HYPERCUBE ROUTING ALGORITHM
Testing Depopulation Schemes

Full Hypercubes (possessing a number of nodes equal to a power of 2) allow the greatest number of busy nodes without path failures (e.g. 64 node hypercube).

For less nodes than a full hypercube, a configuration with full, neighboring subcubes proves most fault tolerant (e.g. 56, 58, 40 node hypercubes).

For hypercubes that must be depopulated less than a full subcube, a configuration maintaining at least 4 nodes in a subcube, produces the fewest path failures.

The C language models of the routing algorithm began as a test of the algorithm's ability to withstand many failed nodes. The model evolved into a test of depopulation schemes, with the number of possible failed nodes limited to two. (One failure is likely, a second failure occurring before the first is fixed is extremely unlikely, any failures beyond that are too unlikely to test).

The goal of the routing algorithm model was to develop depopulation guidelines to significantly reduce the number of source_to_destination failures, while allowing the most flexibility in depopulation.

Initial testing depopulated 4 nodes at a time (quadrant or face). At first, the quadrant or face was chosen at random. By dictating which quadrant or face to depopulate, problems were significantly reduced.

Because a quadrant has an anchor node, and it is the anchor node that is most susceptible to problems, depopulation by faces produced fewer path failures than depopulation by quadrants.

A more desireable scenario, depopulating two nodes at a time, was tested next. Only structured depopulation was tested. Two_by_face2.c produces the fewest path failures, provides the most depopulation flexibility, and allows two node failures.

In Two_by_face2.c, a 64 node cube is depopulated two nodes at a time. The two nodes with the highest addresses are depopulated unless that would leave only two nodes in a subcube. Whenever two nodes would be left in a subcube, nodes 30 and 31 are depopulated. Depopulation continues until the hypercube is reduced to 34 nodes.

The 64 node hypercube was only depopulated down to 34 nodes. Any hypercube depopulated down to or below the next lower power of two is reconfigured to be of the next lower order.

Of the ten failures from Two_by_face2, ALL were resolved when an additional step was allowed. Seven_step.c extended the original limit of 6 steps (= order of hypercube) to seven steps.

An unexpected benefit of the depopulation schemes was the elimination of the two_unused_dimension test (going out and back two unused dimensions).

HYPERCUBE ROUTING ALGORITHM
Testing Depopulation Schemes

Testing busy node combinations:

Fail.c tested every combination of 4 busy nodes in a 32 node hypercube.

4 busy nodes can surround a single node, leaving only one "1st hop". There exists then, a destination that is NUMBEROFBITS hops away from the 1st hop and NUMBEROFBITS + 1 hops away from the source.

Fail5.c tested every combination of 5 busy nodes in a 32 node hypercube.

Five nodes can completely surround a single node, isolating that node from every other.

Testing depopulation 4 nodes at a time:

Random depopulation:

Quadrant.c tested every depopulated quadrant in a 32 node hypercube, allowing one "failed" (or busy) node.

There are no failures.

Six.c tested every depopulated quadrant in a 32 node hypercube, allowing two "failed" (or busy) nodes.

One quadrant and two busy nodes can surround a single node, leaving only one "1st hop". There exists then, a destination that is NUMBEROFBITS hops away from the 1st hop and NUMBEROFBITS + 1 hops away from the source.

Newsix.c tested every depopulated quadrant in a 40 node hypercube, allowing two "failed" (or busy) nodes.

Both quadrants in the first subcube (nodes 0 - 7) are the only link from the fifth subcube (nodes 32 - 39) to the rest of the hypercube. Depopulating a quadrant in the first subcube jeopardizes every path to the fifth subcube. There are many problems.

TwoQuad.c tested every combination of two depopulated quadrants in a 32 node hypercube, allowing no "failed" nodes.

Random depopulation of multiple quadrants lead to many problems.

Two quadrants, in "opposing corners" of a part of the hypercube, make travel within that part impossible. E.g. if quadrants are numbered 0 - 7, quadrant 5 and 8 limit travel between quadrant 6 and quadrant 7.

HYPERCUBE ROUTING ALGORITHM
Testing Depopulation Schemes

TwoQuad64.c tested every combination of two depopulated quadrants in a 40 node hypercube, allowing no "failed" nodes.

The fifth subcube, once again, has a weak link. Any quadrant depopulation there and/or in the first subcube causes path failures.

Face.c tested every combination of two depopulated faces in a 32 node hypercube, allowing two "failed" nodes.

Two faces, in "opposing corners" of a part of the hypercube, make travel within that part impossible. Because a quadrant has an anchor node, and it is the anchor node that is most susceptible to problems, depopulating by faces is better than depopulating by quadrants.

Structured depopulation:

Newq_by_q.c tested the quadrant-by-quadrant depopulation of a 64 node hypercube. The quadrant with the highest addresss was removed until the hypercube was reduced to 36 nodes. One "failed" node was allowed..

Path failures occur only when the hypercube has been depopulated down to 36 nodes, leaving 4 nodes in the fifth subcube. A non-anchor busy node in the first quadrant (1, 2, or 4) forces an additional hop onto the path from non-anchor nodes in the fifth subcube to node 31 (5 hop path from fifth subcube anchor node). There were 3 path failures.

Newq_by_q2.c tested the quadrant-by-quadrant depopulation of a 64 node hypercube. The quadrant with the highest addresses was removed until the hypercube was reduced to 36 nodes. Two "failed" nodes were allowed..

A depopulated quadrant can affect a live node on two dimensions.Two busy nodes could further restrict that node in two dimensions. Depopulation could cut off links to the live node in either one or two dimensions. A node that is restricted in five of its six dimensions cannot reach its destination without additional steps. A node that is restricted in six of its six dimensions is **isolated**.

E.g. Nodes 51, 53, 54... 63 are depopulated. Node 17 and 33 are busy. Node 49 cannot "go out" dimensions 2, 3, 4 (depopulation), or dimensions 0 and 1 (busy nodes).

There were many path failures.

HYPERCUBE ROUTING ALGORITHM
Testing Depopulation Schemes

Newf_by_f.c tested the face-by-face depopulation of a 64 node hypercube. The face with the highest addresses was removed until the hypercube was reduced to 36 nodes. One "failed" node was allowed..

There were **no path failures**.

Newf_by_f2.c tested the face-by-face depopulation of a 64 node hypercube. The face with the highest addresses was removed until the hypercube was reduced to 36 nodes. Two "failed" nodes were allowed..

Path failures occur only when the hypercube has been depopulated down to 36 nodes, leaving a single face in the fifth subcube. Any node in the last face is restricted in three dimensions because of depopulation. Two busy nodes can further restrict a node in two more dimensions. A node that is restricted in five of its six dimensions cannot reach its destination without additional steps.

E.g. Nodes 36 - 63 are depopulated. Node 3 and 33 are busy. Node 35 cannot "go out" dimensions 1, 2, 3 (depopulation), or dimensions 0 and 4 (busy nodes).

There were 16 path failures.

F_by_f.c tested the face-by-face depopulation of a 64 node hypercube. Alternating removal of the face with the highest addresses with the face with the highest addresses less than 32, until the hypercube was reduced to 36 nodes. One "failed" node was allowed..

There were **no path failures**.

F_by_f2.c with the addresses less hypercube

Depopulated a 64 node cube face by face, removing, every other time, the face highest addresses, alternating with removing the face with the highest than 32. Depopulation continues until down to a 36 node

any two nodes busy

Path failures occur only when the hypercube has been depopulated down to 36 nodes, leaving a single face in the third subcube. The configuration of the 36 node hypercube leaves two subcubes in the higher addresses (nodes 32 - 47) and two and one half subcubes in the lower addresses (nodes 0 - 19). There exists a weak...

why is this configuration worse than other 36 node configurations??? i started to think that there exists a weak link between the four higher addressed subcubes and the lower

There were 30 path failures.

HYPERCUBE ROUTING ALGORITHM
Testing Depopulation Schemes

F_by_f_b.c
F_by_f2_b.c
Depopulated a 64 node cube face by face, removing, every other time, the face with the highest addresses, alternating with removing the face with the lowest addresses. Depopulation continues until down to a 36 node hypercube
any one node busy

When the hypercube has been depopulated down to 36 nodes, a single face links two subcubes in the higher addresses (nodes 32 - 47) and two nodes in the lower addresses (16 - 31). Any busy nodes in the linking face is crippling.
There were many path failures.

Testing depopulation 2 nodes at a time:

Structured depopulation:

New2_by_2.c Depopulated a 64 node cube two nodes at a time, removing the two nodes with the highest addresses, until down to a 34 node hypercube
any one node busy

Path failures occur only when the hypercube has been depopulated down to 34 nodes, leaving just two nodes (32, 33) in the fifth subcube. The only link from these nodes to other nodes is through two nodes (0,1) in the first subcube. If node 0 or 1 is busy, an additional hop is forced onto paths between the fifth and fourth subcubes.
There were 2 path failures.

New2_by_22.c Depopulated a 64 node cube two nodes at a time, removing the two nodes with the highest addresses, until down to a 34 node hypercube
any two nodes busy

In addition to the same failures discovered in Newf_by_f2.c, path failures occur when just two nodes are left in a subcube (e.g. nodes 50 - 63 depopulated). With two nodes busy, a live node in a partial subcube can conceivable have only one link. An additional hop is forced onto the path between that node and one that is six hops from the link.

There were many path failures.

Two_by_two.c Depopulated a 64 node cube two nodes at a time, removing, every other time, the two nodes with the highest addresses, alternating with removing the two nodes with the highest addresses less than 32. Depopulation continues until down to a 34 node hypercube.
any one node busy

HYPERCUBE ROUTING ALGORITHM
Testing Depopulation Schemes

why is this configuration worse than other 34 node configurations??? i started to think that there exists a weak link between the four higher addressed subcubes and the lower

There were 2 path failures.

Two_by_face.c Depopulated a 64 node cube two nodes at a time, removing the two nodes with the highest addresses unless that would leave only two nodes in a subcube. Whenever two nodes would be left in a subcube, depopulate instead nodes 30 and 31. Depopulation continues until down to a 34 node hypercube
any one node busy

There were **no path failures.**

Two_by_face2.c Depopulated a 64 node cube two nodes at a time, removing the two nodes with the highest addresses unless that would leave only two nodes in a subcube. Whenever two nodes would be left in a subcube, depopulate instead nodes 30 and 31. Depopulation continues until down to a 34 node hypercube
any two nodes busy

Path failures occur only when the hypercube has been depopulated down to 34 nodes, leaving the front face in the fifth subcube (nodes 30, 31, 36-63 depopulated). With two nodes busy, a live node in a partial subcube can conceivable have only one link. An additional hop is forced onto the path between that node and one that is six hops from the link.

There were 10 path failures.

Testing path length limit extended one hop (to NUMBEROFBITS + 1):
Structured depopulation:

Seven_step.c Depopulated a 64 node cube two nodes at a time, removing the two nodes with the highest addresses unless that would leave only two nodes in a subcube. Whenever two nodes would be left in a subcube, depopulate instead nodes 30 and 31. Depopulation continues until down to a 34 node hypercube
any two nodes busy

This program is identical to Two_by_face2.c except for the fact that seven-step paths can be used from source to destination nodes. All failures from Two_by_face2.c were resolved.

There were **no path failures.**

HONEYWELL Interoffice Correspondence

Date: 28 June 1990

Subject: Hypercube Router Algorithm

To: Kevin Driscoll

From: Chris A. Geilfuss
Organization: Marine Systems Division
HED: WA34
M.S: 4D16
HVN: 356-3676

As requested by Norm Tinklepaugh, I am sending you the attached overview of the latest router algorithm. The modeling for this algorithm has been done in C on a Sun 386i.

The algorithm began as follows:

Given a source node, destination node, and busy nodes:
List all dimensions that must be traveled.
Travel each dimension, highest order to lowest.
If a busy node is encountered,
move that dimension to the end of the list.

The algorithm has evolved into:

For every source, destination and combination of busy nodes:
List all dimension that must be traveled.
Until destination has been reached,
or every rearrangement of dimension list has been tried.
Travel each dimension.
If a busy node is encountered, go back to source node
rearrange the list of dimensions;
If every rearrangement of dimension list has been tried,
Add an unused dimension to the beginning and end of the dimension list.
Travel each dimension.
If a busy node is encountered,
rearrange the list of dimensions;
~~If every rearrangement of dimension list (with extra dimension) has been tried,~~
~~Add two unused dimensions to the beginning and end of the dimension list.~~
~~Travel each dimension.~~
~~If a busy node is encountered,~~
~~rearrange the list of dimensions.~~

I can supply you with copies of any of the code. Please call if you have any questions.

HYPERCUBE ROUTER ALGORITHM

The Hypercube Router Algorithm finds the shortest path, if one exists, from source node to destination node within any size hypercube. The length of the path is limited to the number of dimensions in a hypercube.

If a direct path is not available, and the length of the path would not exceed the limit, the algorithm will go out and back an unused dimension. All paths will be tested in all unused dimensions until the destination is reached.

If a path still is not available, and the length of the path would not exceed the limit, the algorithm will go out and back two unused dimensions. All paths will be tested in all combinations of two unused dimensions until the destination is reached.

The size of a hypercube is designated by the number of its nodes. (Each hypercube is a power of 2), e.g. 2, 4, 8, 16, 32, 64, etc. (**NUMBEROFNODES**).

The nodes of a hypercube are given binary addresses corresponding to decimal 0 thru (number of nodes minus one). In a 64 node hypercube, addresses range from 000000 to 111111 (0 - 63).

The number of dimensions in a hypercube directly corresponds to the length of any node's binary address. A 64 node hypercube has 6 dimensions. (**NUMBEROFBITS**).

A Permutation Table is computed containing every possible way to rearrange the numbers 0 thru x where x is the number of dimensions minus 1,

e.g. in a 64 node hypercube (6 dimensions) the permutation table contains rearrangements of "012345".

```
Perm[0][0]                    = 0
  Perm[0][1]                    = 1
    Perm[0][2]                    = 2
      Perm[0][3]                    = 3
        Perm[0][4]                    = 4
          Perm[0][5]                    = 5
Perm[1][0] thru Perm[1][5]      = 012354
...
Perm[719][0] thru Perm[719][5]  = 543210
```

The number of elements in the Permuation Table is the factorial of the number of dimensions, in this case 6! = 720. (**PERM[NUMBEROFBITS!][NUMBEROFBITS]**). The second index in the Permuatation Table specifies **Bit**.

A Busy Table is maintained containing the busy state of every node. A node could be busy because of intentional depopulation or because of unexpected failure. (**BUSY[NUMBEROFNODES]**). Only long-term busy nodes are considered in this algorithm. Yet to be discussed are temporarily busy nodes.

HYPERCUBE ROUTER ALGORITHM

1) Algorithm 1 - DIRECT PATH

For a given source node and destination node:

a) Calculate **BINARYWORD** to be the exclusive or of the binary addresses of source and destination.

b) Construct **Bit_Sequence** by assigning **Bit** for each 1.

c) Assign **length_of_sequence** to be the number of elements in Bit_Sequence.

d) Construct **Unused_Dimensions** by assigning **Bit** for each .

e.g.

source	13	001101		
destination	62	111110		
BINARYWORD	xor	110011		
		110011		
Bit_Sequence		01 45	= "0145"	**length_of_sequence** = 4
Unused_Dimensions		23	= "23"	

e) Get the first Permutation Table entry, Perm[0].

HYPERCUBE ROUTER ALGORITHM

1) Algorithm 1 - DIRECT PATH (continued)

f) Until (a path is found from source to destination) or (the "end" of the Permutation Table is reached*):

i) Strip off elements of the Permutation Table entry, from the beginning, until the length of the entry = length_of_sequence.

e.g. Perm[x][NUMBEROFBITS - length_of_sequence]
thru Perm[x][NUMBEROFBITS -1]
Perm[2][2] thru Perm[2][5] = 2 4 3 5

ii) Offset each element in the resulting portion of the Permutation Table entry by (NUMBEROFBITS - length_of_sequence).

e.g. Perm[0][2] thru Perm[0][5] = 2 4 3 5
subtract (NUMBEROFBITS - length_of_sequence)
from each element 2 2 2 2
= 0 2 1 3

iii) Use the given sequence to rearrange the order of elements in **Bit_sequence.**

e.g. = 0 2 1 3
elements 0 1 2 3
Bit_sequence was 0 1 4 5 **= 0 4 1 5**

iv) Set **node** to source.

v) For each element in Bit_Sequence:
If node is busy
get another entry in the Permutation Table
go back to the f) Until statement.

If Bit_Sequence[x] = 0
XOR node and 32 (100000) to get next node.

If Bit_Sequence[x] = 1,
XOR node and 16 (010000) to get next node.

If Bit_Sequence[x] = 2
XOR node and 8 (001000) to get next node.

If Bit_Sequence[x] = 3
XOR node and 4 (000100) to get next node.

If Bit_Sequence[x] = 4
XOR node and 2 (000010) to get next node.

If Bit_Sequence[x] = 5
XOR node and 1 (000001) to get next node.

HYPERCUBE ROUTER ALGORITHM

1) Algorithm 1 - DIRECT PATH (continued)

e.g **NODE 61** is busy.
Perm[0][2] thru Perm[0][5] results in Bit_sequence 0145.

	Node	012345	Bit_Sequence[Perm[0][x]] = Bit To Change	
Source	**13**	**001101**	Bit_Sequence[0]	= 0
xor	32	100000		
	45	**101101**		
	45	101101	Bit_Sequence[1]	= 1
xor	16	010000		
	61	**111101**	**BUSY!**	

Algorithm determines next Permutation Table entry based on length_of_sequence and on which port led to a busy node. In this case, next entry is Perm[2].

Perm[2][2] thru Perm[2][5] results in Bit_sequence 0415.

	Node	012345	Bit_Sequence[Perm[0][x]] = Bit To Change	
Source	**13**	**001101**	Bit_Sequence[0]	= 0
xor	32	100000		
	45	**101101**		
	45	101101	Bit_Sequence[2]	= 4
xor	2	000010		
	47	**101111**		
	47	101111	Bit_Sequence[1]	= 1
xor	16	010000		
	63	**111111**		
	63	111111	Bit_Sequence[3]	= 5
xor	1	000001		
Destination	**62**	**111110**		

* The last element of the Permutation Table = factorial of length_of_sequence. For this example, 13 to 62, where length_of_sequence = 4, "end" = Perm[4!] = Perm[24].

HYPERCUBE ROUTER ALGORITHM

2) Algorithm 2 - ADDED DIMENSION

For a given source node and destination node:

a) This is the same as Algorithm 1.
b) This is the same as Algorithm 1.
c) This is the same as Algorithm 1.
d) This begins in the same way as Algorithm 1.
d1) If length_of_sequence <= 4

I) For each Unused_Dimension

A) Modify Bit_Sequence by adding the Unused_Dimension to the beginning and end of Bit_Sequence.

B) Add 2 to length_of_sequence.

e.g. **Bit_Sequence** 01 45 = "0145" **length_of_sequence** = 4
Unused_Dimensions 23 = "23"

Bit_Sequence 201452 length_of_sequence = 6
Bit_Sequence 301453 length_of_sequence = 6

C) do e) as in Algorithm 1.

D) do f) as in Algorithm 1, adding the following after item ii)

ii2) If the resulting sequence (the offset portion of the Permutation Table) has the digit 0 next to (length_of_sequence - 1)
go back to the source
get another entry in the Permutation Table
go back to the f) Until statement.

explanation: Given that the first and last digits of bit_sequence are the same, any Permutation Table entry having (NUMBEROFBITS - length_of_sequence) next to (NUMBEROFBITS - 1), when offset by (NUMBEROFBITS - length_of_sequence), would have 0 next to (length_of_sequence - 1). This would mean that the same dimension is "traveled" back-to-back. These Permutation Table entries are skipped.

e.g. length_of_sequence = 4

Permutation Table entry	element 2 thru 5	offset by 2	0 next to 3?
0	2345	0123	
1	2354	0132	
2	2453	0231	
3	2435	0213	
4	2534	0312	SKIP
5	2543	0321	SKIP
6	3452	1230	SKIP
7	3425	1203	SKIP
...			

HYPERCUBE ROUTER ALGORITHM

2) Algorithm 2 - ADDED DIMENSION (continued)

e.g. Algorithm 2, **Node 54 busy**

Node			
source	38	100110	
destination	42	101010	
BINARYWORD		101110	
Bit_sequence		23	length_of_sequence 2
Unused_dimensions		01 45	
Bit_sequence		0230	length_of_sequence 4
and		1231	length_of_sequence 4
and		4234	length_of_sequence 4
and		5235	length_of_sequence 4

for Bit_sequence 1231

Perm[0] = 2345 offset 0123 rearranged Bit_sequence 1231.

	Node	012345	Bit_Sequence[Perm[0][x]] = Bit To Change
Source	**38**	**100110**	Bit_Sequence[0] = 1
xor	16	010000	
	54	**110110**	**BUSY!**

Algorithm determines next Permutation Table entry based on length_of_sequence and on which port led to a busy node. In this case, next entry is Perm[6].

Perm[6] = 3452	offset 1230	SKIP, 0 and 3 are next to each other
Perm[7] = 3425	offset 1203	SKIP, 0 and 3 are next to each other
Perm[8] = 3524	offset 1302	SKIP, 0 and 3 are next to each other
Perm[9] = 3542	offset 1320	rearranged Bit_sequence 2131.

	Node	012345	Bit_Sequence[Perm[9][x]] = Bit To Change
Source	**38**	**100110**	Bit_Sequence[1] = 2
xor	8	001000	
	46	**101110**	
	46	101110	Bit_Sequence[3] = 1
xor	16	010000	
	62	**111110**	
	62	111110	Bit_Sequence[2] = 3
xor	4	000100	
	58	**111010**	
	58	111010	Bit_Sequence[0] = 1
xor	16	010000	
Destination	**42**	**101010**	

HYPERCUBE ROUTER ALGORITHM

3) Algorithm 3 - TWO ADDED DIMENSIONS

For a given source node and destination node:

a) This is the same as Algorithm 1.
b) This is the same as Algorithm 1.
c) This is the same as Algorithm 1.
d) This begins in the same way as Algorithm 1.
d1) If length_of_sequence <= 2

I) Determine combinations of two unused_dimensions.

e.g. Unused_Dimensions 0235
combinations 02, 03, 05, 23, 25, 35

II) For each combination of two Unused_Dimensions

A) Modify Bit_Sequence by adding each of the Unused_Dimensions to the beginning and end of Bit_Sequence.

B) Add 4 to length_of_sequence.

e.g. **Bit_Sequence** 1 4 = "14" **length_of_sequence** = 2
Unused_Dimensions 0 23 5 = "0235"

Bit_Sequence 021420, 031430, 051450, 231432, 251452, 351453
length_of_sequence = 6

C) do e) as in Algorithm 1.

HYPERCUBE ROUTER ALGORITHM

3) Algorithm 3 - TWO ADDED DIMENSIONS (continued)

D) do f) as in Algorithm 1, adding the following after item ii)

ii2) If the resulting sequence (the offset portion of the Permutation Table) has
either the digit 0 next to (length_of_sequence - 1)
OR the digit 1 next to (length_of_sequence - 2)

go back to the source
get another entry in the Permutation Table
go back to the f) Until statement.

explanation:

Given that the first and last digits of bit_sequence are the same, any Permutation Table entry having (NUMBEROFBITS - length_of_sequence) next to (NUMBEROFBITS - 1), when offset by (NUMBEROFBITS - length_of_sequence), would have 0 next to (length_of_sequence - 1). Given also that the second and second-to-last digits of Bit_sequence are the same, any Permutation Table entry having (NUMBEROFBITS - length_of_sequence + 1) next to (NUMBEROFBITS - 2), when offset by (NUMBEROFBITS - length_of_sequence), would have 1 next to (length_of_sequence - 2).

This would mean that the same dimension is "traveled" back-to-back. These Permutation Table entries are skipped.

Permutation Table entry	element 0 thru 5	offset by 0	0 next to 5 or 1 next to 4?
...			
11	013254	013254	
12	014523	014523	SKIP
13	014532	014532	SKIP
...			
31	024531	024531	
32	024135	024135	SKIP
33	024153	024153	SKIP
...			
95	043215	043215	
96	051234	051234	SKIP
...			

HYPERCUBE ROUTER ALGORITHM

3) Algorithm 3 - TWO ADDED DIMENSIONS (continued)

e.g. Algorithm 3, **Node 43 busy**

source	11	001011	
destination	25	011001	
BINARYWORD		010010	
Bit_sequence		1 4	length_of_sequence 2
Unused_dimensions		0 23 5	

Bit_Sequence 021420, 031430, 051450, 231432, 251452, 351453
length_of_sequence = 6

for Bit_sequence 051450

Perm[0] = 012345 offset 012345 rearranged Bit_sequence 051450.

	Node	012345	Bit_Sequence[Perm[0][x]] = Bit To Change
Source	**11**	**001011**	Bit_Sequence[0] = 0
xor	32	100000	
	43	**101011**	**BUSY!**

Algorithm determines next Permutation Table entry based on length_of_sequence and on which port led to a busy node. In this case, next entry is Perm[120].

Perm[120] = 123450 offset 123450 SKIP, 0 next to 5
Perm[121] = 123405 offset 123405 SKIP, 0 next to 5
Perm[122] = 123504 offset 123504 SKIP, 0 next to 5
Perm[123] = 123540 offset 123540

Bit_Sequence 051450
Rearranged 514050

	Node	012345	Bit_Sequence[Perm[123][x]] = Bit To Change
Source	**11**	**001011**	Bit_Sequence[1] = 5
xor	1	000001	
	10	001010	Bit_Sequence[2] = 1
xor	16	010000	
	26	011010	Bit_Sequence[3] = 4
xor	2	000010	
	24	010000	Bit_Sequence[5] = 0
xor	32	100000	
	56	110000	Bit_Sequence[4] = 5
xor	1	000001	
	57	110001	Bit_Sequence[0] = 0
xor	32	100000	
Destination	**25**	**010001**	

April 18, 1990

IR&D Hypercube Interconnectivity Improvement

Charge # 9806-A01 Acct after 5/6/90

for Norm Tinklepaugh

Task: Create a reporting program showing "paths" traveled within a 4 dimensional Hypercube when given a source address, a destination address, a busy table, and an algorithm for getting from source to destination.

Definition: 4D Hypercube

Input: Algorithms - travel address bits left to right, skipping busy node bit until last
travel address bits right to left, skipping busy node bit until last
travel address bits left to right, skipping busy node bit until after next bit
travel address bits right to left, skipping busy node bit until after next bit
? others depending on number of busy nodes or other sequence ??

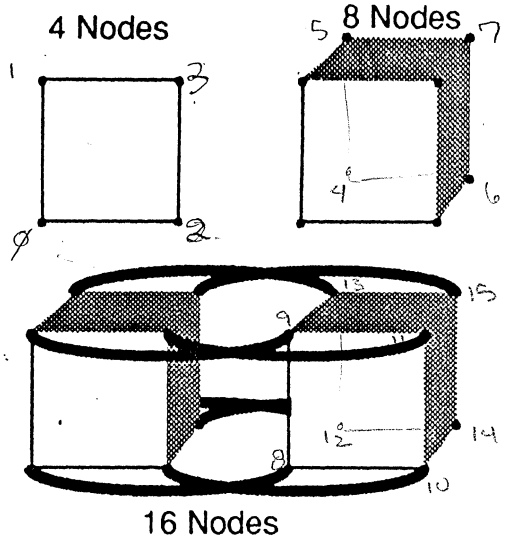
Address - 5 bits 0 0 000
l/r
t/b
within cube
1st bit, left or right quadrant
2nd bit, top or bottom quadrant
last 3 bits, node within cube

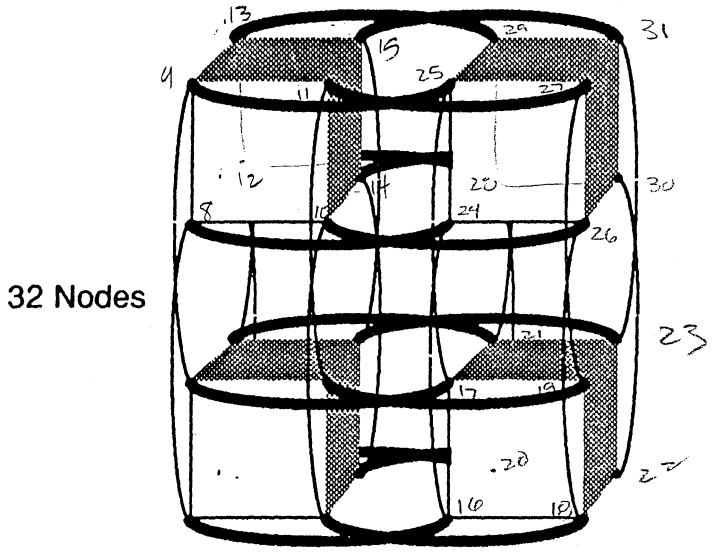
all nodes (computers) have access to busy/unbusy state of their neighbors

single node failure - only one to worry about, the possibility of 2 failures is to small
BUT
unpopulated nodes must be treated as "busy" too (unpopulated = not enough nodes to make up cube, e.g. 20 vs 32)

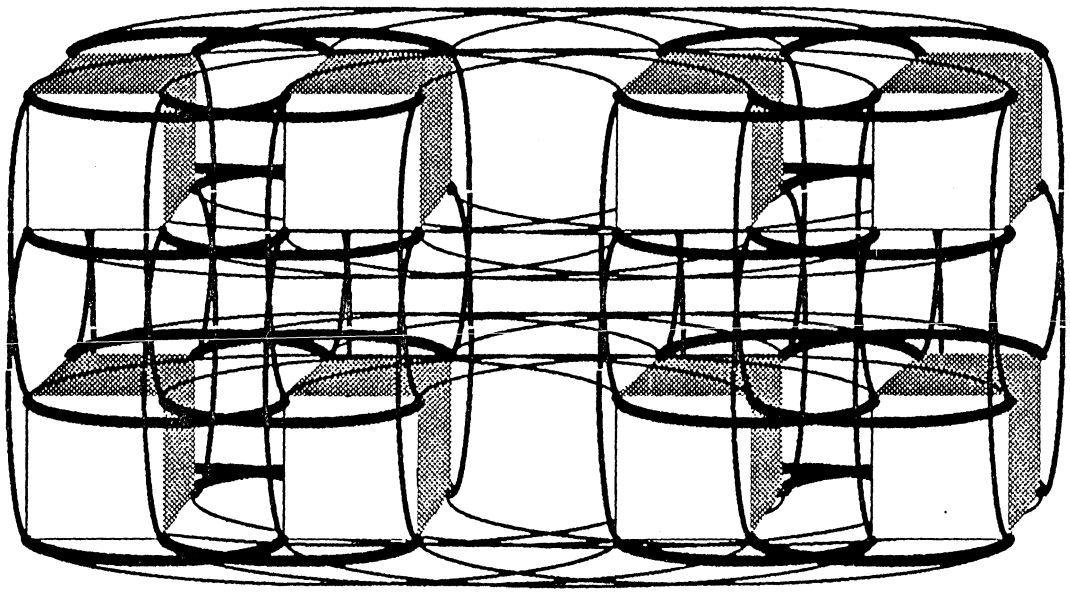
Output
source
destination
path show binary address and decimal equivalent or tree or diagram
algorithm
failed or not after 5 steps

