Fault-tolerant message routing in computer networks

Lev Zakrevski, Mark Karpovsky[♣]

Reliable Computing Laboratory,
Boston University, Department of Computer Engineering,
8 Saint Mary's St., Boston, MA 02215, USA
zakr@bu.edu, mr@enga.bu.edu

Abstract

We are considering a problem of fault-tolerant wormhole unicast routing in computer networks, consisting of nodes (processor units), connected by bi-directional links. Each node can be viewed as the combination of a router and a processor with some RAM, bus and I/O circuitry. We concentrate on networks with arbitrary (irregular) topologies.

The problem of routing in the presence of faults is divided into two sub-problems: diagnosis and reconfiguration of the network and deadlock-free routing in the new network. In this paper we consider non-adaptive routing, not taking the sizes of corresponding message queues into account.

1. Introduction

The problems of fault detection (testing) and fault location (diagnosis) have been mostly studied under the general area of system-level diagnosis [7,21,29]. System-level diagnosis techniques model the system as a test graph, whose vertices denote the nodes and an edge or test link ($\mathbf{p}_i$,$\mathbf{p}_j$) from node $\mathbf{p}_i$ to node $\mathbf{p}_j$ indicates that $\mathbf{p}_i$ tests $\mathbf{p}_j$.

While the test graph in is a subgraph of the system graph, it generally contains as many vertices as there are nodes in the system. For large systems, this approach requires the execution of the test program on every node and can involve significant overhead. Also, it implies that for centralized testing the amount of information sent to the host is relatively large and requires rather complex analysis of the test results. Clearly, a distributed testing strategy that requires fewer tester nodes is desirable. The fault model in system-level diagnosis has generally been limited to processor (node) faults and link and router faults have rarely been considered [21,30,31].

Recently, several authors have proposed techniques for testing and diagnosis of multiprocessors based on self-testing [25]. These techniques can be useful for diagnosis of a large number of processor faults but they are less efficient for detection and location of link and router faults.

Most of the drawbacks mentioned above can be alleviated by combining self-test and system test. The main idea of our approach is to select some nodes, called monitors, which execute a comprehensive self-test. Fault-free monitors are then used to test non-monitor nodes at the second stage of the test. This approach allows simple identification of faulty processors. From the mathematical point of view, this leads to the problem of covering the system graph by balls of some given radius R (usually R=1) centered at monitors for detection of processor (node) faults.

To detect all single node faults, we need to cover each node by at least one ball. The corresponding problem of finding the minimal covering system of balls is known as the *1-covering problem*. We note that this problem is still open even for such regular topologies as binary n-cubes [3].

To detect and locate faults in the communication block of the system, instead of

♣ This work was supported by the NSF under Grant MIP 9630096

using coverings by balls, path based techniques utilizing shorter messages travelling longer distances can be used. This approach leads to the problem of covering links, nodes and pairs of neighboring links by minimal sets of paths [2,18,19]. The major criterion for selecting a system of covering monitors or covering paths is the time required for testing. Other criteria include memory for storing the testing programs in monitors and complexity of the analysis of testing results.

The following classes of faults are considered in this paper:

1. *Processor faults*.
2. L*ink faults* (including links between routers and processors). For a bi-directional link two faults may occur when a message cannot be transmitted in either direction. The number of link faults can then be evaluated as $N_L = 2N + \sum_{i=1}^{N} d_i$, (1)
   where d_i is the degree (number of neighbors) of node **i** in the network graph.
3. *Router faults.* We consider two types of router faults
   3.1 *Total Router Faults* (TRFs). In this case, no messages can go through the router (removal fault, [23]). If every node consists of a router and a processor then the number of these faults is equal to *N*.
   3.2 *Partial Router Faults* (PRFs). In this case, messages cannot be transmitted from one incoming link to one outgoing link of the router. For example, for 2D meshes a message can be transmitted in any direction but cannot make a turn. The bypass fault models presented in [23] is the special case of PRFs. The number of different single PRFs is:

$$N_{PRF} = \sum_{i=1}^{N} d_i (d_i + 1). \quad (2)$$

We note that a TRF can be represented as a multiple link fault, and a link fault as a multiple PRF fault.

