A NOVEL ROUTING ALGORITHM FOR K-ARY N-CUBE INTERCONNECTION NETWORKS

ERIK D. DEMAINE AND SAMPALLI SRINIVAS
Division of Computing Science
Department of Mathematics, Statistics, and Computing Science
Dalhousie University, Halifax, Nova Scotia B3H 3J5, Canada

ABSTRACT

This paper proposes a novel routing algorithm, called direction-first e-cube, for routing on k-ary n-cube interconnection networks. It is an adaptive, partially-minimal algorithm based on the wormhole routing strategy and effectively extends the basic e-cube technique. It has been proved by a set of theorems that the proposed algorithm is deadlock-, livelock- and starvation-free. In the absence of faults, the algorithm is fully minimal. Even in the presence of faults and network congestion, the number of extra hops required to route a message is minimal. The algorithm is also simple to implement, since it utilizes a small header node of $2\log_2 n + n\log_2 k + 1$ bits. Simulation results are given to validate the proposed algorithm.

Keywords: High-performance parallel architectures, Interconnection networks, k-ary n-cubes, Parallel communications, Routing algorithms.

Short title: A Novel Routing Algorithm

1. Introduction. Distributed memory message passing multiprocessors represent a dominant trend in the design of high-performance parallel computers since they offer the advantages of a powerful computing environment, ease in implementation and good scalability to a large number of processors. In such architectures, each processor has an associated local memory and data is passed between processors through an interconnection network that is typically characterized by static links conforming to a specific topology. Of the various topologies proposed in the literature, the k-ary n-cube has emerged as one of the most popular networks as is evidenced by the numerous parallel computers that have been designed based on this topology (e.g., CM5, Cosmic Cube, iWarp, Ncube 2, Symult 2010 [16]). The motivations for its popularity are due to its advantages of high connectivity, small diameter, and small node degree.

One of the most important issues in the design of such networks is the development of an efficient routing algorithm that provides high throughput and low latency in communication. Furthermore, deadlock-, livelock- and starvation-freedom and fault tolerance are natural requirements of any routing strategy.

Despite the popularity of the k-ary n-cube, few routing algorithms exist for it. As communication is often a bottleneck, a supercomputer degrades significantly in performance without an efficient algorithm to guide messages. Even worse, the network may come to a complete halt if deadlocks occur. The few algorithms that exist for routing on k-ary n-cubes such as e-cube routing [15] and more recent techniques including west-first [9], and adaptive minimal routing [8] do not effectively address all important issues, and do not work for all values of k and n.

This paper proposes a novel algorithm, called the direction-first e-cube routing algorithm, for routing on k-ary n-cubes[1]. It is an adaptive, partially-minimal algorithm based on the wormhole routing strategy and effectively extends the basic e-cube routing technique.

It has been proved by a set of theorems that the proposed algorithm is deadlock-, livelock- and starvation-free. It does not make use of computationally expensive virtual channels to avoid deadlocks unlike the previously proposed algorithms. In the absence of any faults, the algorithm is fully minimal. Even in the presence of faults and network congestion, the number of extra hops required is minimal. This is validated by the simulation results which indicate that the average latency (the number of extra hops) scales logarithmically with the network size and that the presence of faults does not significantly affect latency. The algorithm is simple to implement, since it utilizes a small header node of size $2\log_2 n + n\log_2 k + 1$.

The rest of the paper is organized as follows: Section 2 introduces the important issues of routing algorithms on interconnection networks; Section 3 outlines the definition and features of k-ary n-cube; Section 4 surveys previously proposed routing algorithms, e-cube, double e-cube and west-first routing; Section 5 proposes the direction-first e-cube routing algorithm for k-ary n-cubes; Section 6 describes the experiments carried out and the simulation results obtained in order to validate the proposed approach. Section 7 draws concluding remarks.

[1] A version of this paper is accepted for presentation at the High Performance Computing Symposium '95, Montreal, Canada, July 1995 [7]

2. Routing Issues. In this section, we discuss the important issues to be considered while designing routing algorithms.

An *interconnection network* is a connected undirected graph whose nodes represent processors and links represent *channels* by which two nodes can communicate with each other [16]. Using this, messages can be sent among processors by "forwarding" between intermediate nodes to reach the destination using the channels. To accomplish this, a *routing algorithm* is required, which *routes* a message received at a particular node to the next node that would (most likely) bring the message closer to the destination.