HYPERCUBE ARCHITECTURE



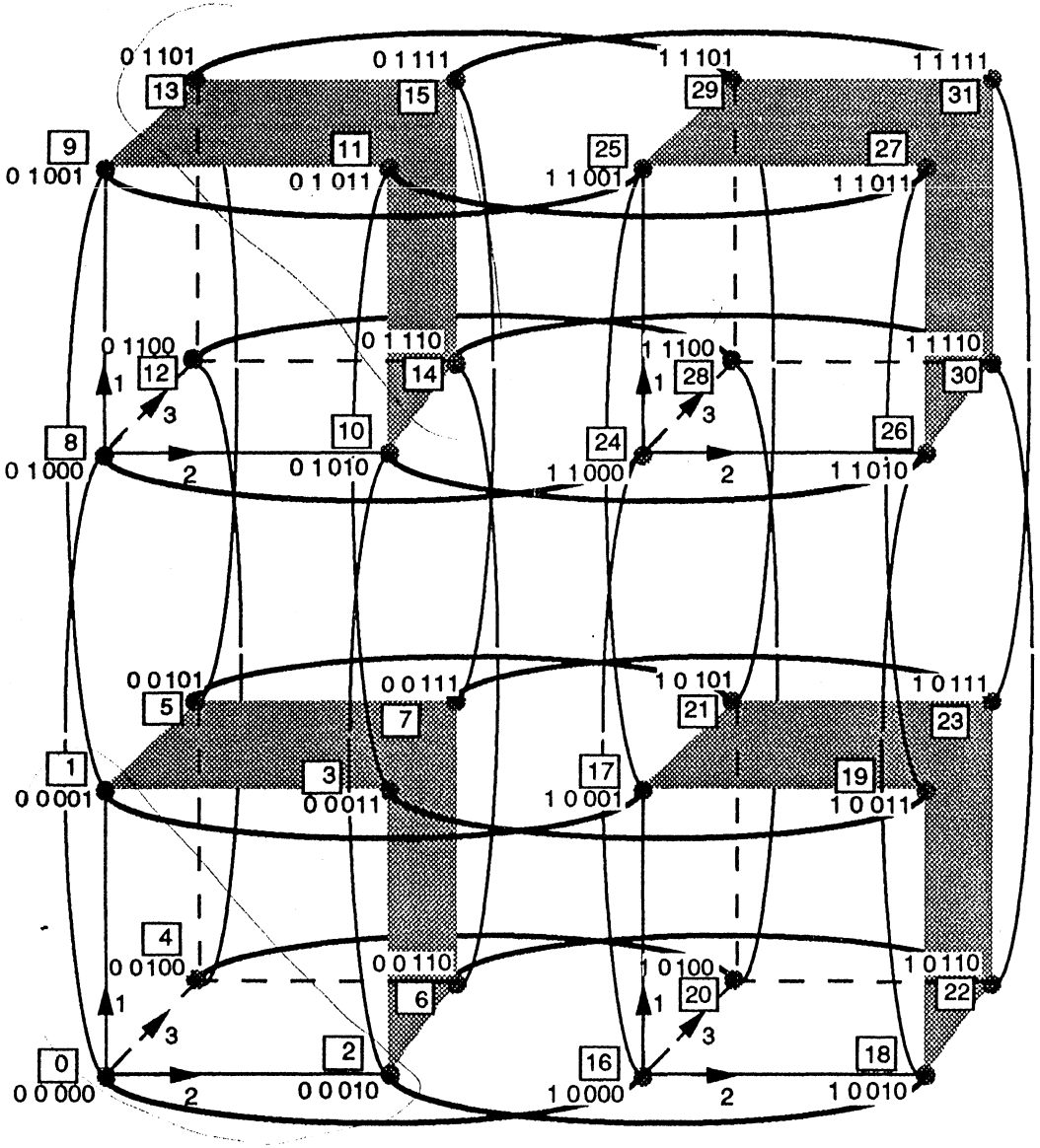


64Nodes

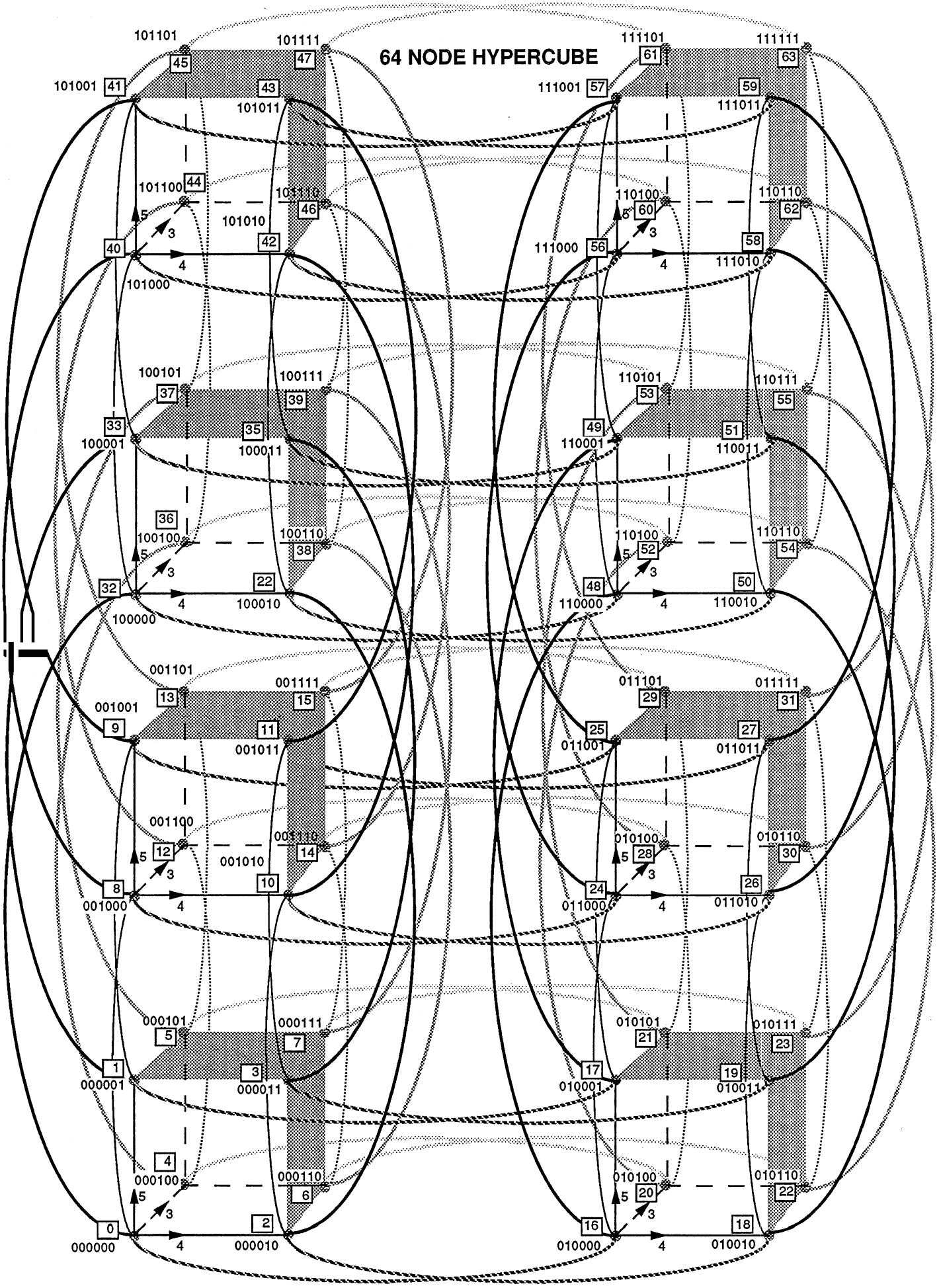


H/C Arch Dwg 3/2990
Rev 2

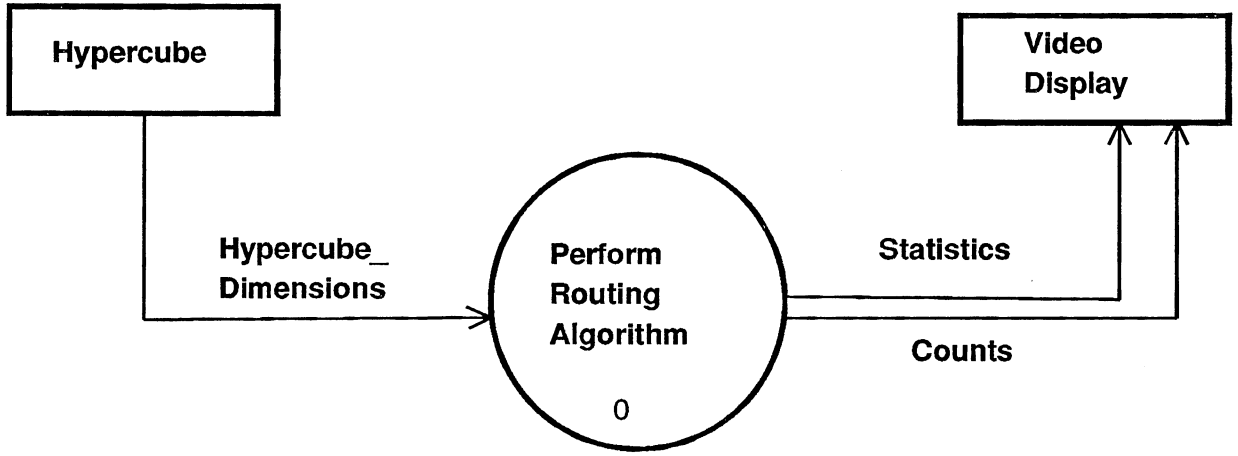
HYPERCUBE ARCHITECTURE



Bit #		Direction
MSB	1	→ } between subcubes
	2	↑ } between subcubes
	3	+ } within subcube
	4	→ } within subcube
LSB	5	↑ } within subcube


64 NODE HYPERCUBE
000000
000001
000010
000011
000100
000101
000110
000111
001000
001001
001010
001011
001100
001101
001110
001111
010000
010001
010010
010011
010100
010101
010110
010111
011000
011001
011010
011011
011100
011101
011110
011111
100000
100001
100010
100011
100100
100101
100110
100111
101000
101001
101010
101011
101100
101101
101110
101111
110000
110001
110010
110011
110100
110101
110110
110111
111000
111001
111010
111011
111100
111101
111110
111111

Context-Diagram;4
Routing Algorithm

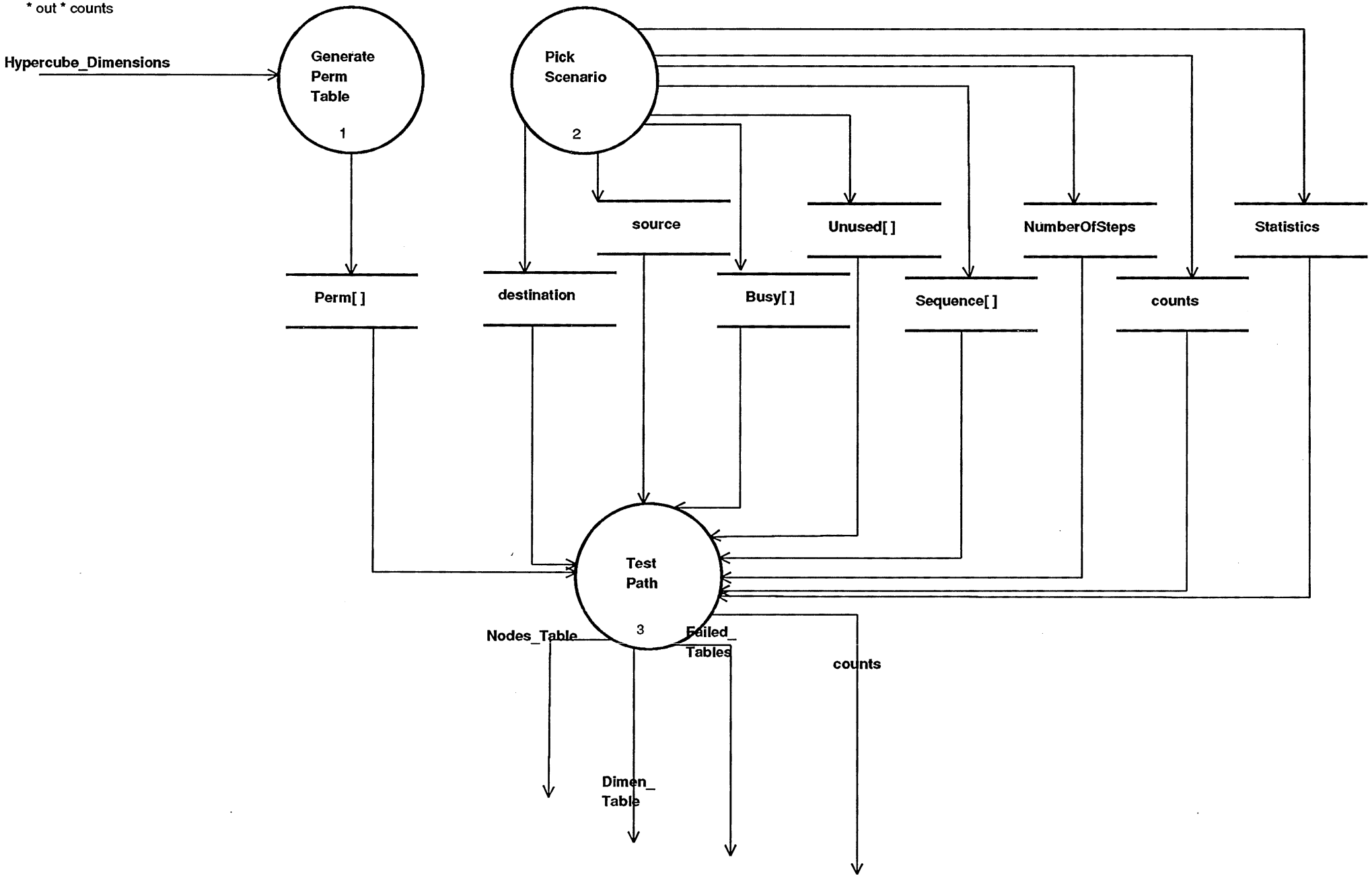


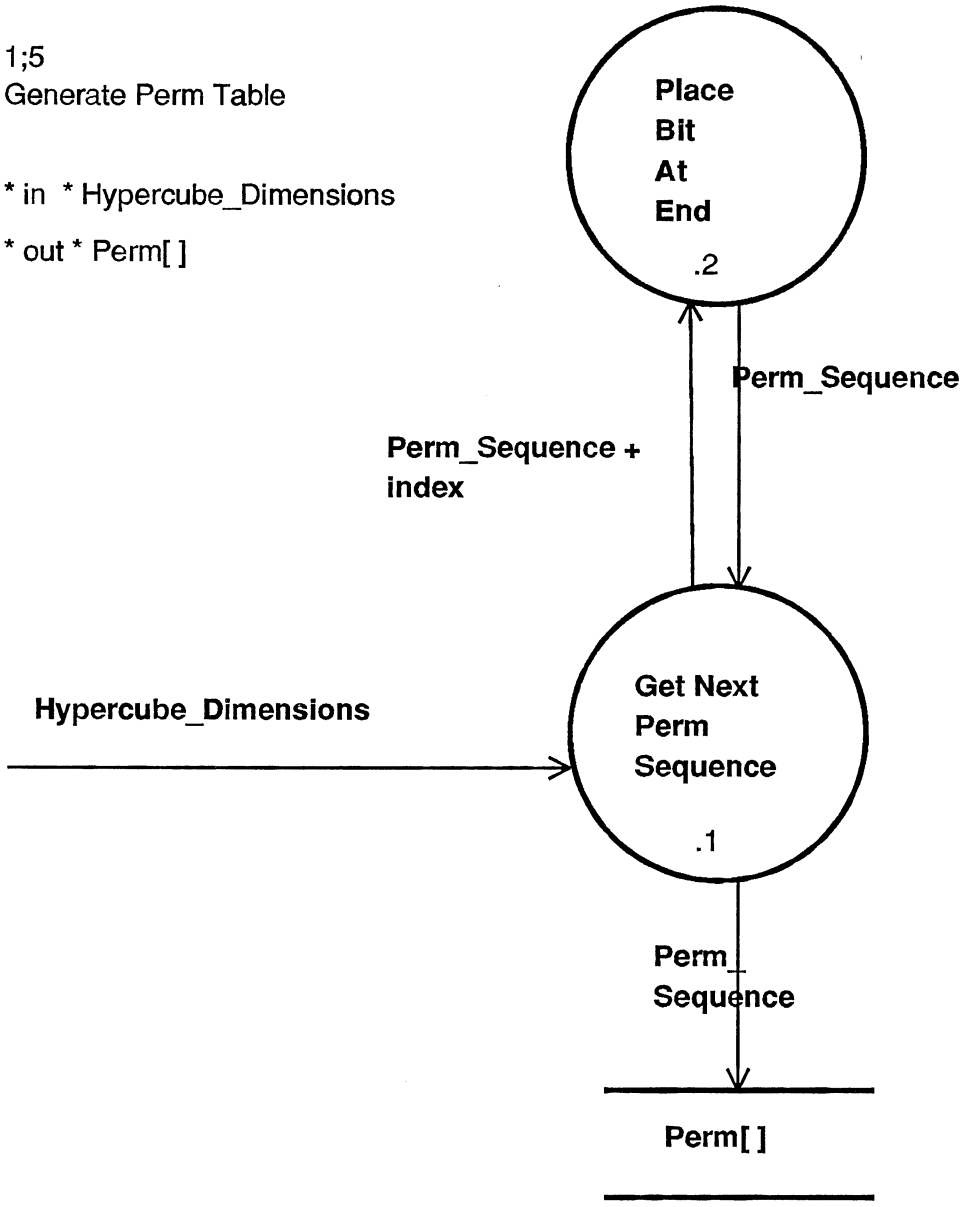
0;7
Perform Rout[redacted]thm

* in * Hypercube_Dimensions

* out * Statistics

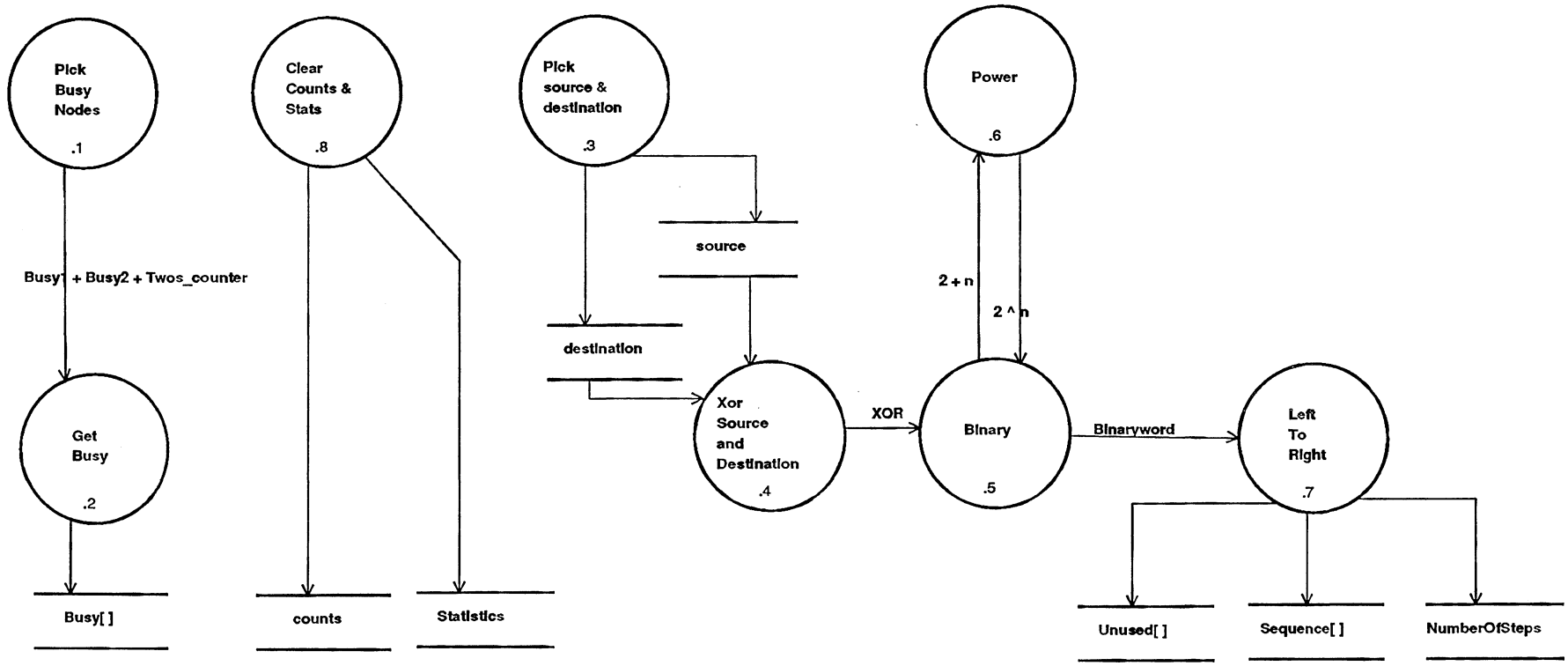
* out * counts




1;5
Generate Perm Table
* in * Hypercube_Dimensions
* out * Perm[ ]
Place
Bit
At
End
.2
Perm_Sequence
Perm_Sequence +
index
Hypercube_Dimensions
Get Next
Perm
Sequence
.1
Perm_
Sequence
Perm[ ]

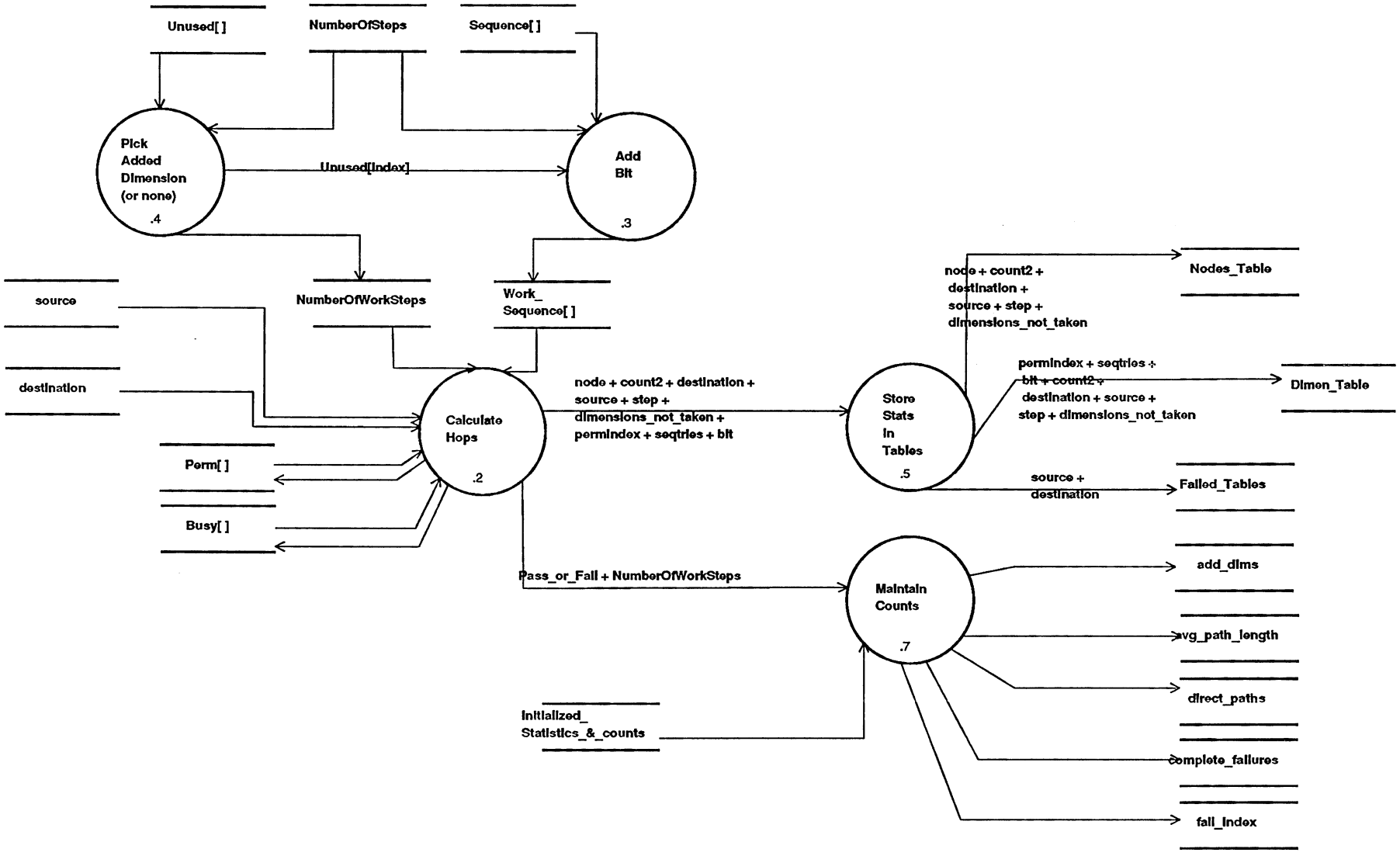
2;7
Pick Scenario

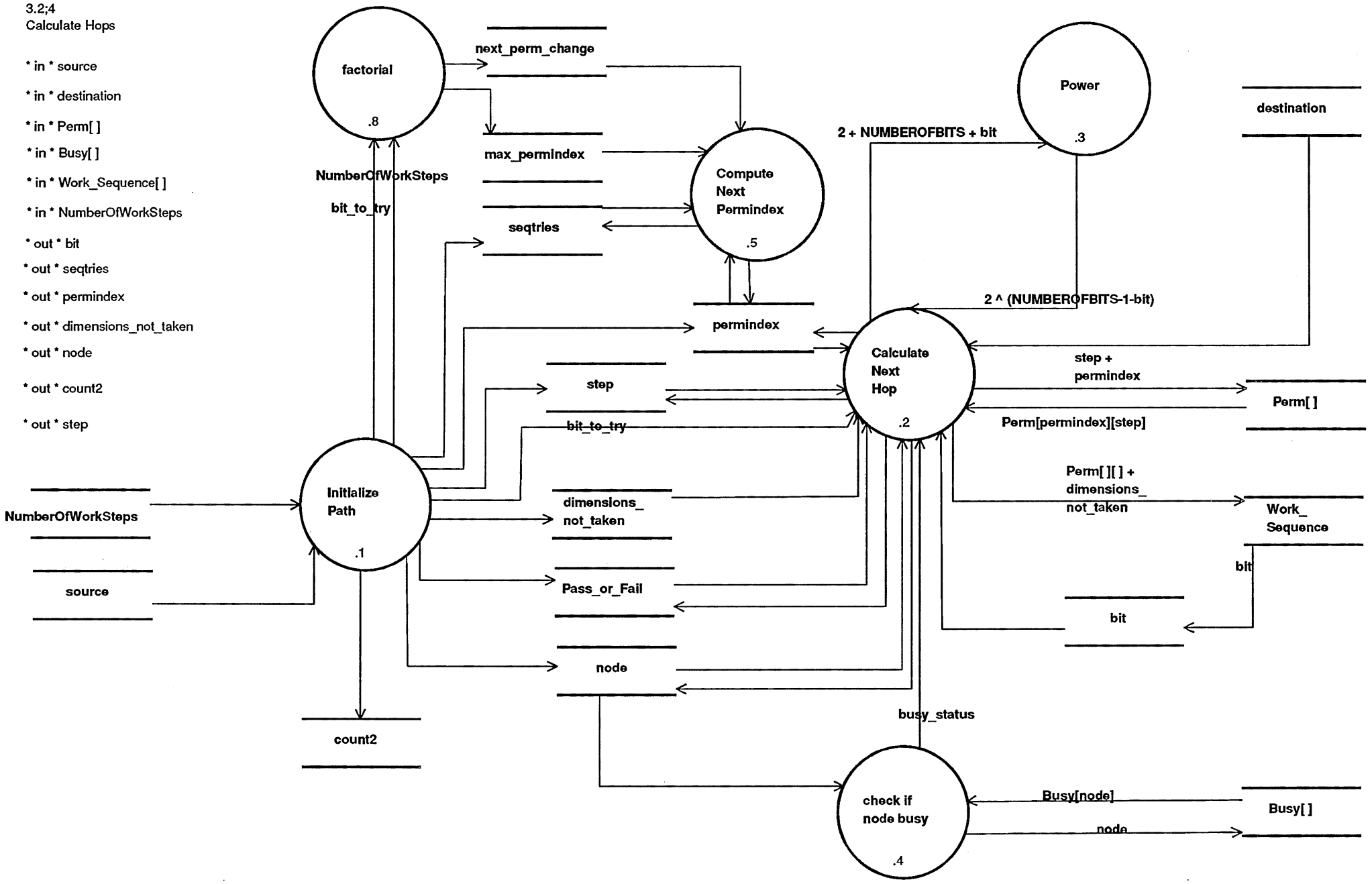
* out * source
* out * destination
* out * Busy[ ]
* out * Unused[ ]
* out * Sequence[ ]
* out * NumberOfSteps
* out * counts
* out * Statistics



3;6
Test Path

* In * source
* In * destination
* In * Busy[ ]
* In * Sequence[ ]
* In * Unused[ ]
* In * NumberOfSteps
* In * Perm[ ]
* In/out * Statistics
* In/out * counts




