FAULT TOLERANT UNICAST WORMHOLE ROUTING IN IRREGULAR COMPUTER NETWORKS

Mark Karpovsky, Mehmet Mustafa, Raman Mathur
Dept. of Computer Engineering, Reliable Computing Laboratory
Boston University, 8 St. Mary's Street, Boston, MA 02215
United States

ABSTRACT

For Networks with *t+1* link disjoint spanning trees, we propose a fault tolerant deterministic and adaptive unicast wormhole routing technique which can tolerate up to 100% of all faults involving up to *t* link faults and high percentage of faults involving more than *t* links. This technique provides deadlock prevention with message delivery times that is almost identical to the case without fault tolerance. The proposed algorithm consists of two stages. At the first stage, we minimize the set of turns in the network graph, which are prohibited for deadlock prevention and fault tolerance. At the second stage, routing tables are constructed based on the set of prohibited turns that minimize average message path lengths for adaptive routing. Routing tables constructed this way produce a set of output ports sorted in ascending order in terms of the distance to the destination. To route the messages adaptively we select the next available shortest path to the destination. We also present results of simulation experiments for average delivery time for uniform traffic pattern and saturation points.

KEY WORDS

Fault-tolerant, adaptive, wormhole, turn model, deadlock prevention

1. INTRODUCTION

Because of the proliferation of inexpensive workstations, ad hoc clustering of workstations, routers or switches and communication links became very popular. Such clustering of workstations forms an irregular topology referred to as Network Of Workstations or NOWs. In order to achieve high bandwidth communications, recent experimental and commercial switches for NOWs implement wormhole routing [1, 2]. Since wormhole routing is susceptible to *deadlocks* considerable body of work has been dedicated to designing routing algorithms that prevent deadlocks from occurring [3-10].

Existing routing strategies can be classified as deterministic [6, 9, 11, 12] and adaptive [3, 5, 8, 9, 13]. In deterministic routing the path between a source and destination is unique, whereas in adaptive routing network state is considered during routing of worms. Adaptive routing algorithms that use all permitted ports in every router are called maximal [6]. Maximal adaptive routing is not fully adaptive, since to prevent deadlocks some ports are not available for routing. In our earlier work we explored a new deadlock prevention algorithm based on turn prohibitions and demonstrated its superior performance for low latency message delivery in irregular computer networks [14-16]. In this paper, we will consider application of turn prohibitions to adaptive fault-tolerant routing.

Main shortcoming of the deterministic routing is its inability to respond to network conditions, such as congestion, dynamically [9]. Once the routing tables are established, they are stationary until network topology changes take place at which time routing tables are recomputed. Therefore when an incoming worm is determined to use a particular output port, worm is blocked until the output port is freed up. This is the case whether there are other available output ports to the destination or not. Our current studies in this paper explore all possible output ports, even at the expense of non-minimal distances to the destination. The only guidepost is prevention of cycle formation and an output port will not be selected if it could lead to deadlocks by creating cycles in the channel dependency graph. With this approach worms would dynamically avoid potential hotspots in the network.

Adaptive routing has been studied extensively. Use of virtual channels to provide adaptive routing has been studied in both regular [5, 17-20] and irregular [21, 22] network topologies. However virtual channel approaches involve adding additional buffers and control circuitry. In [23], author devised an adaptive routing algorithm based on Odd-Even Turn Model in Meshes without virtual channels. However this approach is suitable in meshes only with non uniform traffic model. To guarantee freedom from deadlock formation, in our approach we use the *TP* or *Turn Prohibition* algorithm [24] to prohibit a minimal or near minimal number of input/output pairs of ports at various nodes of the network. Minimizing the