Many different techniques have been adopted to implement routing algorithms, popular among them include minimal path, adaptive, wormhole routing, and packet switching. *Minimal path* algorithms [15] will only take *optimal* or *minimal* routes between nodes. Minimal or near minimal paths are preferable because the latency of messages is kept low. However, to achieve fault tolerance, some routing algorithms will backtrack, taking non-minimal paths. In *adaptive* algorithms [9] the route that the algorithm takes can vary, depending on the current state of the network. This is a necessity for fault tolerance. If some links in the network are faulty, the routing algorithm must avoid them or else the message will get lost. Consequently, most algorithms are adaptive. In a *packet switching* algorithm, the whole message is sent between nodes [8]. This is a communication intensive task and for this reason *wormhole routing* was introduced [6]. In this type of algorithm, the message is split into a series of fixed size *flits.* These flits are sent separately, in order, through the network. This drastically increases performance of message passing. Only wormhole routing algorithms are discussed here, because most can be adapted into packet switching algorithms.

An important issue in designing any routing algorithm is to minimize the network *latency* or delay. Even though the diameter of the specific interconnection network determines the latency, the routing algorithm must be effective in reducing the latency in the network for different communication patterns.

Another important issue of routing is *deadlock.* Deadlock occurs when, due to interdependencies in the network, the system comes to a halt and no messages can be sent [6, 12]. As such, guaranteeing deadlock freedom is also a problem when designing a routing technique. The routing algorithm must have a way to avoid deadlock or a way to recover from it. A good description of the different deadlock situations and their avoidance techniques can be found in [10]. There are two main ways to overcome deadlock. The first is to increase the buffer size in each node to some limit that can never be reached. Because messages using wormhole routing leave a "trail" of flits behind them, this option can only be employed in packet switching algorithms. It is imperative that the routing algorithm must be designed so that it is deadlock free. This makes the routing algorithms harder to formulate. Wormhole routing is particularly susceptible to deadlock, and precautions must be made.

Closely related to the issue of deadlock are livelock- and starvation-freedom [8]. The livelock problem arises whenever a message can be denied getting delivered to its destination forever. A node is said to suffer *starvation* when it cannot place a packet on the network for transmission. The policy adopted by the routing technique must prevent livelocks and starvation in the network.

The routing algorithm must also provide fault-tolerance. Typically, the network is made fault-tolerant by one of two methods: partitioning (graceful degradation) [1] or the allocation of spares [1, 2]. A network partition is the same type of network with fewer nodes. The work is then distributed among these processors. This type of fault tolerance drastically increases the total time to complete a task [1]. The method of using spares will not degrade performance as in the previous technique, but when there are not enough spares something must be done. To maintain the network in such a situation, the routing algorithm needs to be fault tolerant by traveling around the faults.

Of course, the best routing algorithm would be minimal, adaptive, fault tolerant to any degree, deadlock free, and always find efficient paths, uniformly distributing the messages to provide the lowest possible latency. This is not possible, however, so some compromises must be made. For example, some routing algorithms are only partially minimal, taking a few extra hops at times to get to the destination, but provide full fault tolerance if a path exists [14].

3. k-ary n-cube. A k-ary n-cube is a very popular interconnection network with $n \cdot k^n$ nodes, where each processor is given a n-digit address, where each digit d satisfies $0 \leq d < n$ [4]. Each node whose address is $i_{n-1}i_{n-2}\ldots i_1 i_0$ is connected to the nodes

$$(3.1) \qquad i_{n-1}i_{n-2}\ldots i_{j+1}(i_j \pm 1 \bmod n)i_{j-1}\ldots i_1 i_0 \text{ for all } 0 \leq j < n.$$

There are many properties that illustrate the benefits of k-ary n-cubes. First, it has high connectivity, offering a built-in amount of fault-tolerance. It has diameter that only increases logarithmically (to the base k), which gives an automatic low latency of messages. Its small degree of nodes makes extendibility an easy option.

Some common subsets of a k-ary n-cube are *hypercubes* and *two-dimensional tori*. A n-dimensional hypercube is a binary ($k = 2$) n-cube [3, 9, 11], whereas a k by k torus is a k-ary 2-cube [16]. A 4-dimensional hypercube, 4 by 4 tori, and 3-ary 3-cube are shown in Fig. 1, 2, and 3, respectively.