3.2;4
Calculate Hops
* in * source
* in * destination
* in * Perm[ ]
* in * Busy[ ]
* in * Work_Sequence[ ]
* in * NumberOfWorkSteps
* out * bit
* out * seqtries
* out * permindex
* out * dimensions_not_taken
* out * node
* out * count2
* out * step
factorial
.8
next_perm_change
max_permindex
NumberOfWorkSteps
bit_to_try
seqtries
Compute
Next
Permindex
.5
Power
.3
2 + NUMBEROFBITS + bit
destination
2 ^ (NUMBEROFBITS-1-bit)
permindex
Calculate
Next
Hop
.2
step
bit_to_try
step +
permindex
Perm[ ]
Perm[permindex][step]
Perm[ ][ ] +
dimensions_
not_taken
Work_
Sequence
bit
bit
Initialize
Path
.1
NumberOfWorkSteps
source
dimensions_
not_taken
Pass_or_Fail
node
count2
busy_status
check if
node busy
.4
Busy[node]
node
Busy[ ]

```
 /* depopulate 2 nodes by 2 nodes, never leave cube with
    just 2 nodes, 1 busy */
* allow one extra step */

nclude <stdio.h>

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                       */
/*      GLOBAL VARIABLES                                 */
/*                                                       */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#define NUMBEROFBITS 6
                        /* number of bits in Node
                           Address (0, 1, 2, ... \n, \0) */
#define NUMBEROFNODES 64
                        /* number of nodes
                           in hypercube */
#define NUMBEROFPERMUTATIONS 5040
                        /* factorial (NUMBEROFBITS+1) */

FILE *fileptr;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                       */
/*      FACTORIAL                                        */
/*  input : degree                                       */
/* returns: degree!                                      */
/*                                                       */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
factorial(degree)
int degree;

        int c, fac;

        fac = 1;
        for (c = degree; c >= 2; c--)
                fac = fac * c;

        return(fac);
}
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                       */
/*      GENERATE_PERM_TABLE                              */
/*      create PERM[][] table, all rearrangements of     */
/*         digits 0123456                                */
/*                                                       */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
generate_perm_table(p_Perm_table)
int *p_Perm_table;
{
  int i, i2, i3, i4, i5, i6, i7;
  int p = 0;
  int permsequence[NUMBEROFBITS+1];
  int *p_permsequence;

  for (i = 0; i <= NUMBEROFBITS; i++)
        permsequence[i] = i;


  for (i = 0; i <= NUMBEROFBITS; ++i)
  {
    for (i2 = 1; i2 <= NUMBEROFBITS; ++i2)
    {
      for (i3 = 2; i3 <= NUMBEROFBITS; ++i3)
      {
        for (i4 = 3; i4 <= NUMBEROFBITS; ++i4)
        {
          for (i5 = 4; i5 <= NUMBEROFBITS; ++i5)
          {
```

```
            for (i6 = 5; i6 <= NUMBEROFBITS; ++i6)
            {
              for (i7 = 0; i7 <= NUMBEROFBITS; ++i7)
                   *p_Perm_table++ = permsequence[i7];
              ++p;
              p_permsequence = permsequence;
              place_bit_at_end(p_permsequence,5);
            }
            p_permsequence = permsequence;
            place_bit_at_end(p_permsequence,4);
          }
          p_permsequence = permsequence;
          place_bit_at_end(p_permsequence,3);
        }
        p_permsequence = permsequence;
        place_bit_at_end(p_permsequence,2);
      }
      p_permsequence = permsequence;
      place_bit_at_end(p_permsequence,1);
    }
    p_permsequence = permsequence;
    place_bit_at_end(p_permsequence,0);

  }

  return;
}
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                       */
/*      PLACE_BIT_AT_END                                 */
/*  input : pointer to permsequence, which bit to move   */
/* output : pointer to permsequence (with bit at end)    */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */

place_bit_at_end(p_perm,bit_to_move)
int *p_perm;
int bit_to_move;
{
        int i2, temp;

        temp = *(p_perm+bit_to_move);
        for (i2 = bit_to_move;
             i2 <= (NUMBEROFBITS-1);
             ++i2)
                *(p_perm+i2) = *(p_perm+(i2+1));
        *(p_perm+NUMBEROFBITS) = temp;

        return;
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                       */
/*      GETBUSY                                          */
/*  input : number of groups of two, extra busy node,    */
/*          extra busy node                              */
/* output : pointer to busy[] table                      */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */

                                                /* * * * GET BUSY * * * * */
getbusy(twos, busynode1, busynode2, p_busy)
int twos, busynode1, busynode2;
int *p_busy;
{
  int i, upper_busy_range, middle_busy1,
              middle_busy2;
  char ch;

  if (((twos + 1) % 4) > 0)
              /* not third group of two from subcube */
  {
```

```
        upper_busy_range = NUMBEROFNODES - (2 * twos);
        middle_busy1 = middle_busy2 = busynode2;
    }
    else                            /* 3rd group of two */
    {
        upper_busy_range =
                NUMBEROFNODES - (2 * (twos - 1));
        middle_busy1 = (NUMBEROFNODES / 2) - 1;
        middle_busy2 = (NUMBEROFNODES / 2) - 2;
    }
    for (i = 0; i < NUMBEROFNODES; ++i)
    {
        if ((i == busynode2) || (i == busynode1)
          || (i >= upper_busy_range)
          || (i == middle_busy1)
          || (i == middle_busy2))
        {
          *(p_busy+i) = 1;
        }
        else
          *(p_busy+i) = 0;

    }

    return;
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                       */
/*      LEFT_TO_RIGHT                                    */
/*  input : pointer to binaryword                        */
/* output : pointer to unused, pointer to sequence       */
/* returns: NumberOfSteps                                */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */

                                                /* * * * LEFT TO RIGHT * * * * */
left_to_right(p_bin,p_unused,p_seq)
char *p_bin;
int *p_unused;
int *p_seq;
{
    int i;
    int counter = 0, counter1 = 0, counter2 = 0;

    for (counter2 = 0; counter2 <= (NUMBEROFBITS-1); ++counter2)
    {
      if (*(p_bin+counter2) == '1')
      {
          *(p_seq+counter) = counter2;
          ++counter;
      }
      else
      {
          *(p_unused+counter1) = counter2;
          counter1++;
      }

    }
    return(counter);
}


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                       */
/*      ADDBIT                                           */
/*  input :                                              */
/*           number of steps,                            */
/*           which unused dimension to add,              */
/*           pointer to unused, pointer to work_sequence*/
/* output : pointer to work_sequence                     */
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */

                                                    /* * * * ADD BIT * * * * */
adbit(number_of_steps, index, p_u, p_w)
t   number_of_steps;
int   index;
int *p_u;               /* pointer to unusedbits */
int *p_w;               /* pointer to work_sequence[0]*/
{
  int temp_sequence[NUMBEROFBITS+1]; /* temporary     */
  int i2;

  for (i2 = 0; i2 < number_of_steps; i2++)
    temp_sequence[i2] = *(p_w+i2);

              /* last element in work_sequence is
                                  unused dimension */
  *(p_w+number_of_steps+1) = *(p_u+index);

  for (i2 = number_of_steps; i2 > 0; --i2)
    *(p_w+i2) = temp_sequence[i2-1];

  *p_w = *(p_u+index);

                      /* last element in add_dim_sequence
                              is unused dimension */


  return;
}



 * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
                                                        */
/*      POWER                                           */
/*  input : base, n                                     */
/* returns: base ^ n                                    */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */


                                                    /* * * * POWER * * * * */
/* power: raise base to n-th power; n >= 0 */

power(base,n)
int   base, n;
{
        int i, p;
        p = 1;
        for (i = 1; i <= n; ++i)
                p = p * base;
        return p;
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                      */
/*      BINARY                                          */
/*  input : decimal d                                   */
/*  output: pointer to binaryword                       */
/*                                                      */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */

                                                    /* * * * BINARY * * * * */
 * binary: convert decimal number d to binaryword */

binary(d, p_bin)
int    d;
char *p_bin;
{
  int bit, test;
```

```
  test = d;
  for (bit = 0; bit < NUMBEROFBITS; bit++)
  {
                                                    /* * * * POWER * * * * */
        if (test >= power(2,NUMBEROFBITS-1-bit))
        {
          *(p_bin+bit) = '1';
          test = test - power(2,NUMBEROFBITS-1-bit);
        }
        else
        {
          *(p_bin+bit) = '0';
        }
  }
  *(p_bin+NUMBEROFBITS) = '\0';

  return;

}


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                       */
/*      HEADER                                           */
/*                                                       */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */

header()
{
  fprintf(fileptr, "\f\n%s", " BUSY NODES    ");
  fprintf(fileptr, "\t\t%s\t%s\t%s\t%s",
        " FAILURES/", "FAILURES/",
        "COMPLETE", "AVG PATH");
  fprintf(fileptr, "\n\t\t\t%s\t%s\t%s\t%s\t%s\n",
        " DIRECT", "PATHS", "ADDED DIMS",
         "FAILURES", "LENGTH");
  printf("\f\n%s", " BUSY NODES    ");
  printf("\t\t%s\t%s\t%s\t%s", "   FAILURES/",
        "FAILURES/", "COMPLETE", "AVG PATH");
  printf("\n\t\t\t%s\t%s\t%s\t%s\t%s\n", " DIRECT",
        "PATHS", "ADDED DIMS",
         "FAILURES", "LENGTH");
  return;
}




/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                       */
/*      MAIN                                             */
/*                                                       */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */

                                                    /* * * * PERFORM ROUTING ALGORITHM * * * * */

main()
{                               /* start main loop */


  int destination;              /* destination  */
  int source;                   /* source node  */
  int node;                     /* current node in path */
  char binaryword[NUMBEROFBITS+1];
                                      /* binary
                                      equivalent of
                                      difference
                                      vector          */
```

```
char *p_binaryword;
                                    /* pointer to
                                     binaryword     */

int unusedbits[NUMBEROFBITS];   /* 0 bits in
                                     difference
                                     vector         */
int *p_unusedbits;              /* pointer to
                                     unusedbits     */

int sequence[NUMBEROFBITS+1];
                                    /* 1 bits in
                                     difference
                                     vector         */
int *p_sequence;                /* pointer to
                                     sequence   */

int work_sequence[NUMBEROFBITS+1];
                                /* 1 bits in
                                 difference
                                 vector         */
int *p_work_sequence;           /* pointer to
                                   work sequence   */

int Perm[NUMBEROFPERMUTATIONS][NUMBEROFBITS+1];
                                /* every possible
                                   permutations of
                                   (NUMBEROFBITS+1) step
                                   sequence             */


int *p_Perm;                    /* pointer to Perm table*/
int permindex = 0;              /* index to Perm table  */

/* BUSY COMBINATIONS                                    */
int b1 = 0, b2 = 0;             /* busy nodes   */
int twos_counter = 0;           /* group of two */
int busy[NUMBEROFNODES];        /* table of busy
                                     addresses
                                     1 busy,
                                     0 not busy     */
int *p_busy;                    /* pointer to
                                     busy table     */


/* CONTROL                                              */
int PathFailed;                     /* loop control */
int ToTestPath;                     /* loop control */
int GoodPermindex;

/* MISC                                                 */
int p;
int max_permindex;              /* end of Perm table    */
int next_perm_change;
int bit_to_try, bit;
int step;                       /* step number in path                                  */
int x, y;
int i, m, count2, count3, c;
      int counter, counter2, counter3;

/* RESULTS                                              */

int Nodes[NUMBEROFBITS][NUMBEROFNODES][NUMBEROFNODES][NUMBEROFBITS+2];
int Dimen[NUMBEROFBITS][NUMBEROFNODES][NUMBEROFNODES][NUMBEROFBITS+3];

int Failed[][2];                /* Failed[x][0]
                                        dest,
                                   Failed[x][1]
                                    source      */
                                /* failed paths
```

```
                                    for given busy
                                    config          */
/* STATS                                            */
int NumberOfSteps;              /* length of sequence   */
                                /* number of steps from
                                   dest, source
                                  (-1 not tested, 0-5) */
int NumberOfWorkSteps;        /* length of work_sequence*/
int dimensions_not_taken;     /* number of elements in
                                 unusedbits             */
int path[8], dim[7];
int direct_path_failures, add_dim_failures;
int direct_paths, add_dims;
int complete_failures, fail_index;
int seqtries;
float avg_path_length;

/* FORMAT OUTPUT                                        */
int linecount;                          /* count lines
                                        printed on page */


fileptr = fopen("hypercube.d", "w");

                                                        /* * * * GENERATE PERM TABLE * * * */

p_Perm = &Perm[0][0];
generate_perm_table(p_Perm);  /* generate Perm table  */

linecount = 99;               /* force header()       */

                                                        /* * * * PICK SCENARIO * * * */
/* * * * * * * * * * * * * * * * * * * * * * * * * *  */
/*                                                     */
/*    FOR EACH BUSY COMBO                              */
/*                                                     */
/* * * * * * * * * * * * * * * * * * * * * * * * * *  */



                                                        /* * * * PICK BUSY NODES * * * */
for (twos_counter = 1; twos_counter <= 16; twos_counter++)

{
  for (b1 = 0; b1 < NUMBEROFNODES; b1++)
  {
    for (b2 = b1 + 1; b2 < NUMBEROFNODES; b2++)
      {
        p_busy = busy;
                                                        /* * * * GET BUSY * * * */
        getbusy(twos_counter, b1, b2, p_busy);

        /* * * * * * * * * * * * * * * * * * * * * * * */
        /*                                             */
        /*    STATISTICS INITIALIZATION                */
        /*                                             */
        /* * * * * * * * * * * * * * * * * * * * * * * */

                                                        /* * * * CLEAR COUNTS * * * */
        Failed[0][0] = -1;
        fail_index = 0;

        direct_path_failures = add_dim_failures = 0;
        direct_paths = add_dims = complete_failures = 0;
        avg_path_length = 0;


        for (counter = 0; counter < NUMBEROFBITS;
               ++counter)
          for (counter2 = 0; counter2 < NUMBEROFNODES;
```

```
      ++counter2)
    for (counter3 = 0; counter3 < NUMBEROFNODES;
      ++counter3)
      Nodes[counter][counter2][counter3][0] = -1;

/* * * * * * * * * * * * * * * * * * * * * * * * */
/*                                               */
/*    FOR EACH PATH                              */
/*                                               */
/* * * * * * * * * * * * * * * * * * * * * * * * */



                                                  /* * * * PICK SOURCE & DESTINATION * * * * */
for (destination = 0; destination < NUMBEROFNODES;
      ++destination)
{
  if (busy[destination])
            continue;

  for (source = destination + 1;
       source < NUMBEROFNODES; source++)
  {
    if (busy[source])
      continue;


    /* * * * * * * * * * * * * * * * * * * * * * */
    /*                                           */
    /*        INITIALIZE PATH                    */
    /*                                           */
    /* * * * * * * * * * * * * * * * * * * * * * */

                                                  /* * * * XOR SOURCE & DESTINATION * * * * */
                                                  /* * * * BINARY * * * * */
    binaryword[0] = '\0';
    p_binaryword = binaryword;    /* nothing */
    binary((source)^(destination),p_binaryword);

                                                  /* * * * LEFT TO RIGHT * * * * */
    p_unusedbits = unusedbits;
    p_sequence = sequence;
    NumberOfSteps =
      left_to_right(p_binaryword,
        p_unusedbits, p_sequence);



                                                  /* * * * TEST PATH * * * * */
    if (NumberOfSteps >= 1)
    {
      ToTestPath = 1;
      count2 = 0;     /* no extra dimension
                         direct path (Algor)   */
    }

    while (ToTestPath)
    {

      /*  * * * * * * * * * * * * * * * * * * * */
      /*                                         */
      /*      TEST PATHS                         */
      /* count2 = 0, direct path                 */
      /*  results in Nodes[0]                    */
      /* count2 = 1, 1st unused dimension        */
      /*  results in Nodes[1]                    */
      /* count2 = n, nth unused dimension        */
      /*  results in Nodes[n]                    */
      /*  * * * * * * * * * * * * * * * * * * * */
```

```
                                        /* * * * PICK ADDED DIMENSION * * * */
/* copy sequence to work_sequence */
for (i = 0; i < NumberOfSteps; i++)
  work_sequence[i] = *(p_sequence+i);
p_work_sequence = work_sequence;
NumberOfWorkSteps = NumberOfSteps;

/* initialize results, set work_sequence*/
if (count2 > 0)
{
                                        /* * * * ADD BIT * * * */
  addbit(NumberOfWorkSteps,
         count2-1, p_unusedbits, p_work_sequence);
  NumberOfWorkSteps = NumberOfWorkSteps + 2;
}

                                        /* * * * CALCULATE HOPS * * * */
                                        /* * * * INITIALIZE PATH * * * */
Nodes[count2][destination][source][0] = 1;


/* * * * * * * * * * * * * * * * * * * * * * * * */
/*                                               */
/*      TRY_PATH                                 */
/*                                               */
/* * * * * * * * * * * * * * * * * * * * * * * * */

seqtries = 1;
node = source;
permindex = 0;
dimensions_not_taken = NUMBEROFBITS - NumberOfWorkSteps;

for (bit_to_try = NumberOfWorkSteps;
     bit_to_try >= 1;
     bit_to_try--)
{

  while (1)
  {
    step = NUMBEROFBITS +1 - bit_to_try;
    next_perm_change = factorial(bit_to_try - 1);
    max_permindex = factorial(NUMBEROFBITS - step);
                                        /* * * * CALCULATE NEXT HOP * * * */
    bit =
        work_sequence[(Perm[permindex][step] -1 - dimensions_not_taken)];
                                        /* * * * POWER * * * */
    node = (node ^ power(2,NUMBEROFBITS-1-bit));

                                        /* * * * CHECK IF NODE BUSY * * * */
    if (!(busy[node]))
        break;

    node = source;

    /* NUMBEROFBITS -1 = step 6 change every element
        -2 = step 5 change every element
        -3 = step 4 change every 2
        -4 = step 3 change every 6
        -5 = step 2 change every 24
        -6 = step 1 change every 120
        -7 = step 0 change every 720
    */
                                        /* * * * COMPUTE NEXT PERMINDEX * * * */
    if (step < NUMBEROFBITS - 1)
    {
      p = 1;
      while (permindex >= next_perm_change*p)
        p++;
      permindex = next_perm_change * p;
    }
```

```
        else
          permindex++;

        seqtries++;
        GoodPermindex = 0;

        while ((count2 > 0)
            && (!(GoodPermindex))
            && (permindex < max_permindex))
        {
            for (p = NUMBEROFBITS +1 - NumberOfWorkSteps; p < NUMBEROFBITS; p++)

              if (((Perm[permindex][p] == NUMBEROFBITS)
                || (Perm[permindex][p] == NUMBEROFBITS +1 - NumberOfWorkSteps))
                && ((Perm[permindex][p+1] == NUMBEROFBITS)
                || (Perm[permindex][p+1] == NUMBEROFBITS +1 - NumberOfWorkSteps)))
              {
                permindex++;
                seqtries++;
                GoodPermindex = 0;
                break;
              }
              else
                GoodPermindex = 1;
        }


        if (permindex >= max_permindex)
        {
          Nodes[count2][destination][source][0] = 0;
          bit_to_try = 0;           /* will break from
                                       bit to try loop */
          break;
        }

        bit_to_try = NumberOfWorkSteps;

      }             /* end while (1) loop */
                                              /* * * * STORE STATS IN TABLES * * * * */

      Nodes[count2][destination][source][step - dimensions_not_taken] = node;
      Dimen[count2][destination][source][step - dimensions_not_taken] = bit;
      Dimen[count2][destination][source][0] = permindex;
      Dimen[count2][destination][source][NUMBEROFBITS+2] = seqtries;


    } /* end bit_to_try */



    if (Nodes[count2][destination][source][0] == 1)
        ToTestPath = 0;
    else  if (NumberOfSteps > NUMBEROFBITS-1)
        ToTestPath = 0;
    else if (count2 >=
      (NUMBEROFBITS - NumberOfSteps))
        ToTestPath = 0;
    else
        count2++;
}                           /* end while    */


/*  * * * * * * * * * * * * * * * * * *   */
/*                                        */
/*         CALC STATS FOR SOURCE - DEST   */
/*                                        */
/*  * * * * * * * * * * * * * * * * * *   */
                                          /* * * * MAINTAIN COUNTS * * * * */

PathFailed = 1;
```

```
if (Nodes[0][destination][source][0] == 1)
  /* direct path successful */
{
  PathFailed = 0;
  avg_path_length = avg_path_length +
    NumberOfSteps;

  direct_paths++;
}
else if (Nodes[0][destination][source][0] == 0)
  /* direct path failed */
{
  direct_path_failures++;
  for (counter = 1; counter <= NUMBEROFBITS;
     counter++)

    if (Nodes[counter][destination][source][0] == 1)
                /* extra dimension successful */
    {
      PathFailed = 0;
      avg_path_length = 2 + avg_path_length
            + NumberOfSteps;
      direct_paths++;
      break;
    }
  if (PathFailed)
  {
    Failed[fail_index][0] = destination;
    Failed[fail_index][1] = source;
    complete_failures++;
    fail_index++;
  }
}                /* end else if Nodes[0]... */


for (counter = 1; counter < NUMBEROFBITS;
   counter++)

  if (Nodes[counter][destination][source][0] >= 0)
                /* extra dimension tested */
  {
    add_dims++;
    if (Nodes[counter][destination][source][0] == 0)
      add_dim_failures++;
  }


/*  * * * * * * * * * * * * * * * * * *   */
/*                                        */
/*        PRINT STATS FOR SOURCE - DEST   */
/*                                        */
/*  * * * * * * * * * * * * * * * * * *   */
if (linecount > 65)
{
  header();
  linecount = 3;
}

/*      printf("\n\n%s%3d\t%s%3d\n", " Source ", source, " Destination ", destination);
*/

NumberOfWorkSteps = NumberOfSteps;
*       for (counter3 = 0; counter3 <= NUMBEROFBITS-1; counter3++)
{
    printf("\n%s%2d", "Nodes ", counter3);
    if (Nodes[counter3][destination][source][0]==1)
    {
      printf(" Pass\t");
      for (counter = 1;
           counter <= NumberOfWorkSteps;
```

```
                    ++counter)
                  printf("%3d",
                    Nodes[counter3][destination][source][counter]);
                printf("\nDimen\t\t");
                for (counter = 1;
                     counter <= NumberOfWorkSteps;
                     ++counter)
                  printf("%3d",
                    Dimen[counter3][destination][source][counter]);
                printf("%s%4d%s", "\nPerm[",
                  Dimen[counter3][destination][source][0], "]\t");
                for (counter = 1;
                   counter <= NumberOfWorkSteps;
                   ++counter)
                  printf("%3d",
                    Perm[Dimen[counter3][destination][source][0]][NUMBEROFBITS-(NumberOfWorkSteps
                printf("\t%s%4d", "Tries ", Dimen[counter3][destination][source][NUMBEROFBITS+2])
              }
              else if (Nodes[counter3][destination][source][0] == 0)
              {
                printf(" Fail\n");
                printf("%s%4d%s", "\nPerm[",
                  Dimen[counter3][destination][source][0], "]\t");
                printf("\t\t%s%4d", "Tries ", Dimen[counter3][destination][source][NUMBEROFBITS+2
              }

              printf("\n");
              if (counter3 == 0)
                NumberOfWorkSteps = NumberOfWorkSteps + 2;

            } *//* end for loop */

          }                   /* end source loop */

        }                     /* end destination loop */


        /* * * * * * * * * * * * * * * * * * * * * * *  */
        /*                                              */
        /*    CALC STATS FOR BUSY COMBO                 */
        /*                                              */
        /* * * * * * * * * * * * * * * * * * * * * * *  */
        if (direct_paths > 0)
          avg_path_length = avg_path_length / direct_paths;
        else
          printf("%s%3d%s%3d\n",
                 "no direct paths for dest ", destination,
                 " source ", source);

        /* * * * * * * * * * * * * * * * * * * * * * *  */
        /*                                              */
        /*    PRINT STATS FOR BUSY COMBO                */
        /*                                              */
        /* * * * * * * * * * * * * * * * * * * * * * *  */
/*      if (complete_failures > 0)
        {
*/
          fprintf(fileptr, "\n");
          printf("\n");
          for (counter = 0; counter < NUMBEROFNODES;
               ++counter)
            if (busy[counter])
            {
              fprintf(fileptr, "%3d", counter);
              printf("%3d", counter);
            }

          printf("\n\t");
          fprintf(fileptr, "\n\t");
          linecount++;
```

```
            linecount++;
            fprintf(fileptr,
                "\t\t%4d%s%4d\t%4d%s%4d\t%4d\t\t%1.5f",
                 direct_path_failures,"/", direct_paths,
                add_dim_failures, "/", add_dims,
                complete_failures, avg_path_length);

            printf("\t\t%4d%s%4d\t%4d%s%4d\t%4d\t\t%1.5f",
                 direct_path_failures,"/", direct_paths,
                add_dim_failures, "/", add_dims,
                complete_failures, avg_path_length);

            Failed[fail_index][0] = -1;
            counter = 0;
            while (Failed[counter][0] >=0)
                        /* for each failed path */
            {
              if (linecount > 65)
              {
                header();
                linecount = 3;
              }
              linecount++;
              fprintf(fileptr, "\n\t\t\t\t\t%3d%3d",
                 Failed[counter][0], Failed[counter][1]);
              printf("\n\t\t\t\t\t%3d%3d",
                 Failed[counter][0], Failed[counter][1]);
              counter++;
            }                   /* end for each failed path */
/*        }     */              /* end complete failures */


        }                               /* end b2 */
    }                                   /* end b1 */
  }                                     /* end twos */

}                                       /* end main loop */
```

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	COMPLETE FAILURES	AVG PATH LENGTH
0 33 62 63	1/1770	0/ 1	0	3.05141
0 34 62 63	1/1770	0/ 1	0	3.05028
0 35 62 63	0/1770	0/ 0	0	3.04972
0 36 62 63	1/1770	0/ 1	0	3.05028
0 37 62 63	0/1770	0/ 0	0	3.04972
0 38 62 63	1/1770	0/ 1	0	3.04972
0 39 62 63	1/1770	0/ 1	0	3.05028
0 40 62 63	1/1770	0/ 1	0	3.05028
0 41 62 63	0/1770	0/ 0	0	3.04972
0 42 62 63	1/1770	0/ 1	0	3.04972
0 43 62 63	1/1770	0/ 1	0	3.05028
0 44 62 63	1/1770	0/ 1	0	3.04972
0 45 62 63	1/1770	0/ 1	0	3.05028
0 46 62 63	0/1770	0/ 0	0	3.04802
0 47 62 63	0/1770	0/ 0	0	3.04859
0 48 62 63	1/1770	0/ 1	0	3.05028
0 49 62 63	0/1770	0/ 0	0	3.04972
0 50 62 63	1/1770	0/ 1	0	3.04972
0 51 62 63	1/1770	0/ 1	0	3.05028
0 52 62 63	1/1770	0/ 1	0	3.04972
0 53 62 63	1/1770	0/ 1	0	3.05028
0 54 62 63	0/1770	0/ 0	0	3.04802
0 55 62 63	0/1770	0/ 0	0	3.04859
0 56 62 63	1/1770	0/ 1	0	3.04972
0 57 62 63	1/1770	0/ 1	0	3.05028
0 58 62 63	0/1770	0/ 0	0	3.04802
0 59 62 63	0/1770	0/ 0	0	3.04859
0 60 62 63	0/1770	0/ 0	0	3.04802
0 61 62 63	0/1770	0/ 0	0	3.04859
0 62 63	0/1830	0/ 0	0	3.04918
0 62 63	0/1830	0/ 0	0	3.04918
1 2 62 63	1/1770	0/ 1	0	3.05141

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	COMPLETE FAILURES	AVG PATH LENGTH
1 3 62 63	0/1770	0/ 0	0	3.04972
1 4 62 63	1/1770	0/ 1	0	3.05141
1 5 62 63	0/1770	0/ 0	0	3.04972
1 6 62 63	0/1770	0/ 0	0	3.04972
1 7 62 63	1/1770	0/ 1	0	3.05028
1 8 62 63	1/1770	0/ 1	0	3.05141
1 9 62 63	0/1770	0/ 0	0	3.04972
1 10 62 63	0/1770	0/ 0	0	3.04972
1 11 62 63	1/1770	0/ 1	0	3.05028
1 12 62 63	0/1770	0/ 0	0	3.04972
1 13 62 63	1/1770	0/ 1	0	3.05028
1 14 62 63	1/1770	0/ 1	0	3.05028
1 15 62 63	1/1770	0/ 1	0	3.04972
1 16 62 63	1/1770	0/ 1	0	3.05141
1 17 62 63	0/1770	0/ 0	0	3.04972
1 18 62 63	0/1770	0/ 0	0	3.04972
1 19 62 63	1/1770	0/ 1	0	3.05028
1 20 62 63	0/1770	0/ 0	0	3.04972
1 21 62 63	1/1770	0/ 1	0	3.05028
1 22 62 63	1/1770	0/ 1	0	3.05028
1 23 62 63	1/1770	0/ 1	0	3.04972
1 24 62 63	0/1770	0/ 0	0	3.04972
1 25 62 63	1/1770	0/ 1	0	3.05028
1 26 62 63	1/1770	0/ 1	0	3.05028
1 27 62 63	1/1770	0/ 1	0	3.04972
1 28 62 63	1/1770	0/ 1	0	3.05028
1 29 62 63	1/1770	0/ 1	0	3.04972
1 30 62 63	0/1770	0/ 0	0	3.04859
1 31 62 63	0/1770	0/ 0	0	3.04802
1 32 62 63	1/1770	0/ 1	0	3.05141
1 33 62 63	0/1770	0/ 0	0	3.04972
1 34 62 63	0/1770	0/ 0	0	3.04972

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	COMPLETE FAILURES	AVG PATH LENGTH
1 35 62 63	1/1770	0/ 1	0	3.05028
1 36 62 63	0/1770	0/ 0	0	3.04972
1 37 62 63	1/1770	0/ 1	0	3.05028
1 38 62 63	1/1770	0/ 1	0	3.05028
1 39 62 63	1/1770	0/ 1	0	3.04972
1 40 62 63	0/1770	0/ 0	0	3.04972
1 41 62 63	1/1770	0/ 1	0	3.05028
1 42 62 63	1/1770	0/ 1	0	3.05028
1 43 62 63	1/1770	0/ 1	0	3.04972
1 44 62 63	1/1770	0/ 1	0	3.05028
1 45 62 63	1/1770	0/ 1	0	3.04972
1 46 62 63	0/1770	0/ 0	0	3.04859
1 47 62 63	0/1770	0/ 0	0	3.04802
1 48 62 63	0/1770	0/ 0	0	3.04972
1 49 62 63	1/1770	0/ 1	0	3.05028
1 50 62 63	1/1770	0/ 1	0	3.05028
1 51 62 63	1/1770	0/ 1	0	3.04972
1 52 62 63	1/1770	0/ 1	0	3.05028
1 53 62 63	1/1770	0/ 1	0	3.04972
1 54 62 63	0/1770	0/ 0	0	3.04859
1 55 62 63	0/1770	0/ 0	0	3.04802
1 56 62 63	1/1770	0/ 1	0	3.05028
1 57 62 63	1/1770	0/ 1	0	3.04972
1 58 62 63	0/1770	0/ 0	0	3.04859
1 59 62 63	0/1770	0/ 0	0	3.04802
1 60 62 63	0/1770	0/ 0	0	3.04859
1 61 62 63	0/1770	0/ 0	0	3.04802
1 62 63	0/1830	0/ 0	0	3.04918
1 62 63	0/1830	0/ 0	0	3.04918
2 3 62 63	0/1770	0/ 0	0	3.04859
2 4 62 63	1/1770	0/ 1	0	3.05028
2 5 62 63	0/1770	0/ 0	0	3.04972

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	COMPLETE FAILURES	AVG PATH LENGTH
2 6 62 63	0/1770	0/ 0	0	3.04746
2 7 62 63	1/1770	0/ 1	0	3.04915
2 8 62 63	1/1770	0/ 1	0	3.05028
2 9 62 63	0/1770	0/ 0	0	3.04972
2 10 62 63	0/1770	0/ 0	0	3.04746
2 11 62 63	1/1770	0/ 1	0	3.04915
2 12 62 63	0/1770	0/ 0	0	3.04859
2 13 62 63	0/1770	0/ 0	0	3.04915
2 14 62 63	2/1770	0/ 2	0	3.04915
2 15 62 63	1/1770	0/ 1	0	3.04859
2 16 62 63	1/1770	0/ 1	0	3.05028
2 17 62 63	0/1770	0/ 0	0	3.04972
2 18 62 63	0/1770	0/ 0	0	3.04746
2 19 62 63	1/1770	0/ 1	0	3.04915
2 20 62 63	0/1770	0/ 0	0	3.04859
2 21 62 63	0/1770	0/ 0	0	3.04915
2 22 62 63	2/1770	0/ 2	0	3.04915
2 23 62 63	1/1770	0/ 1	0	3.04859
2 24 62 63	0/1770	0/ 0	0	3.04859
2 25 62 63	0/1770	0/ 0	0	3.04915
2 26 62 63	2/1770	0/ 2	0	3.04915
2 27 62 63	1/1770	0/ 1	0	3.04859
2 28 62 63	1/1770	0/ 1	0	3.04915
2 29 62 63	1/1770	0/ 1	0	3.04972
2 30 62 63	0/1770	0/ 0	0	3.04633
2 31 62 63	0/1770	0/ 0	0	3.04689
2 32 62 63	1/1770	0/ 1	0	3.05028
2 33 62 63	0/1770	0/ 0	0	3.04972
2 34 62 63	0/1770	0/ 0	0	3.04746
2 35 62 63	1/1770	0/ 1	0	3.04915
2 36 62 63	0/1770	0/ 0	0	3.04859
2 37 62 63	0/1770	0/ 0	0	3.04915

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	COMPLETE FAILURES	AVG PATH LENGTH
2 38 62 63	2/1770	0/ 2	0	3.04915
2 39 62 63	1/1770	0/ 1	0	3.04859
2 40 62 63	0/1770	0/ 0	0	3.04859
2 41 62 63	0/1770	0/ 0	0	3.04915
2 42 62 63	2/1770	0/ 2	0	3.04915
2 43 62 63	1/1770	0/ 1	0	3.04859
2 44 62 63	1/1770	0/ 1	0	3.04915
2 45 62 63	1/1770	0/ 1	0	3.04972
2 46 62 63	0/1770	0/ 0	0	3.04633
2 47 62 63	0/1770	0/ 0	0	3.04689
2 48 62 63	0/1770	0/ 0	0	3.04859
2 49 62 63	0/1770	0/ 0	0	3.04915
2 50 62 63	2/1770	0/ 2	0	3.04915
2 51 62 63	1/1770	0/ 1	0	3.04859
2 52 62 63	1/1770	0/ 1	0	3.04915
2 53 62 63	1/1770	0/ 1	0	3.04972
2 54 62 63	0/1770	0/ 0	0	3.04633
2 55 62 63	0/1770	0/ 0	0	3.04689
2 56 62 63	1/1770	0/ 1	0	3.04915
2 57 62 63	1/1770	0/ 1	0	3.04972
2 58 62 63	0/1770	0/ 0	0	3.04633
2 59 62 63	0/1770	0/ 0	0	3.04689
2 60 62 63	0/1770	0/ 0	0	3.04746
2 61 62 63	0/1770	0/ 0	0	3.04802
2 62 63	0/1830	0/ 0	0	3.04809
2 62 63	0/1830	0/ 0	0	3.04809
3 4 62 63	0/1770	0/ 0	0	3.04972
3 5 62 63	1/1770	0/ 1	0	3.05028
3 6 62 63	1/1770	0/ 1	0	3.04915
3 7 62 63	0/1770	0/ 0	0	3.04746
3 8 62 63	0/1770	0/ 0	0	3.04972
3 9 62 63	1/1770	0/ 1	0	3.05028

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	COMPLETE FAILURES	AVG PATH LENGTH
3 10 62 63	1/1770	0/ 1	0	3.04915
3 11 62 63	0/1770	0/ 0	0	3.04746
3 12 62 63	0/1770	0/ 0	0	3.04915
3 13 62 63	0/1770	0/ 0	0	3.04859
3 14 62 63	1/1770	0/ 1	0	3.04859
3 15 62 63	2/1770	0/ 2	0	3.04915
3 16 62 63	0/1770	0/ 0	0	3.04972
3 17 62 63	1/1770	0/ 1	0	3.05028
3 18 62 63	1/1770	0/ 1	0	3.04915
3 19 62 63	0/1770	0/ 0	0	3.04746
3 20 62 63	0/1770	0/ 0	0	3.04915
3 21 62 63	0/1770	0/ 0	0	3.04859
3 22 62 63	1/1770	0/ 1	0	3.04859
3 23 62 63	2/1770	0/ 2	0	3.04915
3 24 62 63	0/1770	0/ 0	0	3.04915
3 25 62 63	0/1770	0/ 0	0	3.04859
3 26 62 63	1/1770	0/ 1	0	3.04859
3 27 62 63	2/1770	0/ 2	0	3.04915
3 28 62 63	1/1770	0/ 1	0	3.04972
3 29 62 63	1/1770	0/ 1	0	3.04915
3 30 62 63	0/1770	0/ 0	0	3.04689
3 31 62 63	0/1770	0/ 0	0	3.04633
3 32 62 63	0/1770	0/ 0	0	3.04972
3 33 62 63	1/1770	0/ 1	0	3.05028
3 34 62 63	1/1770	0/ 1	0	3.04915
3 35 62 63	0/1770	0/ 0	0	3.04746
3 36 62 63	0/1770	0/ 0	0	3.04915
3 37 62 63	0/1770	0/ 0	0	3.04859
3 38 62 63	1/1770	0/ 1	0	3.04859
3 39 62 63	2/1770	0/ 2	0	3.04915
3 40 62 63	0/1770	0/ 0	0	3.04915
3 41 62 63	0/1770	0/ 0	0	3.04859

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	COMPLETE FAILURES	AVG PATH LENGTH
3 42 62 63	1/1770	0/ 1	0	3.04859
3 43 62 63	2/1770	0/ 2	0	3.04915
3 44 62 63	1/1770	0/ 1	0	3.04972
3 45 62 63	1/1770	0/ 1	0	3.04915
3 46 62 63	0/1770	0/ 0	0	3.04689
3 47 62 63	0/1770	0/ 0		

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
0 1 62 63	0/1770	0/ 0	0/ 0	0	3.05085
0 2 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 3 62 63	1/1770	0/ 4	0/ 0	0	3.05141
0 4 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 5 62 63	1/1770	1/ 5	0/ 0	0	3.05141
0 6 62 63	1/1770	1/ 5	0/ 0	0	3.05028
0 7 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 8 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 9 62 63	1/1770	1/ 6	0/ 0	0	3.05141
0 10 62 63	1/1770	1/ 6	0/ 0	0	3.05028
0 11 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 12 62 63	1/1770	1/ 6	0/ 0	0	3.05028
0 13 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 14 62 63	1/1770	1/ 5	0/ 0	0	3.04972
0 15 62 63	1/1770	0/ 5	0/ 0	0	3.05028
0 16 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 17 62 63	1/1770	1/ 6	0/ 0	0	3.05141
0 18 62 63	1/1770	1/ 6	0/ 0	0	3.05028
0 19 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 20 62 63	1/1770	1/ 6	0/ 0	0	3.05028
0 21 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 22 62 63	1/1770	1/ 5	0/ 0	0	3.04972
0 23 62 63	1/1770	0/ 5	0/ 0	0	3.05028
0 24 62 63	1/1770	1/ 6	0/ 0	0	3.05028
0 25 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 26 62 63	1/1770	0/ 4	0/ 0	0	3.04972
0 27 62 63	1/1770	0/ 4	0/ 0	0	3.05028
0 28 62 63	1/1770	0/ 5	0/ 0	0	3.04972
0 29 62 63	1/1770	0/ 5	0/ 0	0	3.05028
0 30 62 63	0/1770	0/ 0	0/ 0	0	3.04802
0 31 62 63	0/1770	0/ 0	0/ 0	0	3.04859
0 32 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 33 62 63	1/1770	1/ 6	0/ 0	0	3.05141

0 34 62 63	1/1770	1/ 6	0/ 0	0	3.05028
0 35 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 36 62 63	1/1770	0/ 4	0/ 0	0	3.05028
0 37 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 38 62 63	1/1770	0/ 4	0/ 0	0	3.04972
0 39 62 63	1/1770	1/ 5	0/ 0	0	3.05028
0 40 62 63	1/1770	1/ 6	0/ 0	0	3.05028
0 41 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 42 62 63	1/1770	0/ 4	0/ 0	0	3.04972
0 43 62 63	1/1770	0/ 4	0/ 0	0	3.05028
0 44 62 63	1/1770	0/ 4	0/ 0	0	3.04972
0 45 62 63	1/1770	0/ 5	0/ 0	0	3.05028
0 46 62 63	0/1770	0/ 0	0/ 0	0	3.04802
0 47 62 63	0/1770	0/ 0	0/ 0	0	3.04859
0 48 62 63	1/1770	1/ 6	0/ 0	0	3.05028
0 49 62 63	0/1770	0/ 0	0/ 0	0	3.04972
0 50 62 63	1/1770	0/ 4	0/ 0	0	3.04972
0 51 62 63	1/1770	0/ 4	0/ 0	0	3.05028
0 52 62 63	1/1770	0/ 4	0/ 0	0	3.04972
0 53 62 63	1/1770	0/ 4	0/ 0	0	3.05028
0 54 62 63	0/1770	0/ 0	0/ 0	0	3.04802
0 55 62 63	0/1770	0/ 0	0/ 0	0	3.04859
0 56 62 63	1/1770	0/ 4	0/ 0	0	3.04972
0 57 62 63	1/1770	0/ 4	0/ 0	0	3.05028
0 58 62 63	0/1770	0/ 0	0/ 0	0	3.04802
0 59 62 63	0/1770	0/ 0	0/ 0	0	3.04859
0 60 62 63	0/1770	0/ 0	0/ 0	0	3.04802
0 61 62 63	0/1770	0/ 0	0/ 0	0	3.04859
0 62 63	0/1830	0/ 0	0/ 0	0	3.04918
0 62 63	0/1830	0/ 0	0/ 0	0	3.04918
1 2 62 63	1/1770	0/ 4	0/ 0	0	3.05141
1 3 62 63	0/1770	0/ 0	0/ 0	0	3.04972
1 4 62 63	1/1770	0/ 5	0/ 0	0	3.05141
1 5 62 63	0/1770	0/ 0	0/ 0	0	3.04972
1 6 62 63	0/1770	0/ 0	0/ 0	0	3.04972

1	7	62	63	1/1770	0/ 5	0/ 0	0 3.05028
1	8	62	63	1/1770	0/ 4	0/ 0	0 3.05141
1	9	62	63	0/1770	0/ 0	0/ 0	0 3.04972
1	10	62	63	0/1770	0/ 0	0/ 0	0 3.04972
1	11	62	63	1/1770	0/ 4	0/ 0	0 3.05028
1	12	62	63	0/1770	0/ 0	0/ 0	0 3.04972
1	13	62	63	1/1770	0/ 4	0/ 0	0 3.05028
1	14	62	63	1/1770	1/ 5	0/ 0	0 3.05028
1	15	62	63	1/1770	0/ 5	0/ 0	0 3.04972
1	16	62	63	1/1770	0/ 4	0/ 0	0 3.05141
1	17	62	63	0/1770	0/ 0	0/ 0	0 3.04972
1	18	62	63	0/1770	0/ 0	0/ 0	0 3.04972
1	19	62	63	1/1770	0/ 4	0/ 0	0 3.05028
1	20	62	63	0/1770	0/ 0	0/ 0	0 3.04972
1	21	62	63	1/1770	0/ 4	0/ 0	0 3.05028
1	22	62	63	1/1770	1/ 5	0/ 0	0 3.05028
1	23	62	63	1/1770	0/ 5	0/ 0	0 3.04972
1	24	62	63	0/1770	0/ 0	0/ 0	0 3.04972
1	25	62	63	1/1770	0/ 4	0/ 0	0 3.05028
1	26	62	63	1/1770	0/ 4	0/ 0	0 3.05028
1	27	62	63	1/1770	0/ 4	0/ 0	0 3.04972
1	28	62	63	1/1770	0/ 5	0/ 0	0 3.05028
1	29	62	63	1/1770	0/ 5	0/ 0	0 3.04972
1	30	62	63	0/1770	0/ 0	0/ 0	0 3.04859
1	31	62	63	0/1770	0/ 0	0/ 0	0 3.04802
1	32	62	63	1/1770	0/ 4	0/ 0	0 3.05141
1	33	62	63	0/1770	0/ 0	0/ 0	0 3.04972
1	34	62	63	0/1770	0/ 0	0/ 0	0 3.04972
1	35	62	63	1/1770	0/ 4	0/ 0	0 3.05028
1	36	62	63	0/1770	0/ 0	0/ 0	0 3.04972
1	37	62	63	1/1770	0/ 4	0/ 0	0 3.05028
1	38	62	63	1/1770	0/ 4	0/ 0	0 3.05028
1	39	62	63	1/1770	1/ 5	0/ 0	0 3.04972
1	40	62	63	0/1770	0/ 0	0/ 0	0 3.04972
1	41	62	63	1/1770	0/ 4	0/ 0	0 3.05028

