# UNICAST MESSAGE ROUTING IN COMMUNICATION NETWORKS WITH IRREGULAR TOPOLOGY

**LEV ZAKREVSKI, SHARAD JAISWAL, MARK KARPOVSKY**

Dept. of Computer Engineering, Boston University,
8, St. Mary's Street, Boston, MA 02215
zakr@bu.edu

**Abstract**. In this paper we consider the problem of deadlock-free unicast wormhole routing in computer and communication networks with irregular topologies. An example of such networks are Network of Workstations (NOWs). In general, the topology of these networks can be quite random. Several methods exist in the literature for wormhole routing in networks/multiprocessors with a regular topology, such as a *n*-dimensional mesh, but very few papers have been published on wormhole routing for irregular networks. Some of these existing techniques require complex signaling hardware at the routers or result in a large amount of congestion at some specific links. The problem of deadlock-free routing consists of two parts. First, all deadlocks must be eliminated. An usual way of doing this, both for regular and irregular topologies, is to forbid some turns. The second part, which is the focus of this paper, is the problem of selecting an optimal (usually the shortest) path after the restrictions on routing have been formulated. We propose three efficient approaches for solving this problem. These approaches (local, global and mixed) differ in a way distances in the network graph are estimated using local information stored in the routers. Our approach for non-adaptive unicast deadlock-free wormhole routing provides for message paths very close to the shortest ones and more uniform distribution of the traffic between communication links in the system. Initial simulation results presented in the paper indicate that the proposed approaches are promising in terms of both throughput and scalability.

**Key Words**. Wormhole routing, deadlock elimination, multiprocessor systems.

## 1. INTRODUCTION

Wormhole routing is efficient because it allows low channel-setup time, low latency communications and reduced communication overhead [2,7,9,17]. It has been adopted in almost all existing inter-connection networks. Gradually, variations of this technique, are being incorporated in commercial NOW implementations like Myrinet [1,13,14,18]. Recently, NOWs have emerged as an inexpensive alternative to massively parallel multiprocessors [16]. NOWs comprise a collection of routing switches, communication links and workstations interconnected in an irregular topology. In order to minimize network latency and achieve high bandwidth communications, recent experimental and commercial switches for NOWs implement wormhole routing [1,13,18]. However, wormhole routing is very susceptible to *deadlocks* [6,7,8,9,10,12,17] because packets are allowed to hold many resources while requesting others. Design of efficient deadlock-free routing algorithms in irregular topologies introduces new challenges, which we shall address in this paper.

Overall, routing strategies can be divided into adaptive [3,6,8,12,17] (taking into account existing queue sizes) and unadaptive techniques [2,5,9,17]. In this paper, we will consider non-adaptive methods, which can however be extended for adaptive routing. Several routing methods currently exist for regular topologies, such as 2-dimensional meshes or hypercubes [2,4,6,7,9,11,12]. In addition several approaches have been developed for the more difficult problem of routing in the presence of faults [2,4,9,11,18,19,20] when the number of faults is relatively small compared to the number of nodes. We note that if the number of faults is large then the fault-tolerant routing problem becomes almost equivalent to routing in an arbitrary topology.

For the case of a general topology, the most widespread routing strategy is based on the spanning tree approach [16,18]. According to this strategy, once a spanning tree is constructed, any two nodes can communicate with each other along the tree without any deadlocks. The main drawbacks of this approach are the long message paths and high load on the edges near the root node [16]. This method can be improved by allowing shortcuts using edges, not belonging to the spanning tree [16]- but this could result in deadlocks due to the formation of cycles in the channel dependency graph. A more general routing strategy is the following. Each edge is labeled by a number, such that there are no cycles, consisting of edges with the same label. Then, for routing, it is allowed to only increase labels along the routing path (or, in a more general case, first increase, then decrease them, but not otherwise). Examples of this strategy are *e-cube* [9,17] and *North-Last* [12] approach for meshes and spanning tree [16,18] approach for an arbitrary network (all edges going up to the root of the tree are marked by 0, all edges going down by 1 etc.). To measure the efficiency of the routing strategy, the average message delivery time can be used [3,5,9,17] as a parameter for comparison. Any good routing strategy aims to increase the maximal sustainable throughput and decrease the delivery time for generation rates below the saturation point.