4. Previous Approaches. The following algorithms are examples of deadlock free and fault-tolerance routing: e-cube [9, 14, 15], double e-cube and west-first [9, 15]. They have disadvantages: The first two are non-adaptive and not fault tolerant, but can be used on hypercubes, whereas the third is fault tolerant, but can only be use on tori. None of these techniques work for k-ary n-cubes in general. To overcome all these disadvantages, direction-first e-cube routing combines the last two algorithms to remove the limitations. The techniques of the sample algorithms are helpful in understanding how direction-first e-cube works.

4.1. e-cube routing. e-cube is a non-adaptive, minimal, deadlock free routing algorithm for n-dimensional hypercubes [9, 14, 15]. It is one of the simplest wormhole routing algorithms that is deadlock free. When a message is received at an intermediate node, the node and destination node's numbering is compared using a bitwise exclusive-or (XOR) operation. Any dimensions (bits) that have a 1 bit need to be "corrected," that is, the message must traverse in that dimension to reach the destination. e-cube routing chooses the least significant bit that is set to 1, and sends the message in the corresponding direction, waiting for the link or node to be free (not busy). Proceeding in this way, e-cube stops routing when there are no more bits that need be corrected. At this point, the destination has been reached.

e-cube has been proven to be deadlock free [9, 14, 15]. However, e-cube provides no fault tolerance. If any link or node in the network goes faulty, and a message attempts to traverse to it, e-cube will then wait for the link or node to be free. Assuming no on-line repairs can be made, the message must wait forever, bringing the network to a full halt. This is the main problem with e-cube routing, besides the fact that it does not work on k-ary n-cubes.

A sample e-cube routing, from node 0000 to 0111 is shown in Fig. 1.

4.2. Double e-cube routing. As an intermediate step toward creating direction-first e-cube, we propose in this section a non-adaptive, minimal routing algorithm for n-dimensional hypercubes for $n > 1$, similar to e-cube. When a message is received at a node, two dimensions (the two least significant) that need to be traversed are chosen. The message is sent to along the first with special instructions to send along the second dimension selected, waiting if necessary. As before, this is repeated until either the destination

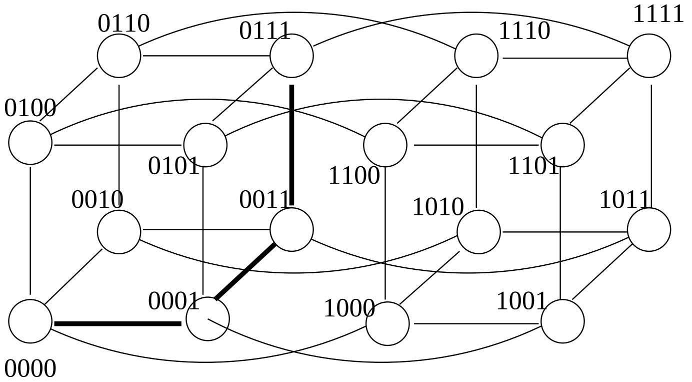

FIG. 1. *A 4-D hypercube with an e-cube routing from 0000 to 0111.*

is reached or only one incorrect dimension exists. In the latter case, the message is simply sent to the destination.

Lemma: Double e-cube is deadlock free for a n-dimensional hypercube where $n > 1$.

Proof: Double e-cube is in fact the same as e-cube, since two dimensions are chosen first, then a routing is made to correct these dimensions. The choice between which dimension first is consistent, and thus the algorithm is the same as e-cube. Since e-cube is deadlock free, double e-cube is also.

□

Although double e-cube is indeed the same as e-cube, the different approach is required for the construction of direction-first e-cube.

4.3. West-first routing. West-first routing is an adaptive, partially minimal, deadlock free routing algorithm for a two-dimensional torus and invented by Glass and Ni [9, 15]. In this routing technique, all *west* (a consistent direction) routing is done first, followed by other direction routing. Once any direction other than west is traversed, west cannot be done again to avoid deadlock. When the west phase is completed, the message is adaptively sent, avoiding faults, toward the destination in a close-to-minimal path. The wraparounds in the torus are only used in the first hop of the routing. Sample routings are given in Fig. 2. If no faults exist in the network, all paths are minimal. Note that the direction "west" can be chosen in any way, so long as the decision does not change during execution, extending to what we call a *direction-first routing.* The low latency of messages and fault tolerance are the main highlights, providing efficient routing for a torus.