1 42 62 63	1/1770	0/	4	0/	0	0	3.05028
1 43 62 63	1/1770	0/	4	0/	0	0	3.04972
1 44 62 63	1/1770	0/	4	0/	0	0	3.05028
1 45 62 63	1/1770	0/	5	0/	0	0	3.04972
1 46 62 63	0/1770	0/	0	0/	0	0	3.04859
1 47 62 63	0/1770	0/	0	0/	0	0	3.04802
1 48 62 63	0/1770	0/	0	0/	0	0	3.04972
1 49 62 63	1/1770	0/	4	0/	0	0	3.05028
1 50 62 63	1/1770	0/	4	0/	0	0	3.05028
1 51 62 63	1/1770	0/	4	0/	0	0	3.04972
1 52 62 63	1/1770	0/	4	0/	0	0	3.05028
1 53 62 63	1/1770	0/	4	0/	0	0	3.04972
1 54 62 63	0/1770	0/	0	0/	0	0	3.04859
1 55 62 63	0/1770	0/	0	0/	0	0	3.04802
1 56 62 63	1/1770	0/	4	0/	0	0	3.05028
1 57 62 63	1/1770	0/	4	0/	0	0	3.04972
1 58 62 63	0/1770	0/	0	0/	0	0	3.04859
1 59 62 63	0/1770	0/	0	0/	0	0	3.04802
1 60 62 63	0/1770	0/	0	0/	0	0	3.04859
1 61 62 63	0/1770	0/	0	0/	0	0	3.04802
1 62 63	0/1830	0/	0	0/	0	0	3.04918
1 62 63	0/1830	0/	0	0/	0	0	3.04918
2 3 62 63	0/1770	0/	0	0/	0	0	3.04859
2 4 62 63	1/1770	0/	5	0/	0	0	3.05028
2 5 62 63	0/1770	0/	0	0/	0	0	3.04972
2 6 62 63	0/1770	0/	0	0/	0	0	3.04746
2 7 62 63	1/1770	0/	5	0/	0	0	3.04915
2 8 62 63	1/1770	0/	4	0/	0	0	3.05028
2 9 62 63	0/1770	0/	0	0/	0	0	3.04972
2 10 62 63	0/1770	0/	0	0/	0	0	3.04746
2 11 62 63	1/1770	0/	4	0/	0	0	3.04915
2 12 62 63	0/1770	0/	0	0/	0	0	3.04859
2 13 62 63	0/1770	0/	0	0/	0	0	3.04915
2 14 62 63	2/1770	1/	9	0/	0	0	3.04915
2 15 62 63	1/1770	0/	5	0/	0	0	3.04859

2 16 62 63	1/1770	0/	4	0/	0	0	3.05028
2 17 62 63	0/1770	0/	0	0/	0	0	3.04972
2 18 62 63	0/1770	0/	0	0/	0	0	3.04746
2 19 62 63	1/1770	0/	4	0/	0	0	3.04915
2 20 62 63	0/1770	0/	0	0/	0	0	3.04859
2 21 62 63	0/1770	0/	0	0/	0	0	3.04915
2 22 62 63	2/1770	1/	9	0/	0	0	3.04915
2 23 62 63	1/1770	0/	5	0/	0	0	3.04859
2 24 62 63	0/1770	0/	0	0/	0	0	3.04859
2 25 62 63	0/1770	0/	0	0/	0	0	3.04915
2 26 62 63	2/1770	0/	8	0/	0	0	3.04915
2 27 62 63	1/1770	0/	4	0/	0	0	3.04859
2 28 62 63	1/1770	0/	5	0/	0	0	3.04915
2 29 62 63	1/1770	0/	5	0/	0	0	3.04972
2 30 62 63	0/1770	0/	0	0/	0	0	3.04633
2 31 62 63	0/1770	0/	0	0/	0	0	3.04689
2 32 62 63	1/1770	0/	4	0/	0	0	3.05028
2 33 62 63	0/1770	0/	0	0/	0	0	3.04972
2 34 62 63	0/1770	0/	0	0/	0	0	3.04746
2 35 62 63	1/1770	0/	4	0/	0	0	3.04915
2 36 62 63	0/1770	0/	0	0/	0	0	3.04859
2 37 62 63	0/1770	0/	0	0/	0	0	3.04915
2 38 62 63	2/1770	0/	8	0/	0	0	3.04915
2 39 62 63	1/1770	1/	5	0/	0	0	3.04859
2 40 62 63	0/1770	0/	0	0/	0	0	3.04859
2 41 62 63	0/1770	0/	0	0/	0	0	3.04915
2 42 62 63	2/1770	0/	8	0/	0	0	3.04915
2 43 62 63	1/1770	0/	4	0/	0	0	3.04859
2 44 62 63	1/1770	0/	4	0/	0	0	3.04915
2 45 62 63	1/1770	0/	5	0/	0	0	3.04972
2 46 62 63	0/1770	0/	0	0/	0	0	3.04633
2 47 62 63	0/1770	0/	0	0/	0	0	3.04689
2 48 62 63	0/1770	0/	0	0/	0	0	3.04859
2 49 62 63	0/1770	0/	0	0/	0	0	3.04915
2 50 62 63	2/1770	0/	8	0/	0	0	3.04915

2 51 62 63	1/1770	0/	4	0/	0	0	3.04859
2 52 62 63	1/1770	0/	4	0/	0	0	3.04915
2 53 62 63	1/1770	0/	4	0/	0	0	3.04972
2 54 62 63	0/1770	0/	0	0/	0	0	3.04633
2 55 62 63	0/1770	0/	0	0/	0	0	3.04689
2 56 62 63	1/1770	0/	4	0/	0	0	3.04915
2 57 62 63	1/1770	0/	4	0/	0	0	3.04972
2 58 62 63	0/1770	0/	0	0/	0	0	3.04633
2 59 62 63	0/1770	0/	0	0/	0	0	3.04689
2 60 62 63	0/1770	0/	0	0/	0	0	3.04746
2 61 62 63	0/1770	0/	0	0/	0	0	3.04802
2 62 63	0/1830	0/	0	0/	0	0	3.04809
2 62 63	0/1830	0/	0	0/	0	0	3.04809
3 4 62 63	0/1770	0/	0	0/	0	0	3.04972
3 5 62 63	1/1770	1/	5	0/	0	0	3.05028
3 6 62 63	1/1770	1/	5	0/	0	0	3.04915
3 7 62 63	0/1770	0/	0	0/	0	0	3.04746
3 8 62 63	0/1770	0/	0	0/	0	0	3.04972
3 9 62 63	1/1770	0/	4	0/	0	0	3.05028
3 10 62 63	1/1770	0/	4	0/	0	0	3.04915
3 11 62 63	0/1770	0/	0	0/	0	0	3.04746
3 12 62 63	0/1770	0/	0	0/	0	0	3.04915
3 13 62 63	0/1770	0/	0	0/	0	0	3.04859
3 14 62 63	1/1770	1/	5	0/	0	0	3.04859
3 15 62 63	2/1770	0/	9	0/	0	0	3.04915
3 16 62 63	0/1770	0/	0	0/	0	0	3.04972
3 17 62 63	1/1770	0/	4	0/	0	0	3.05028
3 18 62 63	1/1770	0/	4	0/	0	0	3.04915
3 19 62 63	0/1770	0/	0	0/	0	0	3.04746
3 20 62 63	0/1770	0/	0	0/	0	0	3.04915
3 21 62 63	0/1770	0/	0	0/	0	0	3.04859
3 22 62 63	1/1770	1/	5	0/	0	0	3.04859
3 23 62 63	2/1770	0/	9	0/	0	0	3.04915
3 24 62 63	0/1770	0/	0	0/	0	0	3.04915
3 25 62 63	0/1770	0/	0	0/	0	0	3.04859

3 26 62 63	1/1770	0/	4	0/	0	0	3.04859
3 27 62 63	2/1770	0/	8	0/	0	0	3.04915
3 28 62 63	1/1770	0/	5	0/	0	0	3.04972
3 29 62 63	1/1770	0/	5	0/	0	0	3.04915
3 30 62 63	0/1770	0/	0	0/	0	0	3.04689
3 31 62 63	0/1770	0/	0	0/	0	0	3.04633
3 32 62 63	0/1770	0/	0	0/	0	0	3.04972
3 33 62 63	1/1770	0/	4	0/	0	0	3.05028
3 34 62 63	1/1770	0/	4	0/	0	0	3.04915
3 35 62 63	0/1770	0/	0	0/	0	0	3.04746
3 36 62 63	0/1770	0/	0	0/	0	0	3.04915
3 37 62 63	0/1770	0/	0	0/	0	0	3.04859
3 38 62 63	1/1770	0/	4	0/	0	0	3.04859
3 39 62 63	2/1770	1/	9	0/	0	0	3.04915
3 40 62 63	0/1770	0/	0	0/	0	0	3.04915
3 41 62 63	0/1770	0/	0	0/	0	0	3.04859
3 42 62 63	1/1770	0/	4	0/	0	0	3.04859
3 43 62 63	2/1770	0/	8	0/	0	0	3.04915
3 44 62 63	1/1770	0/	4	0/	0	0	3.04972
3 45 62 63	1/1770	0/	5	0/	0	0	3.04915
3 46 62 63	0/1770	0/	0	0/	0	0	3.04689
3 47 62 63	0/1770	0/	0	0/	0	0	3.04633
3 48 62 63	0/1770	0/	0	0/	0	0	3.04915
3 49 62 63	0/1770	0/	0	0/	0	0	3.04859
3 50 62 63	1/1770	0/	4	0/	0	0	3.04859
3 51 62 63	2/1770	0/	8	0/	0	0	3.04915
3 52 62 63	1/1770	0/	4	0/	0	0	3.04972
3 53 62 63	1/1770	0/	4	0/	0	0	3.04915
3 54 62 63	0/1770	0/	0	0/	0	0	3.04689
3 55 62 63	0/1770	0/	0	0/	0	0	3.04633
3 56 62 63	1/1770	0/	4	0/	0	0	3.04972
3 57 62 63	1/1770	0/	4	0/	0	0	3.04915
3 58 62 63	0/1770	0/	0	0/	0	0	3.04689
3 59 62 63	0/1770	0/	0	0/	0	0	3.04633
3 60 62 63	0/1770	0/	0	0/	0	0	3.04802

3 61 62 63	0/1770	0/ 0	0/ 0	0	3.04746
3 62 63	0/1830	0/ 0	0/ 0	0	3.04809
3 62 63	0/1830	0/ 0	0/ 0	0	3.04809
4 5 62 63	0/1770	0/ 0	0/ 0	0	3.04859
4 6 62 63	0/1770	0/ 0	0/ 0	0	3.04746
4 7 62 63	1/1770	0/ 5	0/ 0	0	3.04915
4 8 62 63	1/1770	1/ 5	0/ 0	0	3.05028
4 9 62 63	0/1770	0/ 0	0/ 0	0	3.04972
4 10 62 63	0/1770	0/ 0	0/ 0	0	3.04859
4 11 62 63	0/1770	0/ 0	0/ 0	0	3.04915
4 12 62 63	0/1770	0/ 0	0/ 0	0	3.04746
4 13 62 63	1/1770	1/ 5	0/ 0	0	3.04915
4 14 62 63	2/1770	1/ 9	0/ 0	0	3.04915
4 15 62 63	1/1770	0/ 5	0/ 0	0	3.04859
4 16 62 63	1/1770	1/ 5	0/ 0	0	3.05028
4 17 62 63	0/1770	0/ 0	0/ 0	0	3.04972
4 18 62 63	0/1770	0/ 0	0/ 0	0	3.04859
4 19 62 63	0/1770	0/ 0	0/ 0	0	3.04915
4 20 62 63	0/1770	0/ 0	0/ 0	0	3.04746
4 21 62 63	1/1770	0/ 4	0/ 0	0	3.04915
4 22 62 63	2/1770	1/ 9	0/ 0	0	3.04915
4 23 62 63	1/1770	0/ 5	0/ 0	0	3.04859
4 24 62 63	0/1770	0/ 0	0/ 0	0	3.04859
4 25 62 63	0/1770	0/ 0	0/ 0	0	3.04915
4 26 62 63	1/1770	0/ 4	0/ 0	0	3.04915
4 27 62 63	1/1770	0/ 4	0/ 0	0	3.04972
4 28 62 63	2/1770	1/ 10	0/ 0	0	3.04915
4 29 62 63	1/1770	0/ 5	0/ 0	0	3.04859
4 30 62 63	0/1770	0/ 0	0/ 0	0	3.04633
4 31 62 63	0/1770	0/ 0	0/ 0	0	3.04689
4 32 62 63	1/1770	0/ 5	0/ 0	0	3.05028
4 33 62 63	0/1770	0/ 0	0/ 0	0	3.04972
4 34 62 63	0/1770	0/ 0	0/ 0	0	3.04859
4 35 62 63	0/1770	0/ 0	0/ 0	0	3.04915
4 36 62 63	0/1770	0/ 0	0/ 0	0	3.04746

4 37 62 63	1/1770	0/	5	0/	0	0	3.04915
4 38 62 63	2/1770	0/	9	0/	0	0	3.04915
4 39 62 63	1/1770	1/	5	0/	0	0	3.04859
4 40 62 63	0/1770	0/	0	0/	0	0	3.04859
4 41 62 63	0/1770	0/	0	0/	0	0	3.04915
4 42 62 63	1/1770	0/	4	0/	0	0	3.04915
4 43 62 63	1/1770	0/	4	0/	0	0	3.04972
4 44 62 63	2/1770	0/	8	0/	0	0	3.04915
4 45 62 63	1/1770	0/	5	0/	0	0	3.04859
4 46 62 63	0/1770	0/	0	0/	0	0	3.04633
4 47 62 63	0/1770	0/	0	0/	0	0	3.04689
4 48 62 63	0/1770	0/	0	0/	0	0	3.04859
4 49 62 63	0/1770	0/	0	0/	0	0	3.04915
4 50 62 63	1/1770	0/	4	0/	0	0	3.04915
4 51 62 63	1/1770	0/	4	0/	0	0	3.04972
4 52 62 63	2/1770	0/	8	0/	0	0	3.04915
4 53 62 63	1/1770	0/	4	0/	0	0	3.04859
4 54 62 63	0/1770	0/	0	0/	0	0	3.04633
4 55 62 63	0/1770	0/	0	0/	0	0	3.04689
4 56 62 63	1/1770	0/	4	0/	0	0	3.04915
4 57 62 63	1/1770	0/	4	0/	0	0	3.04972
4 58 62 63	0/1770	0/	0	0/	0	0	3.04746
4 59 62 63	0/1770	0/	0	0/	0	0	3.04802
4 60 62 63	0/1770	0/	0	0/	0	0	3.04633
4 61 62 63	0/1770	0/	0	0/	0	0	3.04689
4 62 63	0/1830	0/	0	0/	0	0	3.04809
4 62 63	0/1830	0/	0	0/	0	0	3.04809
5 6 62 63	1/1770	1/	5	0/	0	0	3.04915
5 7 62 63	0/1770	0/	0	0/	0	0	3.04746
5 8 62 63	0/1770	0/	0	0/	0	0	3.04972
5 9 62 63	1/1770	0/	5	0/	0	0	3.05028
5 10 62 63	0/1770	0/	0	0/	0	0	3.04915
5 11 62 63	0/1770	0/	0	0/	0	0	3.04859
5 12 62 63	1/1770	0/	5	0/	0	0	3.04915
5 13 62 63	0/1770	0/	0	0/	0	0	3.04746

5 14 62 63	1/1770	1/ 5	0/ 0	0	0	3.04859
5 15 62 63	2/1770	0/ 9	0/ 0	0	0	3.04915
5 16 62 63	0/1770	0/ 0	0/ 0	0	0	3.04972
5 17 62 63	1/1770	0/ 5	0/ 0	0	0	3.05028
5 18 62 63	0/1770	0/ 0	0/ 0	0	0	3.04915
5 19 62 63	0/1770	0/ 0	0/ 0	0	0	3.04859
5 20 62 63	1/1770	0/ 5	0/ 0	0	0	3.04915
5 21 62 63	0/1770	0/ 0	0/ 0	0	0	3.04746
5 22 62 63	1/1770	1/ 5	0/ 0	0	0	3.04859
5 23 62 63	2/1770	0/ 10	0/ 0	0	0	3.04915
5 24 62 63	0/1770	0/ 0	0/ 0	0	0	3.04915
5 25 62 63	0/1770	0/ 0	0/ 0	0	0	3.04859
5 26 62 63	1/1770	0/ 4	0/ 0	0	0	3.04972
5 27 62 63	1/1770	0/ 4	0/ 0	0	0	3.04915
5 28 62 63	1/1770	0/ 5	0/ 0	0	0	3.04859
5 29 62 63	2/1770	0/ 9	0/ 0	0	0	3.04915
5 30 62 63	0/1770	0/ 0	0/ 0	0	0	3.04689
5 31 62 63	0/1770	0/ 0	0/ 0	0	0	3.04633
5 32 62 63	0/1770	0/ 0	0/ 0	0	0	3.04972
5 33 62 63	1/1770	0/ 5	0/ 0	0	0	3.05028
5 34 62 63	0/1770	0/ 0	0/ 0	0	0	3.04915
5 35 62 63	0/1770	0/ 0	0/ 0	0	0	3.04859
5 36 62 63	1/1770	0/ 4	0/ 0	0	0	3.04915
5 37 62 63	0/1770	0/ 0	0/ 0	0	0	3.04746
5 38 62 63	1/1770	0/ 4	0/ 0	0	0	3.04859
5 39 62 63	2/1770	1/ 10	0/ 0	0	0	3.04915
5 40 62 63	0/1770	0/ 0	0/ 0	0	0	3.04915
5 41 62 63	0/1770	0/ 0	0/ 0	0	0	3.04859
5 42 62 63	1/1770	0/ 4	0/ 0	0	0	3.04972
5 43 62 63	1/1770	0/ 4	0/ 0	0	0	3.04915
5 44 62 63	1/1770	0/ 4	0/ 0	0	0	3.04859
5 45 62 63	2/1770	0/ 10	0/ 0	0	0	3.04915
5 46 62 63	0/1770	0/ 0	0/ 0	0	0	3.04689
5 47 62 63	0/1770	0/ 0	0/ 0	0	0	3.04633
5 48 62 63	0/1770	0/ 0	0/ 0	0	0	3.04915

5 49 62 63	0/1770	0/ 0	0/ 0	0	3.04859
5 50 62 63	1/1770	0/ 4	0/ 0	0	3.04972
5 51 62 63	1/1770	0/ 4	0/ 0	0	3.04915
5 52 62 63	1/1770	0/ 4	0/ 0	0	3.04859
5 53 62 63	2/1770	0/ 8	0/ 0	0	3.04915
5 54 62 63	0/1770	0/ 0	0/ 0	0	3.04689
5 55 62 63	0/1770	0/ 0	0/ 0	0	3.04633
5 56 62 63	1/1770	0/ 4	0/ 0	0	3.04972
5 57 62 63	1/1770	0/ 4	0/ 0	0	3.04915
5 58 62 63	0/1770	0/ 0	0/ 0	0	3.04802
5 59 62 63	0/1770	0/ 0	0/ 0	0	3.04746
5 60 62 63	0/1770	0/ 0	0/ 0	0	3.04689
5 61 62 63	0/1770	0/ 0	0/ 0	0	3.04633
5 62 63	0/1830	0/ 0	0/ 0	0	3.04809
5 62 63	0/1830	0/ 0	0/ 0	0	3.04809
6 7 62 63	0/1770	0/ 0	0/ 0	0	3.04633
6 8 62 63	0/1770	0/ 0	0/ 0	0	3.04859
6 9 62 63	0/1770	0/ 0	0/ 0	0	3.04915
6 10 62 63	1/1770	0/ 5	0/ 0	0	3.04802
6 11 62 63	0/1770	0/ 0	0/ 0	0	3.04746
6 12 62 63	1/1770	0/ 5	0/ 0	0	3.04802
6 13 62 63	0/1770	0/ 0	0/ 0	0	3.04746
6 14 62 63	1/1770	0/ 5	0/ 0	0	3.04633
6 15 62 63	2/1770	0/ 9	0/ 0	0	3.04802
6 16 62 63	0/1770	0/ 0	0/ 0	0	3.04859
6 17 62 63	0/1770	0/ 0	0/ 0	0	3.04915
6 18 62 63	1/1770	0/ 5	0/ 0	0	3.04802
6 19 62 63	0/1770	0/ 0	0/ 0	0	3.04746
6 20 62 63	1/1770	0/ 5	0/ 0	0	3.04802
6 21 62 63	0/1770	0/ 0	0/ 0	0	3.04746
6 22 62 63	1/1770	0/ 5	0/ 0	0	3.04633
6 23 62 63	2/1770	0/ 10	0/ 0	0	3.04802
6 24 62 63	0/1770	0/ 0	0/ 0	0	3.04802
6 25 62 63	0/1770	0/ 0	0/ 0	0	3.04859
6 26 62 63	1/1770	0/ 4	0/ 0	0	3.04746

6 27 62 63	1/1770	0/	4	0/	0	0	3.04802
6 28 62 63	1/1770	0/	5	0/	0	0	3.04746
6 29 62 63	1/1770	0/	5	0/	0	0	3.04802
6 30 62 63	1/1770	0/	4	0/	0	0	3.04576
6 31 62 63	0/1770	0/	0	0/	0	0	3.04520
6 32 62 63	0/1770	0/	0	0/	0	0	3.04859
6 33 62 63	0/1770	0/	0	0/	0	0	3.04915
6 34 62 63	1/1770	1/	5	0/	0	0	3.04802
6 35 62 63	0/1770	0/	0	0/	0	0	3.04746
6 36 62 63	1/1770	1/	5	0/	0	0	3.04802
6 37 62 63	0/1770	0/	0	0/	0	0	3.04746
6 38 62 63	1/1770	0/	4	0/	0	0	3.04633
6 39 62 63	2/1770	2/	10	0/	0	0	3.04802
6 40 62 63	0/1770	0/	0	0/	0	0	3.04802
6 41 62 63	0/1770	0/	0	0/	0	0	3.04859
6 42 62 63	1/1770	0/	4	0/	0	0	3.04746
6 43 62 63	1/1770	0/	4	0/	0	0	3.04802
6 44 62 63	1/1770	0/	4	0/	0	0	3.04746
6 45 62 63	1/1770	0/	5	0/	0	0	3.04802
6 46 62 63	1/1770	1/	5	0/	0	0	3.04576
6 47 62 63	0/1770	0/	0	0/	0	0	3.04520
6 48 62 63	0/1770	0/	0	0/	0	0	3.04802
6 49 62 63	0/1770	0/	0	0/	0	0	3.04859
6 50 62 63	1/1770	0/	4	0/	0	0	3.04746
6 51 62 63	1/1770	0/	4	0/	0	0	3.04802
6 52							

w/ unrevised routing tables

hypercube.d

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	COMPLETE FAILURES	AVG PATH LENGTH
0 1 62 63	0/1770	0/ 0	0	3.05085
0 2 62 63	0/1770	0/ 0	0	3.04972
0 3 62 63	1/1770	0/ 1	0	3.05141
0 4 62 63	0/1770	0/ 0	0	3.04972
0 5 62 63	1/1770	0/ 1	0	3.05141
0 6 62 63	1/1770	0/ 1	0	3.05028
0 7 62 63	0/1770	0/ 0	0	3.04972
0 8 62 63	0/1770	0/ 0	0	3.04972
0 9 62 63	1/1770	0/ 1	0	3.05141
0 10 62 63	1/1770	0/ 1	0	3.05028
0 11 62 63	0/1770	0/ 0	0	3.04972
0 12 62 63	1/1770	0/ 1	0	3.05028
0 13 62 63	0/1770	0/ 0	0	3.04972
0 14 62 63	1/1770	0/ 1	0	3.04972
0 15 62 63	1/1770	0/ 1	0	3.05028
0 16 62 63	0/1770	0/ 0	0	3.04972
0 17 62 63	1/1770	0/ 1	0	3.05141
0 18 62 63	1/1770	0/ 1	0	3.05028
0 19 62 63	0/1770	0/ 0	0	3.04972
0 20 62 63	1/1770	0/ 1	0	3.05028
0 21 62 63	0/1770	0/ 0	0	3.04972
0 22 62 63	1/1770	0/ 1	0	3.04972
0 23 62 63	1/1770	0/ 1	0	3.05028
0 24 62 63	1/1770	0/ 1	0	3.05028
0 25 62 63	0/1770	0/ 0	0	3.04972
0 26 62 63	1/1770	0/ 1	0	3.04972
0 27 62 63	1/1770	0/ 1	0	3.05028
0 28 62 63	1/1770	0/ 1	0	3.04972
0 29 62 63	1/1770	0/ 1	0	3.05028
0 30 62 63	0/1770	0/ 0	0	3.04802
0 31 62 63	0/1770	0/ 0	0	3.04859
0 32 62 63	0/1770	0/ 0	0	3.04972

```
before permutat
worksequence 012345
    rm[  0] 012345
    rm[  1] 012354
    rm[  2] 012453
   Perm[  3] 012435
   Perm[  4] 012534
   Perm[  5] 012543
   Perm[  6] 013452
   Perm[  7] 013425
   Perm[  8] 013524
   Perm[  9] 013542
   Perm[ 10] 013245
   Perm[ 11] 013254
   Perm[ 12] 014523
   Perm[ 13] 014532
   Perm[ 14] 014235
   Perm[ 15] 014253
   Perm[ 16] 014352
   Perm[ 17] 014325
   Perm[ 18] 015234
   Perm[ 19] 015243
   Perm[ 20] 015342
   Perm[ 21] 015324
   Perm[ 22] 015423
   Perm[ 23] 015432
   Perm[ 24] 023451
   Perm[ 25] 023415
   Perm[ 26] 023514
   Perm[ 27] 023541
   Perm[ 28] 023145
   Perm[ 29] 023154
   Perm[ 30] 024513
   Perm[ 31] 024531
   Perm[ 32] 024135
   Perm[ 33] 024153
   Perm[ 34] 024351
   Perm[ 35] 024315
   Perm[ 36] 025134
   Perm[ 37] 025143
   Perm[ 38] 025341
   Perm[ 39] 025314
   Perm[ 40] 025413
   Perm[ 41] 025431
   Perm[ 42] 021345
   Perm[ 43] 021354
   Perm[ 44] 021453
   Perm[ 45] 021435
   Perm[ 46] 021534
   Perm[ 47] 021543
   Perm[ 48] 034512
   Perm[ 49] 034521
   Perm[ 50] 034125
   Perm[ 51] 034152
   Perm[ 52] 034251
   Perm[ 53] 034215
   Perm[ 54] 035124
   Perm[ 55] 035142
   Perm[ 56] 035241
   Perm[ 57] 035214
   Perm[ 58] 035412
   Perm[ 59] 035421
    erm[ 60] 031245
    erm[ 61] 031254
    erm[ 62] 031452
   Perm[ 63] 031425
   Perm[ 64] 031524
   Perm[ 65] 031542
   Perm[ 66] 032451
   Perm[ 67] 032415
```

Now 7 digits!

LENGTH 6

last but

NUMBER OF BITS - 1 △ every element

-2 △ every element

-3 △ every

-4 △ six

-5 △ 24

-6 △ 120

345
354
435
453
534
543

LENGTH 5, 4

1st 120

last 5 digits

1st 24

last 4

6

last 3

24

length +2

no arrangements

PERM 4 <= 3

PERM 5 <= 119

PERM 6 <= 719

2 is not together

1 is not too

0 is too

Perm[68] 032514
Perm[69] 032541
rm[70] 032145
rm[71] 032154
rm[72] 045123
Perm[73] 045132
Perm[74] 045231
Perm[75] 045213
Perm[76] 045312
Perm[77] 045321
Perm[78] 041235
Perm[79] 041253
Perm[80] 041352
Perm[81] 041325
Perm[82] 041523
Perm[83] 041532
Perm[84] 042351
Perm[85] 042315
Perm[86] 042513
Perm[87] 042531
Perm[88] 042135
Perm[89] 042153
Perm[90] 043512
Perm[91] 043521
Perm[92] 043125
Perm[93] 043152
Perm[94] 043251
Perm[95] 043215
Perm[96] 051234
Perm[97] 051243
Perm[98] 051342
Perm[99] 051324
rm[100] 051423
rm[101] 051432
rm[102] 052341
Perm[103] 052314
Perm[104] 052413
Perm[105] 052431
Perm[106] 052134
Perm[107] 052143
Perm[108] 053412
Perm[109] 053421
Perm[110] 053124
Perm[111] 053142
Perm[112] 053241
Perm[113] 053214
Perm[114] 054123
Perm[115] 054132
Perm[116] 054231
Perm[117] 054213
Perm[118] 054312
Perm[119] 054321
Perm[120] 123450
Perm[121] 123405
Perm[122] 123504
Perm[123] 123540
Perm[124] 123045
Perm[125] 123054
Perm[126] 124503
Perm[127] 124530
Perm[128] 124035
Perm[129] 124053
erm[130] 124350
erm[131] 124305
erm[132] 125034
erm[133] 125043
Perm[134] 125340
Perm[135] 125304
Perm[136] 125403
Perm[137] 125430