This paper is organized as following. In Section 2 a general mathematical model of the network, and the unicast "spanning tree" based routing approach is discussed and the global, local and mixed strategies are introduced. Section 3 presents the experimental results for these approaches. In Section 4 we consider performance enhancements in these routing strategies by the addition of virtual networks. Section 5 is devoted to conclusions.

## 2. UNICAST SPANNING-TREE BASED DEADLOCK-FREE ROUTING

We assume that the given network consists of $\boldsymbol{N}$ nodes connected by $\boldsymbol{E}$ edges. Also, we assume that all nodes are connected (for any two nodes there exists a path between them). In general, a network graph $\boldsymbol{G}$ can be considered to be a multigraph- their exist several edges between the same two nodes [7,9,17]. In particular, if $\boldsymbol{k}$ virtual networks are used, each two nodes are connected either by 0, or by $\boldsymbol{k}$ edges. (Each physical channel is split into $\boldsymbol{k}$ virtual channels using time multiplexing [6,7]). Each virtual channel has its own buffer. Sum of the capacities of these virtual channels is restricted by the capacity of the original physical channel (so large $\boldsymbol{k}$ will lead to performance degradation.) Usually this multigraph is characterized by a given network graph and the parameter $\boldsymbol{k}$. In general the number of virtual channels can be different for different links.

In the case of deterministic routing, a routing strategy will be a function on the set of pairs ($\boldsymbol{s,d}$) where $\boldsymbol{s}$ and $\boldsymbol{d}$ are nodes. For each such pair, the value of this routing function will be either 0 (message will not be transmitted), or a vector of edges representing the path from $\boldsymbol{s}$ to $\boldsymbol{d}$**.** The set of all path vectors will be denoted by $\boldsymbol{P}$ [6,7]**.**

As indicated earlier, a major consideration for any wormhole-based routing strategy is to demonstrate it to be deadlock free. The condition for deadlock elimination can be checked by analyzing $\boldsymbol{P}$**.** Based on $\boldsymbol{P}$, the channel dependancy graph can be constructed, with the nodes of this graph corresponding to edges in $\boldsymbol{G}$. For the deterministic case there must be no cycles in the channel dependancy graph [6,7,8,9,10].

The whole problem of deadlock-free wormhole routing can be divided into two parts. First, we have to prevent all deadlocks (eliminate all cycles in channel dependency graph. It can be done using either the *node*, *link* or *turn* models [6,7,8,9,10,12,17]:

*Node model*: each node has some label. It is prohibited to have three sequential nodes in a path, with labels $p_1$, $p_2$, $p_3$ , if $p_1 < p_2$ and $p_3 < p_2$.

*Link model*: each link has some label. It is prohibited to have two sequential links with labels $p_1$, $p_2$ in a path, if $p_1 < p_2$.

*Turn model*: any turn is either permitted, or prohibited.

The link model covers the node model (all links can be labeled with two labels, corresponding to UP and DOWN direction in node model). Similarly, the turn model covers the link model. The advantage of the node model is its compactness - if the number of nodes is $\boldsymbol{N}$, number of links can be of the order of $\boldsymbol{N}^2$ and number of turns of the order of $\boldsymbol{N}^3$**.**

We note that a set of prohibited turns preventing deadlocks does not completely specify the routing strategy, i.e. several routing strategies can satisfy the same set of restrictions on turns in the network graph.