Although west-first routing appears much better than e-cube and double e-cube, it is only designed for tori, which is not very practical in many applications. It is this motivation that led to the proposal of the direction-first e-cube routing algorithm.

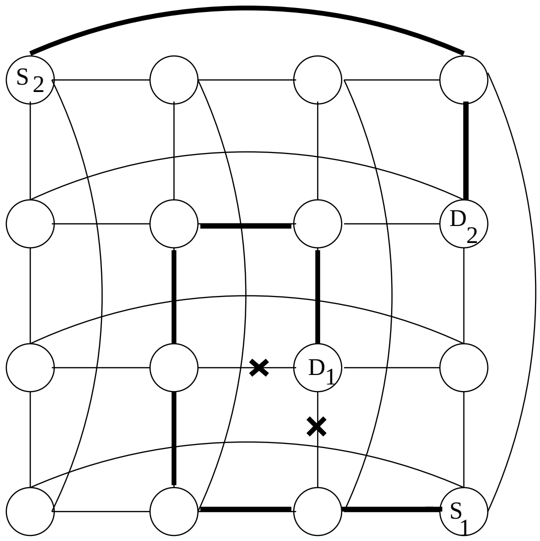

FIG. 2. *A 4 by 4 torus with west-first routing.*

5. Direction-first e-cube. In this section, we describe the proposed routing algorithm.

As mentioned before, few algorithms exist for k-ary n-cubes, and those that have accomplished this are generally neither fault tolerant nor deadlock free. To solve this problem, we propose a new adaptive, partially-minimal algorithm for k-ary n-cubes, $k > 1$, which combines direction-first routing and double e-cube. The strategy adopted in this algorithm consists of first selecting the torus defined by the two dimensions corresponding to the least significant differing bits between the source and destination nodes. Then the direction-first routing technique is employed to correct these dimensions using the designated torus. This is done by traversing in any particular direction first, and then traversing other directions adaptively. The wraparound links are optionally used in the first hop. The above two steps are repeated until the destination is reached.

The following is the pseudo-code description of the algorithm.

- *Input*: k-ary n-cube with local channel information; the source (current) node $s_{n-1}...s_0$; the destination node $d_{n-1}...d_0$; the direction; a flag f which equals 1 if and only if the direction has been traversed in the previous dimensions x, y.
- *Output*: The next node to traverse $a_{n-1}...a_0$; an updated f, x, and y. This algorithm should then be applied with $s = a$ on node a.
- *Initial values:* The first time direction-first e-cube is used for a particular message, use $f = x = y = 0$.