We consider two main methods to detect and locate these faults. First of them is based on covering the network graph by minimal set of balls, centered on monitors, which perform self-check and system check of neighboring nodes [2]. Second method is based on covering all nodes/links by minimal set of paths.

After location and isolation of faulty components the system should go through the pre-routing stage. At the end of this stage the routing table will be constructed in routers of every node. This table for node **I** should indicate for every destination **J** a neighbor of **I** where the message should be send in such a way that no deadlocks can appear in the system.

In this paper we consider the case when wormhole routing [26] is used. According to this method, each message is divided into flits and all flits follow the same path. The major consideration is the absence of deadlocks [4,8]. Most of the known methods for wormhole routing were designed for a specific architecture, such as *n*-dimensional mesh [10,31], while the design of deadlock-free routing algorithms in irregular topologies introduces new challenges [23].

Very few papers [23,28] have been published on routing for networks with irregular topologies. The proposed techniques based on spanning trees may result in traffic contentions near roots of these trees. In this paper we describe another approach for non-adaptive unicast routing deadlock-free wormhole routing, which provides for message paths very close to the shortest ones and more uniform distribution of the traffic between communication links in the system.

2. Testing and diagnosis of computer networks

2.1 Testing and diagnosis of processor faults

In this section, we investigate *testing and diagnosis of faulty nodes* combining self-test and system test. For testing, we select some nodes as monitors. These monitors perform a self-test first. If a monitor passes the self-test it then starts testing its neighbors. An example of monitor placement for processor testing for the irregular topology is shown in Fig.1.

The testing problem can be formulated in the following way: select a minimal set of nodes (monitors) such that balls of radius 1 (radius *R* in the general case) centered at monitors cover all other processors. (A ball with a center **P** of radius *R* is the set of processors connected by paths of length at most *R* to **P**). This problem

was analyzed in [2,15,16,17]. We note that to decrease the test time, balls should be disjoint. Thus, not all nodes at distance at most R from a monitor are included in the corresponding ball. For Example of Fig.1 the balls may be selected as {**2**,**4**,**6**}, {**7**,**1**,**5**}, {**8**,**3**,**9**}.

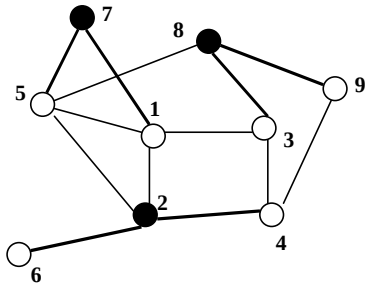

Fig.1 Monitor placement for node testing (monitors are shown in black)

For *diagnosis* monitor placement can be the same as for testing, and diagnosis is done by comparing results of the tests (signatures) from a monitor and its neighbors. In this case, every monitor sends a message to the other nodes (or to the host, if centralized testing is used), which indicates the faulty node in its ball (among its neighbors). For single node faults this message can be encoded in $\lceil \log_2(d_i+1) \rceil$ bits, where d_i is the degree of monitor **I**.

For this approach we have the following lower bound for a minimal number of monitors, M, required for diagnosing a fault involving a single node:

$M \geq K$, where K is the smallest integer such that $\sum_{i=1}^{K}(d_i+1) \geq N$ (3)

(we assume $d_1 \geq d_2 \geq \ldots \geq d_N$). In particular,

$M \geq N/(d_1+1)$. (4)

We also have the following upperbound for the best monitor placement:

$M \leq N/2 \quad (N>1)$. (5)

To prove (5), one can construct a spanning tree in the network graph and select as monitors all nodes on either odd, or even levels.

In Fig. 2 the results of simulations for the number of monitors normalized to numbers of nodes are given (here and below all results are averaged for 1,000 randomly generated graphs).

Fig.2 illustrates advantages of our approach based on combining self-test and system test which results in a reduction of a number of testers from the total number of nodes, N, in the original graph to the number, M, of monitors.