number of prohibited turns has been shown to correlate to lower average message latency in meshes and tori [8] and in irregular graphs [10, 15]. After turn prohibition, routing tables are computed based on the shortest path between source and destination pairs. All paths computed this way contain no prohibited turns. Instead of keeping just the shortest distance to the destinations, we sort distances in increasing order. During on-line routing of messages, we pick the unused output port in this sorted list. For very light traffic conditions always the output port with the shortest distance to the destination will be selected. Hence the behavior of the adaptive system under light traffic conditions is identical to that of the predictive routing. As the traffic generation rate is increased, routing function selects the next available output port, which may or may not be minimum distance to the destination. In network topologies where there are multiple output ports to the destination at minimum distance, subsequent selections of the output ports will result in choosing the ports with optimal distances to destinations. On the other hand as the traffic generation rate is increased further where all optimal distance ports are used, the routing function begins selecting non-optimal ports. In addition, we studied adaptivity available in networks that contain multiple, *t* edge-disjoint spanning trees. We try to use this property to our advantage during routing. We apply turn prohibition rules to break all cycles and therefore prevent deadlocks as follows. First, at each node all turns from links of one spanning tree to the other trees are prohibited. Second, for all cross links that belong to no spanning trees, all turns at both ends of the cross-links are prohibited. Third, *Turn Prohibition* algorithm [24] is applied to the sub-graph obtained by deleting the edges of the *t+1* link-disjoint spanning trees from the original graph. In the rest of the paper we discuss the *Turn Prohibition* or *TP* algorithm and *TP*-based adaptive routing algorithm and adaptivity in graphs with *t+1* link-disjoint spanning trees in Section 2. In Section 2 we also discuss the theoretical basis for the existence of edge disjoint spanning trees in connected graphs. In Section 3 we present our experimental simulation results and finally we offer our conclusions in Section 4.

2. Turn Prohibition and Theoretical Basis for Adaptive Routing

In this section we describe briefly the *Turn Prohibition* or the *TP*-algorithm for creating set of prohibited turns Z(***G***), for a given network graph ***G*** with *N*(***G***) nodes. The *TP*-algorithm is recursive, in which at each step one node is selected and all turns at the selected node are prohibited. After turns are prohibited at the selected node, the node and all of the incident edges are deleted. For example, if after deleting a node ***a*** with degree d_a and all edges incident on it, the remaining network graph ***G-a*** is still connected, then we prohibit all $d_a(d_a-1)/2$ turns (*c, a, b*) and permit all turns (*a, b, c*). Algorithm is invoked recursively as long as there are nodes with degree greater than two in the remaining graph.

Following properties have been shown [14] to hold:

1. Any cycle in ***G*** contains at least one turn included in *Z*(***G***) and hence all cycles are broken.
2. Fraction of prohibited turns is upper-bounded by 1/3.
3. *TP*-algorithm maintains graph's connectivity. For any two connected nodes ***a*** and ***b*** in the original graph, there exists at least one path between ***a*** and ***b***, without any turns from Z(***G***).
4. Set Z(***G***) is irreducible. Deletion of any turn from ***Z(G)*** a new cycle in ***G*** containing no turns from Z(***G***).

We note that the *TP*-algorithm has a complexity of $O(N^2d)$, where *N* is the number of nodes in ***G***, and *d* is the maximal degree of the node. After the set Z(**G**) is computed a routing matrix $\boldsymbol{R}_x(i,\boldsymbol{k})$ is computed for each node *x*, where *i* is the input port number for an incoming message and ***k*** is the destination node number. Thus for a graph with *N* nodes where each node is of degree *d* the routing matrix is of size *(d+1)xN*. Each matrix element is a pointer to a list of output ports, where the output ports are sorted in an ascending order in terms of the distance of the destination node from the present routing node. At the head of the list is the output port number with the shortest distance to the destination. As an example we use the graph in **Fig. 1** in which the turn prohibition has been applied and prohibited turns are depicted by arcs between the two edges involved in the turn. For example we see that there are no prohibited turns at node ***1*** but all turns at node ***7*** are prohibited. At node ***11*** we also show the port numbers for the links. For example for all messages arriving at node ***11*** from input ports 1, 2 and 3 will be routed out on port number 4 since all other turns are prohibited at this node. The list at this node $\boldsymbol{R}_{11}(1,\boldsymbol{10})$ contains just one element, namely port number 4. Similarly the list for $\boldsymbol{R}_{11}(4,\boldsymbol{14})$ is < 2, 1, 3>. Port numbers are sorted as shown since the distances from node ***11*** to node ***14*** are 1, 2, 2 hops from output ports 2, 1, and 3 respectively. No differentiation is made between output ports that are equidistant to the destination node. When an output port is in use, it is flagged as such. For example if both output port 2 is busy when a message arrives at node ***11*** from input port 4, then output port 1 will be used for routing the message destined for node ***14***. Similarly if port 1 is also in use when message arrives then output port 3 will be used.