This paper is devoted to routing strategies satisfying selected sets of prohibited turns and minimizing average path lengths and average delivery time for given restrictions on local memories in routers. In particular, we investigate the case when the popular up/down algorithm [16,18] is used to construct a set of prohibited turns. We note that the developed methods can be used when a set of prohibited turns is selected in a different way.

According to up/down approach, we first construct a spanning tree ***T(G)*** for the network graph ***G***. Then, we label the nodes preserving the partial order defined by ***T(G)*** . Labels are unique (different nodes have different labels). It can be shown, that if labels are repeated, it will either lead to deadlock, or the set or permitted turns will be the same, as for some unique labeling.

To avoid deadlocks, any allowable routing path consists of two phases - at the first phase labels can not decrease, and at the second - they cannot increase (TOP-DOWN restriction). For example, 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 $p_3 < p_2$. Since any path containing links from ***T(G)*** satisfies this restriction, any message will be delivered if ***G*** is a connected graph.

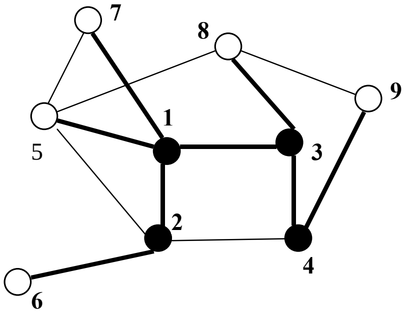

**Fig.1** Example of a spanning tree and its labeling (links of ***T(G)*** are shown in bold).

An example of a random graph, its spanning tree and labeling of its nodes is shown in Fig. 1.

We now address the problem of routing in ***G.*** Lets assume 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 $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_m$ ($\boldsymbol{a}_1=\boldsymbol{s}$, $\boldsymbol{a}_m=\boldsymbol{d}$) among all possible paths, satisfying up/down restrictions.

For any intermediate node ***i*** of a packet path a routing protocol estimates the length of the shortest path between neighbors of ***i*** and the destination, satisfying the restrictions on turns imposed by up/down algorithm, and routes the packet to neighbor ***j***, which has the lowest estimated length (providing that the corresponding turns in ***i*** and ***j*** are permitted). Sizes of local memories in routers will determine the accuracy of these estimations and performance of the corresponding routing strategies.

For the ***local*** approach the distance between any two nodes is estimated as the tree distance (in links) in ***T***(***G***)**.** In this case, the size of the local memory in routers required for storing ***T***(***G***) and node labeling is $O(N)$ (***N*** is a number of nodes) and $O(N^2)$ steps will be required to compute the distances.

For the ***global*** approach the distance between two nodes is the length of the shortest path between these nodes such that this path satisfy the restrictions imposed by up/down algorithm. The size of the local memory required for the global approach is $O(N^2)$ and $O(N^3)$ steps are needed to compute the distances.

The ***mixed*** (hierarchical) approach is a combination of the local and global approaches.

First, we consider the *global* routing algorithm, based on global knowledge (this algorithm is efficient if a number of nodes is small). For this algorithm, in 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 length of the shortest of these paths (if the paths exist, information about the next node along the shortest path can also be stored). 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 the shortest distance as $d(\boldsymbol{s},\boldsymbol{d}) = \min\,(E(\boldsymbol{s},\boldsymbol{i})+F(\boldsymbol{i},\boldsymbol{d}))$ (for all ***i*** such that $E(\boldsymbol{s},\boldsymbol{i})$ and $F(\boldsymbol{i},\boldsymbol{d})$ are defined). The corresponding value of $L_i$ shows the minimal label, which will be reached. To maximize the saturation point, a criterion based on $\max(L_i)$ (for all ***i*** such that $E(\boldsymbol{s},\boldsymbol{i})$ and $F(\boldsymbol{i},\boldsymbol{d})$ are defined are considered) can be used.

