Flexible, Fault-Tolerant Routing Criteria for Circuit-Switched Hypercubes

BY

Ge-Ming Chiu, Suresh Chalasani and C.S. Raghavendra

CEng Tech Report 90-30

Electrical Engineering Systems

University of Southern California

Los Angeles, CA. 90089-0781

Flexible, Fault-Tolerant Routing Criteria for Circuit-Switched Hypercubes *

Ge-Ming Chiu, Suresh Chalasani † and C. S. Raghavendra

Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089-0781

Abstract

Circuit-switched communication has been increasingly used in hypercube multiprocessors [1, 5]. Reserve-and-hold strategy is commonly used to establish a circuit-switched path between a pair of nodes in the hypercube. With this strategy, a routing algorithm has to be designed with sufficient care to avoid deadlocks. A commonly-used, deadlock-free routing algorithm for establishing a circuit-switched shortest path between any two nodes in a hypercube is the *e*-cube routing algorithm. The *e*-cube routing algorithm allows only one shortest path between each source-destination pair, and, hence, does not utilize the flexibility provided by the hypercube. In this paper, we propose a new set of routing criteria for circuit-switched hypercubes that exploit the flexibility provided by the hypercube. Our routing criteria are provably deadlock-free and route messages along shortest paths. The number of shortest paths allowed by our routing criteria is more than one for most source-destination pairs. We show that the flexibility provided by our routing criteria can be used to limit the negative effects due to component-failures. We derive the exact number of disrupted source-destination pairs in the presence of a single faulty link or a single faulty node. We show that these numbers can be minimized using the relabeling techniques proposed in this paper. We evaluate the performance of our routing criteria, with respect to the *e*-cube routing algorithm, using simulation results. We show that our criteria, if used effectively, lead to an improvement in performance over the *e*-cube routing strategy.

*This research is supported in part by the Presidential Young Investigator Award No. MIP-8452003.
†This author's research is supported by an IBM Graduate Fellowship.

I Introduction

Recently, considerable research effort has been directed towards hypercube multiprocessors [2, 3, 9]. Hypercube multiprocessors, owing to their regular structure and low diameter, are well suited for parallel processing [6]. In addition, hypercube systems offer high connectivity and exhibit ability to tolerate faults in the system. Several commercially available hypercube multiprocessors, such as Intel's iPSC-860 and iPSC-2 machines, have been built in recent years [5].

Circuit-switched mode of operation is often used in hypercube systems for effective transmission of long messages without incurring large overhead. In circuit-switched mode, communication resources required for the transfer of a message are reserved before the actual transfer of the message begins. Links that interconnect the processing nodes are the primary communication resources in a hypercube. A link, once reserved for a source-destination (s-d) pair, cannot be used by other source-destination (s-d) pairs until it is released after message transmission is completed. There are two different policies that are generally used for establishing a circuit-switched path between a source and a destination. A commonly-used policy to establish a circuit-switched path between two nodes in the system is the *reserve-and-hold* policy as per which the establishment of the path is achieved by reserving one link at a time. A partial (incomplete) path between an s-d pair may have to wait at an intermediate node, if the next link along the path is reserved for another s-d pair. In another policy, known as the *backtrack-and-retry* policy, a partial path, if it can not be advanced at an intermediate node due to the unavailability of the required link(s), is released and a fresh retrial for establishing the path begins at the source node. Backtrack-and-retry policy may lead to indefinite starvation of some requests and does not perform very well as the cube-size increases [7]. Many commercially available hypercube multicomputers that use circuit-switched communication, such as Intel's iPSC-2, adopt reserve-and-hold strategy. For the rest of this paper, we assume that reserve-and-hold policy is used for establishing a circuit-switched path. A *routing algorithm* precisely states how the links in the network are to be reserved to establish a circuit-switched path between a source and a destination.

In a system that uses the reserve-and-hold policy to establish circuit-switched paths, *deadlocks* may occur unless routing algorithms are designed with sufficient care. A deadlocked state corresponds to a set of s-d pairs such that each s-d pair is waiting for link(s) that are currently reserved for other s-d pairs in the same set. In other words, advancement of no partial path corresponding to an s-d pair in a deadlocked state is possible unless some exceptional action is taken. Thus there is a need to use routing algorithms that are provably deadlock free in multiprocessor systems. In a circuit-

switched environment, link-utilization and, hence, the performance of the system can be enhanced if the routing algorithm always establishes a shortest path for any given s-d pair. In this paper, we concern ourselves with deadlock-free algorithms that route messages along shortest paths in a hypercube multiprocessor.

Deadlock-free algorithms have been studied in [4], with respect to wormhole routing. A general method using the concept of virtual channels for constructing deadlock-free routing algorithms are presented. However, these routing algorithms are designed so as to avoid deadlocks caused by limited buffer resources instead of communication links.

A commonly used routing algorithm in hypercube systems is the *e*-cube routing algorithm. This algorithm reserves links required for an s-d pair in a strictly increasing order of dimensions in which the binary representations of the source and destination nodes differ. It can be easily proved that the *e*-cube algorithm establishes a shortest path for every s-d pair without introducing deadlocks in the system. However, the *e*-cube algorithm establishes only one unique path for each s-d pair although there may be other alternate paths. That is, the *e*-cube routing algorithm does not utilize the complete flexibility provided by the hypercube.

In this paper, we will propose a routing criterion, as per which flexible routing algorithms can be designed which meet the two requirements, namely, freedom from deadlocks and routing messages along shortest paths. The routing criterion that we propose in this paper divides the set of dimensions in which the source and the destination differ into two sets. The first set, known as the set of up-transitions, contains the dimensions in which the source address has zeros and the destination address has ones. The second set, known as the set of down-transitions, contains the dimensions in which the source address has ones and the destination address has zeros. Our routing criterion, to establish a circuit-switched path between a source and a destination, can now be informally described as shown below:

1. A required link corresponding to a dimension in the set of *up-transitions* can be traversed in any order.

2. A required link corresponding to a dimension in the set of *down-transitions* can be traversed only if all required links of lower dimensions are already traversed.

For example, let us consider the source node $5 = (0, 1, 0, 1)$ and the destination node $10 = (1, 0, 1, 0)$ in a 4-cube. The *e*-cube routing algorithm always routes messages using the path $5 \rightarrow 4 \rightarrow 6 \rightarrow 2 \rightarrow 10$

between nodes 5 and 10. Our routing criterion allows several paths between source node 5 and destination node 10. Two such paths are $5 \rightarrow 7 \rightarrow 6 \rightarrow 14 \rightarrow 10$ and $5 \rightarrow 13 \rightarrow 15 \rightarrow 14 \rightarrow 10$. However, the path $5 \rightarrow 13 \rightarrow 12 \rightarrow 8 \rightarrow 10$ is not allowed between nodes 5 and 10 by our criterion. An interesting feature of our routing criterion is that it utilizes the flexibility provided by the hypercube and, hence, allows more than one shortest path for many s-d pairs. In fact, the *e*-cube algorithm can be derived as a special case of our routing criterion. Further, any algorithm that complies with our routing criterion can be implemented in a distributed fashion with very little hardware-overhead.

We first investigate the faut-tolerance provided by our routing criterion. It is shown that the ability to tolerate faults is significantly improved if our routing criterion is used. Moreover, we show that the negative effects due to component failures can be significantly reduced by relabeling the dimensions of the hypercube. Fault-tolerance issue is also studied from the point of view of number of node-disjoint paths provided by our criterion between an s-d pair. We also study how the additional flexibility allowed by our routing criterion can be used to improve system performance in terms of average setup time for a message-path. We show that our criterion, if used effectively, can improve performance over the *e*-cube strategy for most of the operational range of the system. Although we derive our results with respect to circuit-switched mode of communication, these results can be applied to other modes of communication, such as wormhole routing, as well.

The rest of the paper is organized as follows. In Section II, we present the notations and the definitions used in this paper; in this section, we formally introduce the concept of a deadlocked state. Our routing criteria are introduced in Section III. In this section, we also prove that any routing algorithm that adopts our routing criteria is deadlock-free. Fault-tolerance provided by our routing algorithms is analyzed in Section IV. We study the performance of the proposed set of routing algorithms in Section V. Section VI concludes this paper.

II Preliminary Definitions and Notations

In this section, we introduce the preliminary definitions and notations. We consider an n-dimensional hypercube with $N = 2^n$ processors (or nodes). Each processor X, for $0 \leq X \leq N-1$, can be represented by a sequence of n binary digits $(x_{n-1}, ..., x_0)$ where $x_i \in \{0,1\}$ for $0 \leq i \leq n-1$. The bit x_i, $0 \leq i \leq n-1$, is sometimes called the i-dimnesional bit of node X, and x_0 is the least significant bit. Two processors are connected by a pair of links, one in each direction, if and only if the binary representations of their labels differ in exactly one bit. Hence each node would have n neighbors. A

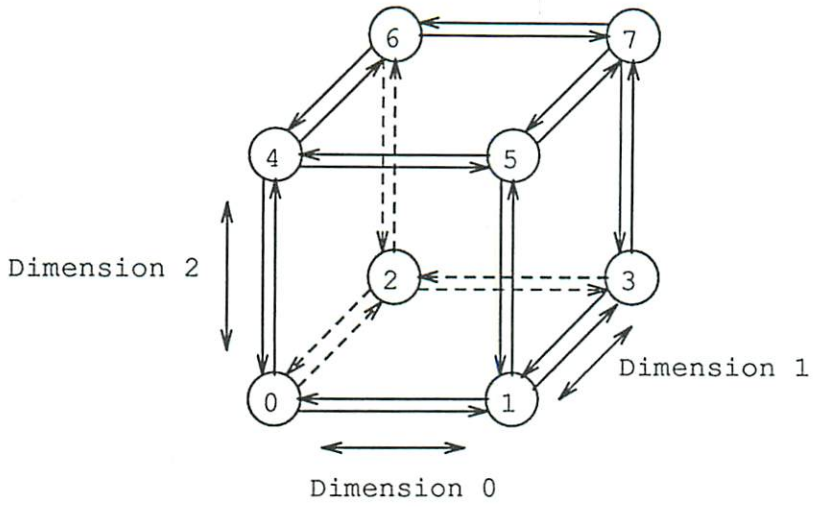

Figure 1: 3-dimensional hypercube with circuit-switched communication mode.

link is said to be an i-dimensional link, $0 \leq i \leq n-1$, if it connects two nodes that differ in their i-dimensional bits. A link of dimension i from node X is sometimes denoted as X_i. Further, an outgoing i-dimensional link at node $X = (x_{n-1}, ..., x_0)$ is normally denoted as $X_i \uparrow$ if bit $x_i = 0$. Similarly, this link is denoted as $X_i \downarrow$ if $x_i = 1$. These notations are used for representing link directions. In general, a link is called an *up*-link if it can be represented as $X_i \uparrow$ for some X and i, where $0 \leq X \leq N-1$ and $0 \leq i \leq n-1$. On the other hand, it is called a *down*-link if it can be represented as $X_i \downarrow$ for some X and i, where $0 \leq X \leq N-1$ and $0 \leq i \leq n-1$. Figure 1 shows a 3-dimensional hypercube. For example, the link from node $5 = (1,0,1)$ to node $7 = (1,1,1)$ is an up-link and is denoted as $5_1 \uparrow$, and the link from node $7 = (1,1,1)$ to node $5 = (1,0,1)$ is a down-link and is denoted as $7_1 \downarrow$. A path between two nodes is sometimes represented as a sequence of links for convenience. For example, in Figure 1, the path from node $4 = (1,0,0)$ to node $2 = (0,1,0)$, with node $6 = (1,1,0)$ as the only intermediate node, is represented using $4_1 \uparrow 6_2 \downarrow$.