For the network topology in **Fig. 2** in which multiple edge-disjoint spanning trees can be constructed, we explored an alternative adaptive routing. For example in the network in **Fig. 2**, we have shown two spanning trees, one drawn with the edges shown as dotted line segments and the other with dashed line segments. The third edge

type shown as solid line between nodes ***1*** and ***3*** is what is referred to as crosslinks.

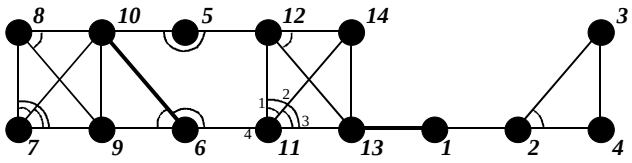

Fig. 1 A Network Topology With Prohibited Turns Illustrating The Generation Of The Routing Tables for *TP* and Adaptive Routing

In this figure we show the prohibited turns again as arcs. The figure on the right is by a straightforward application of the *TP*-Algorithm as discussed above. The set of prohibited turns in the graph on the left is derived as follows:

1) First all turns at every node between edges on different trees are prohibited.
2) For each crosslink, we prohibit the turns between the crosslink and all tree edges. We do this on both ends of the crosslink.
3) We then delete the edges corresponding to the tree edges and apply the *TP*-algorithm to the remaining subgraph.

For the simple graph in **Fig. 2**, In the first step all turns except turns (***3,1,2***) and (3,1,6) at node ***1*** and turns except (***1,3,2***), (***1,3,4***), and (***1,3,6***) at node ***3*** are prohibited. At the second step turns (***3,1,2***), (***3,1,6***), (***1,3,2***), (***1,3,4***), and (***1,3,6***) are prohibited. Finally, at the third step no turns are prohibited because the remaining subgraph is acyclic. After the identification of the set of prohibited turns, similar to the previous case, routing matrices are constructed for each node where each matrix entry is again a pointer to a list of output ports.

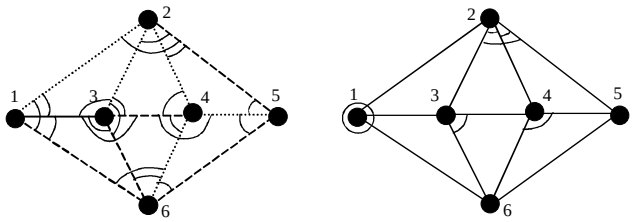

Fig. 2 Network Topology With Two Edge-Disjoint Spanning Trees. Turn prohibitions shown are based on the spanning trees on the left and *TP* on the right.

Spanning tree based adaptive routing relies on the fault-tolerant properties of the underlying network graph G. We define S(G) to be a set of turn prohibitions breaking all cycles in G and call S(G) to be *t-fault-tolerant* if there exists *t+1* edge-disjoint paths containing no turns from S(G) between any two nodes of G. If S(G) is generated by the *TP*-algorithm, S(G) is *0-fault-tolerant* since *TP* guarantees only one path between any two nodes of the graph G. Therefore, if a network topology is such that multiple edge-disjoint spanning trees can be constructed, we could choose to use the additional paths as alternate paths between a source node and a destination node, if we discover the first path to be in use. We therefore expect that for high density graphs, where there are many more crosslinks than tree edges, messages would be delivered at least as effectively as the *TP*. We expect such network topologies to perform better than deterministic *TP* approach due to (i) existence of multiple edge-disjoint paths between any two nodes and (ii) large number of crosslinks. Property (i) assures multiple paths for adaptivity and fault tolerance and large number of crosslinks assures that during step 3) above, fraction of prohibited turns due to crosslinks will not exceed 1/3 due to *TP* property 2. These assumptions are not unreasonable. Commercial routers with large numbers of ports easily available for large workgroups and inter-working workgroups. We can therefore assert that if S(G) is *t-fault-tolerant* then it is possible to use adaptive routing. Routing function $R_x(i,k) = <p_1, p_2,\ldots, p_{t+1}>$ now provides a list of *t+1* output port numbers $p_1, p_2,\ldots, p_{t+1}$, instead of just one, as the case is for deterministic routing.