For the above example (see Fig.1) we have the following matrices ***E*** and ***F***:

$$E=\begin{bmatrix} 0 & 1 & 1 & - & - & - & - & - & - \\ 1 & 0 & 2 & - & - & - & - & - & - \\ 1 & 2 & 0 & - & - & - & - & - & - \\ 2 & 1 & 1 & 0 & 3 & - & 4 & 2 & 1 \\ 1 & 2 & 1 & 3 & 0 & - & 1 & 1 & 2 \\ 2 & 3 & 1 & - & - & 0 & - & - & - \\ 1 & 2 & 2 & 4 & 1 & - & 0 & 2 & 3 \\ 2 & 1 & 2 & 2 & 1 & - & 2 & 0 & 1 \\ 3 & 2 & 2 & 1 & 2 & - & 3 & 1 & 0 \end{bmatrix},$$

$$F=\begin{bmatrix}0 & 1 & 1 & 2 & 1 & 2 & 1 & 2 & 3\\ 1 & 0 & 2 & 1 & 2 & 3 & 2 & 1 & 2\\ 1 & 2 & 0 & 1 & 1 & 1 & 2 & 2 & 2\\ - & - & - & 0 & 3 & - & 4 & 2 & 1\\ - & - & - & 3 & 0 & - & 1 & 1 & 2\\ - & - & - & - & - & 0 & - & - & -\\ - & - & - & 4 & 1 & - & 0 & 2 & 3\\ - & - & - & 2 & 1 & - & 2 & 0 & 1\\ - & - & - & 1 & 2 & - & 3 & 1 & 0\end{bmatrix}.$$

One can see from these matrices that a path of length 2 (**5-8-9**) is already indicated in the ***E*** and ***F*** matrices and this is the shortest for all possible ***i.***

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 which is 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, let us assume that we need to find a routing path from node **5** to node **9** in Fig.1. Since these nodes have the same label, at the first step nodes **1**,**2**,**7** will be analyzed. The corresponding distances to node **9** are 3 (for **1**), 4 (for **2**) and 4 (for **7**). So, node **1** is selected. Using this approach iteratively, we construct path **5-1-3-4-9**. (The global approach generates the shortest path **5-8-9,** it is worth pointing out that the local approach has no information about the legal shortcuts between **8-9** and **2-4**).

We will now describe a *mixed* approach, which will be a combination of the local and the global routing approaches. Let us partition spanning tree ***T(G)*** into ***Q*** disjoint connected components (clusters). Every node will have information about the structure of all clusters directly above it in ***T***(***G***) (including the cluster it belongs to). Also, the structure of ***T***(***G***) with node labels is stored (for the whole graph ***G***)**.** For the example of Fig.1 if nodes **1**,**2**,**3**,**4** form one cluster and all other clusters consist of one node, path **5-2-4-9** will be generated using mixed approach, since it has no information about the inter-cluster link **8-9**.

We note that global and local approaches are extreme cases of the mixed approach with ***Q***=1 and ***Q=N.*** We note also that this approach provides for a tradeoff between the sizes of local memories in routers and the throughput of the system. If more memory is available, the number of clusters ***Q*** can be decreased, with the proportional increase in sizes of clusters. This will provide for more information about shortcuts in ***T***(***G***)**.** To store the information about one cluster of size ***H***, we need at most $H^2/2$ bits (binary memory cells). If we suppose that tree ***T***(***G***) is a balanced binary tree, and all clusters have the same number of nodes (so $Q=N/H$), then, each node has information about at most $\log_2(Q)$ clusters, and we have size $M_r$ of memory in routers $M_r=\log_2(Q)N^2/(2Q^2)$. For example, if $M_r=10^4$ and $N=10^3$, then $Q\approx 15$ and each cluster has about 60 nodes. Typically, resources available for routing are known in advance, so the most difficult problem in application of the mixed approach is to find a good partition of the spanning tree of the original graph into clusters in such a way that global information about clusters can be stored in local memories of routers.