As described earlier, in the circuit-switched mode of operation, a path must first be set up before a source node is allowed to transmit a message to a destination node. A commonly-used strategy to establish a circuit-switched path between two nodes is the *reserve-and-hold* strategy as per which the establishment of a path is achieved by reserving one link at a time. A link, once reserved, cannot be used by other source-destination (s-d) pairs until it is released after message transmission is completed. A partial (incomplete) path corresponding to an s-d pair of nodes may have to wait at an intermediate node, if the next required link along the path is reserved for another s-d pair. A path, whether partial or complete, consists of all the links that have already been reserved

for this path.

An outgoing link from any intermediate node I is said to be *legal*, with respect to source S and destination D, if the link can be used to advance the path from S to D as per the routing algorithm under consideration. In this paper, we concern ourselves with routing algorithms for which the following two criteria can be satisfied:

1. The routing algorithms do not cause any deadlocks in the system, and
2. Each path established from a source to a destination is a shortest path.

Given a set of m, $m > 0$, s-d pairs of nodes (S_i, D_i), $0 \leq i \leq m-1$, let P_i denote the partial (incomplete) path established from S_i to an intermediate node I_i. Further, let L_i represent the set of legal outgoing links from I_i with D_i as the destination node. A deadlock situation is defined as follows:

Definition 1 *The set of source-destination pairs of nodes* (S_i, D_i), $0 \leq i \leq m-1$, *are in a* deadlocked *state if* $\forall i$, $0 \leq i \leq m-1$, $|L_i| > 0$ *and* $\forall l \in L_i$, $\exists j \neq i$, $0 \leq j \leq m-1$, *s.t.* $l \in P_j$.

In other words, a deadlocked configuration corresponds to a set of s-d pairs such that each s-d pair is waiting for some links that are currently reserved for some other pairs in the same set. This definition of a deadlocked state is used, in turn, to define a deadlock-free routing algorithm.

Definition 2 *A routing algorithm is defined as deadlock-free if it does not lead to a deadlocked state while routing any arbitrary set of source-destination pairs.*

We next introduce the necessary framework under which the properties of a deadlocked state can be analyzed. As before, let us assume there are a set of m, $m > 0$, s-d pairs of nodes (S_i, D_i), $0 \leq i \leq m-1$, to be conneted in the system. As before, let P_i represent the path corresponding to the s-d pair (S_i, D_i). Also, recall that L_i represents the set of legal outgoing links from current intermediate node I_i.

Definition 3 *A dependency graph corresponding to the situation described above is a directed graph* $G = (V, A)$ *such that*

1. $V = \{P_i \mid 0 \leq i \leq m-1\}$, *and*
2. $A = \{(P_i, P_j) \mid \exists l \in L_i$ *such that* $l \in P_j\}$.

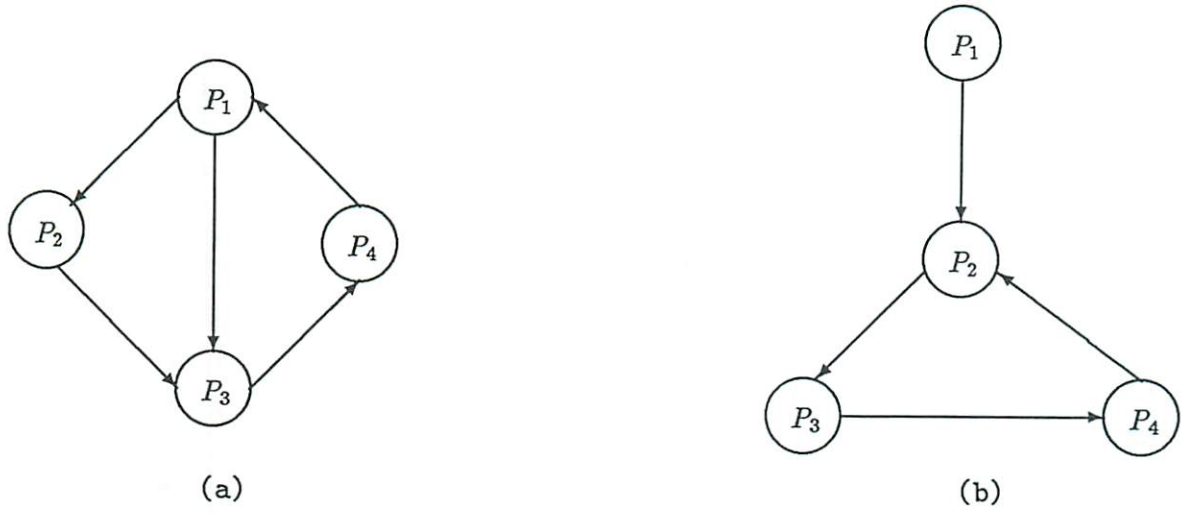

Figure 2: Two dependency graphs with each corresponding to a deadlocked state among 4 partial paths.

In other words, each vertex in the dependency graph corresponds to a path in the system. There exists an arc from vertex P_i to vertex P_j in the graph if and only if P_j contains a link that is legal with respect to s-d pair $(S_i,\ D_i)$ at intermediate node I_i. From the fact that each path is a shortest path, we can infer that there exist no self-loops or multiple arcs between two vertices in the dependency graph. For example, consider the dependency graph shown in Figure 2(a). S-d pair 1 needs to use either a link currently reserved by s-d pair 2 or a link currently reserved by s-d pair 3, and there is no other legal link for s-d pair 1. It is easy to observe that, in this figure, the four s-d pairs are in a deadlocked state. Figure 2(b) shows another example of a deadlocked configuration. Next we will show that there exists at least one cycle in the dependency graph associated with a deadlocked set of s-d pairs.

Lemma 1 *Given a set T of m, $m > 0$, source-destination pairs of nodes that corresponds to a deadlocked configuration, there exists at least one cycle in the associated dependency graph.*

Proof: We prove the lemma by contradiction. Assume that there exists no cycle in the dependency graph. Observe that there are m vertices in the dependency graph with each vertex corresponding to a partial path. Owing to the fact that the m s-d pairs are in a deadlocked state, each incomplete path must be waiting for at least one link that is reserved by another incomplete path. Thus, there should be at least one arc that emanates from every vertex in the dependency graph. Also, every legal link for an s-d pair is reserved for another s-d pair in T, which means every arc from a vertex

must lead to another vertex that corresponds to another s-d pair in T. Now we will generate a sequence of vertices as follows. Choose an arbitrary vertex and label it as vertex 1. From this vertex, we arbitrarily choose an outgoing arc. Since no self loops exist, this arc should lead to an unlabeled vertex. Let us label it as vertex 2. Now, from vertex 2, select an outgoing arc as we did earlier. As argued earlier, the new vertex cannot be vertex 2. Since we assumed that there exists no cycle in the dependency graph, the new vertex cannot be vertex 1 either; otherwise, the sequence of vertices $1 \rightarrow 2 \rightarrow 1$ forms a cycle. Let us label this new vertex as vertex 3. We can repeat the same process and label new vertices in increasing order. Now assume we are visiting vertex i, $i > 1$. Let us select an outgoing arc from this vertex. Owing to our earlier assumption, this arc cannot lead to a vertex which has been labeled earlier. This is because, if it leads to a labeled vertex, say vertex j, $j < i$, then the sequence of vertices $j \rightarrow j+1 \rightarrow ... \rightarrow i-1 \rightarrow i \rightarrow j$ forms a cycle, contradicting our assumption. By the same argument, once we label the m-th vertex, any outgoing arc from the vertex.with label m should lead to a new unlabeled vertex. That means there should be at least $m+1$ vertices (m labeled ones and a new unlabeled one) in the dependency graph. However, this contradicts the fact that there are exactly m vertices in the associated deadlocked configuration. This concludes the proof of the lemma. □

III A Class of New Deadlock-Free Routing Algorithms

A commonly used routing algorithm in circuit-switched hypercube multicomputers is the e-cube algorithm. In the e-cube routing algorithm, if the partial path, which is destined for node $D = (d_{n-1}, ..., d_0)$ from source node S, reaches an intermediate node $X = (x_{n-1}, ..., x_0)$, the link X_i, if available, is reserved for the path from S to D, where i is the least significant bit position in which X and D differ; if this link X_i is unavailable, then the partial path from S to D must wait at the intermediate node X until the link is released. Recall that X_i represents the outgoing i-dimensional link from node X. In other words, the e-cube routing algorithm reserves links in the order of increasing dimension.

The e-cube routing algorithm is deadlock-free and routes messages along shortest paths. However, it does not use the flexibility offered by the hypercube multicomputers to the full extent. There are circumstances under which one would like to utilize the flexibility allowed by the hypercubes to as high an extent as possible. For example, a single link failure might disconnect a source-destination pair, despite the existence of alternative paths, if e-cube routing algorithm is to be used

to route messages. Moreover, utilizing the flexibility of the hypercube system might lead to an improved performance under different traffic conditions. In the following, we propose a set of criteria by which one can develop routing algorithms that are more flexible than the *e*-cube algorithm, while maintaining the two characteristics, namely, freedom from deadlocks and routing along shortest paths.

Consider an n-dimensional hypercube. Given a pair of nodes $X = (x_{n-1}, ..., x_0)$ and $Y = (y_{n-1}, ..., y_0)$, let $UT(X, Y)$ denote the set of dimensions in which X and Y differ such that X has zeros along these dimensions and Y has ones. In other words, $UT(X, Y)$ is defined as $UT(X, Y) = \{i \mid 0 \leq i \leq n-1, x_i = 0, y_i = 1\}$. We shall refer to $UT(X, Y)$ as the set of *up-transitions* between X and Y. Similarly, we define $DT(X, Y) = \{i \mid 0 \leq i \leq n-1, x_i = 1, y_i = 0\}$ as the set of dimensions in which X and Y differ such that X has ones and Y has zeros. $DT(X, Y)$ is referred to as the set of *down-transitions* between X and Y. Observe that $|UT(X, Y)|$ and $|DT(X, Y)|$ together add up to the shortest (Hamming) distance between X and Y. Observe that any algorithm that establishes a shortest path from node X to node Y should reserve one link of a dimension that belongs to the set $UT(X, Y) \cup DT(X, Y)$.

Our routing criterion does not impose any constraints on reserving a link whose dimension belongs to the set of up-transitions between a source and a destination. However, it allows a link, whose dimension belongs to the set of down-transitions between a source and a destination, to be reserved only when all links with lower dimensions have been obtained. Our routing criterion is shown below.

Up-Preference (UP) Routing Criterion

Given a source-destination pair of nodes (S, D), the following rules are applied at any intermediate node I (including source node S):

- An i-dimensional link can be reserved if $i \in UT(I, D)$
- An i-dimensional link, where $i \in DT(I, D)$, can be reserved only if all links of dimension j, such that $j < i$ and $j \in UT(S, D) \cup DT(S, D)$ have been reserved
- An i-dimensional link, where $i \notin UT(I, D) \cup DT(I, D)$, cannot be reserved.