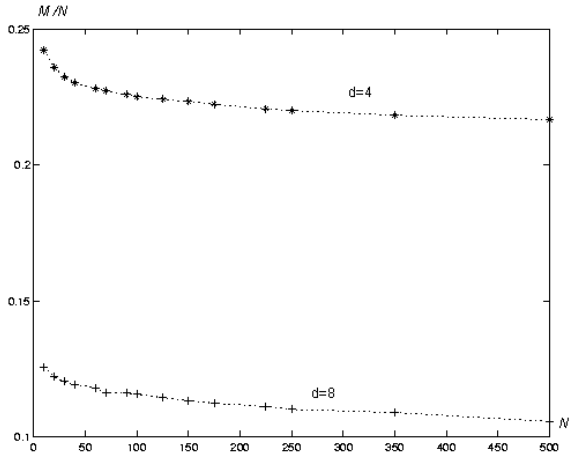

Fig.2 Densities of monitors M/N required for node covering as functions of numbers N of nodes for randomly generated graphs with average degrees equal to 4 (stars) and 8 (pluses).

Monitor placement based on covering of nodes and aimed at single node faults provides for a near-complete diagnosis of faults of higher multiplicity. We have shown [17] that as the number of nodes grows, the diagnosabililty (probability of correct location) of faults of multiplicity exactly t (t nodes are faulty) approaches 1 when N is growing if $t \leq O(N^{1/2})$. From the practical point of view, this implies that for large systems, the number of faulty nodes can grow at the rate of up to $N^{1/2}$ without loss of diagnosability.

2.2. Testing and diagnosis of link faults

In this section, we outline two approaches for testing and diagnosis of link faults - the first is based on covering all the links by balls centered at monitors and the second is based on covering of links by paths starting at monitors.

The first approach provides for testing and diagnosis of all single and multiple link faults and isolation of these faults by disabling the ports connecting monitors and faulty links. This fault isolation allows for graceful degradation and facilitates system reconfiguration. The main disadvantage of this approach is a large number of required monitors.

For the second approach based on covering of links by paths starting at monitors, the

number of monitors is going down, however the location of multiple link faults becomes difficult.

Link covering by balls

Let M be the number of monitors required for detection of link faults. In a d-regular graph, with L links and N nodes, *perfect link testing* is achieved with $M=\lceil L/d \rceil = \lceil N/2 \rceil$ monitors if every link is tested by exactly one monitor. If perfect monitor placement is unavailable, we are looking for an optimal monitor placement, minimizing the number of monitors. An example of this optimal monitor placement is given in Fig.3 (monitors are shown in black).

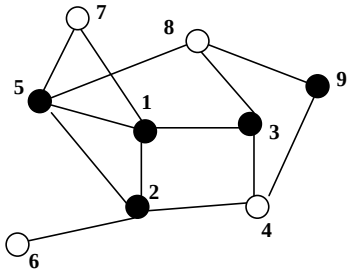

Fig.3 Monitor placement for link covering by balls for the network of Fig.1

The monitor placement problem for link testing is related to the problem of determining the point covering number of the system graph G. A vertex and an edge in G *cover* each other if they are incident, and the size of the smallest set of vertices that covers all the edges in G is called the *point covering number* $\alpha(G)$ [7]. Clearly, $M = \alpha(G)$ and the processors in the smallest cover are selected as monitors. A set of vertices in G is *independent* if no two of them are adjacent. Let $\beta(G)$ be the size of the maximum independent set for G. The maximum independent set can be constructed using a simple procedure described in [7]. For any nontrivial connected graph G with N vertices, $\alpha(G)+ \beta(G)= N$. This equation can be used to determine $\alpha(G)$.

The experimental results on numbers of monitors for link covering for randomly generated graphs are shown in Fig.4.

We note that if the average degree increases, the number of monitors increases for link covering and decreases for ball covering but in both cases considerable savings can be obtained by using as testers monitors only.