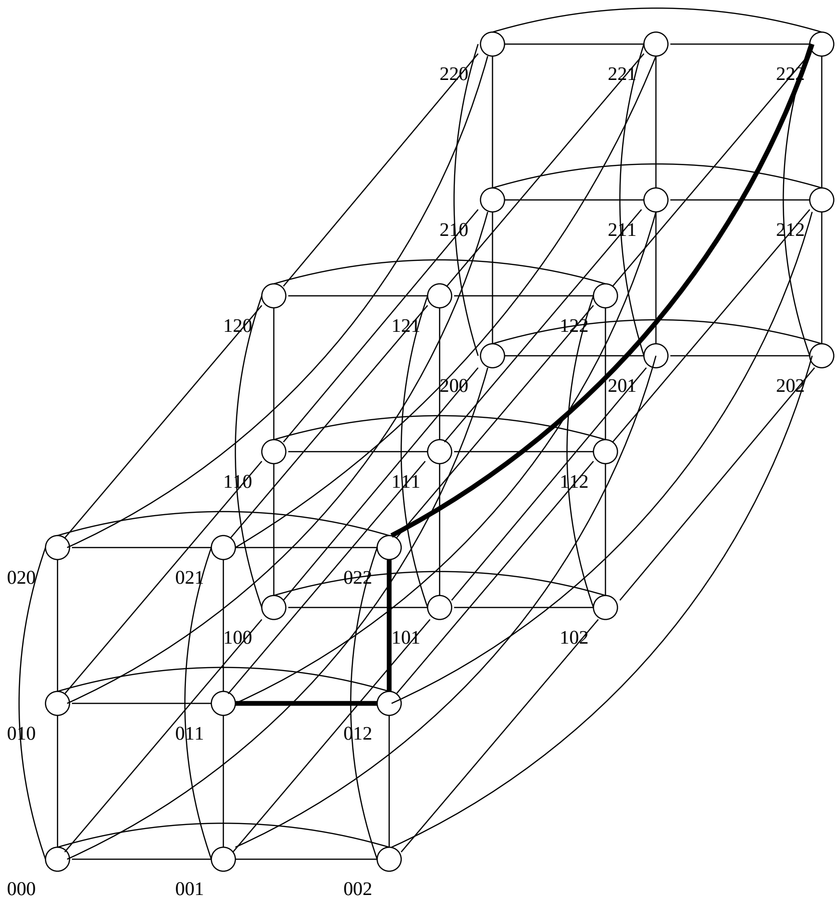

FIG. 3. *A 3-ary 3-cube with a direction-first e-cube routing from 011 to 222.*

- *Algorithm*:
  1. (double e-cube phase) Let i, j be dimensions such that
     (a) $s_i \neq d_i$ and $s_j \neq d_j$. (i and j are incorrect dimensions.)
     (b) If $0 \leq l \leq n-1$, $l \neq i$, $l \neq j$, and $s_l \neq d_l$, then $i < j < l$. (i and j are minimal.)

     If no such j exists, choose it to be y if $y \neq i$ or else a random dimension. If no such i exists either, the destination has been reached (stop).
  2. If $i \neq x$ or $j \neq y$, then set $f \leftarrow 0$, $x \leftarrow i$, $y \leftarrow j$.
  3. Employ direction-first routing given $f, (s_i, s_j)$, and (d_i, d_j).
  4. If the direction was used, set $f \leftarrow 1$.

Fig. 3 shows a 3-ary 3-cube with a routing from 011 to 222 using west-first e-cube. The first two differing dimensions are 0 and 1 (the two least significant ones). We take 0 and 1 to be the X and Y axes, respectively, defining a 2-D torus. The west direction is negative on the X, so no west routing need be done. Suppose the algorithm adaptively takes the route 011-012-022 which corrects dimensions 0 and 1 (any such path may be chosen adaptively). Then dimension 2 is the last one to fix. Assume that dimension 1 is chosen as the other (Y) dimension. The message can take a wraparound link in the west direction, since this is the first traversal made for this torus. The destination is then reached by the path 022-222.

The following theorems validate the important characteristics of the proposed algorithm.

THEOREM 5.1. *Direction-first e-cube is deadlock free for k-ary n-cubes with $k \geq 2$.*

Proof. For each torus that the algorithm uses, the routing is deadlock free since direction-first is guaranteed deadlock free [9, 15]. Thus, we can view the k-ary n-cube as a n-dimensional hypercube and use double e-cube routing with the same degree of deadlock. We can do this because double e-cube is deadlock free on any side of the cube and traversing through one of these is a direction-first routing which is also deadlock free. Since double e-cube is deadlock free for hypercubes (Lemma), we have that direction-first e-cube is also deadlock free. ∎

THEOREM 5.2. *Direction-first e-cube is livelock-free.*

Proof. First of all, note that direction-first routing is livelock free, since it does not apply misrouting (so that a message returns to a location it has previously traversed). Since all routing in direction-first e-cube uses this method, it is also livelock-free. ∎

COROLLARY 5.3. *Direction-first e-cube is starvation-free.*

Proof. Since the algorithm is deadlock- and livelock-free, for any given message, the individual buffers in the channel will not be permanently reserved. By giving new messages higher priorities, it can be ensured that each message will be put on the link within a finite time. ∎

THEOREM 5.4. *The required size of the header node in bits, including the destination node, is $2\log_2 n + n\log_2 k + 1$.*