In essence, the UP routing criterion allows complete freedom in obtaining the up-links and restricts the order in which down-links can be acquired. This criterion is much more flexible than

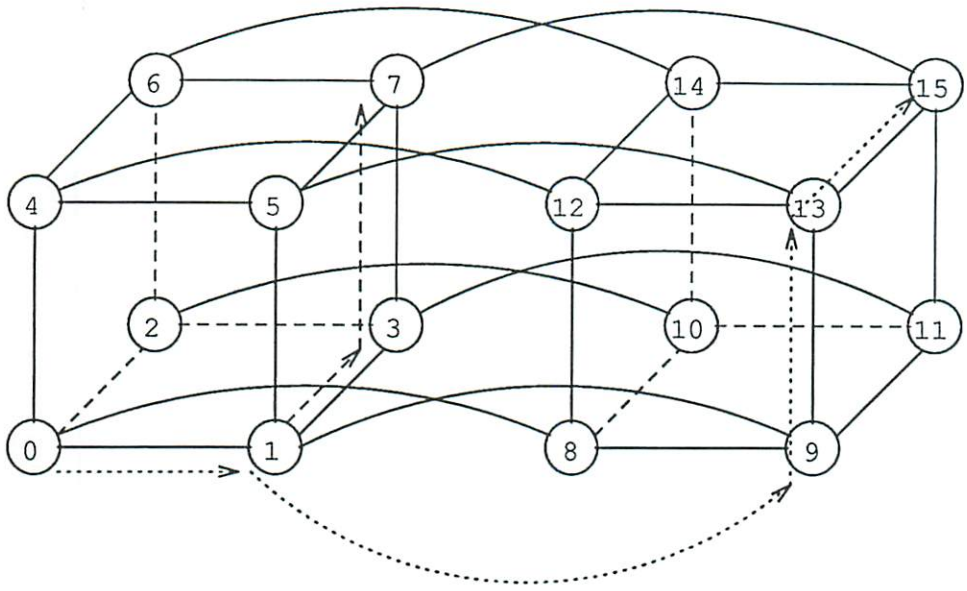

Figure 3: Path establishment in a 4-cube.

the *e*-cube strategy. With the UP routing criterion, very often more than one legal link is allowed at an intermediate node. Notice that the *e*-cube strategy allows only one legal link at an intermediate node for an s-d pair. For example, consider the 4-cube shown in Figure 3. Let us assume that the path $1_1 \uparrow 3_2 \uparrow$, shown by dashed lines, is established in the 4-cube from source 1 to destination 7. Now assume that node 0 wants to realize a circuit-switched path to node 15. Using *e*-cube routing algorithm, the partial path, $0_0 \uparrow$, corresponding to the s-d pair $(0,15)$ has to wait at the intermediate node 1, owing to the fact that link $1_1 \uparrow$ is currently reserved by the s-d pair $(1,7)$. However, with the UP criterion, at source node 0, the set of legal links for the s-d pair $(0,15)$ will be $\{0_0 \uparrow, 0_1 \uparrow, 0_2 \uparrow, 0_3 \uparrow\}$ and every link in this set is currently available. Suppose that, out of these links, 0_0 is reserved for this s-d pair. Hence, the next intermediate node will be 1. At node 1, the set of legal links for the s-d pair $(0,15)$ will be $\{1_1 \uparrow, 1_2 \uparrow, 1_3 \uparrow\}$. Although the link $1_1 \uparrow$ is currently reserved for the s-d pair $(1,7)$, there are two other legal links that are available. These links can be used according to our criterion. As shown in Figure 3, the dotted lines indicate one of the paths from node 0 to node 15 allowed by the UP criterion.

A routing algorithm, which is based on the UP routing criterion, would have to specify some detailed policies, such as a policy for selecting a legal link to reserve if there are more than one available legal links, and a policy for determining the waiting discipline if none of the legal links are available at an intermediate node. These policies, however, can be specified by system designers

depending on their needs.

Notice that we can define another routing criterion by simply interchanging the roles of $UT(I,\ D)$ and $DT(I,\ D)$ in the UP routing criterion. We will refer to this criterion as Down-Preference (DP) routing criterion. In other words, the DP criterion does not restrict reservation of down-links in any fashion. Obviously, the DP routing criterion have the same characteristics as the UP routing criterion. Therefore, in the following discussion, unless otherwise mentioned, we will refer to the UP routing criterion only.

It can be inferred that only the links corresponding to the dimensions in which the source and destination differ can be reserved, with respect to the UP routing criterion. Hence the path established by a routing algorithm, which meets the UP routing criterion, is always a shortest path.

Next, we prove that any routing algorithm that complies with the UP routing criterion is deadlock-free. For a set of m s-d pairs $(S_i,\ D_i)$, $0\ \leq\ i\ \leq\ m-1$, P_i is used to denote the incomplete path established from S_i to an intermediate node I_i. If there is an arc from P_i to P_j in the associated dependency graph, P_i is waiting for a link that is currently reserved for P_j; we denote the link that P_i is requesting and is currently reserved for P_j as l_{ij}. First, we prove the following lemma with respect to a routing algorithm that routes messages along shortest paths (but not necessarily deadlock-free).

Lemma 2 *Consider m, $m > 1$, partial paths P_i, $0 \leq i \leq m-1$, corresponding to m s-d pairs, (S_i, D_i), $0 \leq i \leq m-1$. Further, assume that these partial paths are established using an algorithm that routes messages along shortest paths. Assume that there exists a cycle involving the m vertices P_i, $0 \leq i \leq m-1$, in the associated channel dependency graph. Moreover, assume that the set of arcs in this cycle is given by $\{(P_i,\ P_{i+1 \bmod m}) \mid 0 \leq i \leq m-1\}$. Then, for any partial path P_j such that $l_{j,j+1 \bmod m}$ is an up-link of some dimension k, there exists a partial path P_s, $0 \leq s \neq j \leq m-1$, such that P_s contains a down-link of dimension k.*

Proof: We prove the lemma by contradiction. Without loss of generality, let l_{01} be an up-link of dimension k for some k. Now let us assume $\forall i,\ i \neq 0$, P_i does not contain a down-link of dimension k. Consider that the n-cube is partitioned, along dimension k, into two $(n-1)$-subcubes, N_0 and N_1, such that N_0 (respectively, N_1) contains processors whose binary representations have a zero (respectively, one) in bit position k. Notice that a path that is routed by a shortest-path routing algorithm can never traverse any dimension more than once. Since P_0 is waiting for an up-link of dimension k, P_0 could not have reserved any k-dimensional down-link, and P_1 must have reserved an up-link of dimension k. Also, the intermediate node I_1 for P_1 must be in N_1. Since P_1 is waiting

for some link currently reserved for P_2 at I_1, P_2 must have visited I_1 in N_1. By our assumption, P_2 cannot contain a down-link of dimension k. Therefore, the current intermediate node I_2 for P_2 must be in N_1. The same argument can be repeated to show that the current intermediate node I_{m-1} for P_{m-1} must be in the subcube N_1. Since P_{m-1} is waiting for some link reserved for P_0 at node I_{m-1}, P_0 must have visited I_{m-1}. However, I_0 is in N_0, which means P_0 must have reserved a down-link of dimension k. But P_0 could not have reserved a down-link of dimension k, since it is currently waiting for a k-dimensional up-link and messages are routed aong shortest paths. This yields us the required contradiction and, hence, the lemma. □

Now we are ready to prove that any algorithm that complies with the UP criterion is deadlock-free.

Theorem 1 *Any routing algorithm that meets the UP routing criterion is deadlock free.*

Proof: Again we prove by contradiction. Assume that there exists some routing algorithm which meets the UP routing criterion and caused a deadlocked state among some set of s-d pairs. From lemma 1, we know that there exists at least one cycle in the associated dependency graph with respect to the set of s-d pairs. Without loss of generality, let us assume that one of the cycles involves m, $m > 1$, partial paths which correspond to m s-d pairs of nodes. Let us label these paths such that the cycle can be represented by having an arc from P_i to $P_{i+1 \ mod \ m}$, $0 \leq i \leq m-1$, where P_i denotes the partial path for the i-th s-d pair. Also, let l_{ij} represent the link associated with the arc from P_i to P_j; in other words, l_{ij} denotes the link that P_i is waiting and is reserved for P_j. Now we have to consider two cases for proving the theorem.

Case 1: Link $l_{i,i+1 \ mod \ m}$ at every intermediate node I_i, $0 \leq i \leq m-1$, is a down-link.
That is, in this case, partial path P_i, $0 \leq i \leq m-1$, is currently waiting at intermediate node I_i for a down-link $l_{i,i+1 \ mod \ m}$. Let d_i, $0 \leq i \leq m-1$, denote the dimension of the link $l_{i,i+1 \ mod \ m}$. Let p denote the maximum number among the m d_i's. That is, $p = \max\{d_j \mid 0 \leq j \leq m-1\}$. Without loss of generality, let us assume $d_0 = p$, which means P_0 is waiting for a down-link of dimension p which is now reserved for P_1. We must have $p \geq d_1$, since p is the maximum among the dimensions d_j, $0 \leq j \leq m-1$. Because the UP routing criterion requires that a down-link of dimension j can be reserved by a path only when all required links of dimensions lower than j are acquired, we infer that $d_1 > p$. This contradicts with the previous result that $p \geq d_1$. Hence it is impossible for this case to happen.

Case 2: There exists at least an i, $0 \leq i \leq m-1$, such that the link $l_{i,i+1 \ mod \ m}$ is an up-link.

Let p denote the maximal dimension among all dimensions of the links $l_{j,j+1 \bmod m}$, $0 \leq j \leq m-1$, which are up-links. Without loss of generality, let l_{01} be a p-dimensional up-link. That means, P_0 is waiting for a p-dimensional up-link, which is currently reserved for P_1. From Lemma 2, there must exist some $k \neq 0$ such that P_k already reserved a down-link of dimension p. According to the UP routing criterion, we infer that P_k must have traversed all dimensions less than or equal to p, in which its associated source and destination nodes differ. Therefore, the dimension of $l_{k,k+1 \bmod m}$, denoted as d_k, must be greater than p. Here we have to to consider two situations: 1) $l_{k,k+1 \bmod m}$ is an up-link, 2) $l_{k,k+1 \bmod m}$ is a down-link. If $l_{k,k+1 \bmod m}$ is an up-link, then the fact that $d_k > p$ conflicts with the assumption that p is the maximum dimension of all up-links. Hence this link can only be a down-link. Since a down-link is reserved only if all required links of lower dimensions are reserved, we infer that $d_k > p$. By a repeated application of the same argument, we can show that $l_{m-1,0}$ must be a down-link of dimension $d_{m-1} > p$. This means P_0 would have already reserved a down-link of dimension d_{m-1}. However, this cannot happen, in accordance to the UP routing criterion, since P_0 is currently requesting an up-link of dimension p, which is less than d_{m-1}. This completes the proof of the theorem. □

IV Fault-Tolerance Analysis of the Flexible Routing Criteria

One major disadvantage of the e-cube routing algorithm is that it allows only one path between a source-destination pair. If any link or an intermediate node on this unique path allowed by the e-cube routing algorithm fails, the corresponding source can not send messages to it's destination using the e-cube algorithm. Hence, in a hypercube system that uses the e-cube routing algorithm, the ability to tolerate faults is virtually nonexistent.

As opposed to e-cube strategy, the UP criterion proposed in this paper allows more than one path from source S to destination D unless the distance between S and D is one or the number of up-links that require to be reserved in a path from S to D equals zero. Similarly, the DP criterion allows more than one path from source S to destination D unless the distance between S and D is one or the number of down-links that require to be reserved in a path from S to D equals zero. Hence, link failures would have much less impact on the system's ability to route messages, if the system adopts our routing criteria. Moreover, the fault-tolerance capability provided by our algorithms increases as the size of the hypercube increases, owing to the fact that in a larger cube the average distance between an s-d pair is larger.

In this section, we first analyze the fault-tolerance capability of our algorithms when a single link fails in the system. The number of source-destination pairs that fail to communicate when a single link fails in the system is chosen as the measure of fault-tolerance in our study. We propose techniques to relabel the nodes in the hypercube in such a way that the number of s-d pairs that fail to communicate under a single link-failure is brought down to one.

Another measure that we investigate in this section is the number of *node-disjoint* shortest paths provided by our algorithm between a source and a destination. Two paths from source S to destination D are said to be node-disjoint, if they do not have any node other than S and D in common. It is a well-known fact that the total number of node-disjoint shortest paths between S and D in an n-cube equals the shortest (Hamming) distance between S and D. The number of node-disjoint paths allowed by the e-cube algorithm between source S and destination D is always one. We show that our routing criteria allow multiple node-disjoint shortest paths for an s-d pair.

Finally, we analyze the fault-tolerance capability of our algorithms when a single node fails in the system. We estimate the number of source-destination pairs that fail to communicate when a single node fails. We then propose a technique to relabel the nodes in the hypercube so that the number of affected s-d pairs is minimized.