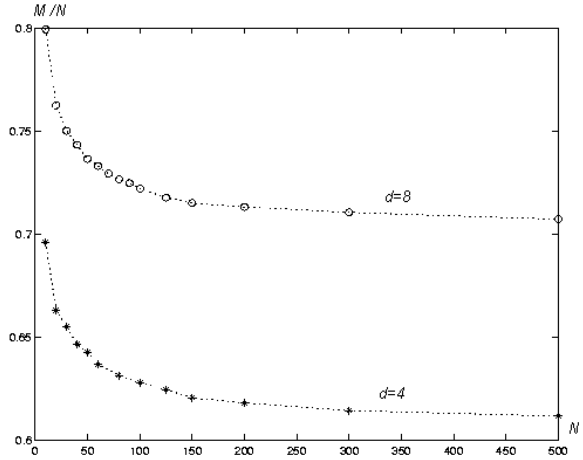

Fig.4 Densities of monitors M/N required for link covering as functions of numbers N of nodes for randomly generated graphs with average degrees equal to 4 and 8.

Link covering by paths

For detection of link faults using this approach, we need to cover every link by at least one message path starting at a monitor. Each node (and therefore, each link) can belong to the path only once. To avoid the contention we will partition the set of covering paths into W groups (phases), such that paths within a group do not intersect (do not have common nodes). The minimal number of phases for link faults is $\lceil 0.5 \max_i(d_i) \rceil$. For Example shown at Fig.1, the following 2 paths will cover all links (in this case W=2): (*Path 1*): **6-2-5-1-3-8-9-4**; (*Path 2*): **3-4-2-1-7-5-8**.

As a criterion for selection of a set of paths covering all links for detection of link faults we will use test time T that can be estimated as

$$T = \sum_{i=1}^{W} (t_M^{(i)} + \max_j T_j^{(i)} + t_H^{(i)}), \qquad (6)$$

where $t_M^{(i)}$ is a time required to initiate phase i of the test, $T_j^{(i)}$ is a time required to perform a test by monitor j and $t_H^{(i)}$ is a time to send test results from monitors to the host at phase i.

We note, that if every link in the original graph belong to a cycle and β is a cyclomatic number of the graph [11] then we have for the minimal number R of required paths

$$R \le \beta + 1 . \qquad (7)$$

To *locate* a single link fault, it is necessary that each link is covered by a *unique combination of paths*. The lower bound on the number of paths (messages) is $\lceil \log_2(L+1) \rceil$, where L is the number of links. For Example of Fig.1 L=13 and one may select 5 paths, which are represented by the following covering matrix $\boldsymbol{C}=(C_{ij})$, where C_{ij} =1 iff link i is covered by path j (i=1,…,13; j=1,…,5):

	1-2	1-3	1-5	1-7	2-4	2-5	2-6	3-4	3-8	4-9	5-7	5-8	8-9
	0	0	0	1	0	0	0	1	0	1	1	1	1
	0	1	0	1	1	1	0	0	1	1	0	1	0
$\boldsymbol{C}=$	0	1	1	0	0	1	1	0	1	1	0	0	1
	1	1	0	0	0	1	0	1	0	1	0	1	0
	1	1	0	0	0	0	0	0	0	0	0	0	0

In the general case, all columns in the covering matrix should be different and $\boldsymbol{C}$ should not contain the all-zeros columns. We note that the number of phases W required for diagnosis is upperbounded by the number of rows in the covering matrix. The following lower bound for the minimal number of paths R covering all links can be obtained, similar to the bounds for node covering by balls:

$$R \geq 2L/(P_{max}+1), \quad (9)$$

where P_{max} is the length of the maximum path (in links).