Proof. There are four inputs to the algorithm: destination node, x, y and f. Since there are k^n possible values for the destination node, $\log_2 k^n$ bits are required to store it. Similarly, there are n possible values for x (a dimension), so x and y each take $\log_2 n$ bits. f is only a flag, and fits in one bit. Summing these results gives a header node size of

$$2\log_2 n + \log_2 k^n + 1, \tag{5.1}$$

$$i.e., 2\log_2 n + n\log_2 k + 1. \tag{5.2}$$

∎

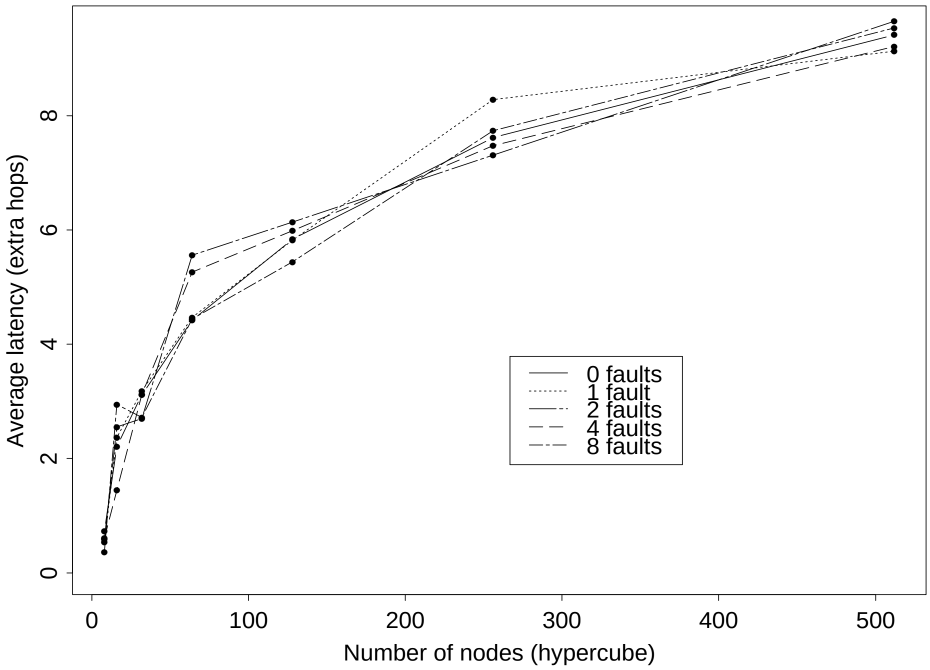

FIG. 4. *Average latency figures for a k-ary n-cube.*
Message generating probability of a node per time step: 20.
Number of time steps: 30.

Direction-first e-cube is seen to remove most of the disadvantages of west-first and double e-cube routing. It is simple, general, has low latency, adaptive, and very fault tolerant. Combinations are often the best type of routing algorithms.

6. Simulation Results. A simulation program was written for direction-first e-cube using CSIM, a process-oriented simulation language [17] that can be effectively used to simulate parallel environments. The simulation was carried out for hypercubes of different dimensionalities. A static injection model was used, that is, time is discrete and operations are synchronized to execute only at certain time intervals. In each run, the average latency (extra hops) was obtained for messages sent between random source and destination nodes. In each time step, every processor decides if it will generate a message using a random number based on a probability of 20%. The number of time steps run was 30.

Fig. 4 summarizes the experimental results.

It can be observed that the latency of messages in direction-first e-cube increases logarithmicly with the size of the network. This is a very important feature, since parallel machines are continually growing. The results also show that the number of faults in the network effects the latency of messages very little.

6.1. Performance in the Presence of Faults. An interesting feature of the graph is that the algorithm in some cases performs better when more faults are present in the network. This is entirely possible – when faulty links are added, the adaptive algorithm (direction-first routing) will attempt to route around these links. This may lead to faster paths, and perhaps less traffic. For example, if a particular link is used for many routes (causing high traffic), then the routing algorithm would most likely perform much better if that link was faulty.

It may be noted that a one fault network performs worst for 256 nodes, but best for 512 nodes, comparing to other numbers of faults. This is due to the randomness in the network. In these tests, the random source was not fixed, yielding better simulation of a real supercomputer environment. It can be seen that, taking into account some variability, the latency is about the same for the various number of faults.

7. Conclusion. The k-ary n-cube is an increasingly popular network for fast communication between nodes on a distributed-memory parallel system. Recently, however, few routing algorithms have been proposed to fully utilize its potential. Often these algorithms do not work in general and provide constraints on k and n.

