LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

Directed Cube Networks:

A Practical Investigation

CSG Memo 253

8 July 1985

Steve Heller

This research was supported by Advanced Research Projects Agency of the Department of Defence under Office of Naval Research contract no. N00014-75-C-0661.

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

Abstract

As research proceeds in the area of multi-processor systems, more emphasis is being placed on the networks needed to connect many processors into a single system. In this paper we examine a restriction on networks that are based on boolean n-dimensional hypercubes [6, 4], while preserving the important characteristics of the networks. We restrict communication paths between adjacent nodes to be unidirectional. The restricted networks use as little as half the communication paths and much smaller switches while achieving similar performance. As a result, much larger networks can be constructed for a given switch using the n-cube paradigm.

Key words and phrases: computer architecture, multiprocessor, n-cube, network, packet communication, store-and-forward network

Directed Cube Networks:

A Practical Investigation

1. Introduction

n-Cube networks[1] [6, 2] provide an elegant mechanism for connecting processors: inter-processor distances are small, and routing algorithms are both simple and flexible. n-Cube networks allow an exponential number of processors[2] to be connected for a given switch. The problem is that moderately large switches are difficult to build, and for that reason, current *state of the art* switches are small. The BBN Butterfly Switch [5] (a 4-input 4-output switch) allows us to build an eight processor network ($8 = 2^{4-1}$), as we will see. So, a number of processors exponential in the size of the switch is not very many at this time.

Cube Connected Cycles (CCC) [4], a variation on the hypercube theme, allows an arbitrarily large number of processors to be connected using fixed size switches, but we sacrifice path length, routing flexibility, and resilience to node failure.

CCC can be thought of as a restriction on an n-cube network. We will examine a different restriction allowing us to connect many more processors than in an n-cube while preserving many important properties of the n-cube (*e.g.*, average path length, flexibility and simplicity in routing, and resilience to node failure). We restrict the direction of communication between adjacent nodes in the n-cube to be unidirectional, without otherwise changing the structure of the n-cube. In this way, a hypercube of twice the dimension can be constructed for a given switch.

2. Graphs Versus Networks

Graphs are a convenient and powerful abstraction for dealing with communication networks, but we must be careful to consider all the practical implications of our abstract models. Nodes (in the graph) correspond to switches (in the network), and directed arcs correspond to links. Several caveats remain:

- dealing with undirected arcs
- integrating processors into the network
- finding paths in the network

Network links are typically unidirectional hardware. In order to establish a bidirectional path,

---

[1] Networks based on boolean n-dimensional hypercubes.

[2] exponential in the number of switch inputs

two anti-parallel[3] links are needed. This is certainly true if information is to be sent both ways at the same time. More importantly, bidirectional links require a sender and receiver at each end. We will "implement" undirected arcs as bidirectional links, or equivalently (from the point of view of the end connections), as two anti-parallel unidirectional links.

Integrating processors into a network is a more subtle task. We might view a network as separate from the processors; the processors sit on one side of the room, and the network on the other. However, we will view processors as integrated parts of the network. A processor will be modeled as having a single input and a single output; the input and output can attach to any switch output and input (possibly two different switches).

Can't we just build a network of switches and simply associate a processor with some or all switches? The processor can then siphon off appropriate messages and inject its own on the outputs. Absorbing messages is easy, but the ability to inject messages implies that the underlying n^2 switch has an input and output port devoted to the local processor and is therefore an $(n+1)^2$ switch. Since this is the case, we will model the processors explicitly.

One possibility is to put a processor in every arc. If we replace each arc of an n-cube by a *dual-ported processor* [2], $n \cdot 2^{n-1}$ processors can be connected using an n^2 switch. However, this requires switching at the processors in addition to switching at the switches (see Figure 2-1).

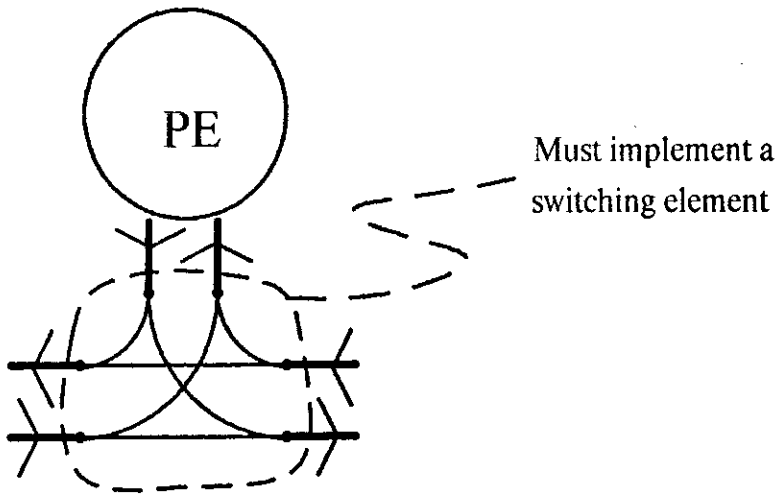

Figure 2-1: A Processing Element (PE) at Each Arc of an n-Cube

However the switching is done, buried in these additional switches will be the electrical arbitration of signals[4]. Since this is a fundamental limit in a switching network (one we are

[3]Between the same nodes but oriented in opposite directions.

[4]This is true even if two PEs are associated with each arc, one for each direction.

concerned with), all path lengths effectively double. We favor connecting one processor to each switch by reserving an input and output port. Examples of this technique will be given (see Figures 3-2 and 3-4).

In a graph, there are usually several paths from one node to another, and several techniques for finding a path. We can find some or all paths in advance and build a table or function that provides paths between two given nodes. This information can be compressed into incremental form; a table or function can tell us the next step of our path. These techniques will be referred to as *complete path generation* and *incremental path generation*, respectively.

Now that the path information is available, how should we decide which path to take? We could pick a path in advance for each pair of nodes (or choose a path at random on demand), or we might make a decision based on traffic and congestion caused by other paths[5]. We call these techniques *static* and *dynamic* routing, respectively.

While complete path generation and incremental path generation are variations on a theme, static and dynamic routing are fundamentally different. Suppose two messages are to be routed simultaneously; we would like to find two arc-disjoint paths. Intuition tells us that dynamic routing will have a much better chance since we have no *a priori* knowledge of which messages will occur simultaneously.