Diagnosis of all link faults can be implemented by first covering all edges by at most $\lceil N/2 \rceil$ disjoint paths or cycles [24], and then by covering each path/cycle of length k (in edges) by $\lceil k/2 \rceil$ paths of the length $\lceil k/2 \rceil$ each. For example, path **1-2-3-4-5** will be covered by 3 paths: **1-2-3**, **2-3-4** and **3-4-5**. It will ensure the diagnosability of single link faults. Since the sum of lengths of all disjoint paths is L, the upperbound for R is:

$$R \leq L/2 + N/2. \quad (10)$$

Similar criteria can be used for diagnosis of more than one link fault. For *location of at most* t link faults the number of paths is lower-bounded by $\log_2 \sum_{i=0}^{t} \binom{L}{i}$. In this case all componentwise ORs of at most t columns of the coverage matrix $\boldsymbol{C}$, should be different, which means that $\boldsymbol{C}$ is a check matrix of a *superimposed* code of length L correcting t errors [20].

2.3. Testing and diagnosis of router faults

To test routers we can adopt the same path-based approach as it was done for link faults.

Let us first consider the case of *total router faults* (TRFs). In this case, we have to cover all router nodes in the network graph by paths. Example of such covering is given in Fig.6.

Thus, the problem of testing TRFs can be reduced to covering of all nodes by paths minimizing test time T, which can be estimated by (6).

If the network graph is connected, one can use for testing a path covering the maximal number of nodes and cover N-P_{max}-1 remaining nodes by at most $\lceil (N-P_{max}-1)/2 \rceil$ paths. Thus, the number of paths sufficient for testing of TRFs is upperbounded by:

$$R \leq \lceil (N-P_{max}+1)/2 \rceil. \quad (11)$$

For diagnosis of TRFs, we can cover all nodes by the set of disjoint paths, as outlined above. It will lead to the formula

$$R \leq R_T \lceil P_{max}/2 \rceil, \quad (12)$$

where R_T is the number of paths suffiicient for testing of TRFs.

For detection of *partial router faults* (PRFs, Section 2.2.3), all possible turns (pairs of neighboring links) should be covered. One solution of this problem for the network in Fig.1 is given by the following 8 paths:

(A): **6-2-1-5**; (C): **3-4-2-1-7-5-8**;
(E): **9-4-3-8-5-1-7**; (G): **1-5-2-4-9**;
(B): **7-5-1-3**; (D): **7-5-2-1-3-8-9-4**;
(F): **7-1-3-4-2-6** (H): **6-2-5-8.**

In this case W=6 (the test will require 6 phases) and the covering is optimal.

For the location of PRFs, the following upperbound holds:

$$R \leq M'_T \lceil P_{max}/2 \rceil, \quad (13)$$

where R'_T is the number of paths necessary for testing of PRFs.

For detection and diagnosis of *all single and multiple* PRFs in irregular networks one can use all nodes as monitors with each monitor sending test messages to all nodes that are at the distance

2 from the monitor. In this case $T=O(d_1^2)+O(N)$, where $d_1 = \max_i d_i$.

3. Deadlock-Free Routing in Computer Networks

In this section we propose a new approach to deadlock-free wormhole unicast routing. We first construct a spanning tree *T(G)* for the network graph *G*. Then, we label the nodes (by unique labels) preserving the partial order defined by *T(G)*.

To avoid deadlocks it is prohibited to have three sequential nodes with labels p_1, p_2, p_3 in the path, if $p_1 < p_2$ and $p_3 < p_2$ (TOP-DOWN restriction). Since any path containing links from *T(G)* satisfies this restriction, any message will be delivered if *G* is a connected graph.

This approach does not require virtual networks. As opposed to the known spanning tree approaches (see e.g. [23]) for our method trees will be used only for labeling of nodes of *G* at the pre-routing (routing table construction) stage. This will result in a decrease of traffic contention near the root of the trees, which is a bottleneck for many existing spanning tree approaches.

The example of the spanning tree and labeling of the nodes, constructed by this algorithm is shown in Fig.1.

Next, we address the problem of routing in *G*, assuming that *T*(*G*) is already constructed and each node **I** is labeled by L_i. For a given source **S** and destination **D** it is necessary to select a shortest routing path $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_m$ ($\boldsymbol{a}_1$=**S**, $\boldsymbol{a}_m$=**D**) among all possible paths, satisfying TOP-DOWN restriction.