To alleviate this problem we have presented a new wormhole routing algorithm, called direction-first e-cube, designed for k-ary n-cube interconnection networks. The algorithm is simple; most others use virtual channels to avoid deadlock [5, 13]. The adaptiveness of the algorithm will route around many faults. Without the presence of faults, the algorithm is fully minimal, and only a few extra hops are required when faults are encountered. Because direction-first e-cube is for a general class of interconnection networks, it works for all subsets such as hypercubes and tori. The simulation results show the routing algorithm scales logarithmically with the size of the network, and that faults have little effect on message latency.

We suggest further development in hybrid routing strategies. Often such methods combine the best features of existing techniques. They offer efficient approaches to more general problems.

Acknowledgment. The authors would like to gratefully acknowledge the support by the Natural Sciences and Engineering Research Council

(NSERC) of Canada under Grant OGP0138637 and the Research Development Fund for the Sciences, Dalhousie University, under Grant RDFS R1-24-3770.

REFERENCES

[1] M. ALAM AND R. MELHEM, "Routing in modular fault-tolerant multiprocessor systems", in *Proc. 22nd Int'l Symp. on Fault- Tolerant computing*, July 1992, pp. 185-193.

[2] J. BRUCK, R. CYPHER, AND C. HO, "Fault-tolerant meshes and hypercubes with minimal number of spares", *IEEE Trans. Computers*, 42 (Sept. 1993), pp. 1089-1104.

[3] M. CHEN AND K. G. SHIN,"Depth-first search approach for fault-tolerant in hypercube multicomputers", *IEEE Trans. Parallel and Distributed Systems*, 1 (Apr. 1990), pp. 152-159.

[4] W. DALLY, "Performance analysis of k-ary n-cube interconnection networks", *IEEE Trans. Computers* 39(6), June 1990, pp. 775-785.

[5] ———, "Virtual-channel flow control", *IEEE Trans. Parallel and Distributed Systems*, 3 (Mar. 1992), pp. 194-205.

[6] W. DALLY AND C. SEITZ, "Deadlock-free message routing in multiprocessor interconnection networks", *IEEE Trans. Computers*, C-36 (May 1987), pp. 547-553.

[7] E. DEMAINE AND S. SRINIVAS, "Direction-First e-cube: A New Routing Algorithm for k-ary n-cube Networks," in *High Performance Computing Symposium* (July 1995), Montreal, Canada.

[8] S. A. FELPERIN, L. GRAVANO, G. D. PIFARRE AND J.L.C.SANZ, "Routing techniques for massively parallel communication", *Proc. IEEE*, 79 (Apr. 1991), pp. 488-502.

[9] P. T. GAUGHAN AND S. YALAMANCHILI,"Adaptive routing protocols for hypercube interconnection networks", *IEEE Computer*, 26 (May 1993), pp. 12-23.

[10] K. GUNTHER, "Prevention of deadlocks in packet switched data transport system", *IEEE Trans. Commun.*, COM-29 (Apr. 1981).

[11] S. LATIFI, "Combinatorial analysis of the fault-diameter of the hypercube", *IEEE Trans. Computers*, 42 (Jan. 1993), pp. 27-33.

[12] Q. LI, "Minimum deadlock-free message routing restrictions in binary hypercubes", *Jl. of Parallel and Distributed Computing*, 15 (1992), pp. 153-159.

[13] D. H. LINDER AND J. C. HARDEN, "An adaptive and fault-tolerant wormhole routing strategy for k-ary n-cubes", *IEEE Trans. Computers*, 40 (Jan. 1991), pp. 2-12.

[14] J. KIM AND K. G. SHIN, "Deadlock-free fault-tolerant routing in injured hypercubes", *IEEE Trans. Computers*, 42 (Sept. 1993), pp. 1078-1088.

[15] L. M. NI AND P. K. MCKINLEY, "A survey of wormhole routing techniques in direct networks", *IEEE Computer*, 26 (Feb. 1993), pp. 62-79.

[16] D. A. REED AND D. C. GRUNWALD,"The performance of multicomputer interconnection networks", *IEEE Computer*, June 1987, pp. 63-73.

[17] H. SCHWETMAN, *CSIM Reference Manual*, Rev. 16, Microelectronics and Computer Technology Corp., 1992.