## 3. SIMULATION EXPERIMENTS

The experiments were conducted for randomly generated connected networks with 64 and 256 nodes and varying node degrees ranging from 5 to 20. The following assumptions for our experiments are similar to those used by [5,11].

All network channels are bi-directional and symmetric. The buffer size for each input/output port is 1 flit. Nodes operate asynchronously and memory capacities of nodes are unlimited - this assumption permits us not to consider loss of messages (or packets) and consequent re-transmission due to insufficient memory space. Messages that are blocked from immediately entering the network are queued at the source node (there are no limitations on a queue length). Messages arriving at a destination node are immediately consumed. When multiple messages are waiting for the same channel, the message that has arrived first gets to use the channel first. All messages have lengths equal to 200 flits. The flit size (in bits) is equal to the number of physical channels that compose a link. Each flit is transmitted in a single cycle link ("hop") time- this time represents the basic temporal unit of the model.

Communications arising from the nodes are independent and identically distributed by a 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). In our simulations, we considered *uniform* traffic only - i.e., each node sends a message to any other node with equal probability.

Performance of routing algorithms are measured in terms of the average message latency (average delivery time) and saturation point (highest sustainable message generation rate).

All results have been averaged over 1,000 randomly generated connected graphs, with 150,000 different messages for each graph.

Fig.2. shows the improvement in the performance of the up/down routing with increase in the information available at each node (increase in the size of the local memory at the routers). The global, local and mixed approaches are compared for graph size of (N) 64 nodes and the average degree per node, *d*= 6. The average path length is much smaller for the global approach (since in this case more information about available shortcuts is stored in local memories) as compared to the local one. The mixed approach, as expected, falls somewhere in between. This decrease in the average path length translates into the global approach having a maximum sustainable throughput 5 times larger than the local approach.

Fig. 3 demonstrates the scalability of the global, local and mixed approaches. For these experiments, *d=6*. The global approach shows the maximum scalability: increasing the size of a network 8 times (from 32 to 256), increases the maximum throughput by 500%. For local and mixed approaches, the corresponding increase is between 200% and 400%.