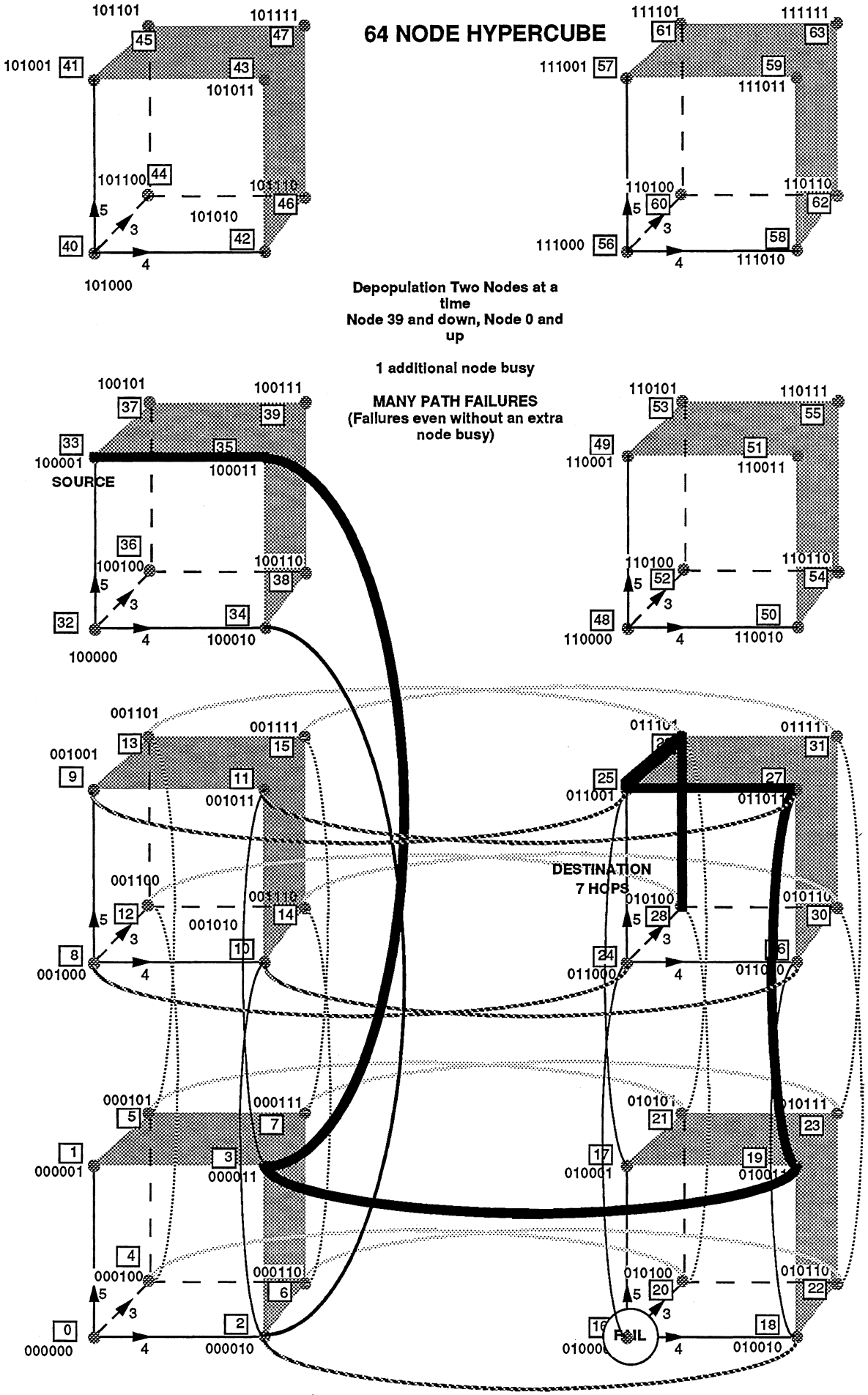
Perm[138] 120345
Perm[139] 120354
Perm[140] 120453
Perm[141] 120435
Perm[142] 120534
Perm[143] 120543
Perm[144] 134502
Perm[145] 134520
Perm[146] 134025
Perm[147] 134052
Perm[148] 134250
Perm[149] 134205
Perm[150] 135024
Perm[151] 135042
Perm[152] 135240
Perm[153] 135204
Perm[154] 135402
Perm[155] 135420
Perm[156] 130245
Perm[157] 130254
Perm[158] 130452
Perm[159] 130425
Perm[160] 130524
Perm[161] 130542
Perm[162] 132450
Perm[163] 132405
Perm[164] 132504
Perm[165] 132540
Perm[166] 132045
Perm[167] 132054
Perm[168] 145023
Perm[169] 145032
Perm[170] 145230
Perm[171] 145203
Perm[172] 145302
Perm[173] 145320
Perm[174] 140235
Perm[175] 140253
Perm[176] 140352
Perm[177] 140325
Perm[178] 140523
Perm[179] 140532
Perm[180] 142350
Perm[181] 142305
Perm[182] 142503
Perm[183] 142530
Perm[184] 142035
Perm[185] 142053
Perm[186] 143502
Perm[187] 143520
Perm[188] 143025
Perm[189] 143052
Perm[190] 143250
Perm[191] 143205
Perm[192] 150234
Perm[193] 150243
Perm[194] 150342
Perm[195] 150324
Perm[196] 150423
Perm[197] 150432
Perm[198] 152340
Perm[199] 152304
Perm[200] 152403
Perm[201] 152430
Perm[202] 152034
Perm[203] 152043
Perm[204] 153402
Perm[205] 153420
Perm[206] 153024
Perm[207] 153042

```
Perm[208] 153240
Perm[209] 153204
  rm[210] 154023
  rm[211] 154032
  rm[212] 154230
Perm[213] 154203
Perm[214] 154302
Perm[215] 154320
Perm[216] 102345
Perm[217] 102354
Perm[218] 102453
Perm[219] 102435
Perm[220] 102534
Perm[221] 102543
Perm[222] 103452
Perm[223] 103425
Perm[224] 103524
Perm[225] 103542
Perm[226] 103245
Perm[227] 103254
Perm[228] 104523
Perm[229] 104532
Perm[230] 104235
Perm[231] 104253
Perm[232] 104352
Perm[233] 104325
Perm[234] 105234
Perm[235] 105243
Perm[236] 105342
Perm[237] 105324
Perm[238] 105423
Perm[239] 105432
rm[240] 234501
rm[241] 234510
rm[242] 234015
Perm[243] 234051
Perm[244] 234150
Perm[245] 234105
Perm[246] 235014
Perm[247] 235041
Perm[248] 235140
Perm[249] 235104
Perm[250] 235401
Perm[251] 235410
Perm[252] 230145
Perm[253] 230154
Perm[254] 230451
Perm[255] 230415
Perm[256] 230514
Perm[257] 230541
Perm[258] 231450
Perm[259] 231405
Perm[260] 231504
Perm[261] 231540
Perm[262] 231045
Perm[263] 231054
Perm[264] 245013
Perm[265] 245031
Perm[266] 245130
Perm[267] 245103
Perm[268] 245301
Perm[269] 245310
erm[270] 240135
erm[271] 240153
erm[272] 240351
erm[273] 240315
Perm[274] 240513
Perm[275] 240531
Perm[276] 241350
Perm[277] 241305
```

```
Perm[278] 241503
Perm[279] 241530
 rm[280] 241035
 rm[281] 241053
 rm[282] 243501
Perm[283] 243510
Perm[284] 243015
Perm[285] 243051
Perm[286] 243150
Perm[287] 243105
Perm[288] 250134
Perm[289] 250143
Perm[290] 250341
Perm[291] 250314
Perm[292] 250413
Perm[293] 250431
Perm[294] 251340
Perm[295] 251304
Perm[296] 251403
Perm[297] 251430
Perm[298] 251034
Perm[299] 251043
Perm[300] 253401
Perm[301] 253410
Perm[302] 253014
Perm[303] 253041
Perm[304] 253140
Perm[305] 253104
Perm[306] 254013
Perm[307] 254031
Perm[308] 254130
Perm[309] 254103
 rm[310] 254301
 rm[311] 254310
 rm[312] 201345
Perm[313] 201354
Perm[314] 201453
Perm[315] 201435
Perm[316] 201534
Perm[317] 201543
Perm[318] 203451
Perm[319] 203415
Perm[320] 203514
Perm[321] 203541
Perm[322] 203145
Perm[323] 203154
Perm[324] 204513
Perm[325] 204531
Perm[326] 204135
Perm[327] 204153
Perm[328] 204351
Perm[329] 204315
Perm[330] 205134
Perm[331] 205143
Perm[332] 205341
Perm[333] 205314
Perm[334] 205413
Perm[335] 205431
Perm[336] 213450
Perm[337] 213405
Perm[338] 213504
Perm[339] 213540
 erm[340] 213045
 erm[341] 213054
 erm[342] 214503
Perm[343] 214530
Perm[344] 214035
Perm[345] 214053
Perm[346] 214350
Perm[347] 214305
```

```
Perm[348] 215034
Perm[349] 215043
rm[350] 215340
rm[351] 215304
rm[352] 215403
Perm[353] 215430
Perm[354] 210345
Perm[355] 210354
Perm[356] 210453
Perm[357] 210435
Perm[358] 210534
Perm[359] 210543
Perm[360] 345012
Perm[361] 345021
Perm[362] 345120
Perm[363] 345102
Perm[364] 345201
Perm[365] 345210
Perm[366] 340125
Perm[367] 340152
Perm[368] 340251
Perm[369] 340215
Perm[370] 340512
Perm[371] 340521
Perm[372] 341250
Perm[373] 341205
Perm[374] 341502
Perm[375] 341520
Perm[376] 341025
Perm[377] 341052
Perm[378] 342501
rm[379] 342510
rm[380] 342015
rm[381] 342051
rm[382] 342150
Perm[383] 342105
Perm[384] 350124
Perm[385] 350142
Perm[386] 350241
Perm[387] 350214
Perm[388] 350412
Perm[389] 350421
Perm[390] 351240
Perm[391] 351204
Perm[392] 351402
Perm[393] 351420
Perm[394] 351024
Perm[395] 351042
Perm[396] 352401
Perm[397] 352410
Perm[398] 352014
Perm[399] 352041
Perm[400] 352140
Perm[401] 352104
Perm[402] 354012
Perm[403] 354021
Perm[404] 354120
Perm[405] 354102
Perm[406] 354201
Perm[407] 354210
Perm[408] 301245
Perm[409] 301254
erm[410] 301452
erm[411] 301425
erm[412] 301524
Perm[413] 301542
Perm[414] 302451
Perm[415] 302415
Perm[416] 302514
Perm[417] 302541
```

```
Perm[418] 302145
Perm[419] 302154
 rm[420] 304512
 rm[421] 304521
 rm[422] 304125
Perm[423] 304152
Perm[424] 304251
Perm[425] 304215
Perm[426] 305124
Perm[427] 305142
Perm[428] 305241
Perm[429] 305214
Perm[430] 305412
Perm[431] 305421
Perm[432] 312450
Perm[433] 312405
Perm[434] 312504
Perm[435] 312540
Perm[436] 312045
Perm[437] 312054
Perm[438] 314502
Perm[439] 314520
Perm[440] 314025
Perm[441] 314052
Perm[442] 314250
Perm[443] 314205
Perm[444] 315024
Perm[445] 315042
Perm[446] 315240
Perm[447] 315204
Perm[448] 315402
Perm[449] 315420
 rm[450] 310245
 rm[451] 310254
 rm[452] 310452
Perm[453] 310425
Perm[454] 310524
Perm[455] 310542
Perm[456] 324501
Perm[457] 324510
Perm[458] 324015
Perm[459] 324051
Perm[460] 324150
Perm[461] 324105
Perm[462] 325014
Perm[463] 325041
Perm[464] 325140
Perm[465] 325104
Perm[466] 325401
Perm[467] 325410
Perm[468] 320145
Perm[469] 320154
Perm[470] 320451
Perm[471] 320415
Perm[472] 320514
Perm[473] 320541
Perm[474] 321450
Perm[475] 321405
Perm[476] 321504
Perm[477] 321540
Perm[478] 321045
Perm[479] 321
```


64 NODE HYPERCUBE
Depopulation Two Nodes at a time
Node 39 and down, Node 0 and up
1 additional node busy
MANY PATH FAILURES
(Failures even without an extra node busy)
SOURCE
DESTINATION
7 HOPS
FAIL
101101
101111
101001
101011
101100
101110
101010
101000
111101
111111
111001
111011
110100
110110
111000
111010
100101
100111
100001
100011
100100
100110
100000
100010
110101
110111
110001
110011
110100
110110
110000
110010
001101
001111
001001
001011
001100
001110
001010
001000
011101
011111
011001
011011
010100
010110
011000
011010
000101
000111
000001
000011
000100
000110
000000
000010
010101
010111
010001
010011
010100
010110
010000
010010

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
0 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	3/ 666	2/ 11	0/ 0	0	2.83784
1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	3/ 666	2/ 11	0/ 0	0	2.83784
2 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	7/ 666	9/ 23	0/ 0	0	2.84685
3 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	7/ 666	9/ 23	0/ 0	0	2.84685
4 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	7/ 666	9/ 23	0/ 0	0	2.84685
5 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	7/ 666	9/ 23	0/ 0	0	2.84685
6 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.82282
7 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.82282
8 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81982
9 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81982
10 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81682
11 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81682
12 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81682
13 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81682
14 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81381
15 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81381
16 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81982
17 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81982
18 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81682
19 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81682
20 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81682
21 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81682
22 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81381
23 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81381
24 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81081
25 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.81081
26 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.80781
27 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.80781
28 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.80781
29 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.80781
30 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.80480
31 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.80480

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
32 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	1/ 666	2/ 4	0/ 0	0	2.79279
33 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	1/ 666	2/ 4	0/ 0	0	2.79279
34 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.78679
35 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.78679
36 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.78679
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 666	0/ 0	0/ 0	0	2.78679
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	0/ 703	0/ 0	0/ 0	0	2.81223

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 2 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	25/ 595	33/ 77	0/ 0	0	2.94118
0 1 3 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	25/ 595	33/ 77	0/ 0	0	2.94118
0 1 4 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	25/ 595	33/ 78	0/ 0	0	2.94118
0 1 5 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	25/ 595	33/ 77	0/ 0	0	2.94118
0 1 6 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	13/ 595	15/ 43	0/ 0	0	2.90084
0 1 7 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	13/ 595	14/ 42	0/ 0	0	2.90084
0 1 8 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	10/ 595	11/ 31	0/ 0	0	2.88403
0 1 9 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	10/ 595	11/ 31	0/ 0	0	2.88403
0 1 10 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 595	16/ 45	0/ 0	0	2.89748
0 1 11 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 595	16/ 45	0/ 0	0	2.89748
0 1 12 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 595	16/ 45	0/ 0	0	2.89748
0 1 13 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 595	16/ 45	0/ 0	0	2.89748
0 1 14 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 595	14/ 38	0/ 0	0	2.89076
0 1 15 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 595	14/ 38	0/ 0	0	2.89076
0 1 16 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	10/ 595	11/ 31	0/ 0	0	2.88403
0 1 17 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	10/ 595	11/ 31	0/ 0	0	2.88403
0 1 18 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 595	16/ 45	0/ 0	0	2.89748
0 1 19 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 595	16/ 45	0/ 0	0	2.89748
0 1 20 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 595	16/ 45	0/ 0	0	2.89748
0 1 21 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 595	16/ 45	0/ 0	0	2.89748
0 1 22 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 595	14/ 38	0/ 0	0	2.89076
0 1 23 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 595	14/ 38	0/ 0	0	2.89076
0 1 24 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	11/ 595	14/ 37	0/ 0	0	2.88067
0 1 25 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	11/ 595	14/ 37	0/ 0	0	2.88067
0 1 26 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 595	14/ 38	0/ 0	0	2.88403
0 1 27 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 595	14/ 38	0/ 0	0	2.88403
0 1 28 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 595	14/ 38	0/ 0	0	2.88403
0 1 29 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 595	14/ 38	0/ 0	0	2.88403
0 1 30 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 595	14/ 38	0/ 0	0	2.88403
0 1 31 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 595	14/ 38	0/ 0	0	2.88403

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
0 1 32 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	7/ 595	9/ 23	0/ 0	0	2.84034
0 1 33 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	7/ 595	9/ 23	0/ 0	0	2.84034
0 1 34 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	16/ 595	21/ 50	0/ 0	0	2.87059
0 1 35 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	16/ 595	21/ 50	0/ 0	0	2.87059
0 1 36 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	16/ 595	21/ 51	0/ 0	0	2.87059
0 1 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	16/ 595	21/ 50	0/ 0	0	2.87059
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254
0 1 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	12/ 630	14/ 38	0/ 0	0	2.88254

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
0 1 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 561	50/ 76 28 33 29 32	0/ 0	2	2.85740
0 1 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 561	50/ 76 28 33 29 32	0/ 0	2	2.85740
0 1 2 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	49/ 528	91/ 129 4 32 8 32 12 32 16 32 20 32 24 32 28 32 28 33 28 34 29 32 30 32	0/ 0	11	2.86553
0 1 3 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	49/ 528	91/ 130 5 33 9 33 13 33 17 33 21 33 25 33 28 33 29 32 29 33 29 35 31 33	0/ 0	11	2.86553
0 1 4 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	26/ 528	45/ 69 28 33 29 32	0/ 0	2	2.85985
0 1 5 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	26/ 528	45/ 69 28 33 29 32	0/ 0	2	2.85985
0 1 6 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	30/ 528	52/ 84 4 32 28 33 29 32	0/ 0	3	2.87121
0 1 7 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	30/ 528	52/ 84 5 33 28 33 29 32	0/ 0	3	2.87121
0 1 8 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	26/ 528	45/ 69 28 33 29 32	0/ 0	2	2.85985
0 1 9 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	26/ 528	45/ 69 28 33 29 32	0/ 0	2	2.85985
0 1 10 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	30/ 528	52/ 83 8 32	0/ 0	3	2.87121

64 NODE CUBE

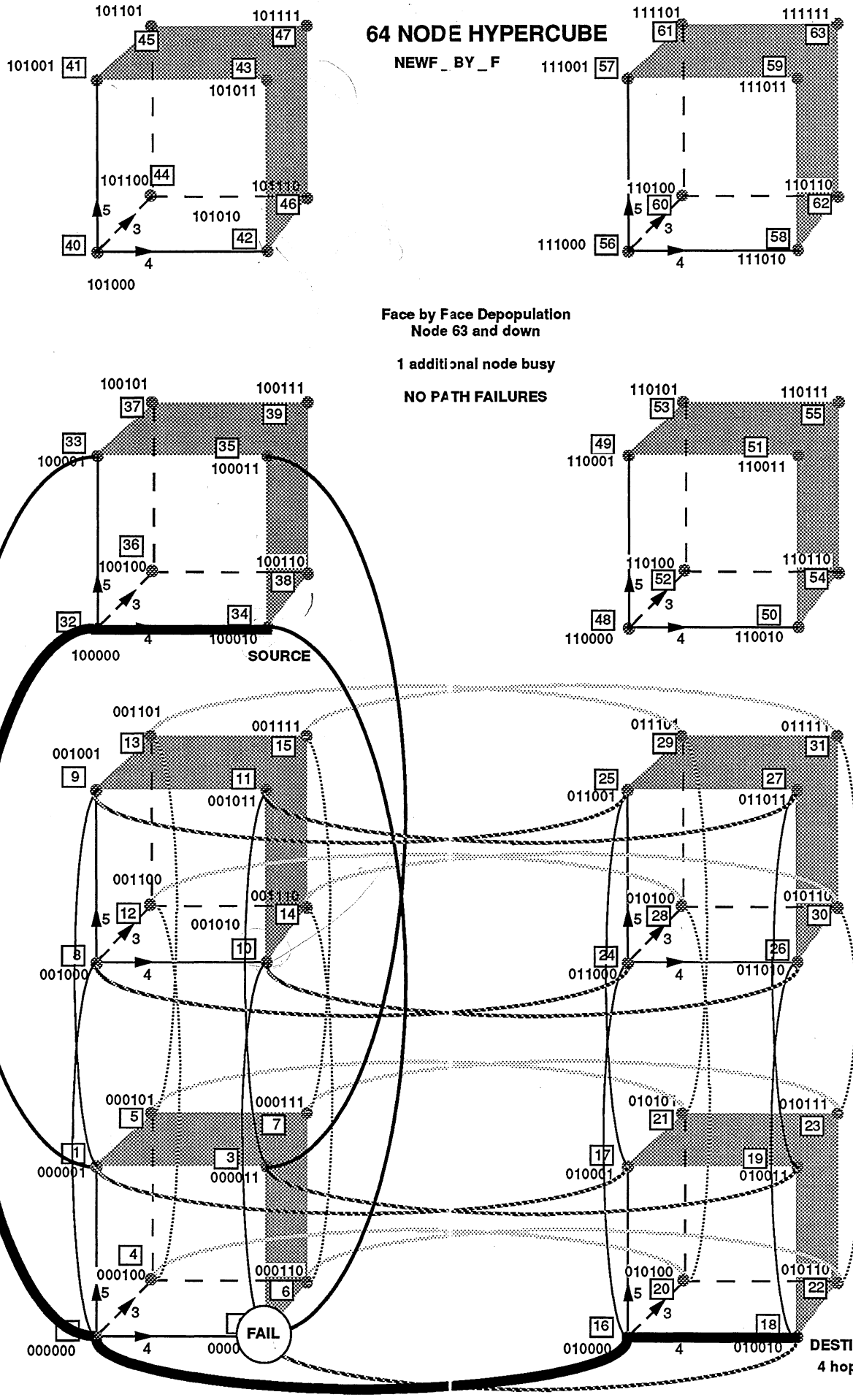
To depopulate a cube face by face, remove the face (4 nodes on one surface of a subcube) with the highest addresses. Reconfigure the hypercube to be less one dimension (the highest order bit) whenever the number of nonbusy nodes goes below the next lower power of 2, ie. reduce a 64 node cube to 32, a 32 node cube to 16, a 16 to 8, and 8 to 4, etc.

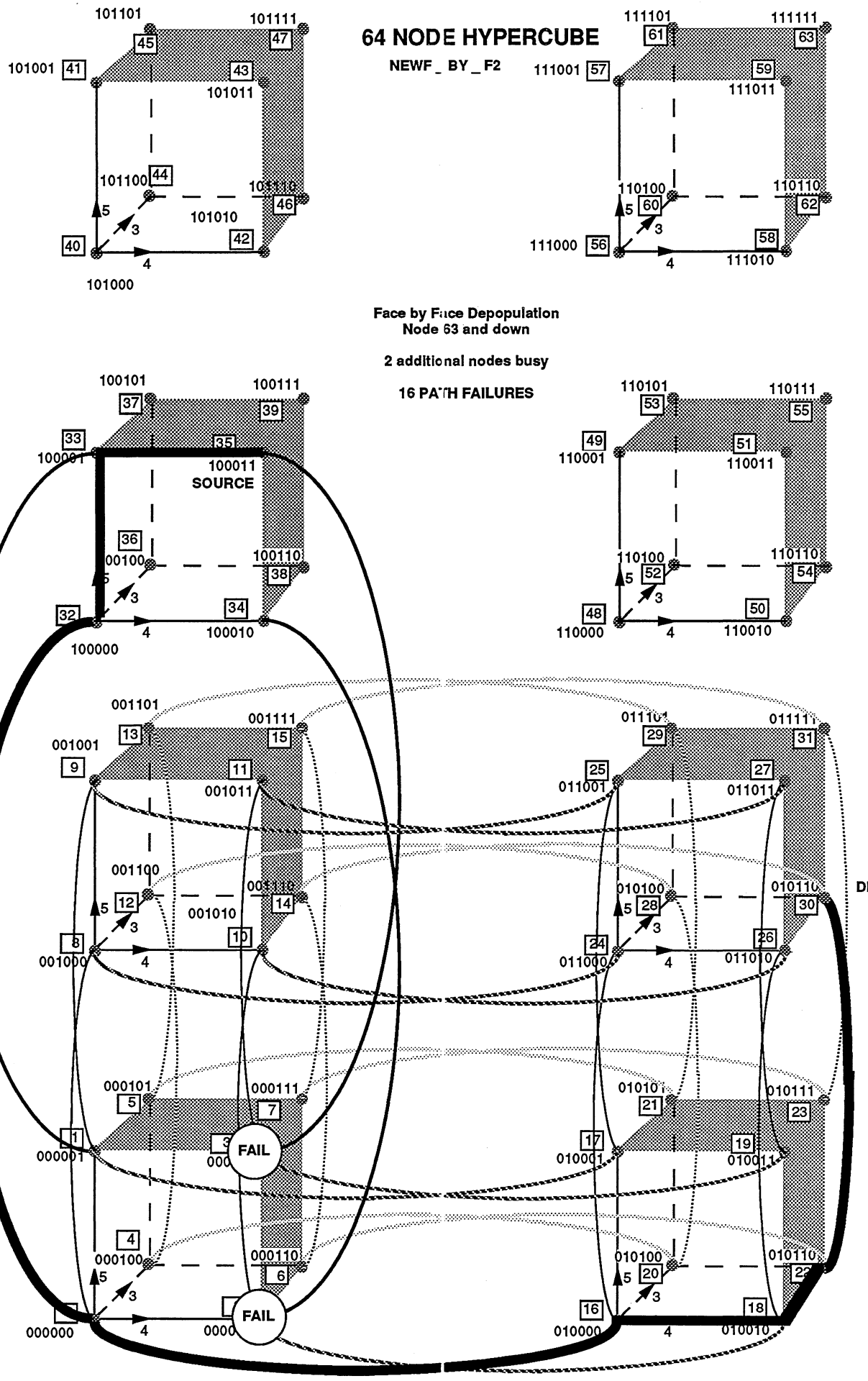
The depopulation of a hypercube face-by-face can survive a single failed node. All source to destination paths can be traversed in 6 or less steps.

The depopulation of a cube face-by-face may not survive two simultaneous failures .

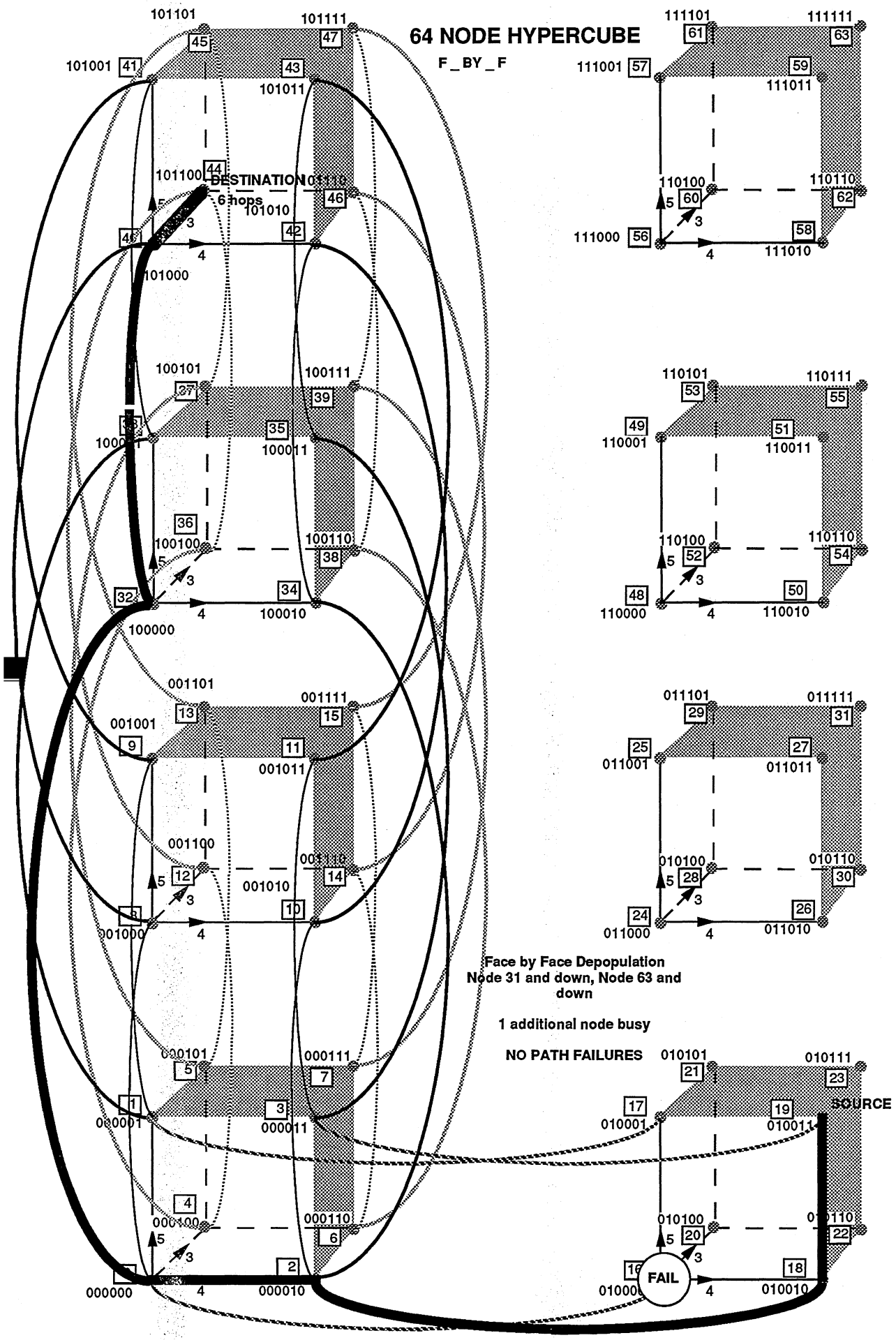
Depopulating a hypercube face-by-face is chosen over a quadrant-by-quadrant depopulation because the latter

1) may not survive a single failed node, and
2) produces more problem paths given two simultaneous failures.

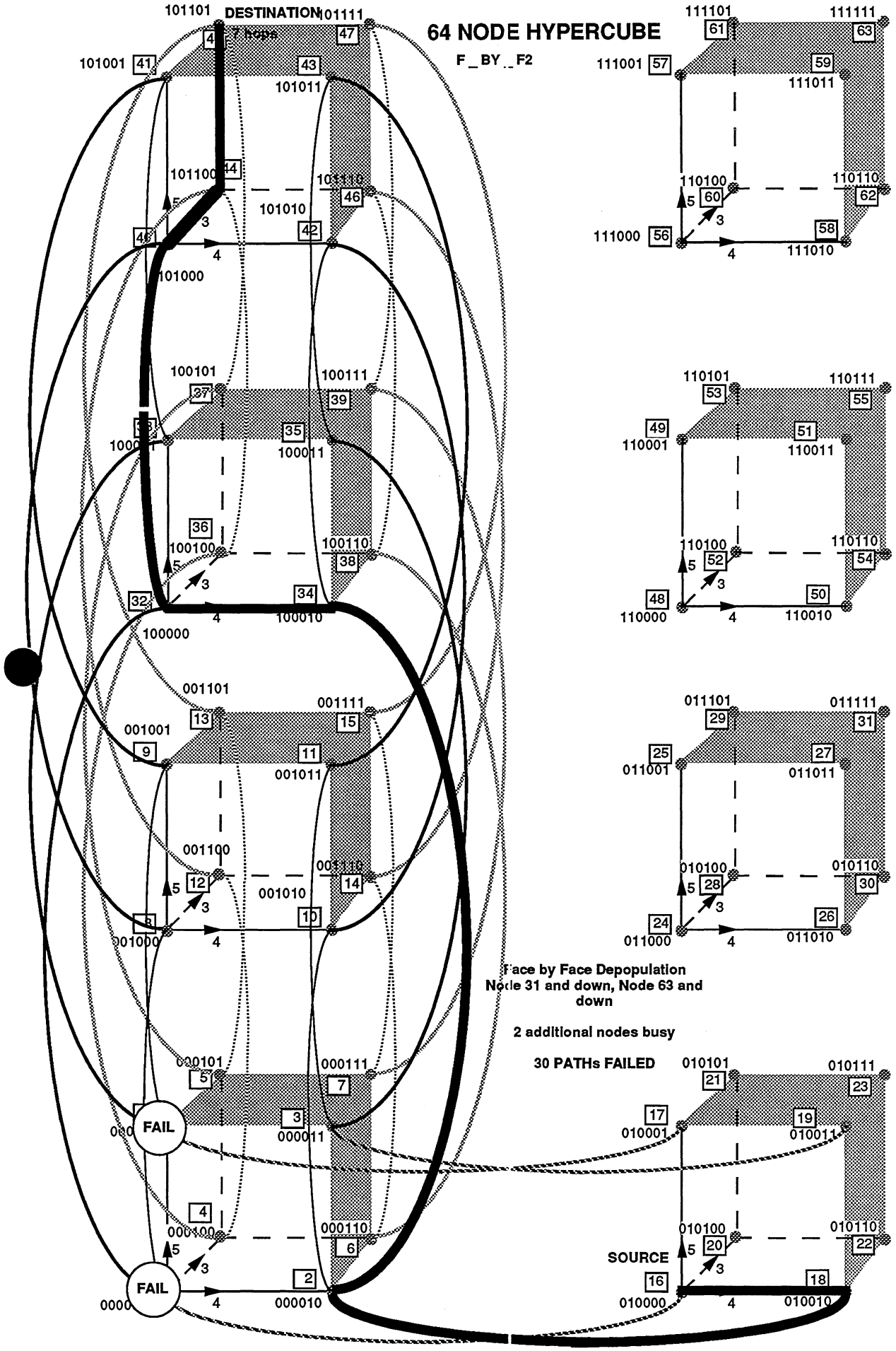

64 NODE HYPERCUBE
NEWF_BY_F
Face by Face Depopulation
Node 63 and down
1 additional node busy
NO PATH FAILURES
SOURCE
FAIL
DESTINATION
4 hops
101101
101111
101001
101011
101100
101110
101010
101000
111101
111111
111001
111011
110100
110110
111000
111010
100101
100111
100001
100011
100100
100110
100000
100010
110101
110111
110001
110011
110100
110110
110000
110010
001101
001111
001001
001011
001100
001110
001010
001000
011101
011111
011001
011011
010100
010110
011000
011010
000101
000111
000001
000011
000100
000110
000000
000010
010101
010111
010001
010011
010100
010110
010000
010010


64 NODE HYPERCUBE
NEWF _ BY _ F2
Face by Face Depopulation
Node 63 and down
2 additional nodes busy
16 PATH FAILURES
SOURCE
DESTINATION
7 hops
FAIL
FAIL
101101
101111
101001
101011
101100
101110
101010
101000
111101
111111
111001
111011
110100
110110
111000
111010
100101
100111
100001
100011
100100
100110
100000
100010
110101
110111
110001
110011
110100
110110
110000
110010
001101
001111
001001
001011
001100
001110
001010
001000
011101
011111
011001
011011
010100
010110
011000
011010
000101
000111
000001
000011
000100
000110
000000
000010
010101
010111
010001
010011
010100
010110
010000
010010

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
0 1 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 561	50/ 76 28 33 29 32	0/ 0	2	2.85740
0 2 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 561	50/ 76 28 34 30 32	0/ 0	2	2.85740
0 33 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 29 32	0/ 0	1	2.77540
0 34 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 30 32	0/ 0	1	2.77540
1 3 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 561	50/ 77 29 35 31 33	0/ 0	2	2.85740
1 32 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 28 33	0/ 0	1	2.77540
1 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 33	0/ 0	1	2.77540
2 3 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 561	50/ 77 30 35 31 34	0/ 0	2	2.85740
2 32 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 28 34	0/ 0	1	2.77540
2 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 34	0/ 0	1	2.77540
3 33 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 29 35	0/ 0	1	2.77540
3 34 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 30 35	0/ 0	1	2.77540


64 NODE HYPERCUBE
F_BY_F
DESTINATION
6 hops
SOURCE
FAIL
101101
101111
101001
101011
101100
101110
101010
101000
100101
100111
100001
100011
100100
100110
100010
100000
001101
001111
001001
001011
001100
001110
001010
001000
000101
000111
000001
000011
000100
000110
000010
000000
111101
111111
111001
111011
110100
110110
111000
111010
110101
110111
110001
110011
110000
110010
011101
011111
011001
011011
010100
010110
011000
011010
010101
010111
010001
010011
010000
010010
Face by Face Depopulation
Node 31 and down, Node 63 and down
1 additional node busy
NO PATH FAILURES

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH


64 NODE HYPERCUBE
F_BY_F2
DESTINATION
101101
7 hops
101111
101001
101011
101100
101110
101010
101000
100101
100111
100011
100100
100110
100010
100000
001101
001111
001001
001011
001100
001110
001010
001000
000101
000111
000011
000100
000110
000010
000000
FAIL
FAIL
111101
111111
111001
111011
110100
110110
111000
111010
110101
110111
110001
110011
110100
110110
110000
110010
011101
011111
011001
011011
010100
010110
011000
011010
Face by Face Depopulation
Node 31 and down, Node 63 and down
2 additional nodes busy
30 PATHs FAILED
010101
010111
010001
010011
010100
010110
SOURCE
010000
010010

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
0 1 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019935389/ 561	36426/ 76 16 15 17 14	0/ 0	2	2.85740
0 2 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019935417/ 561	36476/ 76 16 16 18 14	0/ 0	2	2.85740
0 17 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019935537/ 561	36635/ 42 16 15	0/ 0	1	2.77540
0 18 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019935552/ 561	36663/ 42 16 16	0/ 0	1	2.77540
1 3 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019935723/ 561	36896/ 78 17 17 19 15	0/ 0	2	2.85740
1 16 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019935828/ 561	37036/ 42 17 14	0/ 0	1	2.77540
1 19 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019935850/ 561	37073/ 42 17 17	0/ 0	1	2.77540
0 2 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019935992/ 561	37269/ 76 16 16 18 14	0/ 0	2	2.85740
2 3 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019936035/ 561	37338/ 76 18 17 19 16	0/ 0	2	2.85740
2 16 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019936140/ 561	37482/ 42 18 14	0/ 0	1	2.77540
2 19 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019936162/ 561	37519/ 42 18 17	0/ 0	1	2.77540
1 3 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019936319/ 561	37734/ 78 17 17 19 15	0/ 0	2	2.85740
2 3 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019936347/ 561	37784/ 76 18 17 19 16	0/ 0	2	2.85740
3 17 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019936459/ 561	37933/ 42 19 15	0/ 0	1	2.77540
3 18 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019936474/ 561	37961/ 42 19 16	0/ 0	1	2.77540
1 16 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019937079/ 561	38693/ 42 17 14	0/ 0	1	2.77540
2 16 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019937094/ 561	38721/ 42 18 14	0/ 0	1	2.77540
0 17 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019937117/ 561	38761/ 42 16 15	0/ 0	1	2.77540
3 17 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019937139/ 561	38798/ 42	0/ 0	1	2.77540

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
19 45 0 18 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019937155/ 561	3882?/ 42	0/ 0	1	2.77540
16 46 3 18 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019937177/ 561	38866/ 42	0/ 0	1	2.77540
19 46 1 19 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019937200/ 561	38906/ 42	0/ 0	1	2.77540
17 47 2 19 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	2019937215/ 561	38934/ 42	0/ 0	1	2.77540
18 47 0 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 3	2019940598/ 1	44032/ 4	6/ 6	1	0.00000
1 2 1 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 3	2019940599/ 1	44036/ 4	6/ 6	1	0.00000
0 3 1 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 3	2019940600/ 1	44040/ 4	6/ 6	1	0.00000
0 3					

deleted 7/5/90 because of file size

f_by_f_b.d — depopulate face-by-face
face with highest address followed by
face with highest address less than one
lower power of two

one extra node busy

f_by_f2_b.d — two extra nodes busy

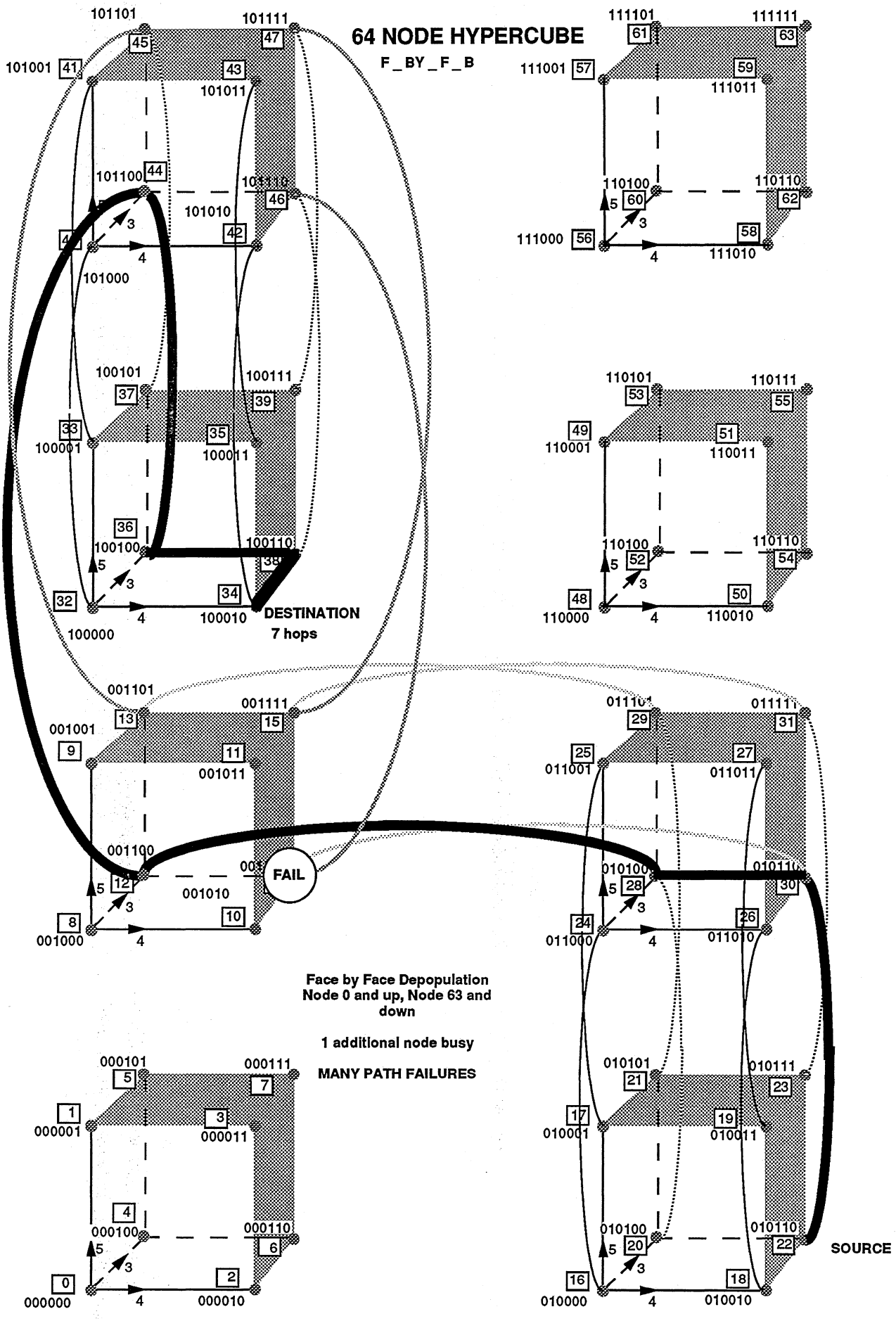
e.g.