Fault-Tolerance Analysis under Single Link-Failure

Consider an n-cube in which the UP criterion is used to realize a circuit-switched path from source $S = (s_{n-1}, ..., s_0)$ to destination $D = (d_{n-1}, ..., d_0)$. In addition, let us assume that the outgoing up-link $X_i \uparrow$ from node $X = (x_{n-1}, ..., x_{i+1}, 0, x_{i-1}, ..., x_0)$ to node $X' = (x_{n-1}, ..., x_{i+1}, 1, x_{i-1}, ..., x_0)$ failed. Further, we assume that every processor in the system knows that link $X_i \uparrow$ failed. We say that an s-d pair (S, D) is *affected* by the failure of a link l if every path from source S to destination D, allowed by the UP criterion, has to acquire link l. We would like to find the number of s-d pairs that are affected by the failure of link $X_i \uparrow$. The following theorem evaluates this number.

Theorem 2 *Assume that link $X_i \uparrow$ from node $X = (x_{n-1}, ..., x_{i+1}, 0, x_{i-1}, ..., x_0)$ to node $X' = (x_{n-1}, ..., x_{i+1}, 1, x_{i-1}, ..., x_0)$ failed in an n-cube that uses the UP criterion. The number of affected source-destination pairs equals 2^p where p is the cardinality of the set*

$$\{j |\ (i+1) \leq j \leq n-1,\ \textit{and}\ x_j = 1\}.$$

Proof: Observe that if the s-d pair $(S,\ D)$ is affected by the failure of link $X_i \uparrow$, then s_i should be zero and d_i should be one. We first show that an s-d pair (S, D) is affected only if S equals X. To

prove this, let us assume that S is not equal to X. Hence, $S_i\uparrow$ is non-faulty. Since $s_i = 0$ and $d_i = 1$, link $S_i\uparrow$ can be reserved as per the UP criterion at source S; hence, link $X_i\uparrow$ will not be required subsequently. Thus, if S is not equal to X, failure of link $X_i\uparrow$ does not affect any s-d pair (S, D), $0 \leq D \leq N-1$. Thus, the only affected source-destination pairs are of the form (X, D). Next, we show, by contradiction, that either $d_j = x_j$, for $0 \leq j \leq i-1$, or $i = 0$, if (X, D) is affected by the failure of link $X_i\uparrow$. Suppose that $i > 0$ and j, where $j < i$, is the least-significant bit in which X and D differ. In this case, if we reserve X_j (it can be either $X_j\uparrow$ or $X_j\downarrow$) at source X, the subsequent path will not require link $X_i\uparrow$; thus, s-d pair (X, D) will not be affected. Hence, the faulty link $X_i\uparrow$ will affect an s-d pair (X, D) only if $d_j = x_j$, for all $0 \leq j \leq i-1$ or $i = 0$. Now, let us consider the highest $n-i-1$ bits $x_{n-1}, ..., x_{i+1}$ under the assumption $i < n-1$. If x_j equals zero, for some j such that $i+1 \leq j \leq n-1$, then d_j must be zero for the s-d pair (X, D) to be affected by the faulty link $X_i\uparrow$. If $x_j = 0$ and $d_j = 1$, for some $j \geq i+1$, then the UP criterion could have reserved link $X_j\uparrow$ at node X, thus avoiding the link $X_i\uparrow$. Thus, we proved that the following must hold if the s-d pair (S, D) is affected by the faulty link $X_i\uparrow$.

1. S must equal X and d_i must equal one.
2. $d_j = x_j = s_j$, for $0 \leq j \leq i-1$, if i exceeds 0.
3. If $s_j = x_j$ equals zero, for some $i+1 \leq j \leq n-1$, then d_j should also equal zero.

From these facts, we infer that an s-d pair (X, D) is affected if and only if i is the least significant bit-position in which X and D differ and all required links of dimension greater than i are down-links. In order to count the number of such s-d pairs, we observe that if x_j, $i+1 \leq j \leq n-1$, equals one, then d_j can be either one or zero for an affected s-d pair. Thus the total number of affected source-destination pairs equals 2^p, where p is the cardinality of the set

$$\{j |\ (i+1) \leq j \leq n-1,\ \text{and}\ x_j = 1\}.$$

This concludes the proof of the theorem. □

For example, if the 1-dimensional link $24_1\uparrow$ from node $24 = (1,1,0,0,0)$ to node $26 = (1,1,0,1,0)$ fails in a 5-cube there will be $2^2 = 4$ affected s-d pairs. The affected s-d pairs, in this example, are (24, 2), (24, 10), (24, 18) and (24, 26). It can seen that, in a system that uses the *e*-cube routing algorithm, communication between sixteen s-d pairs will be disrupted if the up-link $24_1\uparrow$ fails.

Next, we would like to find the number of affected s-d pairs if a down-link fails in a system that uses the UP criterion. The following theorem evaluates this number.

Theorem 3 *Assume that i-dimensional down-link $X_i\downarrow$ from node $X = (x_{n-1}, ..., x_{i+1}, 1, x_{i-1}, ..., x_0)$ to node $X' = (x_{n-1}, ..., x_{i+1}, 0, x_{i-1}, ..., x_0)$ failed in an n-cube that uses the UP criterion. The number of affected source-destination pairs equals 2^{i+p} where p is the cardinality of the set*

$$\{j|\ (i+1) \leq j \leq n-1,\ \textit{and}\ x_j = 1\}.$$

Proof: First observe that an s-d pair (S, D) is affected by the faulty down-link $X_i\downarrow$ only if $s_i = 1$ and $d_i = 0$. According to the UP criterion, before a down-link of dimension i can be reserved for the path from source S to destination D, all required links of dimensions lower than i must be reserved. Hence, for any s-d pair (S, D) affected by the faulty link $X_i\downarrow$, the condition either $x_j = d_j$, for $0 \leq j \leq i-1$ or $i = 0$ must hold. In order to see this fact, assume that $i > 0$ and that $x_j \neq d_j$, for some $j < i$. In this case, the partial path for the s-d pair (S, D), if it visits X, can not use the link $X_i\downarrow$, owing to the fact that x_j and d_j differ for some $j < i$.

Now let us consider the highest $n-i-1$ bits $x_{n-1}, ..., x_{i+1}$ under the assumption that $i < n-1$. We first assume that $x_j = 0$, for some j such that $i+1 \leq j \leq n-1$. Under this condition, we show that both s_j and d_j should be zero. If $s_j = 1$ and $d_j = 0$, then the path from S to D could have reached X only by reserving a j-dimensional down-link before reaching X. However, the j-dimensional down-link can not be reserved before reserving the i-dimensional down-link, owing to the fact that $i < j$. Thus, the case $s_j = 1$ and $d_j = 0$ is not possible. If $s_j = 0$ and $d_j = 1$, then the UP criterion could have reserved link $S_j\uparrow$ at source S; this path does not visit node X owing to the fact that x_j equals zero. If $s_j = d_j = 1$, the path from S to D could not have visited node X, owing to the fact that $x_j = 0$. Next assume that $x_j = 1$, for some j such that $i+1 \leq j \leq n-1$. In this case, s_j can not be zero. If $x_j = 1$ and s_j equals zero, the required down-link of dimension i could have been reserved before reserving j-dimensional up-link. Thus, node X can not be visited since $x_i = 1$. Therefore, s_j must equal one when $x_j = 1$. These observations show that an s-d pair (S, D) is affected by the faulty link $X_i\downarrow$ under the following conditions.

1. $s_i = 1$ and $d_i = 0$.
2. $d_j = x_j$, for $0 \leq j \leq i-1$, if $i > 0$. Observe that s_j can be either zero or one in this case.
3. If $x_j = 0$, for some j such that $i+1 \leq j \leq n-1$ under the assumption that $i < n-1$, then s_j and d_j must both equal zero.

4. If $x_j = 1$, for some j such that $i+1 \leq j \leq n-1$ under the assumption that $i < n-1$, then s_j must be one. Observe that d_j can be either zero or one in this case.

Let p be the cardinality of the set

$$\{j|\ (i+1) \leq j \leq n-1,\ \text{and}\ x_j = 1\}.$$

Observations 2 and 4 presented above give us $2^i \times 2^p = 2^{i+p}$ as the number of s-d pairs affected by the faulty link $X_i \downarrow$. This concludes the proof of the theorem. □

For example, if the 1-dimensional link $26_1 \downarrow$ from node $26 = (1,1,0,1,0)$ to node $24 = (1,1,0,0,0)$ in a 5-cube fails, eight s-d pairs will be affected. The affected s-d pairs, in this example, are (26, 0), (26, 8), (26, 16), (26, 24), (27, 0), (27, 8), (27, 16) and (27, 24).

Results similar to those presented in theorems 2 and 3 can be derived for the routing criterion DP by interchanging the roles of up-links and down-links. The following corollary states these results for the DP routing criterion.

Corollary 1 *Consider an n-cube system that employs the DP routing criterion. Assume that the i-dimensional link emanating from node $X = (x_{n-1}, ..., x_0)$ failed in this system. Let p denote the cardinality of the set*

$$\{j|\ (i+1) \leq j \leq n-1,\ \textit{and}\ x_j = 0\}.$$

The number of s-d pairs affected by the faulty link equals

1. 2^p, *if the failed i-dimensional link is a down-link (that is, if $x_i = 1$), and*
2. 2^{i+p}, *if the failed i-dimensional link is an up-link (that is, if $x_i = 0$).*

Next we calculate the number of s-d pairs affected in a system that adopts the *e*-cube strategy, under the assumption that the link of dimension i at node $X = (x_{n-1}, ..., x_0)$ fails. First, we observe that if the s-d pair (S, D) is affected by the faulty link, then s_i and d_i should equal x_i and $\overline{x_i}$, respectively. In addition, we notice that, as per the *e*-cube strategy, the path from source S to destination D would visit the intermediate destination X if and only if the following conditions are satisfied.

1. $d_j = x_j$, for $0 \leq j \leq i-1$, if $i > 0$ and
2. $s_j = x_j$, for $i \leq j \leq n-1$.

These observations, together with the fact that $s_i = x_i$ and $d_i = \overline{x_i}$ for any affected s-d pair (S, D), prove that the number of source-destination pairs affected by the faulty i-dimensional link at node X equals $2^i \cdot 2^{n-i-1} = 2^{n-1} = N/2$. Thus, a single link-failure affects $N/2$ s-d pairs in a system that employs the e-cube strategy.

Recall that the number of affected s-d pairs, when the UP criterion is employed, equals either 2^p (if the up-link $X_i \uparrow$ is faulty) or 2^{i+p} (if the down-link $X_i \downarrow$ is faulty), where p is the cardinality of the set $\{j |\ i+1 \leq j \leq n-1;\ \text{and}\ x_j = 1\}$. We observe that $p \leq (n-1)$, if the up-link $X_i \uparrow$ is faulty, and $i + p \leq (n-1)$, if the down-link $X_i \downarrow$ is faulty. This shows that, in the worst case, the number of s-d pairs that can be affected by a faulty link in a system that employs the UP routing strategy equals $2^{n-1} = N/2$, which is the number of s-d pairs affected under the e-cube strategy. However, the number of affected s-d pairs, under the UP routing criterion, is dependent on the address of the node from which the faulty link emanates as well as on the dimension of the faulty link. This suggests that we can possibly renumber the dimensions so as to minimize the number of affected s-d pairs when a single link fails in the system. The following procedure renumbers the dimensions, in a hypercube system that uses our routing criterion, so that the number of s-d pairs affected by a faulty link equals one.

Procedure Minimize-the-Number-of-Disrupted-Pairs(X_i)
Comment: Link X_i of dimension i, which emanates from node $X = (x_{n-1}, ..., x_0)$ failed in the hypercube system.

1. Relabel the dimensions of the n-cube such that dimension i, which is the dimension of the faulty link X_i, becomes dimension $(n-1)$ after the relabeling. One simple way to perform this relabeling is by interchanging dimensions $(n-1)$ and i.

2. If link X_i is an up-link (that is, if $x_i = 0$), then use the UP criterion in the system with relabeled dimensions.

3. If link X_i is a down-link (that is, if $x_i = 1$), then use the DP criterion in the system with relabeled dimensions.