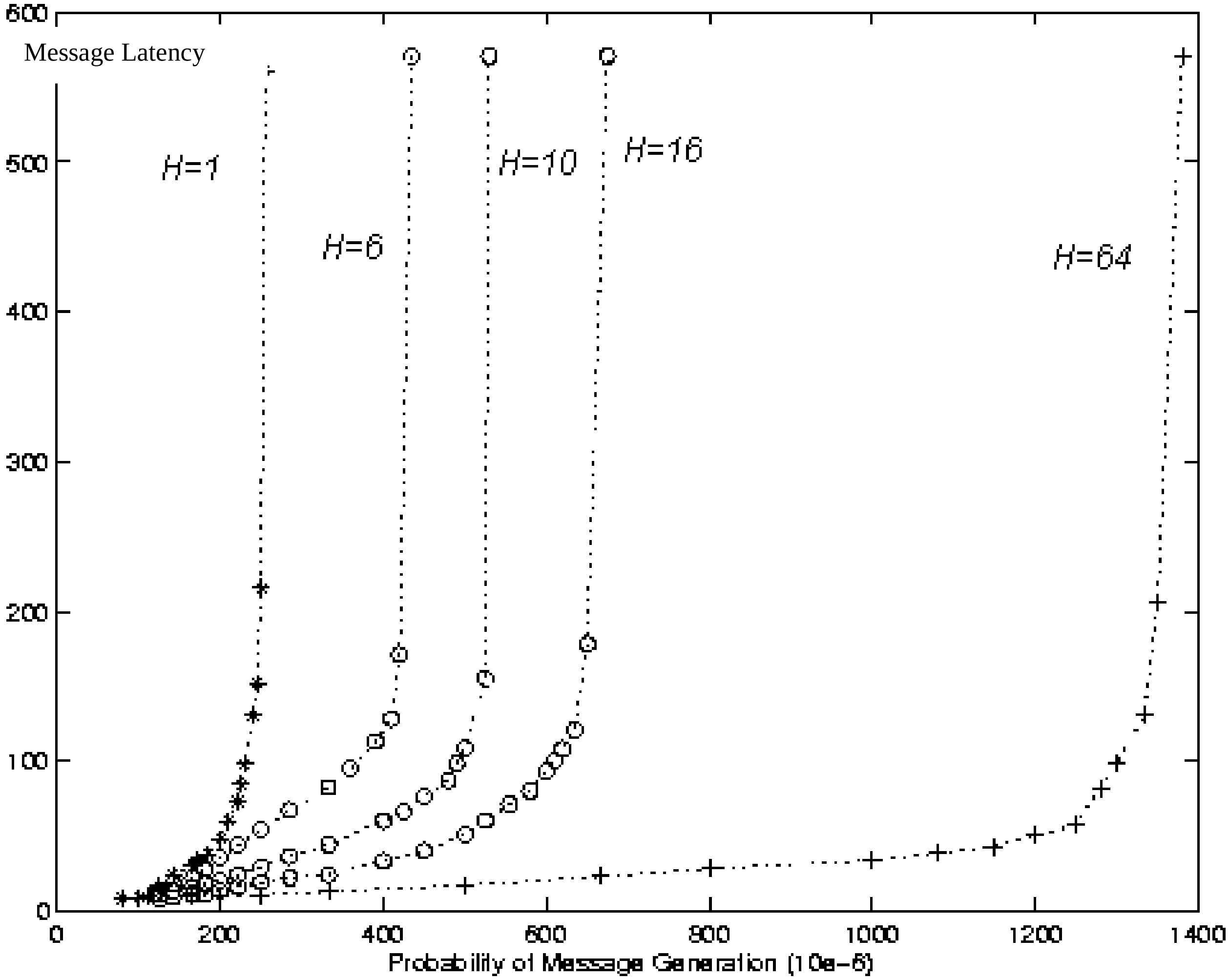

**Fig. 2**. Dependency of average message delivery time (clock cycles) on message generation rate for local (pluses), global (circles) and mixed (circles) approaches. (There are three graphs for the mixed approach for different sizes *H* of clusters).

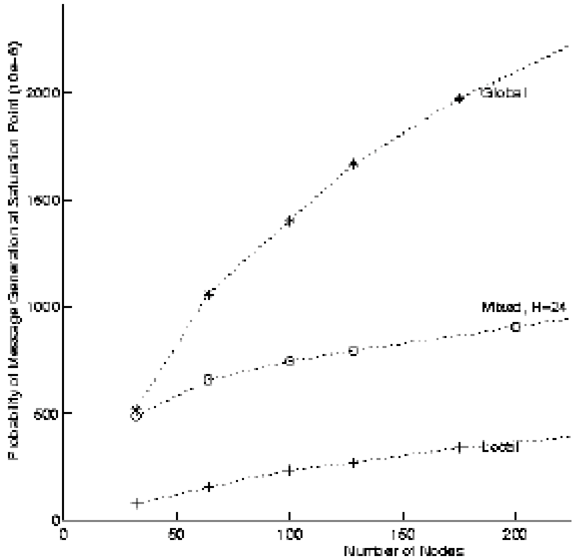

**Fig. 3**. Maximal throughput versus number of nodes for local, global and mixed (fixed cluster size) approaches.

These graphs illustrate a good scalability of the proposed approach. We note that the efficiency of the proposed methods depend on algorithms, used for the constructing of spanning tree and labeling of nodes. In future work, these problems will be addressed.

## 4. EXPANSION OF THE UNICAST ROUTING ALGORITHM FOR THE CASE OF SEVERAL VIRTUAL NETWORKS

Up to now, we have assumed that only one virtual networks is in use. If existing hardware supports the use of several virtual networks (i.e. there are several buffers in the routers for each link), the developed algorithms can be extended. Let us consider using a second virtual network, identical to the first one (each path in this network consists of up and down phases).