Following necessary and sufficient conditions apply to fault tolerance.

1-(Necessary) If S(G) is *t-fault-tolerant* the so is G.

2-(Necessary) If S(G) is *t-fault-tolerant* for t > 0, then deleting any node *a* and its incident edges from the graph G, results in a sub-graph with no more than two components.

3-(Necessary) If S(G) is *t-fault-tolerant* then the number of edges *M* of the graph with *N* nodes satisfies $M > (N-1)(t+1) - 1$.

4-(Necessary) If S(G) is *t-fault-tolerant* then the cut-set size for the graph |C| and minimum node degree in the graph, d_{min} are both greater than *t*.

5-(Sufficient) If G is *t-fault-tolerant*, then graph obtained by adding one new node and arbitrarily connecting it to at least *t+1* nodes of the original graph, results in a new *t-fault-tolerant* graph.

6-(Sufficient) If two *t-fault-tolerant* graphs G1 and G2 are interconnected by at least *t+1* edges then the new graph is also *t-fault-tolerant.*

For example, two-dimensional torus is *1-fault-tolerant*. Similarly a complete graph K_4 and any wheel graph W_n [25] are also *1-fault tolerant*.

3. **Simulation Experiments**

We first studied the existence of multiple spanning trees in randomly generated connected *d-regular* graphs and in graphs with an average degree d. In *d-regular* topologies each node has a fixed degree d router All topologies are irregular. We generated such topologies and then attempted to construct as many spanning trees as

possible. After each spanning tree is generated, edges participating in the tree are deleted and the remaining graph is analyzed again for more spanning trees. At any point if the remaining edges in the graph are fewer than N-1 we terminate the search for more trees. Our results for randomly generated 16-node graphs are shown in **Fig. 3**. In ten experiments we found that for degrees greater than five, we were always able to construct at least two spanning trees.

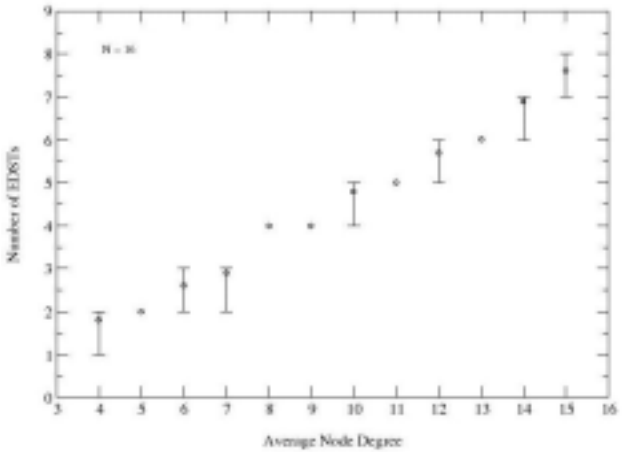

Fig. 3 Number of Edge-Disjoint Spanning Trees In Randomly Generated Connected Graphs

For flit level experiments, an event-driven simulator was used to evaluate the performance of the *TP*-algorithm for adaptive wormhole-routed irregular computer networks. In all experiments the deterministic *TP*-algorithm and the two adaptive routing algorithms are compared. We first generated connected irregular graphs ranging in size from 8 to 256 nodes. All network channels were bi-directional and symmetric. Message queues at each node are of infinite length. Output channel/buffer contention is resolved using the FIFO queuing policy, with each incoming flit being time stamped on its arrival at the router input buffer. In our simulations, we used mostly *uniform* traffic pattern where each node can send a message to any other node with equal probability. Communications arising from nodes are independent and identically distributed by the Poisson process with the generation rate equal to 1/*p* (messages/cycle/node, where *p* is the probability of message generation for any cycle, at any node). The message length was constant and equal to 200 flits and the input/output buffers in the routers were 1-flit deep. In addition a separate experiment has been conducted that investigated the impact of different message lengths on the average latency (delivery time) and on saturation points. The experiments were also performed for different node degrees. A typical simulation would be averaged over a 100 random graphs in each of which 100,000 messages were exchanged. Generally performances of routing algorithms are measured in terms of the average message latencies and saturation points, which are considered as the highest sustainable message generation rates. These experimental assumptions are similar to those reported in [12, 26].