The following is an example of incremental path generation and dynamic routing: a switch considers the set of incoming and queued messages and all possible next destinations. The switch now sends the largest set of messages that can be sent on distinct output ports. Although this example may be too complicated to implement, the technique appears to be quite powerful. Cube networks are quite flexible and lend themselves to this type of routing.

3. Directed n-Cubes

All the cube networks are based on the graph of the boolean n-dimensional hypercube. We begin by presenting the graph for the n-cube of dimension n, NC^n [1].

3.1. n-Cubes

The n-cube [6] is defined as follows. Each of 2^n (N) nodes is labeled from 0 to N-1 by a unique binary string of length n. There is an arc between nodes i and j if i and j differ in exactly one bit position.

---

[5] In general there are several paths of concern at any given time.

$NC^n = (V, E)$
where
$V = \{i \mid 0 \leq i \leq 2^n-1\}$
$E = \{(i,j) \mid i, j \in V;$
$i = j//q^6$ for some q *such that* $0 \leq q < n$

Now we can build a network from the graph. Each node of the n-cube graph is replaced by an n^2 switch, and each arc is replaced by a bidirectional link. The n-cube network requires N n^2-switches, and n•N communication links (bidirectional links count twice). The n-cube of dimension 3 is shown in Figure 3-1.

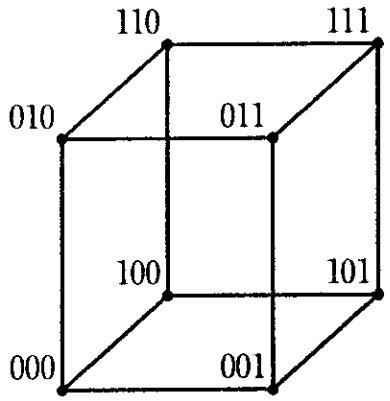

Figure 3-1: n-Cube Network of Dimension 3 (n=3)

If each node of the n-cube were replaced by an $(n+1)^2$ switch, a *Processing Element* (PE) or *external processor*[7] could be attached to each switch using the extra pair of input and output ports. In this way, the n-cube can be used to send messages among 2^n external processors. This kind of network will be referred to as an *n-Cube Processor Network.* An n-cube processor network of dimension 3 is shown in Figure 3-2.

Although the number of processors that can be connected using the n-Cube Processor Network is exponential in the size of the switch, moderately large switches are difficult to build[8]. For example, the BBN Butterfly switch [5] (a 4^2 *state of the art* switch) allows n+1 (the exponent) to be as large as 4. Hence we can build a $2^{4-1} = 2^3 = 8$ processor network using the n-cube. So even an exponential number of processors is not very many at this time.

---

[6] j//q is j with the qth most significant bit flipped.

[7] a processor external to the switch itself

[8] Each switch must act like a $(n+1)^2$ crossbar.

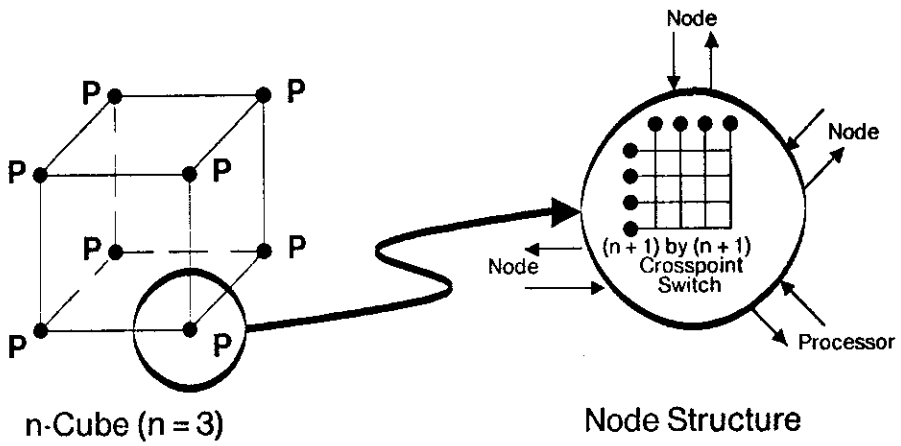

Figure 3-2: A Switch's View of an n-Cube Processor Network

3.2. Directed n-Cubes

Key Idea: Why not direct all the arcs of the n-cube, cutting the in-degree and out-degree of each node in half. In that way, *cubes of larger dimension (twice) can be built to connect more processors (N^2) with the same size switch ($\log_2 N$ by $\log_2 N$).* Conversely, the same number of processors can be connected with half the wires and much simpler switches.

The following questions immediately arise. Can we assign directions to the arcs in a simple and uniform fashion? Is the resulting graph connected? Will the derived network retain the following desirable properties?

- short inter-node distances
- many paths that are easy to generate
- resilience to node failure

Before defining the graph for the directed n-cube, DC^n, we define the *parity of a node* and the *parity of an arc* for n-cubes.

- If a node has an even number of ones in its binary label, the node has *even parity.* Otherwise, the node has *odd parity.*
- Each arc connects two nodes whose labels differ only in the qth *Most Significant Bit* (MSB) ($0 \leq q \leq n\text{-}1$). If q is even, the arc has *even parity.* Otherwise, the arc has *odd*

parity[9].

Node Alternation Lemma: All even nodes are adjacent to odd nodes only, and vice-versa. Equivalently, all arcs have an even node at one end, and an odd node at the other end. (*i.e.*, Parity partitions DC^n into a bipartite graph.)

Proof: Adjacent nodes differ by exactly one bit. Traversing an arc must therefore change the parity of a node. □

The graph for the *Directed n-Cube* of dimension n, DC^n, is derived from NC^n as follows:

- Orient all even arcs from the even node to the odd node, and
- orient all odd arcs from the odd node to the even node.

$DC^n = (V, E)$
where
$V = \{i \mid 0 \leq i \leq 2^n-1\}$
$E = \{(i,j) \mid i, j \in V;$
$\quad$ even-node $(i) \Rightarrow i = j//q$ for some even q *such that* $0 \leq q < n$
$\quad$ odd-node $(i) \Rightarrow i = j//q$ for some odd q *such that* $0 < q < n$

The *Directed n-Cube Network* is constructed by replacing all nodes by $(n/2)^2$ switches[10], and by replacing all arcs by unidirectional links. The directed n-cube network requires N $(n/2)^2$-switches, and n•N/2 communication links. In comparison with the n-cube network of the same dimension, the directed n-cube uses much smaller switches (perhaps 1/4 the size) and half as many links.