Each message starts in the first virtual network. If after going down (i.e. a path in which we are travelling only over equal or increasing labels), going up is needed, then the message is transferred to the second network (but not otherwise). Any path with length not more than 3 is allowed using the described routing strategy (for *k* virtual networks, selected in the similar way, any path up to the length 2*k*-1 is allowed). Since more message paths are available, the load on links can be made more uniform, which allows improved saturation characteristics. It is especially important if the network graph has some long cycles.

For the example shown in Fig.1, path **1-4-3** will be allowed using two virtual networks.

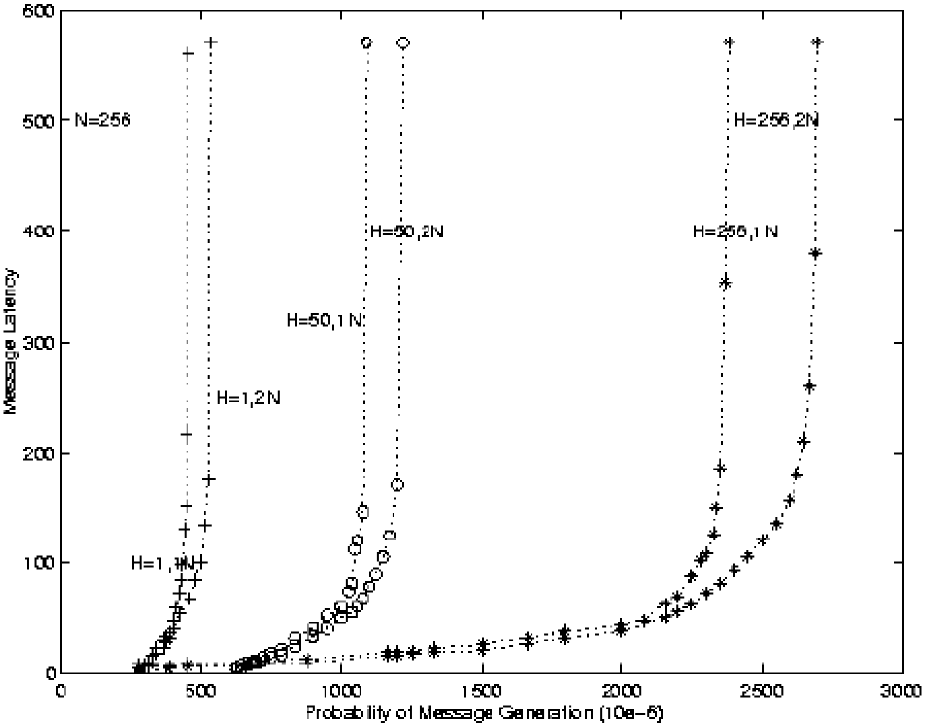

**Fig.4.** Average message delivery time versus packet generation rate for the up/down algorithm, for local (*H*=1), mixed (*H*=50), global (*H*=256) approaches, 1 and 2 (1N,2N) virtual networks (*N*=256).

Fig.4 shows the improvement in the performance of the up/down routing with the addition of a second virtual network. The graph size is 256 nodes and the average degree per node is 6. The corresponding increase in the maximum allowable throughput is around 10%. Fig.4 also illustrates the tradeoff between sizes of local memories $H$ and the throughput. For example, the transition from the block size of $H$=50 to $H$=256 (which requires an increase in the size of the local memory by a factor of 5) results in an increase in the saturation point by 100%.

## 5. CONCLUSIONS

The proposed methods allow performing efficient deadlock-free wormhole routing for networks with irregular topologies. These approaches result in message paths very close to the minimal ones. Complexities of pre-routing stages required for constructing routing tables are proportional to $N^3$ for the global approach and $N^2$ for the local one. The required memory size is of the order $N^2$ for the global approach and $Nlog(N)$ for the local one. A mixed approach was introduced, which combines the advantages of global and local methods. Results of experiments illustrate a good scalability of these techniques.