Our first set of experiments involved comparing the performances of deterministic *TP* and adaptive *TP* approaches as discussed above. We simulated message delivery using the low level simulator for both the deterministic and adaptive *TP*. For network sizes of 8, 16, 32, 64, 128, and 256 nodes we measured the latency and computed average latency. Our results shown in **Fig. 4** and **Fig. 5**, indicate that in all but the smallest 8-node graphs both approaches performed similarly with the deterministic *TP* approach having a slight edge. In the 8-node graphs the adaptive approach performed marginally better. These results are somewhat counter-intuitive until we recall that in our experiments we selected the next available shortest distance port to route the otherwise would be blocked messages. Our experimental results indicate that on the average the distances traveled by majority of the messages were not minimal. Even though we could possibly have acquired the minimal distance port if we were to wait a few clock cycles, we chose not to wait and took the next available and possibly longer distance port. In the figures, ED or Edge Density is the probability of the presence of an edge between any two nodes.

In the next set of experiments we studied the impact of message length on latency. We performed similar simulation experiments in 256 node irregular graphs for message latency, with the message length as the parameter in both the deterministic and adaptive *TP* approaches. We definitely see the slight edge (11% better) of the deterministic *TP* clearly with the longest, 200-flits long messages. These results are shown in **Fig. 6** and **Fig. 7** below. We see that shorter messages have the tendencies to reach saturation further away than the longer messages.

Our last set of experiments explored performance in high-density graphs with multiple edge disjoint spanning trees. We used 32-node graphs with two edge-disjoint spanning trees and varied the edge density between 0.50 and 0.95 for all graphs. We than prohibited every turn at every node between the two spanning trees. Furthermore, we prohibited the turns on both ends of all crosslinks that do not belong to any one of the spanning trees. Finally we deleted the edges corresponding to the spanning trees and applied the *TP*-algorithm to the remaining network graph. After breaking all possible cycles in the graph we then selected the port with the shortest distance as the routing port and constructed the deterministic routing table. We simulated message delivery in such a network and compared the results with the deterministic *TP* approach. Our results are shown in **Fig. 8** and **Fig. 9.** In **Fig. 10** and **Fig. 11** we show the results of adaptive routing using fault tolerant approach closer to origin for 0-fault tolerant adaptive TP and 1-fault tolerant adaptive TP. The simulated network topology is the 32 node irregular graph with 0.5 edge density and two edge

disjoint spanning trees. The performance in both cases is almost identical in terms of message latency with no perceivable difference between the *TP* and the fault-tolerant *TP* approaches.