First, we introduce a *global* routing algorithm, based on global knowledge (this algorithm is efficient if a number of nodes is small). For this algorithm at the first phase two matrices ***E*** and ***F*** are formed, that show for every pair of source-destination nodes if the label-increasing and label-decreasing paths exist and lengths of the shortest paths. This phase requires about N^3 operations (*N* is the number of nodes). Next, if we need to route from **S** to **D**, we can find a shortest distance as *d*(**S**,**D**) = min (*E*(**S**,**I**)+*F*(**I**,**D**)) (for all **I** such that *E*(**S**,**I**) and *F*(**I**,**D**) are defined). The value of $L_{\mathbf{I}}$ shows the minimal label, which will be reached.

If the use of global knowledge is too costly, a *local* algorithm can be used. Let us assume that we need to find a routing path from **S** to **D** and $L_s \geq L_d$. Then we look for all nodes **C**, adjacent to **S**, such that $L_s \geq L_c$ and select the node, closest to **D**. The distance between **C** and **D** is estimated as the tree distance in *T(G)*. At the next step we find a best route between **C** and **D**, etc. For example, finding a routing path from node **5** to node **9** in Fig.5, we construct path **5-1-3-4-9**. (The global approach generates the shortest path **5-8-9**).

To compare the developed methods, computer simulations for randomly generated graphs with *N*=64 nodes have been performed. About 10,000 uniformly distributed messages (with length 200 flits) have been generated for every graph and every message generation rate.

In Fig.6 the dependencies of the average delivery time (latency) from a message generation rate for local and global approaches are shown (*N*=64, average degree 8).

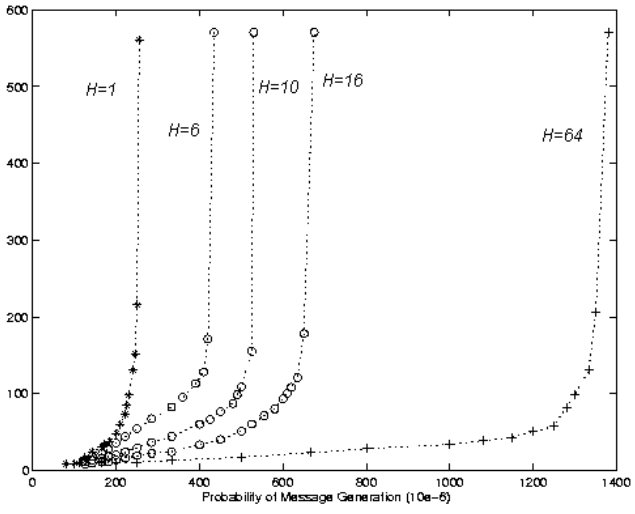

Fig. 6. Dependency of average message delivery time (clock cycles) on message generation rate for local (left) and, global (right) approaches.

This graph illustrates a good scalability of the proposed approach.

Conclusions

In this paper, we investigated the problem of detection and location of permanent faults in

computer networks with arbitrary topologies. For processor faults, we have proposed the use of a monitor-based approach, covering all nodes by balls of radius one, centered at monitors. A maximum of $\lceil N/2 \rceil$ monitors for N processors is required to test all single faults. For detection and location of link and router faults, methods based on covering network graphs by appropriate sets of paths have been developed. The transition from testing to diagnosis of router and link faults results in an increase in a number of covering paths by at most a factor of $\lceil P_{max}/2 \rceil$, where P_{max} is the maximal length of a path in the network.

Also, the problem or deadlock-free wormhole routing was considered. The proposed methods provide for efficient deadlock-free wormhole routing for networks with irregular topologies. These approaches result in message paths very close to the minimal ones. Results of simulation experiments illustrate a good scalability of these techniques.

Acknowledgments

Authors would like to thank Prof. Levitin from Boston University for many useful suggestions and Mr. Sharad Jaiswal from the Reliable Computing Laboratory, Boston University for performing simulations.