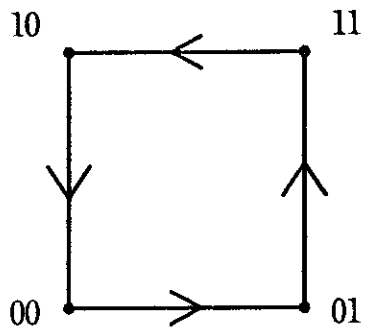

Figure 3-3: The Directed 2-Cube

---

[9]As pointed out by Professor Charles Leiserson, any lexicographic division of the bits, for example low order versus high order bits, is equally acceptable.

[10]For simplicity, assume that n is even. Odd dimensioned directed n-cube networks will be discussed in Section 6. n corresponds to the dimension of the hypercube (as opposed to the size of the switch) in both the bidirectional and the unidirectional cube networks.

To use the directed n-cube to connect external processors, use $(n/2 + 1)^2$ switches at each node. Establish a bidirectional link between the switch and the local PE, but maintain unidirectionality on all links that traverse dimensions of the hypercube. This *Directed n-Cube Processor Network* connects $2^{2(k-1)}$ processors using an equal number of k^2 switches. Using a 4^2 switch, $2^{2(4-1)} = 2^6 = 64$ processors can be connected (see Figure 3-4).

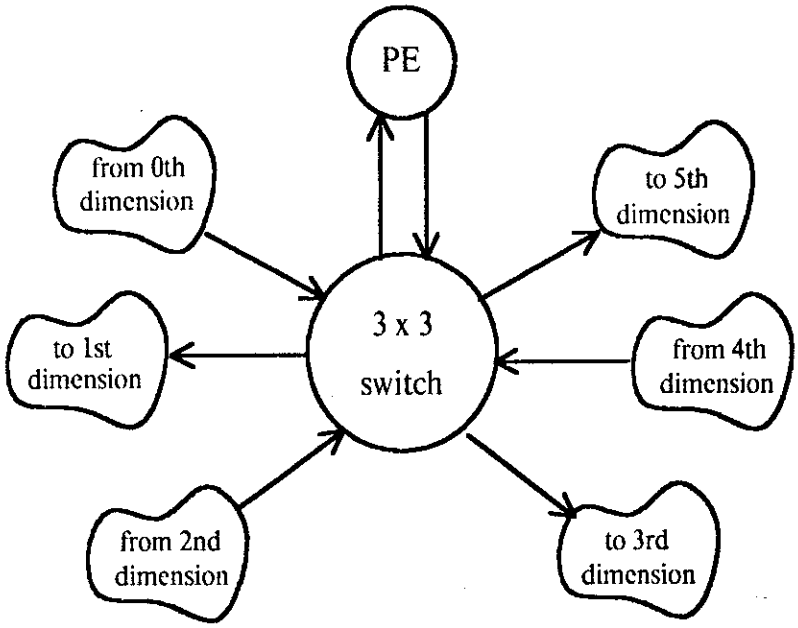

Figure 3-4: A Switch's[11] View of a Directed 4-cube Processor Network

The directed n-cube can be thought of as an n-cube with half of the links uniformly removed. There is no longer as much flexibility in routing packets. However, there are still very many paths, and nodes are about as close as they were in the n-cube, as we shall see.

In the n-cube, any dimension can be traversed from any node, but this is not true in the directed n-cube. How will routing be done? From any even node we can immediately traverse any even arc to an odd node, and from any odd node we can immediately traverse any odd arc to an even node. However, it may not be clear that there is a path from every node to every other node.

Connection Lemma: If s and t are nodes in the directed n-cube ($n \geq 2$), there is a directed path from s to t.

Proof: First we give an inductive construction for DC^k.

---

[11] The switch portrayed is resident at an odd network node.

- Basis: The basis is provided by the directed 2-cube in Figure 3-3[12].

- Induction: Given that we can construct n-cubes of dimension k ($k \geq 2$), construct an n-cube of dimension $k+1$ as follows. Construct two directed k-cubes, DC_0^k and DC_1^k. Reverse the parity of the nodes and arcs (and hence the direction of the arcs) in DC_1^k. Replace the label of each node in DC_0^k by the old label with a "0" concatenated on the left. Replace the label of each node in DC_1^k by the old label with a "1" concatenated on the left (now the parity of the nodes and arcs of DC_1^k are assigned as in the original construction). To synthesize a directed n-cube of dimension $k+1$, connect the corresponding nodes of DC_0^k and DC_1^k (those differing in only the $k+1$st bit) from the even node to the odd node if $k+1$ is even, and from the odd node to the even node if $k+1$ is odd.

We can see inductively that there is a path between every pair of nodes in the n-cube of dimension k since there is in the directed 2-cube, and since constructing a larger cube from two smaller ones orients half of the arcs in each direction between the sub-cubes. □

4. Routing in the n-Cube and the Directed n-Cube

Following a cube arc[13] corresponds to traversing a dimension in the boolean n-space of the node labels. Since following an arc changes exactly one bit of the node label we can think of each traversal as a one bit *correction* in our travel to any destination. So, finding a path from one node to another simply involves changing one bit at a time until we arrive at the destination *i.e.*, the start node label has been transformed into the destination node label by flipping one bit at a time.

Suppose we are at node s and wish to travel to node t by moving along cube edges. After taking τ steps we will have reached node s^τ. Let Δ^τ be the bitwise difference between the current node and the destination, a binary vector of length n. $\Delta^0 = s \oplus t$, the bitwise exclusive-or of s and t, and $\Delta^\tau = s^\tau \oplus t$. The ones in Δ^τ correspond to the dimensions that remain to be traversed after τ steps. If we traverse dimension $q^{\tau+1}$ in the $\tau+1$st step, $s^{\tau+1}$ will be $s^\tau // q^{\tau+1}$), and $\Delta^{\tau+1}$ can be derived from Δ^τ as follows:

- $\Delta^{\tau+1} = \Delta^\tau // q^\tau$

If we reach t after γ steps, $q^1 q^2 \ldots q^\gamma$ is the path[14] we followed.

---

[12] The basis could have been a directed 0-cube (a single node of even parity labeled 0) for the purpose of the construction. However, the Lemma is only true for n-cubes of dimension 2 or higher.

[13] An arc of either the n-cube or the directed n-cube.

[14] A path from a node is denoted by a sequence of dimensions.