Suppose that the above procedure is executed in order to relabel the dimensions such that dimension i, which is the dimension of the faulty link X_i, becomes dimension $n-1$. Thus, in the relabeled system the dimension of the faulty link becomes $n-1$. Further, assume that the failed link is an up-link (respectively, a down-link). Hence, Step 2 (respectively, Step 3) of the above procedure will be

executed. Thus the UP (respectively, DP) criterion will be used in the relabeled system. Therefore, the total number of affected s-d pairs in the relabeled system equals $2^0 = 1$ from theorems 2 and corollary 1. This proves that the procedure presented above reduces the number of s-d pairs affected by a faulty link to one. As an example, let us assume that the 1-dimensional link $26_1 \downarrow$, from node $26 = (1,1,0,1,0)$ to node $24 = (1,1,0,0,0)$, in a 5-cube failed. In this case, we simply exchange the dimensions 4 and 1 in the 5-cube. In other words, the original dimensions 4, 3, 2, 1, and 0 (before the fault occurred) are now relabeled as dimensions 1, 3, 2, 4, and 0, respectively. In the relabeled 5-cube the faulty link is from node $26 = (1,1,0,1,0)$ to node $10 = (0,1,0,1,0)$. Further, this faulty link affects no s-d pair other than $(26,10)$, if the DP criterion is employed in the relabeled hypercube system.

Notice that, if we relabel the dimensions of the n-cube so as to minimize the number of affected s-d pairs due to a faulty link, only the routing function has to use the reordered dimensions. Node addresses used in the program do not require to be changed.

Analysis of Maximum Node-Disjoint Paths

Let the shortest (Hamming) distance between source S and destination D be represented by $H(S,\ D)$. It can be easily proved that the maximum number of node-disjoint shortest paths between S and D in an n-cube equals the $H(S,\ D)$ [8]. Note that only one path is allowed by the e-cube routing algorithm. Our routing criteria, in many cases, allow more than one node-disjoint shortest path between S and D.

In this subsection, we evaluate the maximum number of node-disjoint shortest paths between source $S = (s_{n-1}, ..., s_0)$ and destination $D = (d_{n-1}, ..., d_0)$ allowed by our routing criteria. During this analysis, we assume an n-cube that uses the UP criterion to establish a circuit-switched path between S and D. First, let us consider the case in which no down-link requires to be reserved to establish a path from S to D. That is, all required links in any path from S to D are up-links. Since the UP criterion does not restrict the acquisition of up-links in any way, the maximum number of node-disjoint shortest paths, in this case, equals $H(S, D)$. Hence, during the remainder of this section we consider s-d pairs (S, D) such that at least one down-link requires to be reserved for the path from S to D. For this case, the following theorem evaluates the number of node-disjoint shortest paths allowed by the UP criterion.

Theorem 4 *In an n-cube with UP criterion, consider an s-d pair* $(S,\ D)$ *that requires at least one down-link to be reserved on any shortest path from* S *to* D. *Further, assume that* i *is the largest*

integer such that $s_i = 1$, $d_i = 0$, *and* $0 \leq i \leq n-1$. *Then, the maximum number of node-disjoint shortest paths between* S *and* D *equals* $p+1$, *where* p *is the cardinality of the set*

$$\{j \mid s_j = 0,\ d_j = 1, \quad and \quad i+1 \leq j \leq n-1\}.$$

Proof: First assume that p equals zero. That is, i is the largest dimension in which S and D differ. Since the down-link corresponding to dimension i can be reserved only after all required links of lower dimensions are reserved, there exists exactly one node-disjoint path from S to D in this case.

Let us now assume that p exceeds zero. Without loss of generality, let us assume that $s_{n-1} = \ldots = s_{n-p} = 0$ and $d_{n-1} = \ldots = d_{n-p} = 1$, where $n - p > i$. First we show that there can be at most $p+1$ node-disjoint shortest paths, in this case. In order to see this, let us consider node D' from which node D is reached by reserving one link. That is, node D' is the penultimate node on a path from S to D. Further, observe that in order to reach destination D from node D', the UP criterion could not have reserved a link of dimension less than i; otherwise, link of dimension i would not have been reserved. Thus, the partial path at intermediate node D' would have reached destination D by reserving a link of dimension greater than or equal to i. However, there are exactly $p+1$ dimensions greater than or equal to i (including i) in which S and D differ. Hence, there are at most $(p+1)$ values for the intermediate destination D', which proves that there can be at most $(p+1)$ node-disjoint paths from S to D.

We next show that there exist $(p+1)$ node-disjoint paths from S to D. For simplicity, in the remainder of this proof, we represent a path using a sequence of dimensions. Each sequence corresponds to an ordering (of dimensions) as per which links are reserved. Thus, if links of dimensions $l_1, l_2, \ldots, l_k$ are reserved in that order, to establish the path from S to D, the corresponding path is represented as $l_1, l_2, \ldots, l_k$. Further, let L represent an increasing sequence of all dimensions in the set

$$\{j \mid s_j \neq d_j \quad \text{and} \quad 0 \leq j \leq i\}.$$

Notice that L can not be an empty sequence owing, to the fact that S and D differ in dimension i. Next, let us consider the sequence

$$(n-p) \ \ldots \ (n-1)\ L.$$

From this sequence, we derive $(p+1)$ paths as described below.

1. Path 0 is given by the sequence $(n-p) \ \ldots \ (n-1)\ L$.

2. Path j is given by the sequence $(n-p+j)\ \ldots\ (n-1)\ L\ (n-p)\ \ldots\ (n-p+j-1)$, for $1 \leq j \leq p-1$.

3. Path p is given by the sequence $L\ (n-p)\ \ldots\ (n-1)$.

Next, we prove that the $(p+1)$ paths shown above are node-disjoint. In this proof, we utilize the fact that, if two sequences differ in at least one dimension, the corresponding partial paths with node S as the starting node must be node-disjoint. Observe that Path 0 ends with sub-sequence L and Path p begins with sub-sequence L. Hence, two partial paths from source S corresponding to Path 0 and Path p must be node-disjoint. In fact, the paths shown above are formed by cyclic rotation in which we treat the sequence L as a group of dimensions. By the same arguments as given in [8], we can easily prove that any two partial paths Path j and Path k, $0 \leq j, k \leq p$ and $j \neq k$, are node-disjoint. This fact, together with our earlier result that the number of node-disjoint paths between S and D can not exceed $p+1$, proves the theorem. □

Notice that the number of node-disjoint paths allowed by the UP criterion between S and D depends on the addresses $s_{n-1}, ..., s_0$ and $d_{n-1}, ..., d_0$, and varies from 1 to $H(S,D)$. A value of $H(S,D)$ is achieved only when the UP criterion does not constrain acquisition of any link. The UP criterion allows only one node-disjoint path, when $H(S,D) > 1$, if the link of highest dimension that needs to be reserved is a down-link. We next derive the number of affected s-d pairs when a node fails in the hypercube.

Fault-Tolerance Analysis under Single Node-Failure

Let us now assume that node $X = (x_{n-1}, ..., x_0)$ failed in the n-cube . We are interested in finding the number of s-d pairs affected by the failure of this node. As in the case of link failures, we say that the failure of node X affects an s-d pair (S, D), if every path from S to D must pass through node X. If node X failed, then no path can be established a) from node X to any other node (when X is the source), and b) to node X from any other node (when X is the destination). These cases yield us a lower bound on the number of affected s-d pairs as $2(N-1)$. We investigate the number of affected s-d pairs when X is a strict intermediate node in the theorem given below.

Theorem 5 *Let us assume that node $X = (x_{n-1}, ..., x_0)$ failed in an n-cube. Let L be the set of all bit-positions in which X has ones. In other words,*

$$L = \{k \mid x_k = 1 \quad and \quad 0 \leq k \leq n-1\}.$$

Further, let p_i be the cardinality of the set

$$\{k \mid x_k = 1 \quad and \quad (i+1) \leq k \leq n-1\}.$$

Then the number of s-d pairs affected by the faulty node X equals

$$2(N-1) + \sum_{i \in L} (2^i - 1) \cdot 2^{p_i}.$$

Proof: The factor $2(N-1)$ is obtained, owing to the fact that $(N-1)$ s-d pairs from X and $(N-1)$ s-d pairs to X are affected. We next investigate the case when X is an intermediate node. Let us assume that the path from S to D must pass through the intermediate node X. Further, let us assume that an outgoing link of dimension i requires to be reserved at the intermediate node X. Observe that $s_i = x_i$ and $d_i = \overline{x_i}$. The link of dimension i that requires to be reserved at node X can not be an up-link; if it were an up-link, the UP criterion could have reserved the link of dimension i, $S_i \uparrow$, at source S and could have avoided the faulty node X. Thus the link to be reserved at node X for the partial path from S to D must be a down-link. That is $s_i = x_i = 1$ and $d_i = 0$. Since X is an intermediate node, failure of node X is equivalent to the failure of all outgoing links from X. Let us denote the set of s-d pairs affected by the failure of link $X_i \downarrow$ with A_i. Similarly, A_j denotes the set of s-d pairs affected by the failure of link $X_j \downarrow$. Next we need to show that the sets A_i and A_j of affected s-d pairs are disjoint. To show this, let us, without loss of generality, assume that $i < j$. Observe that for every s-d pair in A_i, $d_i = 0$, whereas for every s-d pair in A_j, $d_i = x_i = 1$ (since $i < j$). Hence, no s-d pair that is present in A_i can be in A_j and vice-versa. Thus, the total number of affected s-d pairs, for the case when X is an intermediate node, equals the sum $\sum_{i \in L} |A_i|$, where $|A_i|$ denotes the cardinality of the set A_i.

Now we concentrate on computing the value $|A_i|$. To compute this value, recall that $s_i = x_i = 1$, and $d_i = 0$. This implies that $d_k = x_k$, for $0 \leq k \leq i$. In summary, bits s_k, d_k, and x_k, for $0 \leq k \leq i$, should satisfy the following constraints.

1. $s_i = 1$ and $d_i = 0$, if $x_i = 1$.

2. $d_k = x_k$, for $0 \leq k \leq i-1$.

Observe that there are no restrictions on the values of s_k, for $0 \leq k \leq i-1$. Thus, s_k, $0 \leq k \leq i-1$, can be either zero or one.

Now let us consider any bit position $k > i$. If $s_k = 0$, then d_k should be zero. To prove this, let us assume that $d_k = 1$. If $x_k = 0$, then an up-link of dimension k can be reserved at S, thus avoiding the faulty node X. If $x_k = 1$, it implies that the up-link of dimension k must have been reserved prior to reaching X. However, the k-dimensional up-link need not be reserved prior to reserving the i-dimensional down-link, since $k > i$. Thus the UP criterion could have avoided the faulty node X by reserving the up-link of dimension k only after reserving the down-link of dimension i. Thus, if s_k equals zero, for some $k > i$, then d_k should be zero. Further, if $s_k = 1$, for some $k > i$, then x_k must be one, for otherwise the partial path from S to D could not have visited node X. Thus, if $x_k = 1$, then s_k must be one; if s_k equals zero, then d_k equals zero, from the earlier argument, and the path from S to D can not visit node X. Similarly, if $x_k = 0$, then s_k must be zero, owing to the fact that s_k can be one only if x_k is one. In summary, bits s_k, d_k, and x_k, for $i+1 \leq k \leq n-1$, should satisfy the following constraints.

1. $s_k = x_k$.

2. If $x_k = 0$, then d_k must be zero.

Notice that, if $x_k = 1$, then there is no restriction on the value of d_k. In other words, if $x_k = 1$, then d_k can be either one or zero.