References

1. R.V. Boppana and S. Chalasani, "Fault-Tolerant Wormhole Routing Algorithms in Mesh Networks," *IEEE Trans. on Comput*., vol. 44, pp.848-864, 1995.
2. K. Chakrabarty, M.G. Karpovsky, L.B. Levitin, "Fault Isolation and Diagnosis in Mutliprocessor Systems with Point-to-Point Connections," *Fault Tolerant Parallel and Distributed Systems* (Editors D.Avresky, D.Kaeli), Kluwer Academic Publishers, 1998, pp.285-301.
3. G.D. Cohen, I. Honkala, S.N. Litsyn and Lobstein, "Covering Radius," North Holland, Amsterdam, The Netherlands, 1997.
4. J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks," *Proc. of Int. Conf. on Parallel Processing*, vol. I., pp. 142-149, August 1994.
5. A.G. Duachkov, V.V. Rykov, "A Survey on Superimposed Code Theory," *Problems of Control and Information Theory*, vol.12, No.4, 1983.
6. E. Fleury and P. Fraigniaud, "A General Theory for Deadlock Avoidance in Wormhole-Routed Networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 9, pp.626-638, July 1998.
7. C. Glass and L. Ni, "The Turn Model for Adaptive Routing," *Proc. of the 19th Annual Int. Symp. on Computer Architecture*, pp. 278-286, May 1992.
8. C. Glass and L. Ni, "Fault-Tolerant Wormhole Routing in Meshes," *Proc. of Int. Symp. on Fault-Tolerant Computing*, 1993.
9. F. Harary, "Graph Theory," *Addison-Wesley*, Reading, Mass, 1969.
10. J.P. Hayes, "A Graph Model for Fault-Tolerant Computing Systems," *IEEE Trans. on Computers*, vol.25, pp.875-884, 1976.
11. M.G. Karpovsky, K. Chakrabarty, L.B. Levitin, "A New Class of Codes for Identification of Vertices in Graphs," *IEEE Trans. Info Theory*, March 1998.
12. M.G. Karpovsky, K. Chakrabarty, L. Levitin, D. Avresky, "On the Covering of the Vertices for Fault Diagnosis in Hypercubes," *Information Processing Letters*, vol.69, pp.99-103, 1999.
13. M.G. Karpovsky, E.A. Moskalev, "Tests for Non-Oriented Graphs," *Automaton and Remote Control*, vol.31, pp.656-665, April 1970.
14. C.R. Kime, "System Diagnosis", In Fault-Tolerant Computing: Theory and Techniques, vol.2, D.K. Pradhan (ed.), Prentice-Hall, New Jersey, 1986.
15. P. LeMahieu, V. Bohossian, J. Bruck, "Fault-Tolerant Switched Local Area Networks," *Proc. of the First Merged Int. Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, pp. 747-751, 1998.
16. R. Libeskind-Hadas, D. Mazzoni and R. Rajagopalan, "Tree-Based Multicasting in Wormhole-Based Irregular Topologies," *Proc. of the First Merged Int. Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, pp.244-249, 1998.
17. F.J. Meyer and D.H. Pradhan, "Dynamic Testing Strategy for Distributed Systems," *IEEE Trans. on Computers*, vol.39, March 1989.
18. L. M. Ni and P.K. McKinley, "A Survey of Wormhole Routing Techniques in Directed Networks," *Computer*, vol. 26, pp. 62-76, Feb. 1993.
19. M. Schroeder et al., "Autonet: A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links," Technical Report 59, DEC SRC, April 1990.
20. C.L. Yang and G.M. Masson, "An Efficient Algorithm for Multiprocessor Fault Diagnosis Using the Comparison Approach", *Proc. of the 16th Annual Int. Symp. on Fault-Tolerant Computer Systems*, pp.238-243, 1986.
21. L. Zakrevski and M.G. Karpovsky, "Fault-Tolerant Message Routing for Multiprocessors." *Parallel and Distributed Processing* (Editors J.Rolim, D.Avresky, D.Kaeli), Springer, 1998, pp.714-731.