It is inconvenient, however, to work with index vectors[15]. Let Q^τ be the set of integers whose index vector is Δ^τ ($q^\tau \in Q^\tau \Leftrightarrow$ the q^τth MSB of Δ^τ is 1). As we move along edges of the cube, Q^τ will be the set of dimensions in which s^τ differs from t (the set of dimensions in which the current node differs from the destination node). If we traverse dimension $q^{\tau+1}$ in the $\tau+1$st step, $Q^{\tau+1}$ can be derived from Q^τ as follows:

- $Q^{\tau+1} = Q^\tau - \{q^{\tau+1}\}$ if $q^{\tau+1} \in Q^\tau$ (correct the $q^{\tau+1}$th bit)
- $Q^{\tau+1} = Q^\tau \cup \{q^{\tau+1}\}$ if $q^{\tau+1} \notin Q^\tau$ (introduce an *artificial transition* on the $q^{\tau+1}$th bit to be corrected later)

Let's derive a path from s to t in an n-cube[16]. If we choose $q^{\tau+1} \in Q^\tau$, we will get closer to t, and if we choose $q^{\tau+1} \notin Q^\tau$, we will get further away from t. Any permutation of the elements of Q^0 is a shortest path. The n-cube allows us to correct in any order the bits in the source node that differ from those in the destination node. The directed n-cube, however, only allows us to correct even bits from even nodes and odd bits from odd nodes, so there are some constraints on the order in which bits can be corrected.

For any τ, let Q_e^τ be the even subset of Q^τ, and let Q_o^τ be the odd subset of Q^τ. Q_e^τ and Q_o^τ enumerate the even and odd dimensions that still need to be traversed after the τth step.

The following lemmas will be helpful in our discussion.

Path Alternation Lemma: All legal paths in the directed n-cube alternate between even and odd nodes and between even and odd arcs.

Corollary 1: The number of even traversals in any directed n-cube path is within one of the number of odd traversals.

Corollary 2: The length of any directed n-cube path is even if the source and destination nodes have the same parity, and odd if the source and destination nodes have different parity.

Proof: The lemma and corollaries follow directly from the Node Alternation Lemma. □

n-Cube Lemma: The shortest directed n-cube path between two nodes is bounded below by $|Q^0| = |Q_e^0| + |Q_o^0|$.

Proof: $|Q^0|$ is the length of the shortest path in the n-cube. The lemma follows since the set of paths in the directed n-cube is a subset of the set of paths in the n-cube of the same dimension, and since Q_e^0 and Q_o^0 partition Q^0. □

Directed n-Cube Lemma: The shortest directed n-cube path between two nodes is bounded below by $2 \cdot \text{MAX}(|Q_e^0|, |Q_o^0|) - 1$.

---

[15] Δ^τ is the index vector for the dimensions in which s^τ differs from t.

[16] Wandering about the n-cube is well understood. This section is to develop intuition for the newcomer.

Proof: The lemma follows from the first corollary of the Path Alternation Lemma and the fact that every dimension in the larger of Q_e^0 and Q_o^0 must be traversed. □

Corollary: The shortest directed n-cube path between two nodes of the same parity is bounded below by twice MAX($|Q_e^0|$, $|Q_o^0|$).

Proof: This tighter bound follows from the lemma and the second corollary of the Path Alternation Lemma. □

Parity Lemma: If the source and the destination in a directed n-cube have the same parity, $|Q_e^0| + |Q_o^0|$ is even. If the source and the destination in a directed n-cube have different parity, $|Q_e^0| + |Q_o^0|$ is odd.

Proof: $|Q_e^0| + |Q_o^0| = |Q^0|$, the length of the shortest path in the n-cube. The lemma follows from the Path Alternation Lemma. □

How to get from s to t.[17] Assume *without loss of generality* (*w.l.o.g.*) that s is an even node. We can traverse any (even) dimension $q_e^0 \in Q_e^0$ (*deleting the traversed dimension from* Q^0), then any (odd) dimension in Q_o^1, alternating until we cannot proceed because one of the sets is empty. If we are at t, stop. Otherwise, when this happens (say after τ steps), we will be at either an even node with Q_e^τ empty, or an odd node with Q_o^τ empty (or else we would not be stuck). Assume *w.l.o.g.* that the former is true. Now choose any even dimension 2g ($0 \leq 2g \leq n\text{-}1$). Traverse 2g while adding 2g both to Q^τ to get $Q^{\tau+1}$ and to Q_e^τ to get $Q_e^{\tau+1}$ (normally we would *delete* the dimension being traversed). Then traverse a member of $Q_o^{\tau+1}$, and once again traverse 2g. If there were exactly one (odd) dimension left at time τ (when we got stuck), we are now finished. If there is more than one dimension left, traverse another odd dimension. Then if necessary, choose a new 2g, and repeat the last step until you reach t.

How long a path is generated? If $|Q_e^0| = |Q_o^0|$, the length of the path is $|Q_e^0| + |Q_o^0|$, the same as the shortest path in the n-cube. However if $|Q_e^0| \neq |Q_o^0|$, a more detailed analysis is necessary.

Suppose t, the destination, is also an even node. When we are ready to chose our first 2g, we are in one of two situations:

- 1. We are at an even node. Since we have made $2 \cdot |Q_e^0|$ (say τ) steps so far, $|Q_o^\tau|$ must be even (by the Parity Lemma, $|Q_e^0| + |Q_o^0|$ = an even number = $2 \cdot |Q_e^0| + |Q_o^\tau|$). The length of the remaining path will be $2 \cdot |Q_o^\tau|$, so the length of the entire path will be $2 \cdot Q_o^0$, (since $|Q_e^0| + |Q_o^\tau| = |Q_o^0|$ when we got stuck), the lower bound as specified by the Directed n-Cube Lemma.

- 2. We are at an odd node. Since we have made $2 \cdot |Q_o^0| + 1$ (say τ) steps so far, $|Q_e^\tau|$ must be odd (by the Parity Lemma, $|Q_e^0| + |Q_o^0|$ = an even number = $2 \cdot |Q_o^0| + 1 + |Q_e^\tau|$). The length of the remaining path will be $2 \cdot |Q_e^\tau| + 1$, so the length of the entire path is $2 \cdot |Q_e^0|$ (since $1 + |Q_o^0| + |Q_e^\tau| = |Q_e^0|$ when we got stuck), the lower bound as specified by the Directed n-Cube Lemma.

---

[17] From this point until Section 6, we will only consider even dimensioned hypercubes.