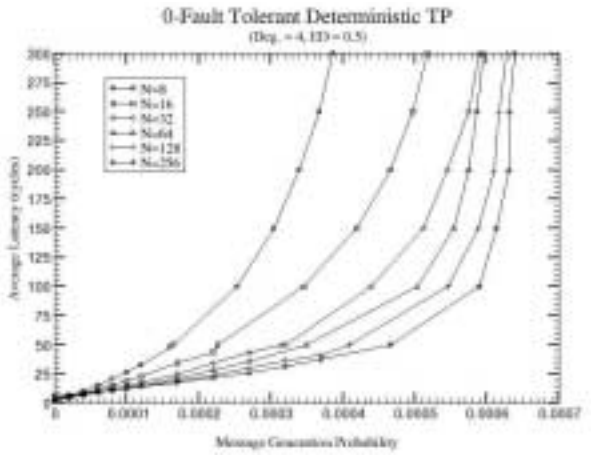

Fig. 4 Average Message Latency Using Deterministic *TP*

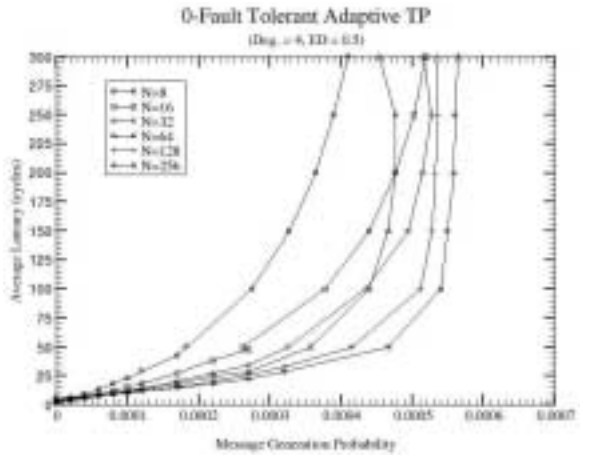

Fig. 5 Average Message Latency Using Adaptive *TP*

4. Conclusions

In this paper we studied deterministic *TP*, adaptive *TP* and deterministic and adaptive fault-tolerance based *TP* algorithms and compared their performances for average message latency. All four algorithms have exhibited about the same performance in terms of message delivery times. In the fault-tolerance based *TP* approach we have increased the fraction of prohibited turns from either the *TP* or the adaptive *TP* approaches but the performance remained the same with the added improved reliability and fault tolerance.

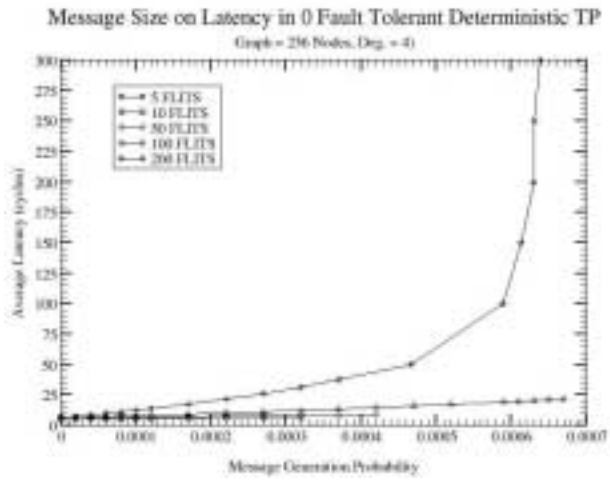

Fig. 6 Average Latency With Message Length As The Parameter Using *TP*

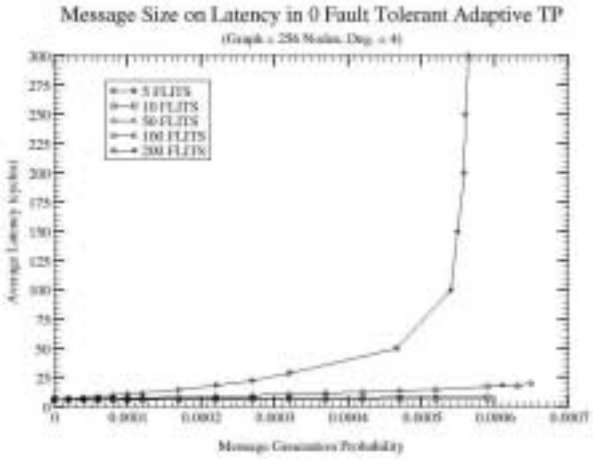

Fig. 7 Average Latency With Message Length As The Parameter Using Adaptive *TP*

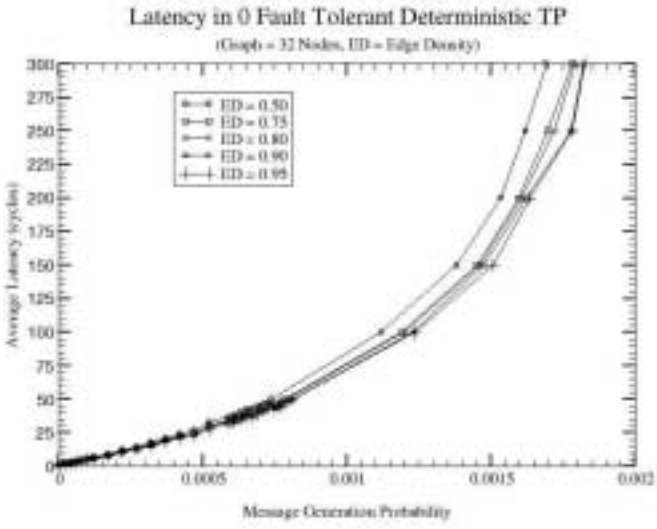

Fig. 8 Average Latency With Edge Density As The Parameter Using Deterministic *TP*