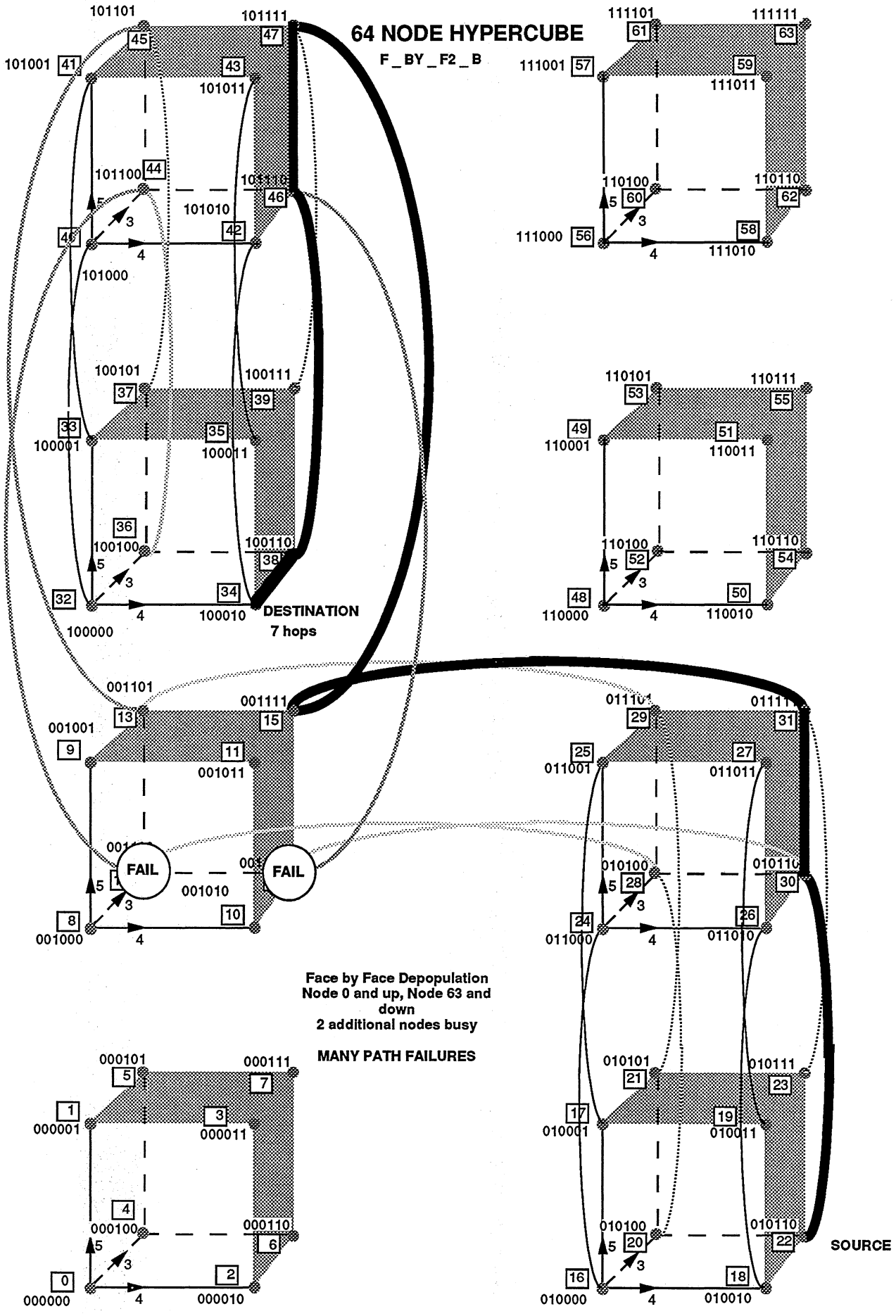
face	addresses busy	
0		
1	60 - 63	
2	60 - 63	28 - 31
3	56 - 63	28 - 31
4	56 - 63	24 - 31
5	52 - 63	24 - 31
6	52 - 63	20 - 31
...		

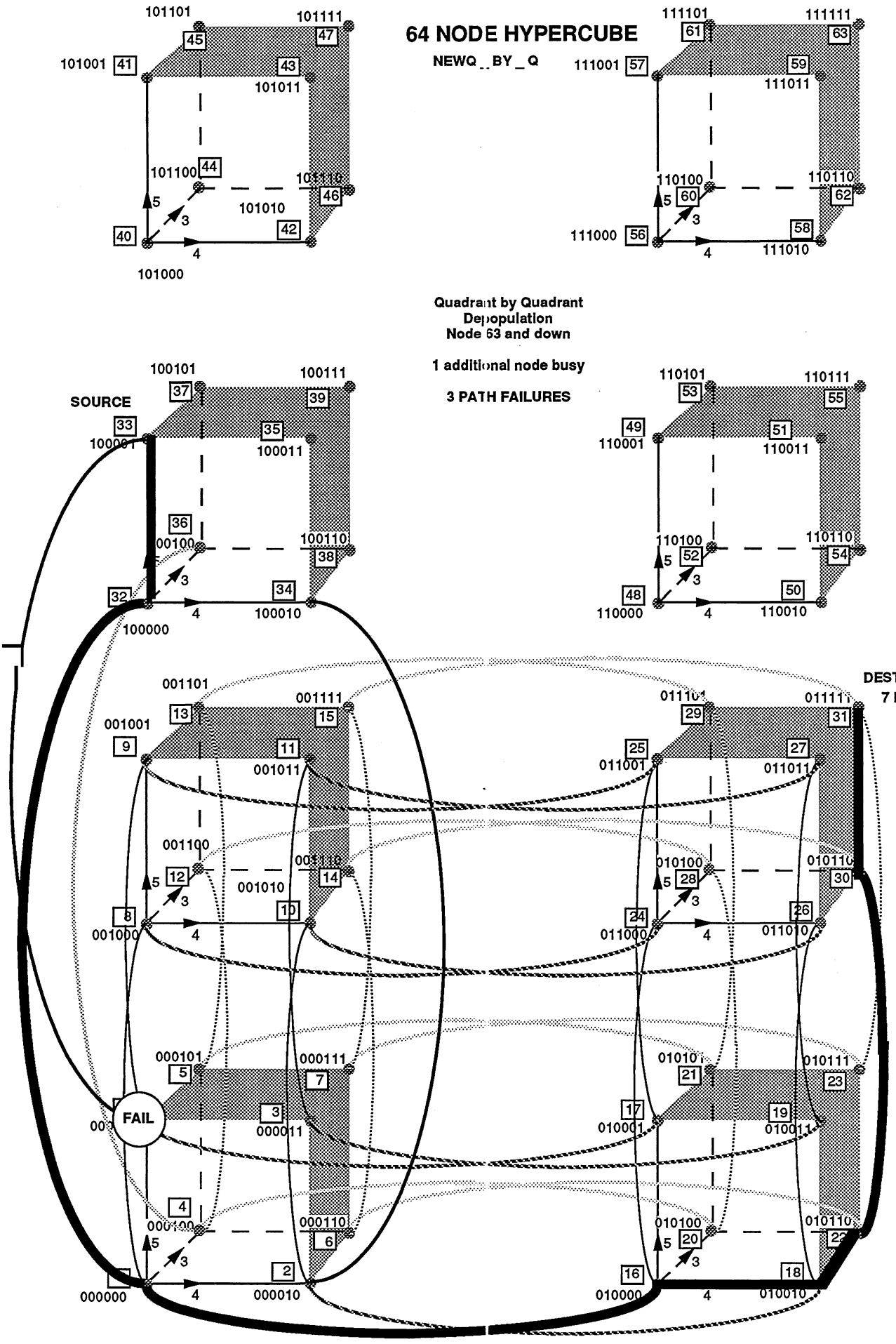
RESULTS:

when depopulated almost to next lower power of two
(36 node cube), the remaining cubes are separated,
linked only by a few nodes

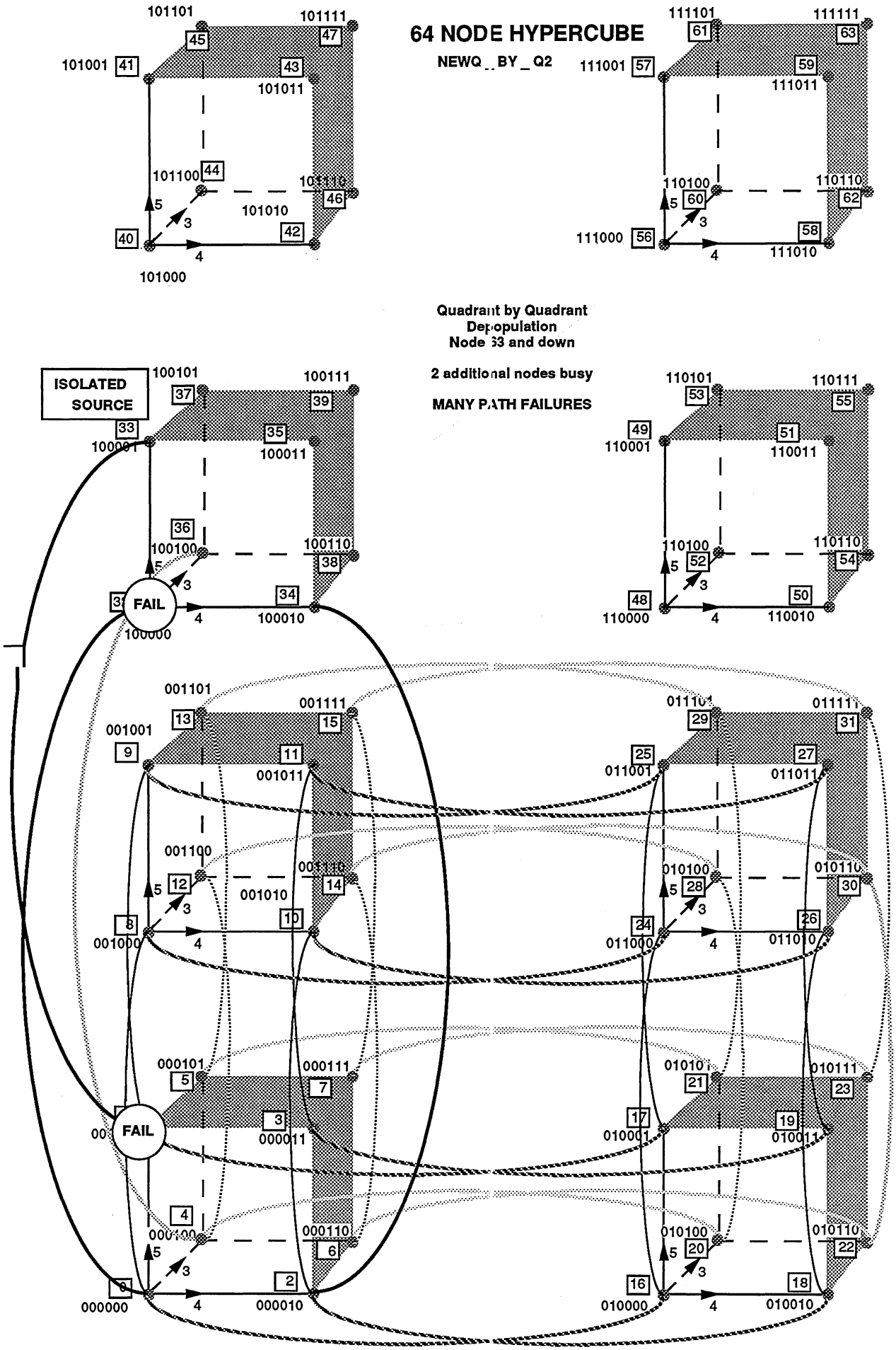
MANY FAILURES WHEN ONLY ONE FACE IS LEFT LINKING
2 SETS OF SUBCUBES


64 NODE HYPERCUBE
F_BY_F_B
101101
101111
101001
101011
101100
101110
101010
101000
111101
111111
111001
111011
110100
110110
111000
111010
100101
100111
100001
100011
100100
100110
100000
100010
DESTINATION
7 hops
110101
110111
110001
110011
110100
110110
110000
110010
001101
001111
001001
001011
001100
001110
001010
001000
FAIL
011101
011111
011001
011011
010100
010110
011000
011010
Face by Face Depopulation
Node 0 and up, Node 63 and down
1 additional node busy
MANY PATH FAILURES
000101
000111
000001
000011
000100
000110
000000
000010
010101
010111
010001
010011
010100
010110
010000
010010
SOURCE


64 NODE HYPERCUBE
F_BY_F2_B
101101
101111
101001
101011
101100
101110
101010
101000
111101
111111
111001
111011
110100
110110
111000
111010
100101
100111
100001
100011
100100
100110
100000
100010
DESTINATION
7 hops
110101
110111
110001
110011
110100
110110
110000
110010
001101
001111
001001
001011
001010
001000
FAIL
FAIL
011101
011111
011001
011011
010100
010110
011000
011010
Face by Face Depopulation
Node 0 and up, Node 63 and down
2 additional nodes busy
MANY PATH FAILURES
000101
000111
000001
000011
000100
000110
000000
000010
010101
010111
010001
010011
010100
010110
010000
010010
SOURCE


64 NODE HYPERCUBE
NEWQ_BY_Q
Quadrant by Quadrant
Depopulation
Node 63 and down
1 additional node busy
3 PATH FAILURES
SOURCE
DESTINATION
7 hops
FAIL
101101
101111
101001
101011
101100
101110
101010
101000
111101
111111
111001
111011
110100
110110
111000
111010
100101
100111
100001
100011
100100
100110
100010
100000
110101
110111
110001
110011
110100
110110
110010
110000
001101
001111
001001
001011
001100
001110
001010
001000
011101
011111
011001
011011
010100
010110
011000
011010
000101
000111
000011
000100
000110
000010
000000
010101
010111
010001
010011
010100
010110
010010
010000

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
1 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 595	28/ 42 31 33	0/ 0	1	2.80840
2 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 595	28/ 42 31 34	0/ 0	1	2.80840
4 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 595	28/ 42 31 36	0/ 0	1	2.80840


64 NODE HYPERCUBE
NEWQ_BY_Q2
Quadrant by Quadrant
Depopulation
Node 33 and down
2 additional nodes busy
MANY PATH FAILURES
ISOLATED
SOURCE
FAIL
FAIL
101101
101111
101001
101011
101100
101110
101010
101000
111101
111111
111001
111011
110100
110110
111000
111010
100101
100111
100001
100011
100100
100110
100010
100000
110101
110111
110001
110011
110100
110110
110000
110010
001101
001111
001001
001011
001100
001110
001010
001000
011101
011111
011001
011011
010100
010110
011000
011010
000101
000111
000011
000100
000110
000010
000000
010101
010111
010001
010011
010100
010110
010000
010010

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
17 33 51 53 54 55 56 57 58 59 60 61 62 63	22/1225	44/ 65 15 49	0/ 0	1	3.02857
17 48 51 53 54 55 56 57 58 59 60 61 62 63	18/1225	38/ 59 30 49	0/ 0	1	3.01388
18 34 51 53 54 55 56 57 58 59 60 61 62 63	22/1225	44/ 65 15 50	0/ 0	1	3.02857
18 48 51 53 54 55 56 57 58 59 60 61 62 63	18/1225	37/ 60 29 50	0/ 0	1	3.01388
20 36 51 53 54 55 56 57 58 59 60 61 62 63	22/1225	44/ 65 15 52	0/ 0	1	3.02857
20 48 51 53 54 55 56 57 58 59 60 61 62 63	18/1225	38/ 60 27 52	0/ 0	1	3.01388
33 48 51 53 54 55 56 57 58 59 60 61 62 63	18/1225	37/ 55 46 49	0/ 0	1	3.01388
34 48 51 53 54 55 56 57 58 59 60 61 62 63	18/1225	37/ 56 45 50	0/ 0	1	3.01388
36 48 51 53 54 55 56 57 58 59 60 61 62 63	18/1225	36/ 54 43 52	0/ 0	1	3.01388
9 33 43 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	18/ 861	36/ 53 23 41	0/ 0	1	2.95238
9 40 43 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	18/ 861	37/ 55 30 41	0/ 0	1	2.95006
10 34 43 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	18/ 861	36/ 53 23 42	0/ 0	1	2.95238
10 40 43 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	18/ 861	36/ 55 29 42	0/ 0	1	2.95006
12 36 43 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	18/ 861	36/ 53 23 44	0/ 0	1	2.95238
12 40 43 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	18/ 861	37/ 56 27 44	0/ 0	1	2.95006
0 1 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	24/ 561	40/ 69 25 33 31 33	8/ 12	2	2.85561
0 2 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	24/ 561	40/ 69 26 34 31 34	8/ 12	2	2.85561
0 4 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	24/ 561	40/ 69 28 36 31 36	8/ 12	2	2.85561
1 2 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	32/ 561	59/ 91 31 33 31 34	6/ 12	2	2.87879
1 3 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	25/ 38	0/ 0	1	2.81462

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
1 4 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	32/ 561	31 33 59/ 92	6/ 12	2	2.87879
1 5 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	31 36 25/ 38	0/ 0	1	2.81462
1 6 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 28/ 42	0/ 0	1	2.82175
1 7 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 27/ 43	0/ 0	1	2.81640
1 8 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 561	31 33 30/ 49	3/ 6	1	2.83066
1 9 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	31 33 25/ 38	0/ 0	1	2.81105
1 10 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 28/ 42	0/ 0	1	2.81818
1 11 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 27/ 43	0/ 0	1	2.81283
1 12 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 28/ 42	0/ 0	1	2.81818
1 13 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 27/ 43	0/ 0	1	2.81283
1 14 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 28/ 42	0/ 0	1	2.81640
1 15 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	31 33 27/ 40	0/ 0	1	2.80749
1 16 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 561	31 33 30/ 49	3/ 6	1	2.83066
1 17 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	31 33 25/ 38	0/ 0	1	2.81105
1 18 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 28/ 42	0/ 0	1	2.81818
1 19 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 27/ 43	0/ 0	1	2.81283
1 20 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 28/ 42	0/ 0	1	2.81818
1 21 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 27/ 43	0/ 0	1	2.81283
1 22 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 28/ 42	0/ 0	1	2.81640
1 23 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	31 33 27/ 40	0/ 0	1	2.80749
1 24 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	31 33 28/ 42	0/ 0	1	2.81462

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
		31 33			
1 25 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43	0/ 0	1	2.80927
		31 33			
1 26 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81283
		31 33			
1 27 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	27/ 40	0/ 0	1	2.80392
		31 33			
1 28 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81283
		31 33			
1 29 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	27/ 40	0/ 0	1	2.80392
		31 33			
1 30 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81105
		31 33			
1 32 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	34/ 561	81/ 82	42/ 42	33	2.53476
		0 33			
		2 33			
		3 33			
		4 33			
		5 33			
		6 33			
		7 33			
		8 33			
		9 33			
		10 33			
		11 33			
		12 33			
		13 33			
		14 33			
		15 33			
		16 33			
		17 33			
		18 33			
		19 33			
		20 33			
		21 33			
		22 33			
		23 33			
		24 33			
		25 33			
		26 33			
		27 33			
		28 33			
		29 33			
		30 33			
		31 33			
		33 34			
		33 36			
1 34 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.77540
		31 33			
1 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.77540
		31 33			
2 3 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	25/ 38	0/ 0	1	2.81462
		31 34			

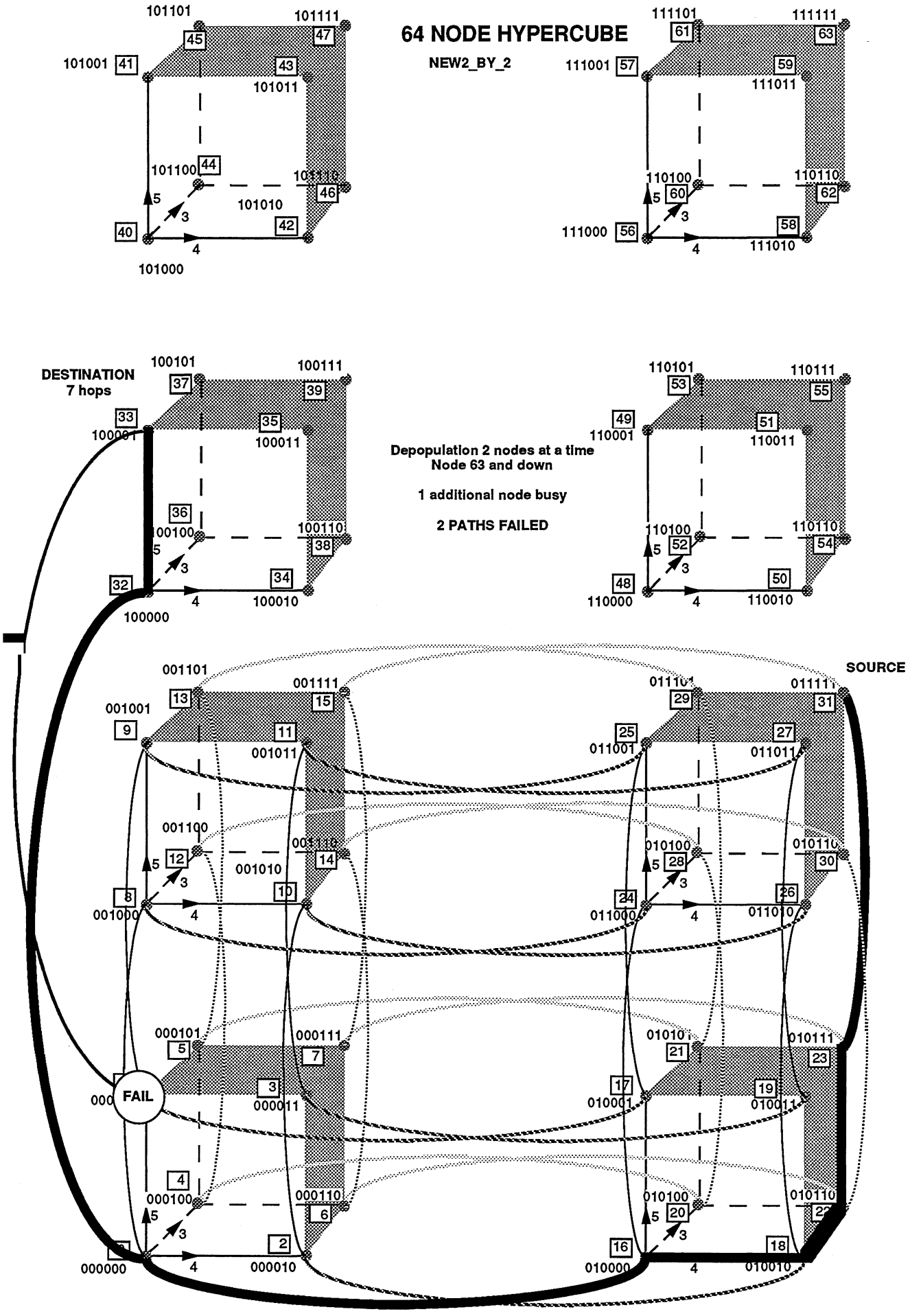
BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
2 4 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	32/ 561	59/ 92 31 34 31 36	6/ 12	2	2.87879
2 5 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 34	0/ 0	1	2.82175
2 6 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	25/ 38 31 34	0/ 0	1	2.81462
2 7 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43 31 34	0/ 0	1	2.81640
2 8 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 561	30/ 49 31 34	3/ 6	1	2.83066
2 9 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 34	0/ 0	1	2.81818
2 10 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	25/ 38 31 34	0/ 0	1	2.81105
2 11 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43 31 34	0/ 0	1	2.81283
2 12 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 34	0/ 0	1	2.81818
2 13 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 34	0/ 0	1	2.81640
2 14 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43 31 34	0/ 0	1	2.81283
2 15 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	27/ 40 31 34	0/ 0	1	2.80749
2 16 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 561	30/ 49 31 34	3/ 6	1	2.83066
2 17 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 34	0/ 0	1	2.81818
2 18 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	25/ 38 31 34	0/ 0	1	2.81105
2 19 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43 31 34	0/ 0	1	2.81283
2 20 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 34	0/ 0	1	2.81818
2 21 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 34	0/ 0	1	2.81640
2 22 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43 31 34	0/ 0	1	2.81283
2 23 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	27/ 40 31 34	0/ 0	1	2.80749
2 24 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81462

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
		31 34			
2 25 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81283
		31 34			
2 26 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43	0/ 0	1	2.80927
		31 34			
2 27 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	27/ 40	0/ 0	1	2.80392
		31 34			
2 28 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81283
		31 34			
2 29 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81105
		31 34			
2 30 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	27/ 40	0/ 0	1	2.80392
		31 34			
2 32 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	34/ 561	81/ 82	42/ 42	33	2.53476
		0 34			
		1 34			
		3 34			
		4 34			
		5 34			
		6 34			
		7 34			
		8 34			
		9 34			
		10 34			
		11 34			
		12 34			
		13 34			
		14 34			
		15 34			
		16 34			
		17 34			
		18 34			
		19 34			
		20 34			
		21 34			
		22 34			
		23 34			
		24 34			
		25 34			
		26 34			
		27 34			
		28 34			
		29 34			
		30 34			
		31 34			
		33 34			
		34 36			
2 33 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.77540
		31 34			
2 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.77540
		31 34			
3 4 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.82175
		31 36			

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
4 5 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	25/ 38	0/ 0	1	2.81462
31 36					
4 6 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	25/ 38	0/ 0	1	2.81462
31 36					
4 7 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43	0/ 0	1	2.81640
31 36					
4 8 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 561	30/ 50	3/ 6	1	2.83066
31 36					
4 9 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81818
31 36					
4 10 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81818
31 36					
4 11 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81640
31 36					
4 12 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	25/ 38	0/ 0	1	2.81105
31 36					
4 13 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43	0/ 0	1	2.81283
31 36					
4 14 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43	0/ 0	1	2.81283
31 36					
4 15 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	27/ 40	0/ 0	1	2.80749
31 36					
4 16 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 561	30/ 50	3/ 6	1	2.83066
31 36					
4 17 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81818
31 36					
4 18 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81818
31 36					
4 19 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81640
31 36					
4 20 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	25/ 38	0/ 0	1	2.81105
31 36					
4 21 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43	0/ 0	1	2.81283
31 36					
4 22 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43	0/ 0	1	2.81283
31 36					
4 23 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	27/ 40	0/ 0	1	2.80749
31 36					
4 24 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81462
31 36					
4 25 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42	0/ 0	1	2.81283
31 36					

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
4 26 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 36	0/ 0	1	2.81283
4 27 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 36	0/ 0	1	2.81105
4 28 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	27/ 43 31 36	0/ 0	1	2.80927
4 29 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	27/ 40 31 36	0/ 0	1	2.80392
4 30 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 561	27/ 40 31 36	0/ 0	1	2.80392
4 32 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	34/ 561	81/ 82 0 36 1 36 2 36 3 36 5 36 6 36 7 36 8 36 9 36 10 36 11 36 12 36 13 36 14 36 15 36 16 36 17 36 18 36 19 36 20 36 21 36 22 36 23 36 24 36 25 36 26 36 27 36 28 36 29 36 30 36 31 36 33 36 34 36	42/ 42	33	2.53476
4 33 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 36	0/ 0	1	2.77540
4 34 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 36	0/ 0	1	2.77540
1 16 19 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 5	18/ 153	53/ 54 0 17 2 17 3 17 4 17 5 17	36/ 36	17	2.07843

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
		6 17			
		7 17			
		8 17			
		9 17			
		10 17			
		11 17			
		12 17			
		13 17			
		14 17			
		15 17			
		17 18			
		17 20			
2 16 19 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 5	18/ 153	53/ 54	36/ 36	17	2.07843
		0 18			
		1 18			
		3 18			
		4 18			
		5 18			
		6 18			
		7 18			
		8 18			
		9 18			
		10 18			
		11 18			
		12 18			
		13 18			
		14 18			
		15 18			
		17 18			
		18 20			
4 16 19 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 5	18/ 153	53/ 54	36/ 36	17	2.07843
		0 20			
		1 20			
		2 20			
		3 20			
		5 20			
		6 20			
		7 20			
		8 20			
		9 20			
		10 20			
		11 20			
		12 20			
		13 20			
		14 20			
		15 20			
		17 20			
		18 20			


64 NODE HYPERCUBE
NEW2_BY_2
DESTINATION
7 hops
Depopulation 2 nodes at a time
Node 63 and down
1 additional node busy
2 PATHS FAILED
SOURCE
FAIL
101101
101111
101001
101011
101100
101110
101010
101000
111101
111111
111001
111011
110100
110110
111000
111010
100101
100111
100001
100011
100100
100110
100000
100010
110101
110111
110001
110011
110100
110110
110000
110010
001101
001111
001001
001011
001100
001110
001010
001000
011101
011111
011001
011011
010100
010110
011000
011010
000101
000111
000001
000011
000100
000110
000000
000010
010101
010111
010001
010011
010100
010110
010000
010010

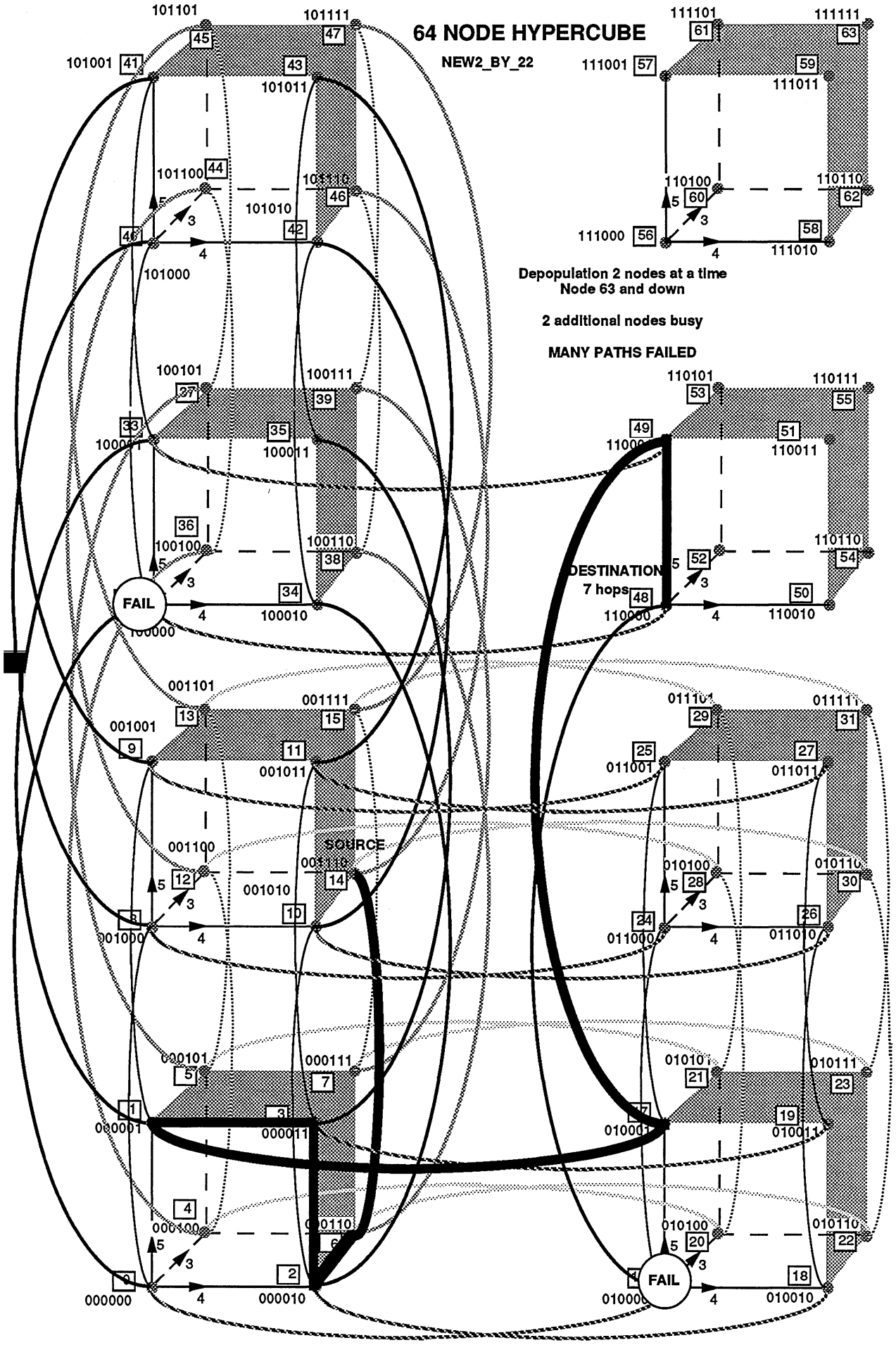
BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
0 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 528	28/ 42	0/ 0	1	2.73674
1 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 528	28/ 42	0/ 0	1	2.73674