Suppose t is an odd node. When we are ready to chose our first 2g, we are in one of two situations:

- 3. We are at an even node. Since we have made $2 \cdot |Q_e^0|$ (say τ) steps so far, $|Q_o^\tau|$ must be odd (by the Parity Lemma). The length of the remaining path will be $2 \cdot |Q_o^\tau| + 1$, so the length of the entire path is $2 \cdot |Q_o^0| + 1$, larger than the derived lower bound.

- 4. We are at an odd node. Since we have made $2 \cdot |Q_o^0| + 1$ (say τ) steps so far, $|Q_e^\tau|$ must be even (by the Parity Lemma). The length of the remaining path will be $2 \cdot |Q_e^\tau|$, so the length of the entire path is $2 \cdot |Q_e^0| - 1$, the lower bound as specified by the Directed n-Cube Lemma.

By the Directed n-Cube Lemma, this procedure generates shortest paths in cases 1, 2, and 4. We will consider case 3 more carefully. Since Q_o^0 is larger than Q_e^0, more odd transitions are indicated than even transitions. However, since s is even and t is odd, both the first and last transitions must be even and there is one more even transition than odd transition in every path from s to t. Since there must be at least $|Q_o^0|$ odd transitions, and at least one more even transition than that, the shortest path must be at least of length $2 \cdot |Q_o^0| + 1$, the length of the path constructed above. Hence, the above procedure only generates shortest paths.

We have generated the paths in an incremental fashion to provide intuition. Also, we have left the extra transitions until the end. But there is no reason why we cannot introduce and correct the necessary dimensions at any time. With this modification, all shortest paths can be generated.

5. N-Cube / Directed n-Cube Tradeoffs

5.1. Average Distances

What have we given up by restricting the directions of the arcs? The longest route[18] in the n-cube is of length n. The longest routes in the directed n-cube (from case 3 above with $|Q_e^0| = n/2 - 1$ and $|Q_o^0| = n/2$, for example) are of length $n+1$.

How many nodes are at the maximum distance from a given node? All longest routes are between nodes of different parity. Assume *w.l.o.g.* that s is even and t is odd. For a long route, $|Q_o^0|$ must be n/2, but $|Q_e^0|$ can take on the values n/2 - 1, n/2 - 3, *etc.* down to 0 or 1, and any choice i for $|Q_e^0|$ can be realized in $\binom{n/2}{i}$ ways.

For n/2 even, the number of *far nodes* (nodes at the maximum distance) for a given node is:

$$\sum_{\substack{1 \le i \le n/2 \\ \text{odd } i}} \binom{n/2}{i}$$

[18] *Route* refers to a shortest path between two nodes.

For n/2 odd, the number of far nodes for a given node is:

$$\sum_{\substack{0 \le i \le n/2 \\ \text{even } i}} \binom{n/2}{i}$$

Both of which can be rewritten (by reversing a summation, using the identity $\binom{n}{k} = \binom{n}{n-k}$, and doing some algebraic manipulation):

$$\sum_{i=1}^{\lceil n/4 \rceil} \binom{n/2}{2i-1}$$

This is a binomial expansion. The number of far nodes from a given node is $2^{n/2-1}$. The number of far nodes from a given node in the n-cube is 1.

What about average path length? Number crunching[19] gave the results in Table 5-1.

Table 5-1: Table of Average Distances for Directed n-Cubes

n	n/2	Average Distance
2	1	1.5
4	2	2.75
6	3	3.9375
8	4	5.09375
10	5	6.23046875

The average path length in the n-cube is n/2. And as we will show, the average path length in the directed n-cube[20] is bounded above and closely approximated by $n/2 + C\sqrt{n}$ for some constant C. Since all even dimensioned graphs are symmetric, the average distance over the whole network is the same as the average distance from *any* single source, say an even node. The distance to any even destination is $2 \cdot \text{MAX}(|Q_e^0|, |Q_o^0|)$ (call this quantity 2•Max). The distance to an odd destination is 2•Max-1 if $|Q_e^0| > |Q_o^0|$, and 2•Max+1 if $|Q_e^0| < |Q_o^0|$, *an equal chance.* So the expected distance between nodes is 2•Max. We turn to probabilistic analysis to find the expected value for Max.

Suppose P(i, m) is the probability that there are i *ones* in a binary string of length m. (i corresponds to either $|Q_e^0|$ or $|Q_o^0|$, and m = n/2 corresponds to the potential maximum sizes of Q_e^0 and Q_o^0.) P(i, m) is a binomial probability mass function (PMF):

---

[19] All path lengths were computed directly from the graph for the directed n-cube.

[20] For simplicity, we are still only considering even dimensioned directed n-cubes

$$P(i, m) = \binom{m}{i} (1/2)^i (1 - 1/2)^{m-i}$$

$$= (1/2)^m \binom{m}{i}$$

The variance σ^2 of this binomial is $m(1/2)(1-1/2) = m/4$, and the standard deviation σ is $(1/2)\sqrt{m}$.

Assume *w.l.o.g.* that we are calculating the expected distance from the node labeled 0. P(i, m) is the PMF for both $|Q_e^0|$ and $|Q_o^0|$. We would like the expected value for the maximum of two trials[21] with PMF P(i, m).

The probability that there are i or less ones in a string of length m ($P_{\leq}(i, m)$) is calculated by taking the cumulative sum (integrating discretely) from 0 to i.

$$P_{\leq}(i, m) = (1/2)^m \sum_{j=0}^{j=i} \binom{m}{j}$$

The probability that the maximum of two trials is less than or equal to i ,$P_{\leq}^{\text{max of 2}}(i, m)$, is $[P_{\leq}(i, m)]^2$. $P_{\leq}^{\text{max of 2}}(i, m)$ is known as the *second order statistic* for a sample of size two with PDF P(i, m) [3]. Hence the probability that the maximum of two trials is exactly equal to i ($P^{\text{max of 2}}(i, m)$) is calculated by taking the stepwise difference (differentiating discretely).

$$P^{\text{max of 2}}(i, m) = P_{\leq}^{\text{max of 2}}(i, m) - P_{\leq}^{\text{max of 2}}(i-1, m)$$

$$= [P_{\leq}(i, m)]^2 - [P_{\leq}(i-1, m)]^2$$

$$= (1/2)^{2m} \binom{m}{i} \{[2 \sum_{j=0}^{j=i} \binom{m}{j}] - \binom{m}{i}\}$$

The Table of Average Distances (Figure 5-1) was generated by finding and then averaging all shortest distances in directed n-cubes. The expected values of $P^{\text{max of 2}}(i,m)$ (Exp(m)) for various values of m should correspond to the Figure 5-1 entries as follows. Let E(n) be the expected distance in the directed n-cube of dimension n (the expected value for 2•Max).