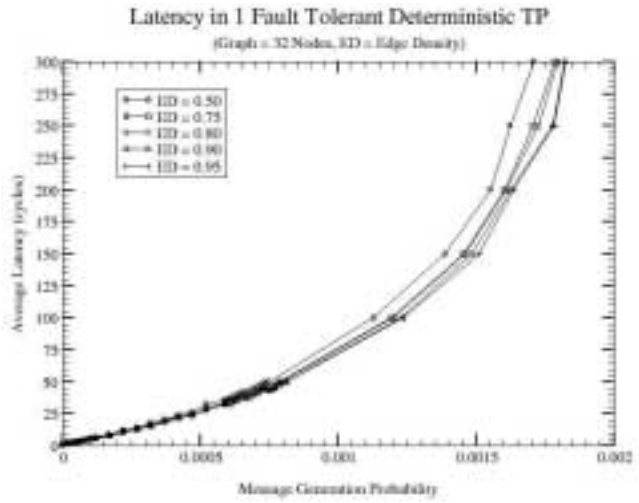

Fig. 9 Average Latency With Edge Density As The Parameter Using Deterministic Fault Tolerance Based Routing

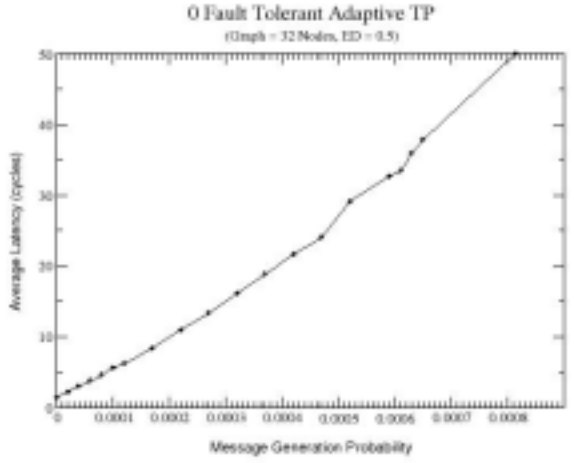

Fig. 10 Average Message Latency inn 32-Node Irregular Topology With ED = 0.5 And Zero Fault Tolerance

5. Acknowledgment

The authors would like to thank Professor Lev Levitin for his insightful discussions and Krishna Gali of Boston University for helping with the fault tolerance based turn prohibition experiments. This research was partially sponsored by NSF grant MIP-96630096.

REFERENCES

[1] N. Boden and e. al., Myrinet: A Gigabit per second Local Area Network, *IEEE Micro* pp. 29-35, 1995.
[2] R. Horst, W., ServerNet(TM) Deadlock Avoidance and Fractahedral Topologies, *Proc. of IEEE Int. Parallel Processing Symp.* pp. 274-280, 1996.

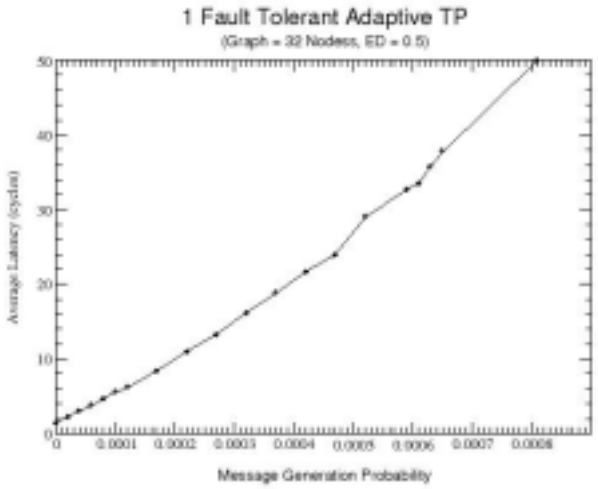

Fig. 11 Average Message Latency in 32-Node Irregular Topology With ED = 0.5 And With One Fault Tolerance