30 32 - Shape ... need 7

31 33 5 hops need 7

new2_by_2

f_by_2


64 NODE HYPERCUBE
NEW2_BY_22
Depopulation 2 nodes at a time
Node 63 and down
2 additional nodes busy
MANY PATHS FAILED
SOURCE
DESTINATION
7 hops
FAIL
FAIL
000000
000001
000010
000011
000100
000101
000110
000111
001000
001001
001010
001011
001100
001101
001110
001111
010000
010001
010010
010011
010100
010101
010110
010111
011000
011001
011010
011011
011100
011101
011110
011111
100000
100001
100010
100011
100100
100101
100110
100111
101000
101001
101010
101011
101100
101101
101110
101111
110000
110001
110010
110011
110100
110101
110110
110111
111000
111001
111010
111011
111100
111101
111110
111111

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
16 17 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/1128	54/ 86 30 49 31 48	0/ 0	2	3.01241
16 32 50 51 52 53 54 55 56 57 58 59 60 61 62 63	22/1128	44/ 65 14 48	0/ 0	1	3.00887
16 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/1128	30/ 47 31 48	0/ 0	1	2.98404
17 33 50 51 52 53 54 55 56 57 58 59 60 61 62 63	22/1128	44/ 65 15 49	0/ 0	1	3.00887
17 48 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/1128	30/ 47 30 49	0/ 0	1	2.98404
32 33 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/1128	51/ 78 46 49 47 48	0/ 0	2	3.01241
32 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/1128	28/ 43 47 48	0/ 0	1	2.98404
33 48 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/1128	29/ 43 46 49	0/ 0	1	2.98404
8 9 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 780	51/ 78 30 41 31 40	0/ 0	2	2.96154
8 32 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	18/ 780	36/ 53 22 40	0/ 0	1	2.92564
8 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 780	28/ 43 31 40	0/ 0	1	2.91026
9 33 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	18/ 780	36/ 53 23 41	0/ 0	1	2.92564
9 40 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 780	29/ 43 30 41	0/ 0	1	2.91026
2 3 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 630	50/ 77 30 35 31 34	0/ 0	2	2.90476
2 32 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	16/ 630	31/ 46 28 34	0/ 0	1	2.84127
2 35 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 630	28/ 42 31 34	0/ 0	1	2.83492
3 33 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	16/ 630	31/ 46 29 35	0/ 0	1	2.84127
3 34 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 630	28/ 42 30 35	0/ 0	1	2.83492
4 5 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 630	50/ 80 30 37 31 36	0/ 0	2	2.90476
4 32 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	16/ 630	31/ 46	0/ 0	1	2.84127

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
		26 36			
4 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 630	28,' 42	0/ 0	1	2.83492
		31 36			
5 33 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	16/ 630	31,' 46	0/ 0	1	2.84127
		27 37			
5 36 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 630	28,' 42	0/ 0	1	2.83492
		30 37			
0 1 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 561	50,' 76	0/ 0	2	2.85740
		28 33			
		29 32			
0 2 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 561	50,' 76	0/ 0	2	2.85740
		28 34			
		30 32			
0 33 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28,' 42	0/ 0	1	2.77540
		29 32			
0 34 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28,' 42	0/ 0	1	2.77540
		30 32			
1 3 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 561	50,' 77	0/ 0	2	2.85740
		29 35			
		31 33			
1 32 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28,' 42	0/ 0	1	2.77540
		28 33			
1 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28,' 42	0/ 0	1	2.77540
		31 33			
2 3 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 561	50,' 77	0/ 0	2	2.85740
		30 35			
		31 34			
2 32 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28,' 42	0/ 0	1	2.77540
		28 34			
2 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28,' 42	0/ 0	1	2.77540
		31 34			
3 33 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28,' 42	0/ 0	1	2.77540
		29 35			
3 34 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28,' 42	0/ 0	1	2.77540
		30 35			
0 1 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	60/ 496	132,' 132	48/ 48	60	2.26210
		2 32			
		2 33			
		3 32			
		3 33			
		4 32			
		4 33			
		5 32			
		5 33			
		6 32			
		6 33			
		7 32			

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
		7 :33			
		8 :32			
		8 :33			
		9 :32			
		9 :33			
		10 :32			
		10 :33			
		11 :32			
		11 :33			
		12 :32			
		12 :33			
		13 :32			
		13 :33			
		14 :32			
		14 :33			
		15 :32			
		15 :33			
		16 :32			
		16 :33			
		17 :32			
		17 :33			
		18 :32			
		18 :33			
		19 :32			
		19 :33			
		20 :32			
		20 :33			
		21 :32			
		21 :33			
		22 :32			
		22 :33			
		23 :32			
		23 :33			
		24 :32			
		24 :33			
		25 :32			
		25 :33			
		26 :32			
		26 :33			
		27 :32			
		27 :33			
		28 :32			
		28 :33			
		29 :32			
		29 :33			
		30 :32			
		30 :33			
		31 :32			
		31 :33			
0 2 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	25,' 38	0/ 0	1	2.73992
		30 :32			
0 3 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 496	30,' 49	3/ 6	1	2.75806
		30 :32			
0 4 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	25,' 38	0/ 0	1	2.73992
		30 :32			
0 5 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 496	30,' 49	3/ 6	1	2.75806
		30 :32			
0 6 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	27,' 43	0/ 0	1	2.74194

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
30 32 0 7 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74395
30 32 0 8 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	25/ 38	0/ 0	1	2.73992
30 32 0 9 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 496	30/ 49	3/ 6	1	2.75806
30 32 0 10 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	27/ 43	0/ 0	1	2.74194
30 32 0 11 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74395
30 32 0 12 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	27/ 43	0/ 0	1	2.74194
30 32 0 13 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74395
30 32 0 14 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	27/ 40	0/ 0	1	2.73589
30 32 0 15 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74194
30 32 0 16 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	25/ 38	0/ 0	1	2.73992
30 32 0 17 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 496	30/ 49	3/ 6	1	2.75806
30 32 0 18 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	27/ 43	0/ 0	1	2.74194
30 32 0 19 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74395
30 32 0 20 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	27/ 43	0/ 0	1	2.74194
30 32 0 21 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74395
30 32 0 22 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	27/ 40	0/ 0	1	2.73589
30 32 0 23 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74194
30 32 0 24 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	27/ 43	0/ 0	1	2.74194
30 32 0 25 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74395
30 32 0 26 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	27/ 40	0/ 0	1	2.73589
30 32 0 27 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74194

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
		30 32			
0 28 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	27/ 40	0/ 0	1	2.73589
		30 32			
0 29 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74194
		30 32			
0 31 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.73992
		30 32			
0 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	31/ 496	70/ 70	30/ 30	31	2.41935
		1 32			
		2 32			
		3 32			
		4 32			
		5 32			
		6 32			
		7 32			
		8 32			
		9 32			
		10 32			
		11 32			
		12 32			
		13 32			
		14 32			
		15 32			
		16 32			
		17 32			
		18 32			
		19 32			
		20 32			
		21 32			
		22 32			
		23 32			
		24 32			
		25 32			
		26 32			
		27 32			
		28 32			
		29 32			
		30 32			
		31 32			
1 2 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 496	30/ 49	3/ 6	1	2.75806
		31 33			
1 3 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	25/ 38	0/ 0	1	2.73992
		31 33			
1 4 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 496	30/ 50	3/ 6	1	2.75806
		31 33			
1 5 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	25/ 38	0/ 0	1	2.73992
		31 33			
1 6 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74395
		31 33			
1 7 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	27/ 43	0/ 0	1	2.74194
		31 33			
1 8 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 496	30/ 49	3/ 6	1	2.75806

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
1 9 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	25/ 38	0/ 0	1	2.73992
1 10 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74395
1 11 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	27/ 43	0/ 0	1	2.74194
1 12 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74395
1 13 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	27/ 43	0/ 0	1	2.74194
1 14 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74194
1 15 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	27/ 40	0/ 0	1	2.73589
1 16 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	17/ 496	30/ 49	3/ 6	1	2.75806
1 17 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	25/ 38	0/ 0	1	2.73992
1 18 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74395
1 19 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	27/ 43	0/ 0	1	2.74194
1 20 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74395
1 21 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	27/ 43	0/ 0	1	2.74194
1 22 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74194
1 23 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	27/ 40	0/ 0	1	2.73589
1 24 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74395
1 25 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	27/ 43	0/ 0	1	2.74194
1 26 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74194
1 27 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	27/ 40	0/ 0	1	2.73589
1 28 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.74194
1 29 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	14/ 496	27/ 40	0/ 0	1	2.73589

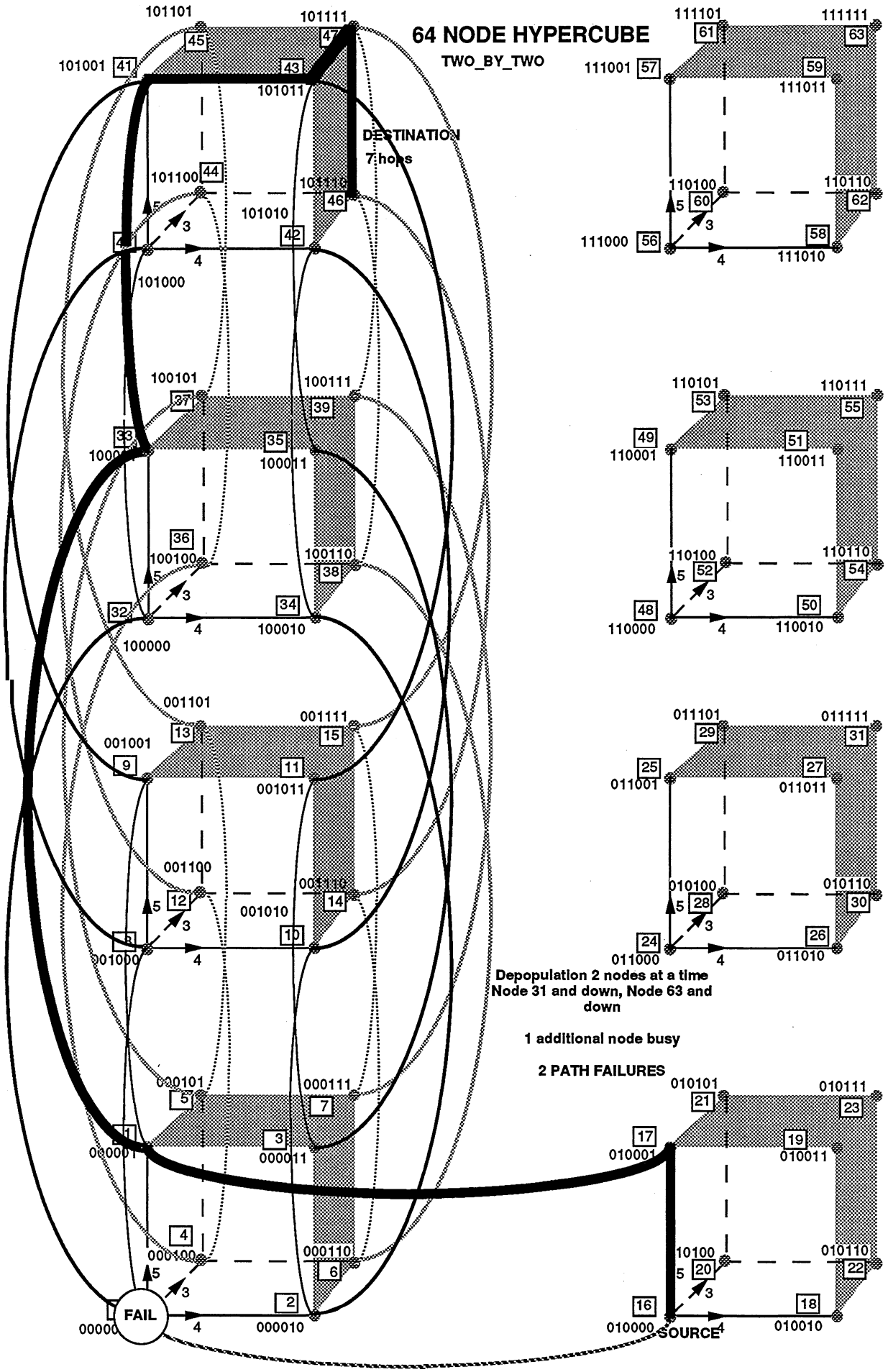
BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
1 30 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42	0/ 0	1	2.73992
1 32 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	31/ 496	70/ 70	30/ 30	31	2.41935
0 1 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 4	28/ 120	76/ 76	36/ 36	28	1.61667

31 33
0 33
2 33
3 33
4 33
5 33
6 33
7 33
8 33
9 33
10 33
11 33
12 33
13 33
14 33
15 33
16 33
17 33
18 33
19 33
20 33
21 33
22 33
23 33
24 33
25 33
26 33
27 33
28 33
29 33
30 33
31 33

2 16
2 17
3 16
3 17
4 16
4 17
5 16
5 17
6 16
6 17
7 16
7 17
8 16
8 17
9 16
9 17
10 16
10 17
11 16
11 17
12 16
12 17
13 16
13 17

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
		14 16			
		14 17			
		15 16			
		15 17			
0 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 4	15/ 120	42/ 42	24/ 24	15	1.86667
		1 16			
		2 16			
		3 16			
		4 16			
		5 16			
		6 16			
		7 16			
		8 16			
		9 16			
		10 16			
		11 16			
		12 16			
		13 16			
		14 16			
		15 16			
1 16 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 4	15/ 120	42/ 42	24/ 24	15	1.86667
		0 17			
		2 17			
		3 17			
		4 17			
		5 17			
		6 17			
		7 17			
		8 17			
		9 17			
		10 17			
		11 17			
		12 17			
		13 17			
		14 17			
		15 17			
0 1 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 4	12/ 28	38/ 38	24/ 24	12	0.92857
		2 8			
		2 9			
		3 8			
		3 9			
		4 8			
		4 9			
		5 8			
		5 9			
		6 8			
		6 9			
		7 8			
		7 9			
0 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 4	7/ 28	23/ 23	18/ 18	7	1.28571
		1 8			
		2 8			
		3 8			
		4 8			
		5 8			
		6 8			
		7 8			
1 8 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 4	7/ 28	23/ 23	18/ 18	7	1.28571

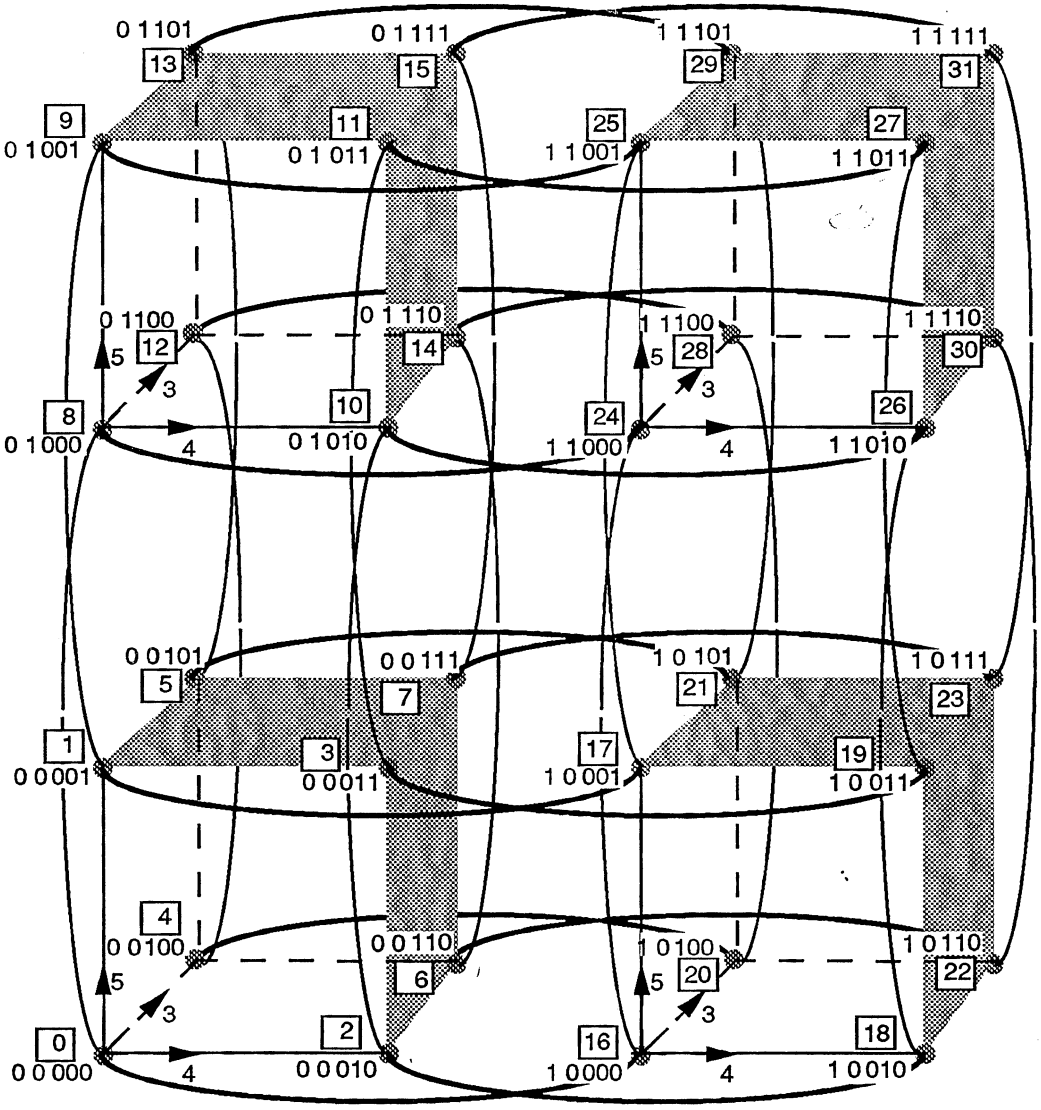
BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
		0 9 2 9 3 9 4 9 5 9 6 9 7 9			
0 1 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 3	4/ 6	14/ 14 2 4 2 5 3 4 3 5	12/ 12	4	0.33333
0 3 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 3	3/ 6	11/ 11 1 2 2 4 2 5	12/ 12	3	0.66667
0 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 3	3/ 6	11/ 11 1 4 2 4 3 4	12/ 12	3	0.66667
1 2 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 3	3/ 6	11/ 11 0 3 3 4 3 5	12/ 12	3	0.66667
1 4 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 3	3/ 6	11/ 11 0 5 2 5 3 5	12/ 12	3	0.66667
0 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 3	1/ 1	4/ 4 1 2	6/ 6	1	0.00000
1 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 3	1/ 1	4/ 4 0 3	6/ 6	1	0.00000


64 NODE HYPERCUBE
TWO_BY_TWO
DESTINATION
7 hops
FAIL
SOURCE
101101
101111
101001
101011
101100
101110
101010
101000
100101
100111
100011
100100
100110
100010
100000
001101
001111
001001
001011
001100
001110
001010
001000
000101
000111
000001
000011
000100
000110
000010
00000
111101
111111
111001
111011
110100
110110
111000
111010
110101
110111
110001
110011
110000
110010
011101
011111
011001
011011
010100
010110
011000
011010
010101
010111
010001
010011
010000
010010
Depopulation 2 nodes at a time
Node 31 and down, Node 63 and down
1 additional node busy
2 PATH FAILURES

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
7 62 63					
7 60 61 62 63					
7 30 31 60 61 62 63					
7 56 57 58 59 60 61 62 63					
7 54 55 56 57 58 59 60 61 62 63					
7 52 53 54 55 56 57 58 59 60 61 62 63					
7 30 31 52 53 54 55 56 57 58 59 60 61 62 63					
7 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63					
7 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63					
7 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63					
7 30 31 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63					
7 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63					
7 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63					
7 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63					
7 30 31 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63					

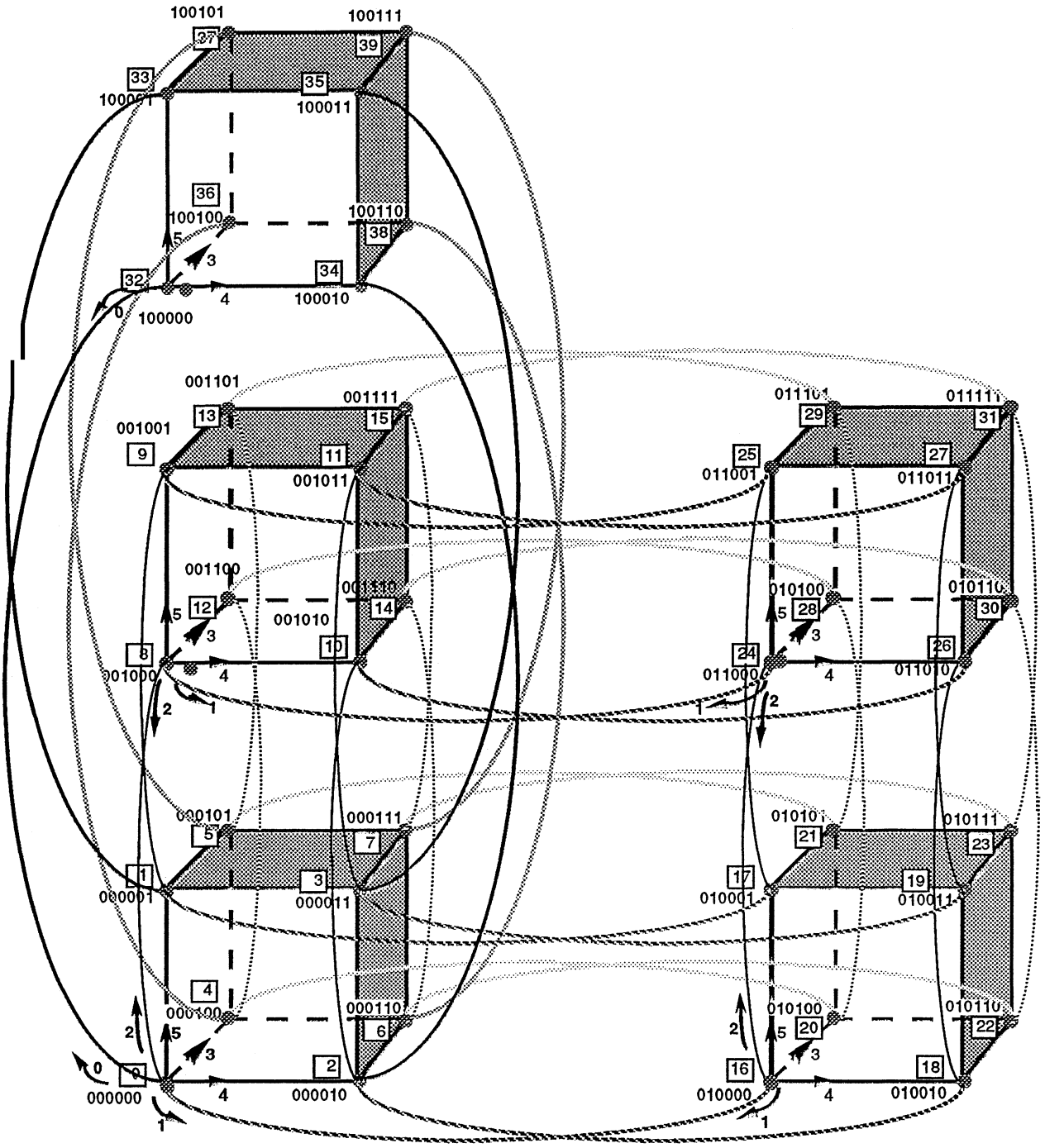
BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
0 18 19 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 528	28/ 42 16 16	0/ 0	1	2.73674
1 18 19 20 21 22 23 24 25 26 27 28 29 30 31 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 528	28/ 42 17 17	0/ 0	1	2.73674

HYPERCUBE ARCHITECTURE

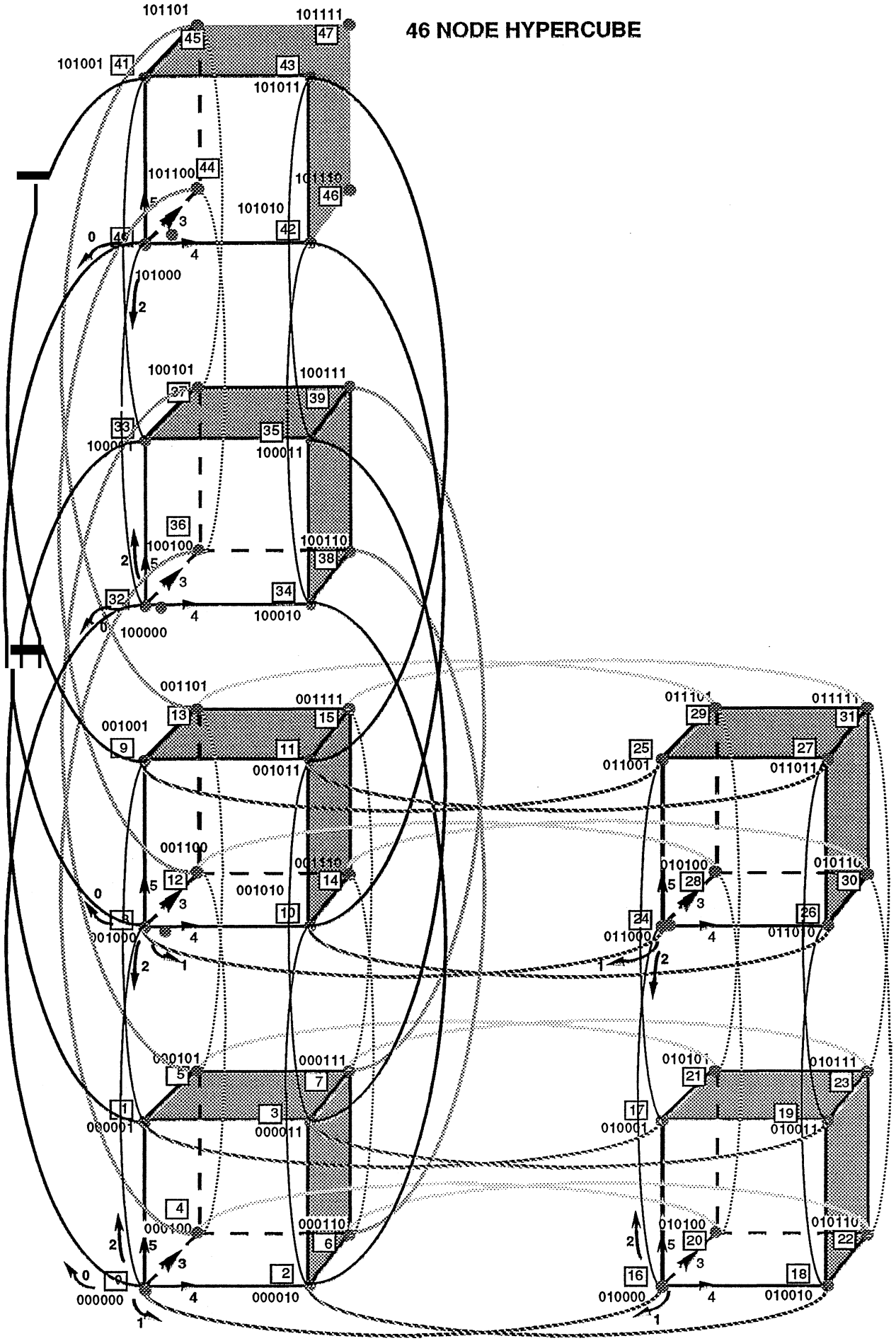


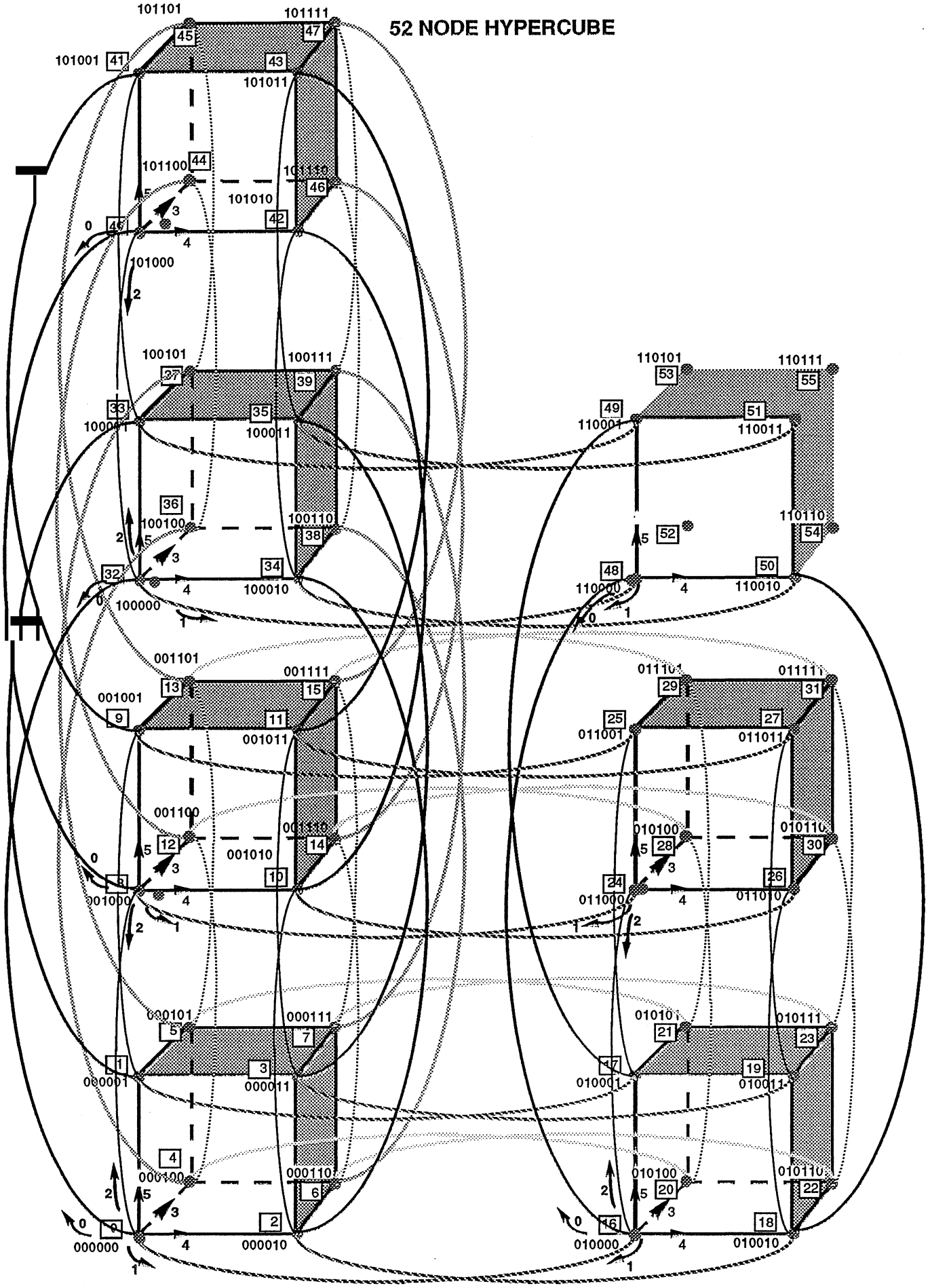
Bit #	Direction	
MSB 1	→	between subcubes
2	↑	
3	+	within subcube
4	→	
LSB 5	↑	