---

[21] The following is an analogous problem (suggested by Professor Morton Tavel of Vassar College). Assume that you know the probability distribution for the time until the first failure of a machine. Given two of these machines, what is the expected time until both machines have failed.

$$Exp(m) = \sum_{i=0}^{i=m} i \cdot P^{\max \text{ of } 2}(i, m)$$

$$E(n) = 2 \cdot Exp(m) = 2 \cdot Exp(n/2)$$

$$E(n) = (1/2)^{n-1} \sum_{i=0}^{n/2} i \cdot \binom{n/2}{i} \{[2 \sum_{j=0}^{j=i} \binom{n/2}{j}] - \binom{n/2}{i}\}$$

The expected distances in the directed cube (E(n)) were calculated and are displayed as *predicted distances* in Table 5-2.

Table 5-2:
Table of Predicted Distances
(rounded to 3 significant figures)

n	E(n)	n	E(n)
2	1.5	40	22.507
4	2.75	50	27.807
6	3.938	60	33.077
8	5.094	70	38.326
10	6.230	80	43.337
12	7.354	90	48.774
14	8.466	100	53.979
16	9.571	110	59.175
18	10.669	120	64.361
20	11.762	130	69.540
30	17.167	140	74.712

The values in Table 5-2 match those in Table 5-1 *exactly* (the unrounded values, that is) as we expected. But is E(n) bounded by $n/2 + C\sqrt{n}$? Table 5-3 gives values for $\{E(n)-(n/2)\}/(\sqrt{n})$.

E(n) is approximated and bounded above for all $n \leq 140$ (up to 2^{140} nodes[22]).

$$E(n) \leq n/2 + .399\sqrt{n}$$

Does the form of this bound make sense? n/2 is the expected value for a single trial (the first order statistic for a sample of size one, and the distance in the undirected cube), and $\sqrt{n}$ is proportional to σ the standard deviation.

---

[22]Dr. Larry Stockmeyer is credited with the following observation: *2^{140} is approximately the number of proton sized objects it takes to compactly fill the known universe.* We do not expect to build any networks with more nodes than this.

Table 5-3: Table to determine C
(rounded to 3 significant figures)

n	$\frac{E(n)-(n/2)}{\sqrt{n}}$	n	$\frac{E(n)-(n/2)}{\sqrt{n}}$
2	.354	40	.396
4	.375	50	.397
6	.383	60	.397
8	.387	70	.398
10	.389	80	.398
12	.391	90	.398
14	.392	100	.398
16	.393	110	.398
18	.393	120	.398
20	.394	130	.398
30	.396	140	.398

We have a bound for the average distance, but how good an approximation is it[23]? Table 5-4 gives the difference between our bound and the actual expected value (bd(n) - E(n), *where* bd(n) = n/2 + .399 $\sqrt{n}$), and the percentage error ({bd(n)-E(n)}/E(n)).

Table 5-4: Table to check out C
(rounded to 3 significant figures)

n	bd(n) - E(n)	% error	n	bd(n) - E(n)	% error
2	.0643	4.28	40	.0161	.0715
4	.0480	1.75	50	.0145	.0521
6	.0398	1.01	60	.0133	.0402
8	.0348	.683	70	.0124	.0323
10	.0313	.502	80	.0116	.0267
12	.0287	.390	90	.0110	.0226
14	.0266	.314	100	.0105	.0195
16	.0250	.261	110	.0101	.0171
18	.0236	.221	120	.00973	.0151
20	.0224	.191	130	.00940	.0135
30	.0184	.107	140	.00910	.0122

Table 5-4 shows that we not only have an upper bound but a good approximation for the average

[23] We used .399 instead of .4 to avoid an apparent inconsistency due to roundoff error that would crop up.

distance for all directed cube networks that we can ever build.

5.2. Other Performance Measures

How fast can the directed n-cube route a permutation? I suspect about as fast as the n-cube can, but more study is required to show this. A permutation is not a very taxing test for an n-cube (or a directed n-cube for that matter).

What about throughput? First we must define what we mean by throughput. Define the *throughput* of a network as the number of messages delivered per unit time. In some abstract sense the directed n-cube has about half the potential throughput of the n-cube of the same dimension (due simply to the number of wires), but it is not at all clear that the additional capacity the n-cube provides is in any way usable. I would like to perform the following tests while recording times and queue sizes.

1. Permutation: Send a packet from each node to a distinct destination

2. Pipelined permutation: Start a permutation every j steps

3. Random routing: Send a packet from each node to a random destination

4. Pipelined random routing: Start a random routing every j steps

5. Local random routing: Send a packet from each node to a random destination in such a way that local destinations are more likely

6. Pipelined local random routing: Start a local random routing every j steps

Another variation of these tests is to return an acknowledge message for each message sent.

While these measures are difficult to derive analytically, they would provide insight into the meaning of throughput in this scenario. A network simulator would help.

5.3. Resilience to Node Failure

What happens to the average path length when a node can no longer be used to route packets? Even though the directed n-cube was generated by deleting links in an n-cube, it is still resilient to node failure. Even if a switch dies, processors can communicate using other paths. Table 5-5 contains information about distances with a failed node.

The percent increase in the average path length goes quickly to zero. Why is this so? There are many paths in a cube. For longer paths, a failed node can be avoided easily. The only pairs of nodes that will be further away are ones that are very close and require the broken node to achieve the shortest path. The fraction of paths that require a specific node is small, and diminishes quickly in larger cubes.

Table 5-6 compares n-cube processor networks with directed n-cube processor networks of the

Table 5-5:
Table of Directed N-cube Distances with a Single Node Failure
(rounded to 3 significant figures)

n	average distance	average distance with one node down	Δ in last two columns	% increase in path length
4	2.75	2.809	.0589	2.14
5	3.438	3.459	.0214	.622
6	3.937	3.946	.00808	.205
7	4.593	4.596	.00269	.0586
8	5.094	5.095	.000906	.0178

same dimension. The dimension of the hypercube is n.