All developed methods can be easily modified for different deadlock prevention strategies. In this paper we have restricted ourselves to the up/down method. Our methods can also be extended for adaptive routing [3,6,8] and for multicasting [9,15,16].

## 6. ACKNOWLEDGMENTS

Authors would like to thank Prof. Levitin from Boston University for many useful suggestions. This work was supported by the NSF under Grant MIP 9630096 and a Presidential University Graduate Fellowship from Boston University.

## 7. REFERENCES

1. N.J. Boden et al., "Myrinet: A Gybabit per Second Local Area Network," *IEEE Micro*, pp.29-35, 1995.
2. R.V. Boppana and S. Chalasani, "Fault-Tolerant Wormhole Routing Algorithms in Mesh Networks," *IEEE Trans. on Comput.*, vol. 44, pp.848-864, 1995.
3. R.V. Boppana and S. Chalasani, "A Comparison of Adaptive Wormhole Routing Algorithms," *Computer Architecture News*, 21(2), pp. 351-360, May 1993.
4. Y.M. Boura and C.R. Das, "Fault-Tolerant Routing in Mesh Networks," *Proc. of Int. Conf. on Parallel Processing*, pp. 106-109, August 1995.
5. B. Ciciani, M. Colajanni and C. Paolucci, “Performance evaluation of deterministic wormhole routing in *k*-ary *n*-cubes”, *Parallel Computing*, No. 24 , pp 2053-2075, 1998.
6. W.J. 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, April 1997.
7. W. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. on Comput.*, vol. 36, pp.547-553, 1987.
8. J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Trans. on PDS*, vol. 4, pp. 1,320-1,331, 1993.
9. J. Duato, S. Yalamanchili, and L.M. Ni, "Interconnection Networks: An Engineering Approach," Los Alamitos, IEEE CS Press, 1997.
10. E. Fleury and P. Fraigniaud, "A General Theory for Deadlock Avoidance in Wormhole-Routed Networks," *IEEE Trans. on PDS*, vol. 9, pp.626-638, July 1998.
11. C. Glass and L. Ni, "Fault-Tolerant Wormhole Routing in Meshes," *Proc. of Int. Symp. on Fault-Tolerant Computing*, 1993.
12. C. Glass and L. Ni, "The Turn Model for Adaptive Routing," *J. of ACM*, vol.5, pp.874-902, 1994.
13. R.W. Horst, "ServerNet$^{TM}$ Deadlock Avoidance and Fractahedral Topologies", *Proc. of IEEE Int. Parallel Processing Symp.*, pp.274-280, 1996.
14. P. Kermani and L. Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks*, pp.267-286, 1979.
15. R. Kesavan and D. Panda, "Multiple Multicast with Minimised Node Contention on Wormhole k-ary n-cube Networks”, *IEEE Trans. on Parallel Distributed Systems*, Vol. 10, No. 4, April 1999.
16. R. Libeskind-Hadas, D. Mazzoni and R. Rajagopalan, "Tree-Based Multicasting in Wormhole-Based Irregular Topologies," *Proc. of Symp. on Parallel and Distr. Proc.*, pp.244-249, 1998.
17. L. M. Ni and P.K. McKinley, "A Survey of Wormhole Routing Techniques in Directed Networks," *Computer*, vol. 26, pp. 62-76, February 1993.
18. 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.
19. L. Zakrevski and M.G. Karpovsky, "Fault-Tolerant Message Routing for Multiprocessors," *Parallel and Distributed Processing* (Editor J.Rolim), Springer, 1998, pp.714-731.
20. L. Zakrevski, M.G. Karpovsky, "Fault-Tolerant Message Routing in Computer Networks," *Proc. of Int. Conf. on PDPA-99*, 1999.