Thus we proved that s_k, $0 \leq k \leq i-1$, can be either one or zero and that, for $i+1 \leq k \leq n-1$, d_k can be either zero or one if $s_k = 1$. Moreover, any s-d pair that meets the constraints on s_k and d_k, $0 \leq k \leq n-1$, proved earlier must visit the faulty node X. Hence, the number of possibilities for source S equals 2^i. But, out of these 2^i choices, one choice corresponds to the case when X is the source (which is a case that we already considered). Hence, the number of possible sources equals $(2^i - 1)$. The number of possible destinations equals 2^{p_i}, where p_i is the cardinality of the set

$$\{k \mid x_k = 1 \quad \text{and} \quad (i+1) \leq k \leq n-1\}.$$

Thus the total number of s-d pairs affected in this scenario is given by $|A_i| = (2^i - 1) \cdot 2^{p_i}$. Hence, the total number of s-d pairs affected by the faulty node $X = (x_{n-1}, ..., x_0)$ is given by

$$2(N-1) + \sum_{i \in L} (2^i - 1) \cdot 2^{p_i},$$

where p_i be the cardinality of the set

$$\{k \mid x_k = 1 \quad \text{and} \quad (i+1) \leq k \leq n-1\}.$$

This concludes the proof of the theorem. □ In comparison, any node failure in an n-cube that employs the e-cube routing algorithm affects $(N-1)+nN/2$ s-d pairs.

As an example, let us consider that node $11 = (0,1,0,1,1)$ failed in a 5-cube. Thus node 11 can not send messages to 31 other nodes and can not receive messages from 31 other nodes. Let us consider the other s-d pairs affected by the faulty node 11. Observe that node 11 has ones in bit-positions 0, 1, and 3. Further, $p_0 = 2$, $p_1 = 1$ and $p_3 = 0$. Thus the number of affected s-d pairs, for which node 11 has to be an intermediate node, is given by $(2^0-1)2^2+(2^1-1)2+(2^3-1)2^0 = 9$. Further, the set of these s-d pairs is given by

$$\{(8,3),(9,3),(10,1),(10,3),(10,9),(12,3),(13,3),(14,3),(15,3)\}.$$

It is easy to observe that 49 s-d pairs (S, D), such that $S \neq 11$ and $D \neq 11$, are affected by the faulty node 11, if e-cube algorithm is used.

From the formula for the number of s-d pairs affected due to the failure of node $X = (x_{n-1},...,x_0)$, we observe that this number depends on the address of the faied node. We also infer that this number is a minimum when the node address contains zeros in the highest $(n-1)$ dimensions. Observe that when node 0 (node 1, respectively) fails in the system, node 0 (node 1, respectively) alone can not communicate with the other nodes in the system. However, a fault-free, circuit-switched path from every node other than node 0 (node 1, respectively) can be established to any node other than node 0 (node 1, respectively). The following procedure, by relabeling the faulty node X as node 0, minimizes the number of affected s-d pairs.

Procedure Minimize-the-Number-of-Disrupted-Pairs(X)
Comment: Node $X = (x_{n-1},...,x_0)$ in the n-cube is faulty.

1. Relabel the nodes in the n-cube such that node X becomes node 0 after relabeling.

V Performance Study

Criteria proposed in the section III provide us with a large amount of flexibility in designing routing algorithms for circuit-switched hypercubes. These criteria allow more than one path for most of the s-d pairs in a hypercube system without causing deadlocks. On the other hand, e-cube strategy allows only a single path between any s-d pair. This motivates us to study whether the additional flexibility allowed by our routing criteria would lead to an improvement in the performance of the system. A

comparative performance-study of our routing criteria with respect to the *e*-cube algorithm is the subject of this section.

We adopt the average *setup time* required to establish a path for an s-d pair as the performance measure in our study. Setup time corresponding to an s-d pair (S, D) is defined as the time required to establish a circuit-switched path from source S to destination D. Transmission time is defined as the time required to transmit a message from source S to destination D, once a circuit-switched path is established between S and D. Observe that the setup time is primarily due to the queueing delays caused by the unavailability of the required legal link(s) at source and intermediate nodes, whereas the transmission time depends on the message length. Since transmission time is independent of the routing algorithm under consideration, we use average setup time as our performance measure. A similar measure was adopted for performance evaluation in [7]. Our goal, in this study, is to assess the effect of the additional flexibility provided by our routing algorithms on the average setup time for a path. The results that we present will be helpful for a system designer to customize our routing criteria to a specific environment.

Extensive simulations were performed to evaluate the system performance. In our simulation model, messages are generated independently by every node according to a Poisson process. Every message generated by a node is destined towards any of the remaining $N - 1$ nodes with uniform probability. Message lengths are assumed to be exponentially distributed, with an average message length that corresponds to one time-unit. Other timing parameters are given with respect to this unit hereafter. In our simulator, buffer size at each node is assumed to be infinite. These assumptions might not correspond to the traffic patterns in a realistic hypercube multiprocessor. Nevertheless, this model provides us a basis for comparing our routing strategy with the *e*-cube strategy.

Before we present our results, we would like to mention that the *e*-cube algorithm performs very well under uniform traffic (that is, when the message generation rate is uniform at every node and messages generated by a node are destined towards every other node with equal probability). Under uniform traffic conditions, the amount of traffic routed through each link by the *e*-cube strategy can be shown to be identical. In other words, traffic load is divided uniformly among the links in the system. Such a uniform distribution of traffic among the links enables the system to operate without experiencing congestion at high traffic rates. It is easy to see that our routing criterion, in general, does not distribute traffic uniformly among the links in the system. However, our routing criterion allows more freedom in reserving a link, which may reduce the setup time for a path.

We categorize our results into two cases. In the first case, which we call *non-hierarchical*, we apply our routing criterion to the entire hypercube. As we shall see, in this case, our algorithms do not perform as well as the *e*-cube algorithm at high traffic rates. The second case, which we call *hierarchical*, strikes a balance between the inflexible *e*-cube strategy and our routing strategy. In this case, our simulation results show that our routing algorithms provide improvement over the *e*-cube routing algorithm.

A. Non-Hierarchical Case

Our routing criteria allow one to devise a variety of algorithms as opposed to a single one provided by the *e*-cube strategy. The assumptions used in our simulations corresponding to the non-hierarchical case are as follows. We use the UP routing criterion to route messages unless otherwise mentioned. In addition to the routing criterion, we require to specify two strategies which are given below.

AVAIL Strategy: For a given partial path, there may be more than one legal, available link at the source or intermediate nodes. The AVAIL strategy specifies how to choose the next link, out of the set of available legal links, at the source and intermediate node in order to advance the partial path.

UNAVAIL Strategy: Suppose that no legal link is available at the source or intermediate node. The UNAVAIL strategy specifies the waiting policy for the partial path to follow at the source or intermediate node.

Observe that there can be many different types of AVAIL strategies, such as selecting the available link of least dimension or selecting an available link randomly. Similarly, there can be many different types of UNAVAIL strategies, such as waiting for a legal link with the shortest queue-length or waiting for any legal link that first becomes available. We first examined the system performance with various AVAIL and UNAVAIL strategies, for cubes of dimensions up to five. Variation in the results that we obtained by varying the AVAIL and UNAVAIL strategies was insignificant. Hence, for the rest of simulations we used a fixed AVAIL and another fixed UNAVAIL strategy. The AVAIL strategy that we adopted can be described as follows: if the link of least legal dimension is available, reserve it; otherwise, choose any available legal link randomly and reserve that link. The UNAVAIL strategy that we used allows a partial path to wait for the first legal link that becomes available. If there is more than one partial path waiting at an intermediate node, they are treated on a first-

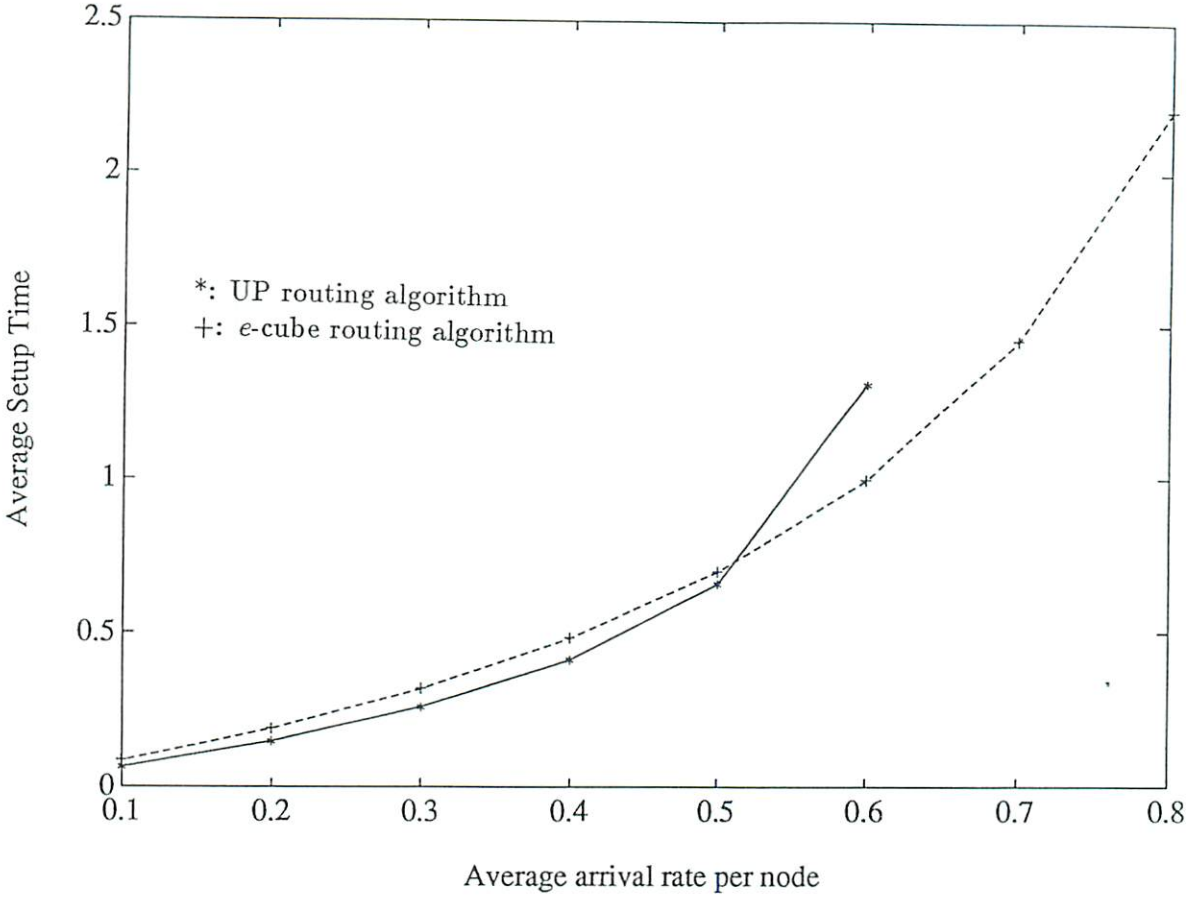

Figure 4: Performance comparison of the UP criterion and e-cube strategy for 3-cube.

come-first-served (FCFS) basis when an outgoing link from the intermediate node is released. For example, let us consider that a message, which is currently at intermediate node (01110), is destined for node (10100) in a 5-cube. There are two legal links, namely $14_1 \downarrow$ and $14_4 \uparrow$. If both of them are available, link $14_1 \downarrow$ is reserved. If both are currently reserved for other s-d pairs, the path will wait for the first one to be released. However, if another path arrived at the intermediate node (01110) earlier and is also waiting for link $14_4 \uparrow$, it will reserve link $14_4 \uparrow$ when the link is released as per the FCFS discipline.

From the simulation results, we found that balanced traffic on each link is a crucial factor in achieving a lower average setup time at high message generation rates. For example, Figure 4 presents the average setup time versus the average generation rate of messages per node in a 3-cube. It shows that our algorithm outperforms the e-cube strategy until a message generation rate of 0.51 per node and deteriorates quickly thereafter. These results can be explained as follows. Flexibility provided by our algorithm leads to uneven distribution of the traffic among the links in the system and, hence, to uneven utilization of the links, while allowing more freedom in reserving a link on the

other hand. These two factors compensate each other. Uneven utilization could lead to excessive traffic (congestion) at some intermediate nodes. Additional freedom allows a path to advance with more alternative routes, hence reducing path setup time. At lower message generation rates, the effect of congestion caused by uneven distribution of traffic is not as significant as that of additional freedom. Hence, flexibility provided by our algorithm helps in reducing the average setup time over the *e*-cube algorithm at lower message generation rates. As the message generation rate increases, our algorithm utilizes the links more unevenly. This uneven utilization of links becomes dominant eventually, and hence setup time increases.