Table 5-6:
Comparison of Undirected and Directed
n-Cube Processor Networks

Aspect	n-Cube	Directed n-Cube
Switch Size	$(n+1)^2$	$((n/2)+1)^2$
Number of Processors	2^n	2^n
Number of Links	$n \cdot 2^n$	$n \cdot 2^{n-1}$
Maximum Distance	n	n+1
Average Distance	n/2	$< n/2 + .399 \sqrt{n}$

5.4. Decomposability

All sub-cubes of an n-cube are proper n-cubes. This is not true in general for directed n-cubes. For a sub-cube of a directed n-cube to be a directed n-cube, the number of even dimensions (that the sub-cube spans in the original cube) must be within one of the number of odd dimensions.

6. Odd Dimensioned Directed n-Cube Networks

As long as we are prepared to provide buffering at each switch, there is no reason that we cannot build directed n-cubes for odd dimensioned hypercubes as well. Let $n^+ = \lceil n/2 \rceil$ and $n^- = \lfloor n/2 \rfloor$. Replace even nodes by n^+-input by n^--output switches, and replace odd nodes by n^--input by

n^+-output switches. The Directed 3-cube is pictures in Figure 6-1.

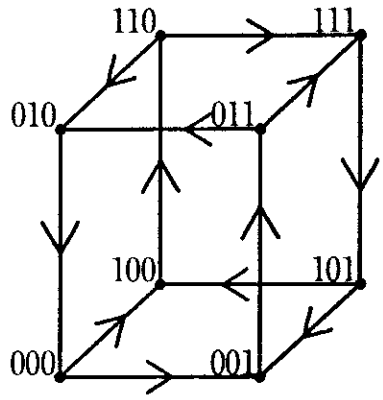

Figure 6-1: The Directed 3-Cube

It turns out[24] that the longest path for odd n is n+1 for even sources and n+2 for odd sources. Furthermore, both the number of distance n+1 nodes from a given even node and the number of distance n+2 nodes from a given odd node seem to be $2^{(n-1)/2}$, and the average distance in the directed n-cube of odd dimension seems to be E(n+1)-1/2. Table 6-1 is a chart for average path length (obtained by number crunching) including hypercubes of both even and odd dimensions.

Table 6-1: Table of Average Distances

n	n/2	Average Distance (overall)	Average Distance (even source)	Average Distance (odd source)
2	1	1.5	1.5	1.5
3	1.5	2.25	2	2.5
4	2	2.75	2.75	2.75
5	2.5	3.4375	3.25	3.625
6	3	3.9375	3.9375	3.9375
7	3.5	4.59375	4.4375	4.75
8	4	5.09375	5.09375	5.09375

In a directed n-cube with odd degree, odd nodes will have more incoming arcs than outgoing arcs. Are these nodes in danger of being overrun by packets? Is the pipeline balanced? This should not be a problem since all odd nodes are fed by even nodes suffering from the reverse problem. Suppose that all of the output arcs from all of the odd nodes are fully occupied. The output arcs

[24] These results from number crunching.

from the even nodes will be on the average (n-1)/(n+1) full, not enough to overload the odd nodes, so the pipeline is balanced.

7. Directed Cube Connected Cycles

Cube Connected Cycles (CCC) [4] are in a sense a restricted class of n-cubes. But the technique of directing arcs is an orthogonal restriction that can be applied to CCC as well. First we review CCC briefly. Then we show cursorily how the technique of directing arcs can be applied to CCC networks.

Construct a CCC as follows. We will need a new building block, a ring. Form a ring of size n by connecting n nodes in a circle. Label the nodes from 0 to n-1 so that consecutive numbers label adjacent nodes. Now, given an n-cube graph as defined above, replace all nodes by rings of size n. The qth node of each ring should be attached to the arc that corresponds to the qth bit changing (spans the qth dimension).

More precisely, each node labeled (r, p) has an n-bit ring label (r, indicating which ring we are on), and a $\lceil \log(n) \rceil$ bit position label (p, indicating our position within the ring)[25]. The node labeled (r, p) is connected to (r, p+1), (r, p-1), and (r//p, p); the two nodes adjacent in the same ring (forward and backward connections) and the corresponding node in another ring (lateral connection). The CCC of dimension 3 is shown in Figure 7-1.

The CCC presented here differ slightly from those presented in [4]. Preparata and Vuillemin only use rings whose size is a power of 2 ($\exp_2 \lceil \log_2 \text{dimension} \rceil$, the least power of two greater than or equal to the dimension of the cube). As a result, the number of nodes in their networks is a power of two. While this is important for their construction of the ASCEND and DESCEND algorithms, their CCC has some nodes with lateral connections and some without. Here, the nodes with no lateral connections have been deleted, and all nodes are equivalent.

CCC provide a technique for building networks that can emulate n-Cube networks while restricting the size of the switches to 3^2. *CCC Processor Networks* can be constructed from CCC networks in the obvious way using switches of size 4^2.

Routing is accomplished in the CCC in two stages:

1. Choose either the increasing or decreasing direction to shift around the rings (the choice will remain in force for the duration of this stage). At any node (say (r, p)) we can either correct the pth bit of r using the lateral connection to (r//p, p), or shift around the ring in the chosen direction: take the forward link to (r, p+1) or the backward link to (r, p-1). In this way, we shift through the values of the ring position p, correcting any bits of r at the appropriate time. In general we will need to shift all the way around the ring (n steps), but we need only correct the bits that are wrong (a maximum of n steps, and an average of n/2 steps). This stage will take 2n steps in the worst case and about 3n/2

[25] All arithmetic with p is *mod* $\lceil \log(n) \rceil$.