[3] W. Dally and H. Aoki, Deadlock-Free Adaptive Routing in Multiprocessor Networks Using Virtual Channels, *IEEE Trans. on Parallel and Distributed Systems vol. 8*, pp. 466-475, 1997.
[4] W. Dally and C. Seitz, L., Deadlock-Free Message Routing in Multiprocessor Interconnection Networks, *IEEE Trans. on Comput. vol. 36*, pp. 547-553, 1987.
[5] J. Duato, A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks, *IEEE Trans. on Parallel and Distributed Systems vol. 4*, pp. 1320-1331, 1993.
[6] J. Duato, S. Yalamanchili and L. Ni, *Interconnection Networks An Engineering Approach*, 1997.
[7] 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, 1998.
[8] C. Glass and L. Ni, The Turn Model for Adaptive Routing, *Journal of ACM* vol. 5, pp. 874-902, 1994.
[9] L. Ni, M. and P. McKinley, K., A Survey of Wormhole Routing Techniques in Directed Networks, *Computer vol. 26*, pp. 62-76, 1993.
[10] L. Zakrevski and M. Karpovsky, G., Fault-Tolerant Message Routing in Computer Networks, *Proc. of Int. Conf. on PDPA-99* pp. 2279-2287, 1999.
[11] R. Boppana, V. and S. Chalasani, Fault-Tolerant Wormhole Routing Algorithms in Mesh Networks, *IEEE Trans. on Comput. vol. 44*, pp. 848-864, 1995.
[12] B. Ciciani and M. Colajani, Paolucci, C., Performance Evaluation of Deterministic Routing in k-ary n-cubes, *Parallel Computing* no. 24, pp. 2053-2075, 1998.
[13] R. Boppana and S. Chalasani, A Comparison of Adaptive Wormhole routing Algorithms, *Computer Architecture News vol. 21*, no. 2, pp. 351-360, 1993.
[14] L. Zakrevski, S. Jaiswal, L. Levitin and M. Karpovsky, A New Method for Deadlock Elimination in

Computer Networks With Irregular Topologies, *Proc. of the IASTED Conf. PDCS-99 vol. 1*, pp. 396-402, 1999.
[15] L. Zakrevski, S. Jaiswal and M. Karpovsky, Unicast Message Routing in Communication Networks With Irregular Topologies, *Proc. of CAD-99* 1999.
[16] L. Zakrevski, M. Mustafa and M. Karpovsky, Turn Prohibition Based Routing in Irregular Computer Networks, *Proc. of the IASTED International Conference on Parallel and Distributed Computing and Systems* pp. 175-179, 2000.
[17] Y. Boura, M. and C. Das, R., Efficient Fully Adaptive Routing in n-Dimensional Meshes, *Proc. International Conf. Distributed Computing Systems* pp. 589-596, 1994.
[18] A. A. Chien and J. Kim, Planar Adaptive Routing: Low-Cost Adaptive Networks for Multiprocessors, *Journal of ACM vol. 42*, no. 1, pp. 91-123, 1995.
[19] Glass, J. C and Ni, M. L., Maximally Fully Adaptive Routing in 2D Meshes, *Proc. 1992 International Conf. Parallel Processing*, pp. 101-104, 1992.
[20] H. Linder and C. Harden, An Adaptive and Fault-Tolerant Wormhole Routing Strategy for k-Ary n-Cubes, *IEEE Trans. Computers vol. 40*, no. 1, pp. 2-12, 1991.
[21] F. Silla and J. Duato, On the Use of Virtual Channels in Networks of Workstations with Irregular Topology, *Proc. of the 1997 Parallel Computing, Routing, and Communication Workshop*, June, 1997.
[22] L. Zakrevski, M. Mustafa and M. Karpovsky, Turn Prohibition Based Routing in Irregular Computer Networks, *Proc. of PDCS-2000* pp. 174-179, 2000.
[23] G.-M. Chiu, The Odd-Even Turn Model for Adaptive Routing, *IEEE Transactions on Parallel and Distributed Systems vol. 11*, no. 7, pp. 729-737, 2000.
[24] L. Zakrevski, PhD Thesis: Fault-Tolerant Wormhole Message Routing in Computer Communication Networks, *College of Engineering* pp. 21-27, 2000.
[25] F. Harary, Graph Theory, *Addison-Wesley Series in Mathematics*, 1998, 43-56.
[26] C. Glass and Ni,L., Fault-Tolerant Wormhole Routing in Meshes, *Proc. of Int. Symp. on Fault-Tolerant Computing* 1993.