In summary, at lower message generation rates the flexibility of the algorithm is advantageous, for it leads to lower queueing delays. At higher message generation rates, the same flexibility is disadvantageous, for it leads to uneven link utilization and, hence, to congestion. Also, as the dimension of the hypercube is increased, the cutoff message generation rate below which our algorithm outperforms the *e*-cube algorithm decreases indicating that the effect due to congestion becomes critical at lower message generation rates for higher cube dimensions. For example, this cutoff message generation rate is less than 0.5 for a 5-cube.

In order to reduce the negative effect due to congestion, we restricted our algorithm in such a way that at most one *non-sequential link* can be reserved for every s-d pair during the course of the routing. An up-link of dimension i that emanates from intermediate node I is termed a *non-sequential* link with respect to the s-d pair (S, D) if i is not the least significant bit position in which I and D differ. In other words, we limited the flexibility provided by our algorithm in reserving up-links. It turned out that this modification is helpful. Even with this modification, the average setup time achieved by our algorithm is significantly higher than that achieved by the *e*-cube algorithm at high message generation rates. However, in the rest of this study, we use this restricted version of our algorithm.

B. Hierarchical Case

We have just seen that a direct application of our routing criteria on the entire hypercube improves performance, with respect to the *e*-cube strategy, only over a limited range of message generation rates. In order to reduce the negative effect due to congestion while taking advantage of the flexibility, we applied our routing criteria in a hierarchical fashion. For convenience, we refer to this type of algorithms as *hierarchical* algorithms. Consider an n-dimensional hypercube. We divide the n-cube into m, $m > 0$, levels, by partitioning the number n into m positive integers

$n_0, n_1, \ldots, n_{m-1}$, such that $n = n_0 + n_1 + \ldots + n_{m-1}$. Let $n_{0,i} = \sum_{j=0}^{i} n_j$ for $0 \leq i \leq m-2$. For notational convenience, let $n_{0,-1} = 0$ and $n_{0,m-1} = n+1$. The hierarchical strategy that we adopt is characterized by the following set of rules:

1. A required link of a dimension greater than or equal to $n_{0,i}$, $0 \leq i \leq m-2$, cannot be reserved, before all required links of dimensions less than $n_{0,i}$ are reserved.

2. Required links of dimensions in the range $[n_{0,i}, n_{0,(i+1)})$, $-1 \leq i \leq (m-2)$, are reserved by applying any deadlock-free routing algorithm.

In other words, we divided the cube into m levels of hierarchy. As a result, links of the n-cube are divided into m independent sets with each set containing links of dimensions that correspond to a particular level. Notice that, at each level i, the n-cube is virtually partitioned into 2^{n-n_i} n_i-subcubes. For example, if a 5-cube is partitioned into $5 = 2 + 3$, there are $8 = 2^3$ 2-subcubes and $4 = 2^2$ 3-subcubes in levels 0 and 1, respectively. The 2-subcubes contain links of dimensions 0 and 1, while the 3-subcubes contain links of dimensions 2, 3 and 4. It is easy to observe that, in such a hierarchical structure, no two subcubes share a link. Within each subcube, any deadlock-free routing strategy can be applied independent of the other subcubes. Hierarchical routing strategy is deadlock-free since links that belong to different levels of the hierarchy are reserved in a strictly increasing order of the levels, and subcubes in the same level do not share links. The proof is straightforward and, hence, is omitted. Basically, we intend to reduce load imbalance by imposing strict order among all levels while allowing freedom at each level.

A large number of routing algorithms that adhere to the hierarchical strategy can be developed. In the results that we present, we used the hierarchical strategy for which the routing algorithm in each subcube follows either the UP routing strategy described in non-hierarchical case or the *e*-cube strategy. Next, we compare the performance of the hierarchical strategy with that of the *e*-cube routing strategy.

B.1 Performance Comparison

Associated with a given hypercube, there is a certain *operational range* of message generation rates. System becomes *unstable* (or the associated average setup time for a path becomes unreasonably large) when the message generation rate goes beyond this operational range. In fact, this range dwindles as the size of the hypercube increases, owing to the fact that the contention for links is higher in larger cubes. As described earlier, *e*-cube strategy maintains good load-balance among all

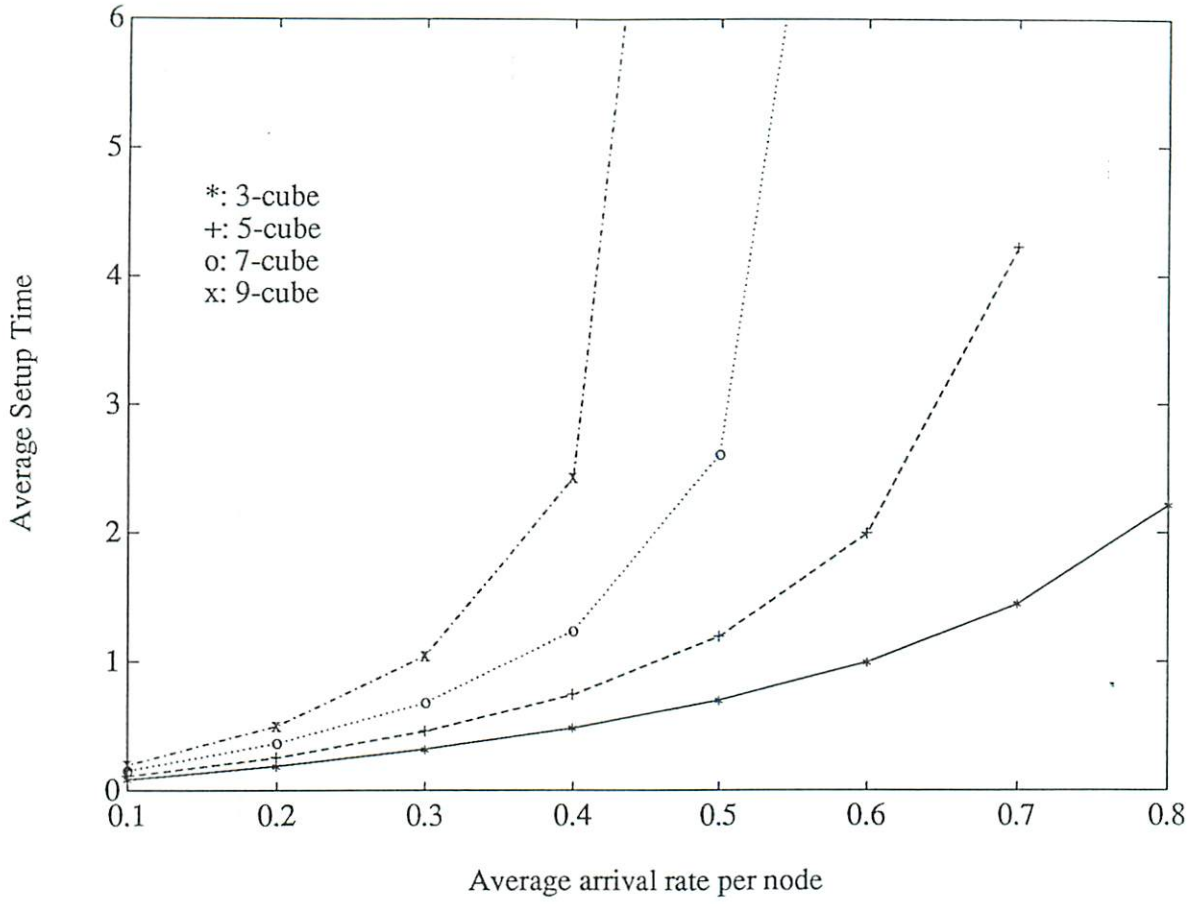

Figure 5: Operational ranges, in terms of message generation rate, with e-cube routing for hypercubes of various sizes.

links in the hypercube under uniform traffic conditions. It is also shown in [7] that e-cube algorithm provides good throughput. Therefore, we can use the e-cube strategy to obtain information about the operational range for a hypercube of a given size. Figure 5 is presented for this purpose. It shows operational ranges for hypercubes of various sizes. For example, for a 9-cube, message generation rate per node should be within 0.5 in order to have a reasonable setup time. Thus, while comparing performance of the hierarchical strategy with that of the e-cube strategy, we restrict ourselves to message generation rates that are within the operational ranges shown in Figure 5.

In the figures shown in this section, we use the notions n_i^f and n_i^e to represent, respectively, that our UP routing algorithm and e-cube strategy are used in all subcubes of level i of size n_i. In Figure 6, we show the performance with the hierarchical strategy for a 5-cube, when the 5-cube is partitioned into two levels ($5 = 2 + 3$) and when the flexible routing algorithm is adopted in each subcube. Hierarchical strategy has lower average setup times compared to the e-cube strategy for message generation rates that are below 0.5. Observe that the range $[0, 0.5]$ of message generation

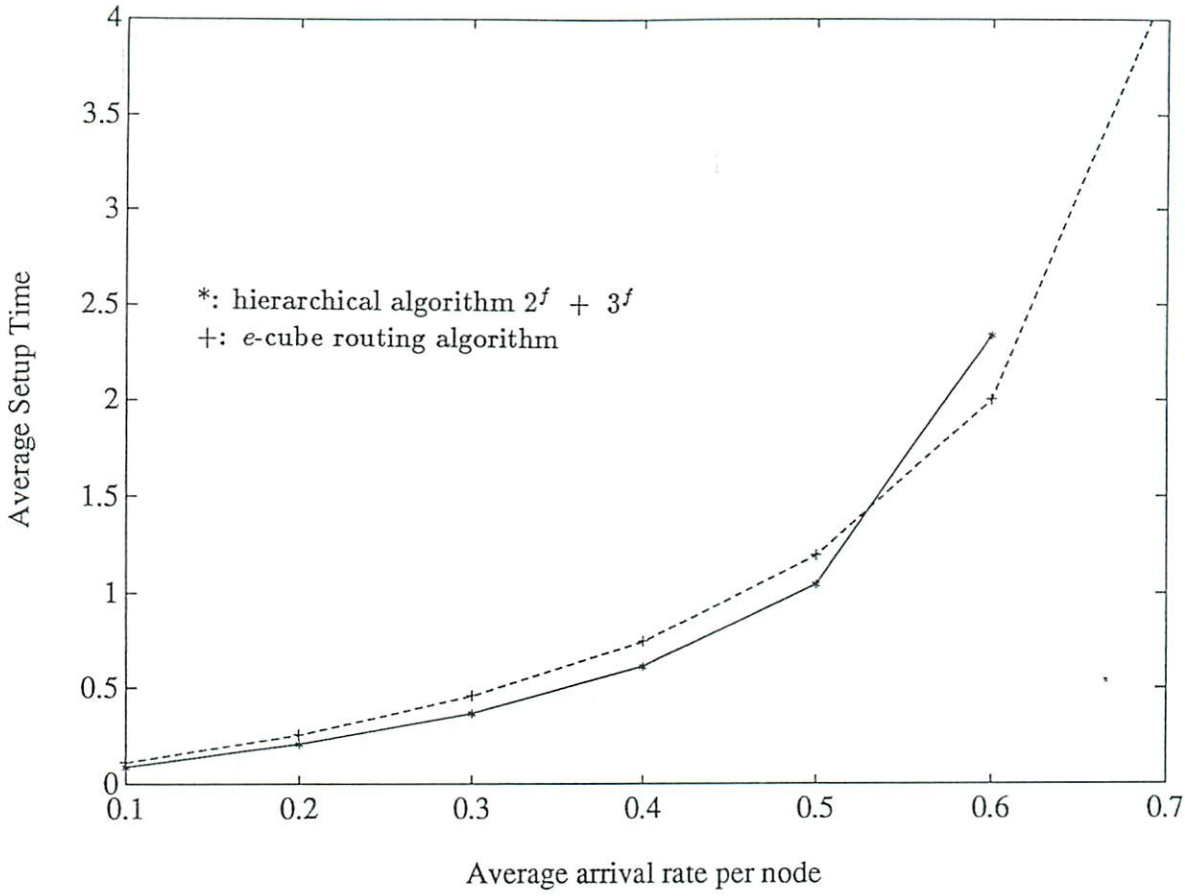

Figure 6: Setup time vs. message generation rate per node using hierarchical routing strategy and *e*-cube routing in 5-cube.

rates constitutes a major portion of the operation range for a 5-cube. For example, at the message generation rate of 0.4, the hierarchical strategy offers an improvement of 17.44% in the setup time over the *e*-cube strategy.