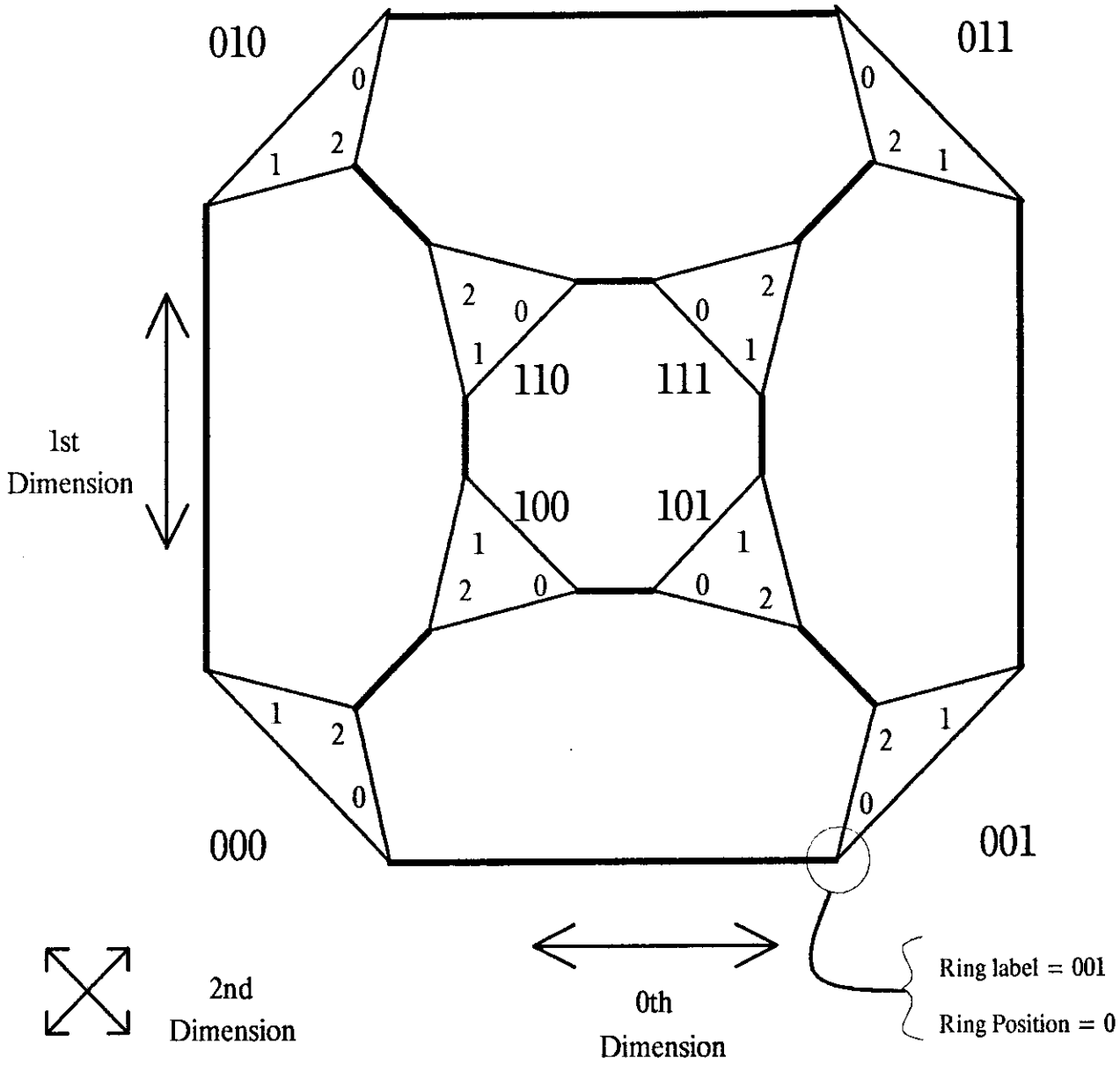

Figure 7-1: The Cube Connected Cycle of Dimension 3

steps on average.

2. At this point we must be on the right ring (r has been corrected), but we may be at the wrong ring position (the current value of p is wrong). Simply take the shortest path

around the ring to the destination (at most n/2 and on average n/4 steps).

The directed CCC (DCCC) is constructed using unidirectional rings (only forward connections, no backward connections) and bidirectional lateral connections. The switch labeled (r, p) is connected to (r, p+1) and (r//p, p). The DCCC is more restricted in its routing than the CCC. In stage 1, we cannot choose the direction to circulate around the ring (and hence the order in which we correct the bits of the ring label). This restriction does not have much effect on path length, but there are fewer paths. In stage 2, however, we will always have to go the same way around the ring for the final correction, doubling both the worst case and average distances for stage two.

There is a trick that allows us to construct a more clever CCC. Replace the undirected transverse arc by two directed arcs to the neighbors: connect (r, p) to the forward and backward nodes as before, and connect to (r//p, p+1) using a directed arc. In this way we can shift around the rings at the same time as we correct or leave untouched successive bits.

Similarly, there is a more clever directed version. Connect (r, p) to (r, p+1) and (r//p, p+1) using directed arcs. Since the clever CCC can only shift in one direction, the clever DCCC only loses ground in stage 2.

Table 7-1 compares CCC processor networks with directed CCC processor networks of the same dimension. The dimension of the hypercube is n.

Table 7-1:
Comparison of Undirected and Directed
Cube Connected Cycle Processor Networks
(These are approximate figures)

Aspect	CCC	DCCC	Clever CCC	Clever DCCC
Switch Size	4^2	3^2	4^2	3^2
Number of Processors	$n \cdot 2^n$	$n \cdot 2^n$	$n \cdot 2^n$	$n \cdot 2^n$
Number of Links	$2n \cdot 2^n$	$n \cdot 2^n$	$2n \cdot 2^n$	$n \cdot 2^n$
Maximum Distance	2.5 n	3 n	1.5 n	2 n
Average Distance	1.75 n	2 n	1.25 n	1.5 n

8. Conclusion

By restricting the direction of communication between adjacent nodes of the n-cube, directed n-cubes with many more nodes (x^2 where we used to have x) can be built. Nodes in directed n-cubes are almost as close as they are in undirected n-cubes (the average distance is bounded

above by $n/2 + .4 \sqrt{n}$), and resilience to node failure is maintained. Also, the technique of directing edges carries over to Cube Connected Cycles.

Simulation or mathematical modeling must be done to understand the impact of the topological restrictions and the choices for routing.

References

[1] Bondy, J. A. and Murty, U. S. R.
Graph Theory with Applications.
Elsevier North Holland, Inc., 1976.
Exercise 1.2.10.

[2] Finkel, R. A., Solomon, M. H.
Processor Interconnection Strategies.
IEEE Transactions on Computers C-29(5):360-371, May, 1980.

[3] Larsen, R. J. and Marx, M. L.
An Introduction to Mathematical Statistics and Its Applications.
Prentice-Hall, 1981.

[4] Preparata, F. P., Vuillemin, J.
The Cube-Connected Cycles: A Versatile Network for Parallel Computation.
Communications of the ACM 24(5):300-309, May, 1981.

[5] Rettberg, R. D.
Development of a Voice Funnel System.
Technical Report Report No. 4149, Bolt, Beranek, and Newman Inc., June, 1979.
Quarterly Technical Report No. 3, 1 February 1979 - 30 April 1979.

[6] Valiant, L. G., Brebner, G. J.
Universal Schemes for Parallel Communication.
In *STOC, ACM Conference Proceedings, Milwaukee.* 1981.