	99.4	99.9
0.7	32.0	53.6	73.3	85.1	35.6	72.3	91.8	97.6
0.8	9.7	11.9	23.7	37.3	11.1	20.8	40.4	61.2
0.9	0.9	0.1	0.1	0.3	0.9	0.1	0.2	0.6

Table 4: **Probability of Success Convergence Rates**

large space overhead since each message must carry with it a complete list of all nodes visited by the message. Transmission of this history information, in turn, would incur large communication time overhead. Table 3 summarizes the results for backtracking, again with fault probabilities greater than 1/2. The greatest benefit of backtracking is in lowering the average path length of a successfully routed message but, overall, the improvements over sidetracking are not great. Lastly, Table 4 shows how fast the probability of success values converge to their asymptotic limits (as $n \to \infty$) using both sidetracking and randomized backtracking.

4 Conclusions

We have presented a framework for the analysis of fault tolerant routing schemes on a hypercube. Our notion of faults is extremely versatile. Our results apply not only to systems with permanent node failures; but since a communications hot spot may be identified as a fault, our methods apply as well to routing in congested hypercubes.

The basic conclusion to be drawn from our study is that randomization is very effective when it is applied to routing. In addition, we have shown that simple routing schemes with simple assumptions, namely that of local information, work very well. Specifically, the sidetracking algorithm performs incredibly well. Consider the following asymptotic behavior. Using minimal path random routing with no local information, the probability of successful message routing is 0%. Using minimal path random routing with one-step local information, the probability of successful message is quite good, but tails off rapidly for a high probability of node failure (see Figure 1). Finally, using sidetracking (or *non*-minimal path random routing with one-step local information), the probability of successful message routing is 100%.

There are many implications of the conjectured asymptotic behavior of sidetracking. Its behavior implies that, given a fixed pair of live antipodal nodes of the hypercube, not only are they in the same connected component, but that sidetracking algorithm actually finds a path connecting them. These consequences further the work of Erdös and Spencer [3] who prove a double jump threshold for the connectivity of a random n-cube. They show that asymptotically, for $p > 1/2$ the n-cube is disconnected, for $p < 1/2$ the n-cube is connected and for $p = 1/2$ the n-cube is connected with probability $1/e$.[1] Our results show that, even when $p \geq 1/2$ and the n-cube is likely disconnected, sidetracking can almost always successfully route a single message. This success can be achieved because it is most likely that only singleton nodes are disconnected and, hence, two randomly chosen live nodes are probably in the same connected component.

References

[1] B. Bollobás, *Random Graphs.* London:Academic Press, 1985.

[2] A. Borodin and J. E. Hopcroft, "Routing, Merging and Sorting on Parallel Models of Computation," *Journal of Computer and System Sciences*, vol. 30, pp. 130-145, 1985.

[3] P. Erdös and J. Spencer, "Evolution of the n-Cube," *Computers and Mathematics with Applications*, vol. 5, pp. 33-39, 1979.

[4] J. P. Hayes, T. N. Mudge, Q. F. Stout, S. Colley and J. Palmer, "A Microprocessor-based Hypercube Supercomputer," *IEEE Micro*, pp. 6-17, Oct. 1986.

[5] D. E. Knuth, *The Art of Computer Programming, Vol 3: Sorting and Searching.* Reading, Massachusetts:Addison-Wesley, 1973.

[6] E. M. Palmer, *Graphical Evolution : An Introduction to the Theory of Random Graphs.* New York:John Wiley and Sons, 1985.

[7] M. C. Pease, III, "The Indirect Binary n-Cube Microprocessor Array," *IEEE Transactions on Computers*, vol. C-26, pp. 458-473, May 1977.

[8] C. L. Seitz, "The Cosmic Cube," *Communications of the ACM*, vol. 28, pp. 22-33, Jan. 1985.

[9] J. S. Squire and S. M. Palais, "Programming and Design Considerations of a Highly Parallel Computer," in *Proceedings of the Spring Joint Computer Conference*, 1963, pp. 395-400.

[10] L. G. Valiant, "A Scheme for Fast Parallel Communication," *SIAM Journal of Computing*, vol. 11, pp. 350-361, May 1982.

[11] L. G. Valiant, "Experiments with a Parallel Communication Scheme," in *Proceedings of the 18th Allerton Conference on Communications, Control and Computing*, University of Illinois, October 8-10, 1980, pp. 802-811.

[12] L. G. Valiant and G. J. Brebner, "Universal Schemes for Parallel Communication," in *Proceedings of the 13th ACM Symposium of Theory of Computing*, 1981, pp. 263-277.

[1] Even though Erdös and Spencer model edge faults, the results for the case of node faults are the same, except for the value at $p = 1/2$.