Figure 7 presents the performance of the hierarchical and the *e*-cube strategies in a 7-cube. The results shown in Figure 7 correspond to a 3-level partition $7 = 2 + 2 + 3$. The e-cube strategy is adopted in the subcubes of level 0 and the UP routing strategy is used in the subcubes that belong to levels 1 and 2. Notice that the routing strategy used for the highest five dimensions in this case is exactly the same as the one previously used in Figure 6 for the 5-cube. From Figure 7, it can be inferred that the hierarchical strategy offers reasonably low average setup times over the entire operational range of the 7-cube. From Figure 8, which compares the performance of the *e*-cube algorithm with those of two hierarchical strategies, it is easy to observe that the hierarchical strategies show performance-improvement over the whole operational range of the 9-cube.

Considering the large number of possible alternatives for choosing a hierarchical strategy, it

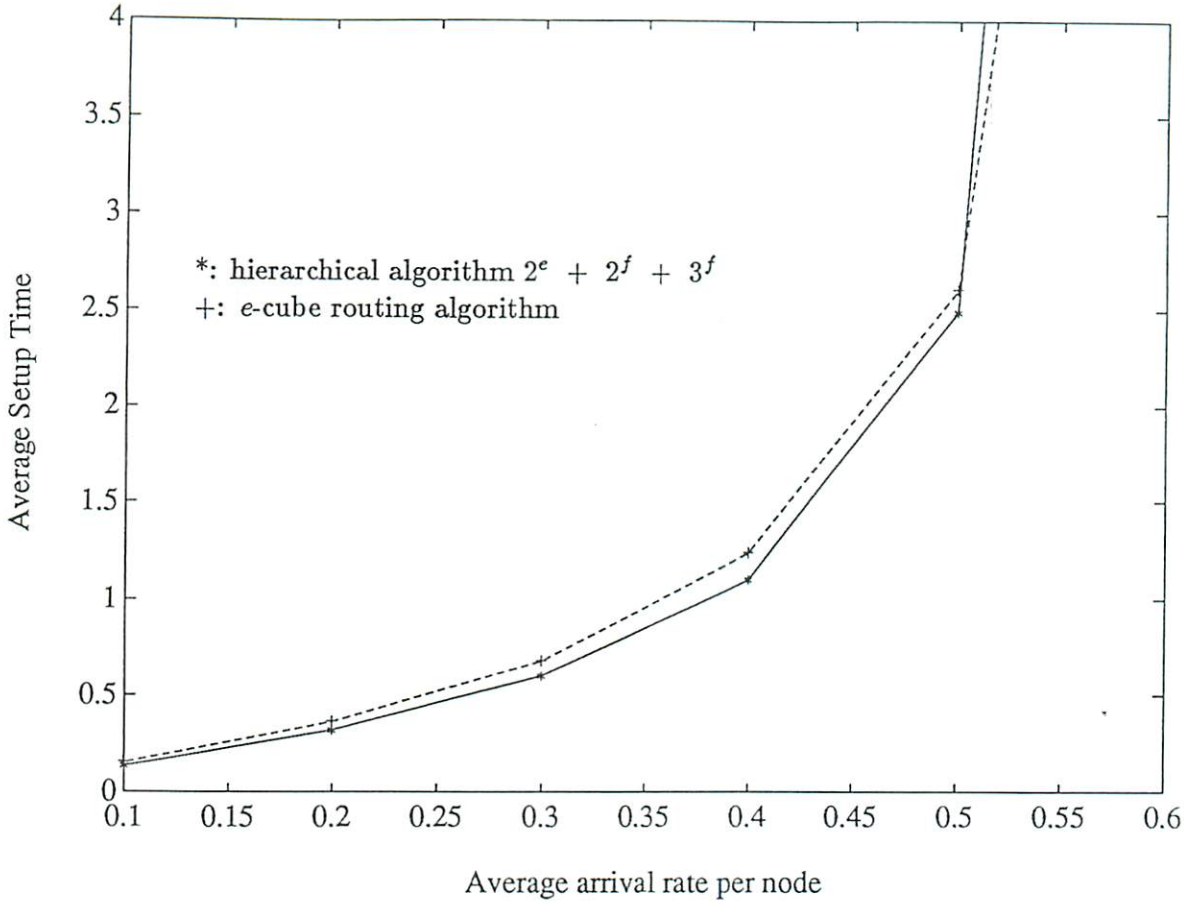

Figure 7: Setup time vs. message generation rate using hierarchical and e-cube routing in 7-cube.

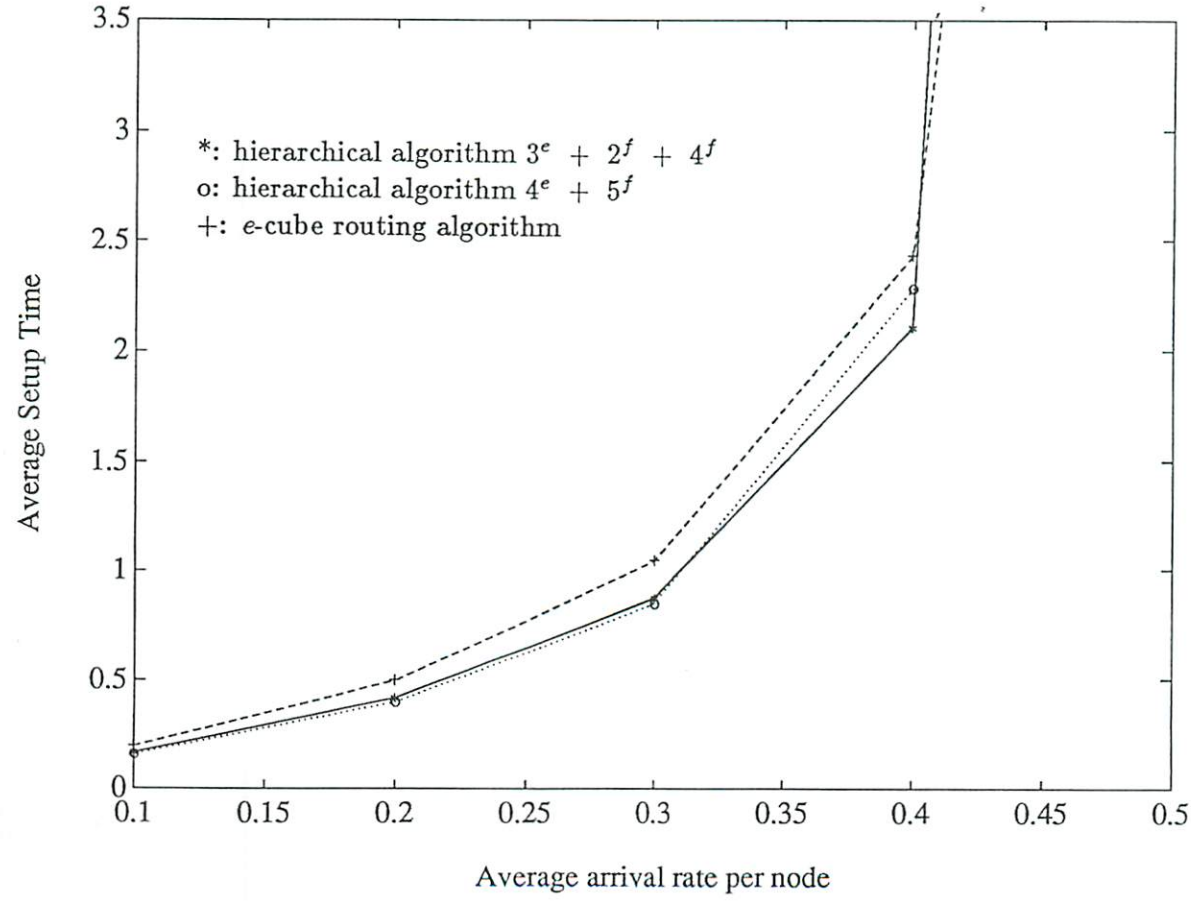

Figure 8: Setup time vs. message generation rate using two different hierarchical and e-cube routing algorithms in 9-cube.

is difficult to provide an algorithm for arriving at an "optimal" hierarchical strategy. However, the following rules, if adopted, can lead to good hierarchical strategies:

- If e-cube strategy is used at all, it should be used at the lowest level, i.e. level 0, to reduce the negative effect due to congestion.
- "Good" partitions for the hypercube of a given size can be constructed from "good" partitions corresponding to hypercubes of smaller sizes.

VI Conclusion Remarks

In this paper, we have proposed a set of routing criteria, based on which flexible, fault-tolerant routing algorithms for circuit-switched communication in hypercubes can be designed. As opposed to the well-known e-cube routing strategy, these criteria allow multiple shortest paths between a large number of source-destination pairs. In fact, the e-cube routing algorithm can be derived as a special case of our criteria. Fault-tolerance capability of our routing criteria has been shown to be superior to that offered by the e-cube routing algorithm. In particular, by relabeling the dimensions and nodes of the hypercube, we can minimize the negative effect due to component failures. We studied the performance of our routing criteria for the case of uniform traffic. By employing our routing criteria in a hierarchical manner, we have been able to demonstrate performance-improvement over the e-cube routing algorithm.

There are still more issues to be investigated with respect to these routing criteria. We are currently evaluating the performance of these criteria for the case of non-uniform traffic patterns. For example, congestion due to hot-spot traffic might be alleviated if our routing criteria are adopted in the hypercube system. Better partitioning-strategies to implement our routing criteria in a hierarchical fashion still need to be investigated. We are also currently investigating the effectiveness of our routing criteria under multiple component failures.

Acknowledgements

The authors remain indebted to Rajendra Boppana for the many insightful, valuable discussions and to Chien-Ming Cheng for providing crucial support on the implementation of the simulator.

References

[1] S. H. Bokhari. "Communication overhead on the Intel iPSC-860 hypercube". Technical report, NASA ICASE Report No. 90-38, May 1990.

[2] R. Boppana and C. S. Raghavendra. "Optimal self-routing of linear-complement permutations in hypercubes". In *The Fifth Distributed Memory Computing Conference*, pages 800–808. The University of Southern Carolina, April 1990.

[3] Ming-Syan Chen and Kang G. Shin. "Processor allocation in an n-cube multiprocessor using gray codes". *IEEE Trans. Computers*, C-36(12):1396–1407, December 1987.

[4] W. J. Dally and C. L. Seitz. "Deadlock-free message routing in multiprocessor interconnection networks". *IEEE Trans. Computers*, C-36(5):547–553, May 1987.

[5] A. L. DeCegama. *"The Technology of Parallel Processing – Parallel Processing Architectures and VLSI Hardware Volume 1"*. Prentice Hall, Inc., 1989.

[6] P. J. Denning. "Parallel computing and its evolution". *Commun. ACM*, 29:1163–1167, December 1986.

[7] N. J. Dimopoulos, D. Radvan, and K. F.Li. "Performance evaluation of the backtrack-to-the-origin-and-retry routing for hypercycle-based interconnection networks". In *Int. Conf. on Dist. Comp. Sys.*, pages 278–284. IEEE, May 1990.

[8] Y. Saad and M. H. Schultz. "Topological properties of hypercubes". *IEEE Trans. Computers*, 37(7):867–872, July 1988.

[9] C. L. Seitz. "The cosmic cube". *Commun. ACM*, 28(1):22–33, July 1985.