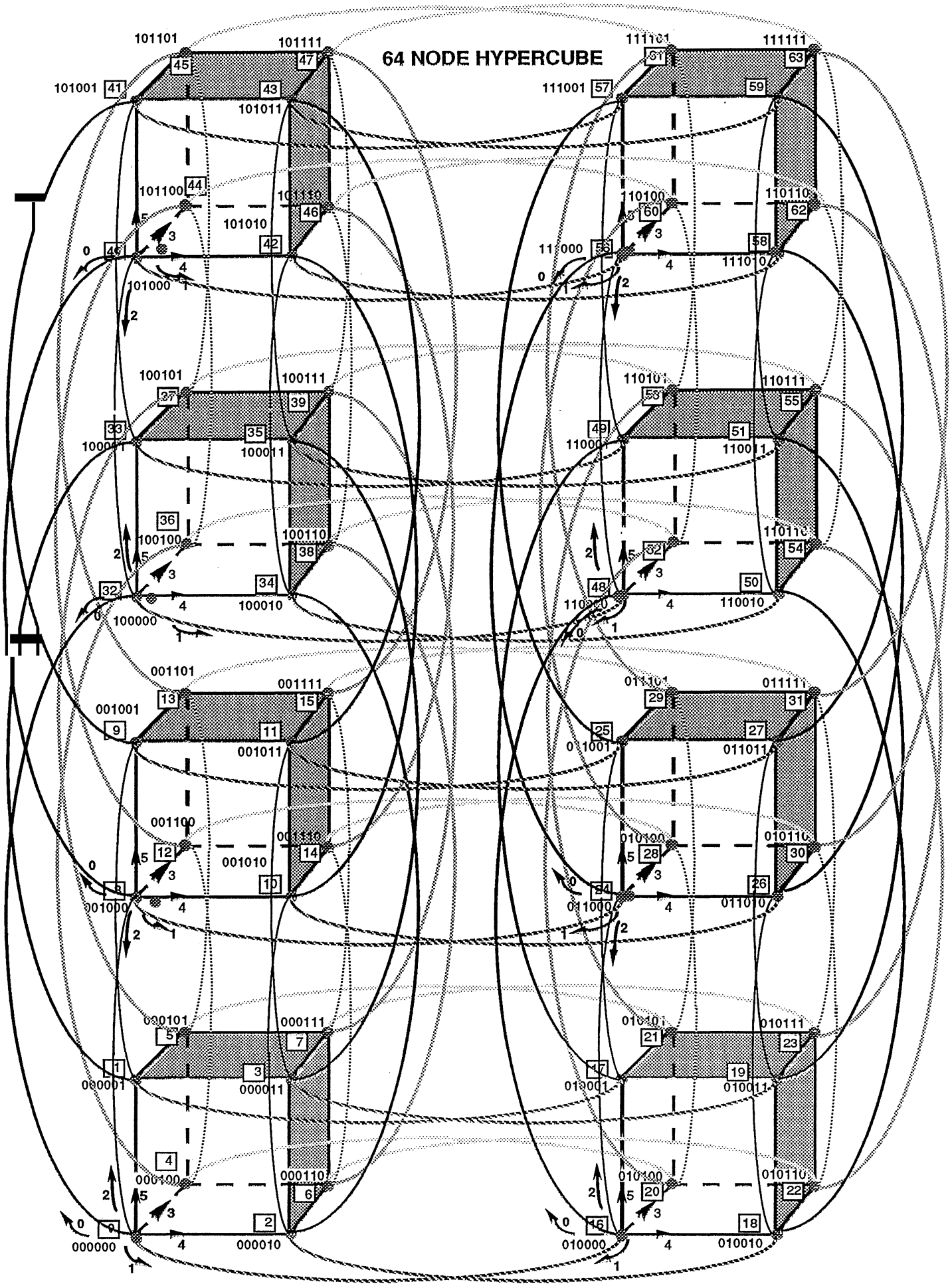
40 NODE HYPERCUBE

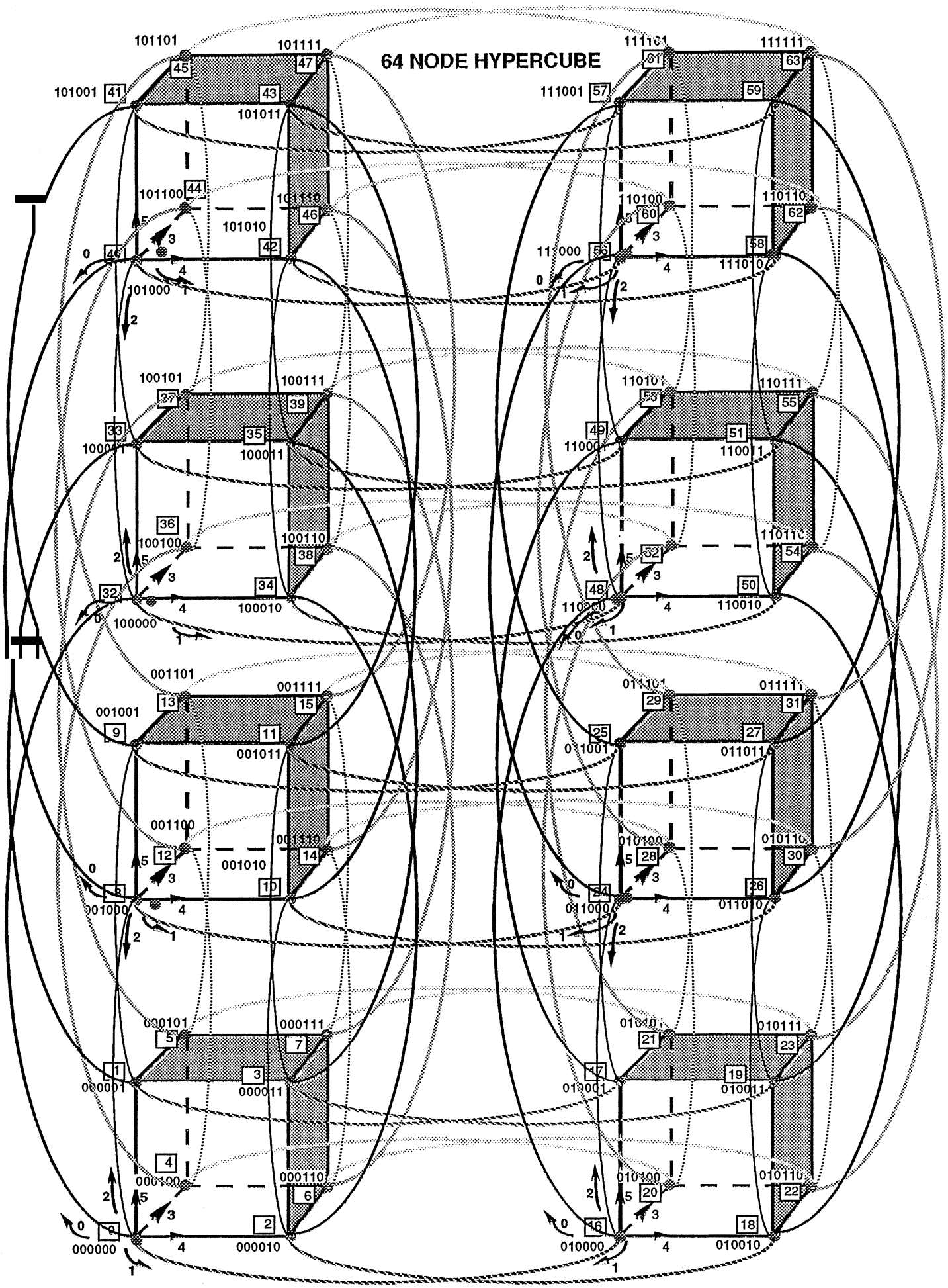


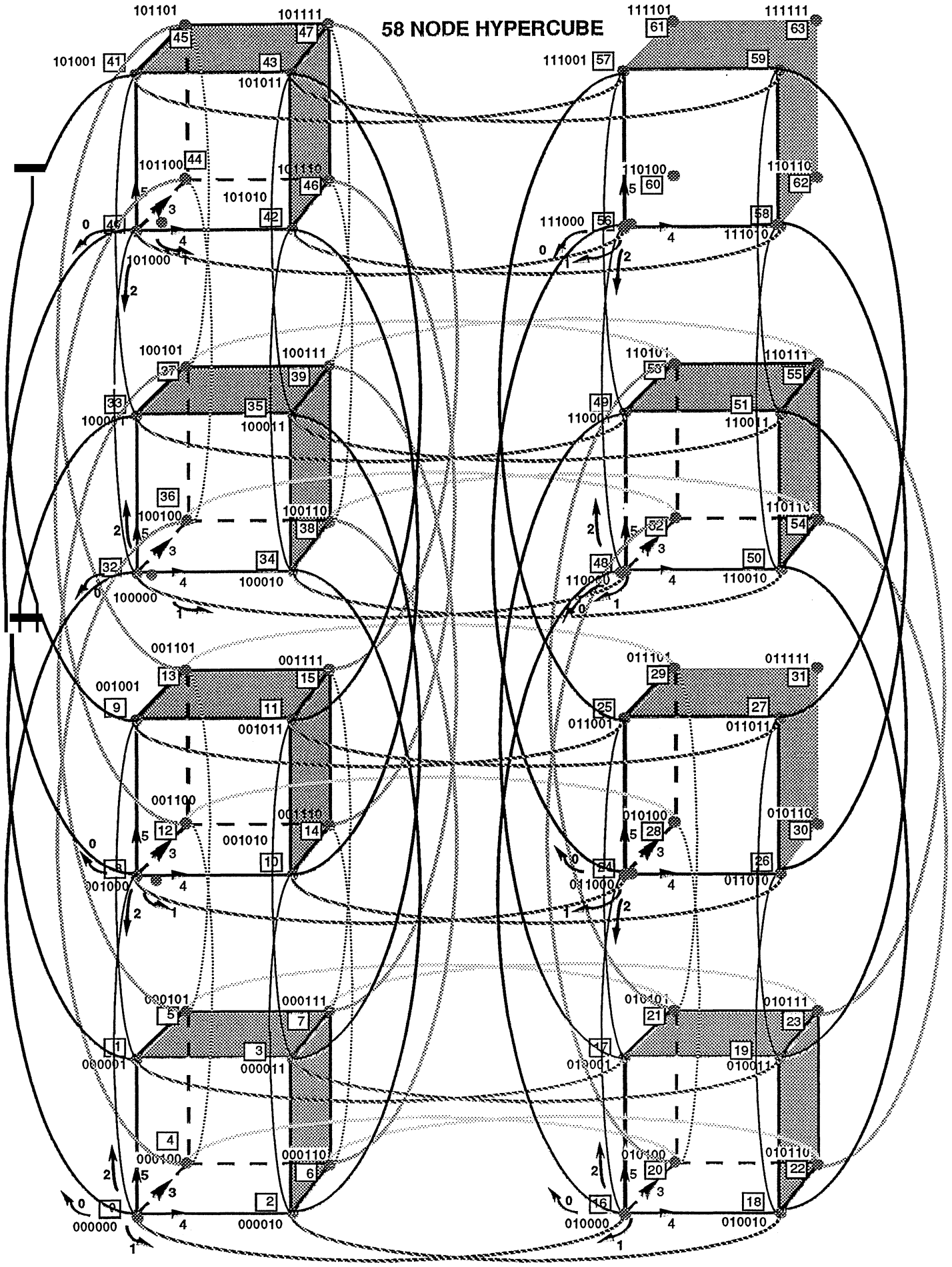
46 NODE HYPERCUBE

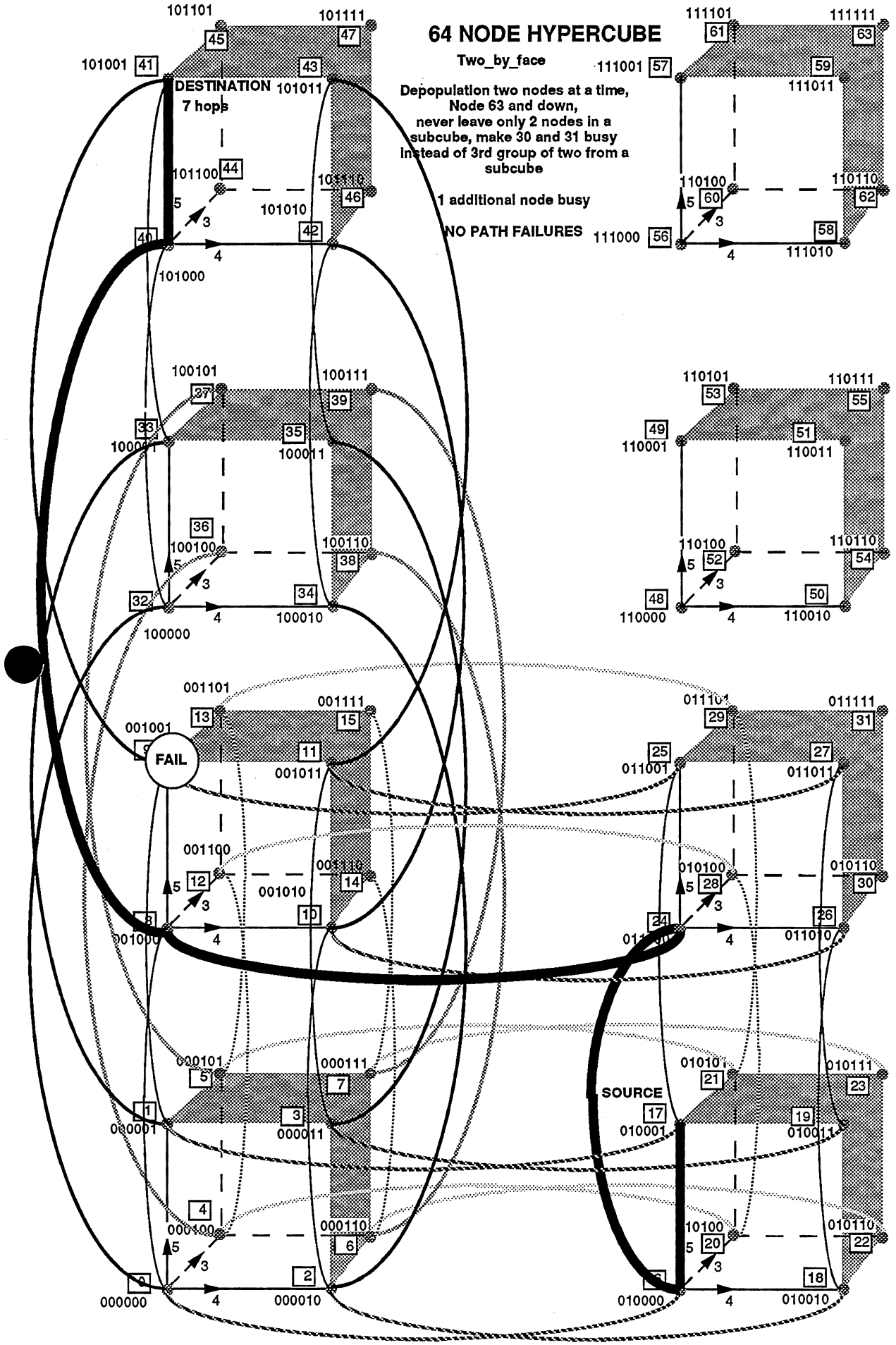



52 NODE HYPERCUBE
101101
101111
101001
101011
101100
101110
101010
101000
100101
100111
100001
100011
100100
100110
100010
100000
110101
110111
110001
110011
110110
110010
110000
001101
001111
001001
001011
001100
001110
001010
001000
011101
011111
011001
011011
010100
010110
011000
011010
000101
000111
000001
000011
000100
000110
000010
000000
010101
010111
010001
010011
010100
010110
010000
010010

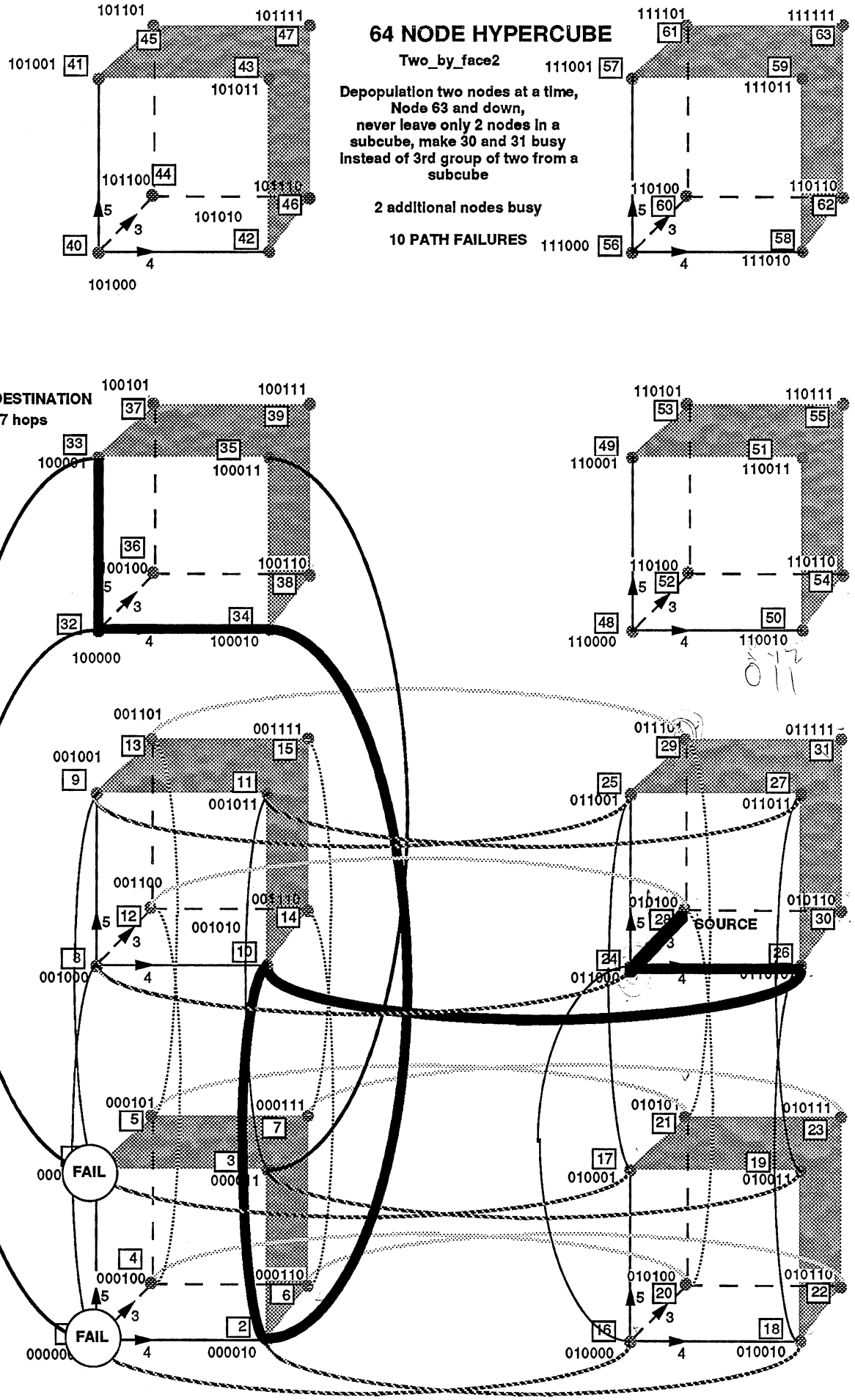

64 NODE HYPERCUBE
101101
101111
101001
101011
101100
101110
101010
101000
100101
100111
100001
100011
100100
100110
100010
100000
001101
001111
001001
001011
001100
001110
001010
001000
000101
000111
000001
000011
000100
000110
000010
000000
111101
111111
111001
111011
111100
111110
111010
111000
110101
110111
110001
110011
110100
110110
110010
110000
011101
011111
011001
011011
011100
011110
011010
011000
010101
010111
010001
010011
010100
010110
010010
010000


64 NODE HYPERCUBE
101101
101111
101001
101011
101100
101110
101010
101000
100101
100111
100001
100011
100100
100110
100010
100000
001101
001111
001001
001011
001100
001110
001010
001000
000101
000111
000001
000011
000100
000110
000010
000000
111101
111111
111001
111011
111100
111110
111010
111000
110101
110111
110001
110011
110100
110110
110010
110000
011101
011111
011001
011011
011100
011110
011010
011000
010101
010111
010001
010011
010100
010110
010010
010000


58 NODE HYPERCUBE
101101
101111
101001
101011
101100
101110
101010
101000
100101
100111
100001
100011
100100
100110
100010
100000
001101
001111
001001
001011
001100
001110
001010
001000
000101
000111
000001
000011
000100
000110
000010
000000
111101
111111
111001
111011
111100
111110
111010
111000
110101
110111
110001
110011
110100
110110
110010
110000
011101
011111
011001
011011
011100
011110
011010
011000
010101
010111
010001
010011
010100
010110
010010
010000


64 NODE HYPERCUBE
Two_by_face
Depopulation two nodes at a time,
Node 63 and down,
never leave only 2 nodes in a
subcube, make 30 and 31 busy
instead of 3rd group of two from a
subcube
1 additional node busy
NO PATH FAILURES
DESTINATION
7 hops
FAIL
SOURCE
101101
101111
101001
101011
101100
101110
101010
101000
111101
111111
111001
111011
110100
110110
111000
111010
100101
100111
100001
100011
100100
100110
100010
100000
110101
110111
110001
110011
110100
110110
110000
110010
001101
001111
001001
001011
001100
001110
001010
001000
011101
011111
011001
011011
010100
010110
011000
011010
000101
000111
000001
000011
000100
000110
000010
000000
010101
010111
010001
010011
010100
010110
010000
010010

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH


64 NODE HYPERCUBE
Two_by_face2
Depopulation two nodes at a time,
Node 63 and down,
never leave only 2 nodes in a
subcube, make 30 and 31 busy
instead of 3rd group of two from a
subcube
2 additional nodes busy
10 PATH FAILURES
DESTINATION
7 hops
SOURCE
FAIL
FAIL
101101
101111
101001
101011
101100
101110
101010
101000
111101
111111
111001
111011
110100
110110
111000
111010
100101
100111
100001
100011
100100
100110
100000
100010
110101
110111
110001
110011
110100
110110
110000
110010
001101
001111
001001
001011
001100
001110
001010
001000
011101
011111
011001
011011
010100
010110
011000
010010
000101
000111
000001
000011
000100
000110
000010
000000
010101
010111
010001
010011
010100
010110
010000
010010

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH
0 1 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 561	50/ 76 28 33 29 32	0/ 0	2	2.85740
1 3 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 561	50/ 77 29 35 31 33	0/ 0	2	2.85740
1 32 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 28 33	0/ 0	1	2.77540
1 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 561	28/ 42 31 33	0/ 0	1	2.77540
0 1 30 31 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	28/ 496	50/ 76 28 33 29 32	0/ 0	2	2.86694
1 3 30 31 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	26/ 496	49/ 74 29 35	0/ 0	1	2.86895
1 30 31 32 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	15/ 496	28/ 42 28 33	0/ 0	1	2.77823

```
Dim2Path    0     0
Perm                        0  1  2  3  4  5  6
source   33 dest  28
111101


Algor1      0
Path1     114     8
Perm                        0  1  6  5  2  3  4

Dim1        1              35  3 19 27 31 30 28
Dim1Path    0     1         4  0  1  2  3  5  4
Perm                        0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                        0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                        0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                        0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                        0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                        0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                        0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                        0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                        0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                        0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                        0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                        0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                        0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                        0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                        0  1  2  3  4  5  6
source   32 dest  29
```

9, 1, 36 →

28 ⇒ 33

```
111101


Algor1      0
Path1     114     8
Perm                       0  1  6  5  2  3  4

Dim1        1             34  2 18 26 30 31 29      32 <- 29
Dim1Path    0     1        4  0  1  2  3  5  4
Perm                       0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                       0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                       0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                       0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6
source  33 dest  29
111100
```

36 nodes, 1,3 busy


```
111110


Algor1      0
Path1     114      8
Perm                          0   1   6   5   2   3   4

Dim1        1                34   2  18  26  30  28  29
Dim1Path    0      1          5   0   1   2   3   4   5
Perm                          0   1   2   3   4   5   6

Dim1       -1
Dim1Path    0      0
Perm                          0   1   2   3   4   5   6

Dim1       -1
Dim1Path    0      0
Perm                          0   1   2   3   4   5   6

Dim1       -1
Dim1Path    0      0
Perm                          0   1   2   3   4   5   6

Dim1       -1
Dim1Path    0      0
Perm                          0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0      0
Perm                          0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0      0
Perm                          0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0      0
Perm                          0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0      0
Perm                          0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0      0
Perm                          0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0      0
Perm                          0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0      0
Perm                          0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0      0
Perm                          0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0      0
Perm                          0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0      0
Perm                          0   1   2   3   4   5   6

  1  3 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
                           2/   3          1/   3          0/   0          0             6.33333
```

29 <-> 35

```
Algor1      0
Path1     114     8
Perm                       0  1  6  5  2  3  4

Dim1        1             32  0 16 24 28 30 31
Dim1Path    0     1        5  0  1  2  3  4  5      33<-->31
Perm                       0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                       0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                       0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                       0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                       0  1  2  3  4  5  6

  1  3 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
                            1/  1          0/     1       0/     0           0         7.00000
  1  3 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
                            2/  3          1/     3       0/     0           0         6.33333
```

36 nodes 1,32 busy

```
BUSY NODES                  FAILURES/     FAILURES/        FAILURES/        COMPLETE          AVG PATH
                          DIRECT PATHS    ADDED DIMS       ADD2 DIMS        FAILURES          LENGTH

 1 32 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
        source   33 dest   28
111101


Algor1       0
Path1        0     0
Perm                        0  1  2  3  4  5  6

Dim1         1              35  3 19 27 31 30 28      28<--->33
Dim1Path     0     1         4  0  1  2  3  5  4
Perm                         0  1  2  3  4  5  6

Dim1        -1
Dim1Path     0     0
Perm                         0  1  2  3  4  5  6

Dim1        -1
Dim1Path     0     0
Perm                         0  1  2  3  4  5  6

Dim1        -1
Dim1Path     0     0
Perm                         0  1  2  3  4  5  6

Dim1        -1
Dim1Path     0     0
Perm                         0  1  2  3  4  5  6

Dim2        -1
Dim2Path     0     0
Perm                         0  1  2  3  4  5  6

Dim2        -1
Dim2Path     0     0
Perm                         0  1  2  3  4  5  6

Dim2        -1
Dim2Path     0     0
Perm                         0  1  2  3  4  5  6

Dim2        -1
Dim2Path     0     0
Perm                         0  1  2  3  4  5  6

Dim2        -1
Dim2Path     0     0
Perm                         0  1  2  3  4  5  6

Dim2        -1
Dim2Path     0     0
Perm                         0  1  2  3  4  5  6

Dim2        -1
Dim2Path     0     0
Perm                         0  1  2  3  4  5  6

Dim2        -1
Dim2Path     0     0
Perm                         0  1  2  3  4  5  6

Dim2        -1
Dim2Path     0     0
Perm                         0  1  2  3  4  5  6

Dim2        -1
```

BUSY NODES | FAILURES/ DIRECT PATHS | FAILURES/ ADDED DIMS | FAILURES/ ADD2 DIMS | COMPLETE FAILURES | AVG PATH LENGTH

1 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
source 33 dest 31
111110

```
Algor1      0
Path1       0     0
Perm                    0  1  2  3  4  5  6        31 ⇐ 33

Dim1        1           32  0 16 24 28 30 31
Dim1Path    0     1     5  0  1  2  3  4  5
Perm                    0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                    0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                    0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                    0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                    0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                    0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                    0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                    0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                    0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                    0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                    0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                    0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                    0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                    0  1  2  3  4  5  6

Dim2       -1
```

30 node 0, 1 busy

```
Dim2Path    0     0
Perm                        0   1   2   3   4   5   6
source  33  dest   28
111101


Algor1      0
Path1     114     8
Perm                        0   1   6   5   2   3   4

Dim1        1               35   3  19  27  26  24  28
Dim1Path    2     2         4   0   1   2   5   4   3
Perm                        0   1   2   3   5   6   4

Dim1       -1
Dim1Path    0     0
Perm                        0   1   2   3   4   5   6

Dim1       -1
Dim1Path    0     0
Perm                        0   1   2   3   4   5   6

Dim1       -1
Dim1Path    0     0
Perm                        0   1   2   3   4   5   6

Dim1       -1
Dim1Path    0     0
Perm                        0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0     0
Perm                        0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0     0
Perm                        0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0     0
Perm                        0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0     0
Perm                        0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0     0
Perm                        0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0     0
Perm                        0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0     0
Perm                        0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0     0
Perm                        0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0     0
Perm                        0   1   2   3   4   5   6

Dim2       -1
Dim2Path    0     0
Perm                        0   1   2   3   4   5   6
source  32  dest   29
```

33↔28

```
111101

Algor1      0
Path1     114     8
Perm                      0  1  6  5  2  3  4

Dim1        1            34  2 18 26 27 25 29      32<-->29
Dim1Path    2     2       4  0  1  2  5  4  3
Perm                      0  1  2  3  5  6  4

Dim1       -1
Dim1Path    0     0
Perm                      0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                      0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                      0  1  2  3  4  5  6

Dim1       -1
Dim1Path    0     0
Perm                      0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                      0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                      0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                      0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                      0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                      0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                      0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                      0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                      0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                      0  1  2  3  4  5  6

Dim2       -1
Dim2Path    0     0
Perm                      0  1  2  3  4  5  6
source  33 dest  29
111100
```

30 node 1,3 busy

BUSY NODES	FAILURES/ DIRECT PATHS	FAILURES/ ADDED DIMS	FAILURES/ ADD2 DIMS	COMPLETE FAILURES	AVG PATH LENGTH

```
  1  3 30 31 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
        source   35 dest   29
 111110


 Algor1      0
 Path1     114     8
 Perm                      0  1  6  5  2  3  4

 Dim1        1            34  2 18 26 24 25 29
 Dim1Path    2     2       5  0  1  2  4  5  3
 Perm                      0  1  2  3  5  6  4

 Dim1       -1
 Dim1Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim1       -1
 Dim1Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim1       -1
 Dim1Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim1       -1
 Dim1Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim2       -1
 Dim2Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim2       -1
 Dim2Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim2       -1
 Dim2Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim2       -1
 Dim2Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim2       -1
 Dim2Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim2       -1
 Dim2Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim2       -1
 Dim2Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim2       -1
 Dim2Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim2       -1
 Dim2Path    0     0
 Perm                      0  1  2  3  4  5  6

 Dim2       -1
```

30 node 1, 32 busy


```
BUSY NODES                    FAILURES/     FAILURES/       FAILURES/       COMPLETE        AVG PATH
                             DIRECT PATHS   ADDED DIMS      ADD2 DIMS       FAILURES        LENGTH

  1 30 31 32 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
        source  33 dest  28
 111101


 Algor1     0
 Path1      0     0
 Perm                     0  1  2  3  4  5  6

 Dim1       1             35  3 19 27 26 24 28        33 <-> 28
 Dim1Path   2     2        4  0  1  2  5  4  3
 Perm                      0  1  2  3  5  6  4

 Dim1      -1
 Dim1Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim1      -1
 Dim1Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim1      -1
 Dim1Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim1      -1
 Dim1Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim2      -1
 Dim2Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim2      -1
 Dim2Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim2      -1
 Dim2Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim2      -1
 Dim2Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim2      -1
 Dim2Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim2      -1
 Dim2Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim2      -1
 Dim2Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim2      -1
 Dim2Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim2      -1
 Dim2Path   0     0
 Perm                      0  1  2  3  4  5  6

 Dim2      -1
```

```
LeftToRight
=RUN(Initialize)
•
=FOR("row",FirstRowofList,LastRowofList)                    • for each source - destination

bitsequence="12345"                                         • process bits left to right
bit1Column=GET.CELL(3,lColumn1ofSource)                     • 1st column of parsed source

=MESSAGE(TRUE,"Processing Left to Right Algorithm for Path  • status
•
=RUN(SetDestinationBits)                                    • name DestinationBit1 thru
                                                            • DestinationBit5
•
=SELECT("RC1")                                              • look in first column for "busy"
=IF(GET.CELL(5)="BUSY",GOTO(NextRow))                       • if cell is blank (<0), skip this row
•
=FOR("bitcounter",1,5)                                      • for each bit

• LoopHereIfFailed                                          • failed attempt at step
=RUN(CopyAndPasteStep)
BitToChange=LEFT(bitsequence,1)                             • strip # of bit to change from
bitsequence=RIGHT(bitsequence,5-bitcounter)                 • bitsequence

=SELECT("RC"&bit1Column+BitToChange-1)                      • replace appropriate source bit with destination bit
=FORMULA(GET.NAME("destinationbit"&BitToChange))

=RUN(DetermineBinaryWord)                                   • put 5 cells together into BinaryWord
=RUN(ToDecimal)                                             •.convert BinaryWord to Decimal

=SELECT("RC"&bit1Column+5)                                  • to put Decimal in cell after 5th bit


=IF(INDEX(lbusytable,Decimal+1,2),,GOTO(NotBusy))           • look up decimal, if not busy, skip to NotBusy
FailCount=FailCount+1                                       • count # of attempts at this step

=IF(FailCount>5,GOTO(Failed))                               • 5 attempts means whole row fails

bitsequence=bitsequence&BitToChange
bit1Column=bit1Column-7
=GOTO(LoopHereIfFailed)                                     • go back and try this step again

• Failed
=FORMULA("*")                                               • put a * in the decimal column
FailCount=0                                                 • done with this row
bit1Column=bit1Column-7
=RUN(CopyAndPasteStep)
bitcounter=5                                                • will exit this loop for this row
=GOTO(NextBit)

• NotBusy
FailCount=0
=FORMULA(Decimal)                                           • put decimal equivalent in 6th column

• NextBit
=NEXT()                                                     • next bitcounter

• NextRow
=NEXT()                                                     • next row (source - destination)


=RETURN()
```

SUBROUTINES In LeftToRight

```
CopyAndPasteStep                                            • copy step (5 columns) given bit1Column
                                                            • paste 7 columns over
=SELECT("RC"&bit1Column&":RC"&bit1Column+4)                 • 1st time - select source
=COPY()                                                     • other times - select step
bit1Column=bit1Column+7                                     • calc new start bit for next step
=SELECT("RC"&bit1Column&":RC"&bit1Column+4)                 • copy to next step
=PASTE()

=RETURN()


Initialize
=WORKSPACE(FALSE,,TRUE,TRUE,TRUE,TRUE,"/",FALSE,TRUE,3)
=ECHO(FALSE)                                                • set workspace
FirstRowofList=3
LastRowofList=34
Failcount=0                                                 • count # of tries at step
=ACTIVATE("hypercube worksheet")

=SELECT(lStepArea)                                          • clear cells to calculate
=CLEAR(3)

=RETURN()

SetDestinationBits
=FOR("counter",1,5)
                                                            • set variable "destinationbit1"
=SELECT("R"&row&"C"&counter+5)                              • thru "destinationbit5"
=SET.NAME("destinationbit"&counter,GET.CELL(5))

=NEXT()                                                     •
=RETURN()


DetermineBinaryWord                                         • given bit1column
BinaryWord=""
=FOR("counter2",0,4)                                        • regroup 5 bits into one word
=SELECT("RC"&bit1Column+counter2)
BinaryWord=BinaryWord&GET.CELL(5)
=NEXT()                                                     • next bit in 5 bit word

=RETURN()
```

DestinationBinaryNDecimal
=MESSAGE(TRUE,"Destination "&Dest)
decimal=Dest
=RUN(ToBinary)

=SELECT(IDestinationList)
=FORMULA("""&BinaryWord&""")
=SELECT("RC")
=COPY()
=SELECT(IDestinationList)
=PASTE()

=SELECT(IDestinationDecimal)
=FORMULA(Dest)
=SELECT("RC")
=COPY()
=SELECT(IDestinationDecimal)
=PASTE()

=RETURN()

• get the binary equivalent
• of Dest

• put the binary # in the 1st
• cell of destination list

• copy binary # to
• destination list

• copy decimal number
• to destination decimal
• column

ToBinary

BinaryWord=""

=FOR("position",length,1,-1)

test=2^(position-1)
=IF(Decimal<(test),GOTO(DoNotSetBit))

• SetBit
=SET.NAME("bit",1)
decimal=Decimal-test
=GOTO(AddBitToWord)

• DoNotSetBit
=SET.NAME("bit",0)

• AddBitToWord
BinaryWord=BinaryWord&bit

=NEXT()
• end of for loop

=RETURN()

decimal=20
length=5
=RUN(ToBinary)
=MESSAGE(TRUE,BinaryWord)
=RETURN()

ToDecimal
length=LEN(BinaryWord)
decimal=0

• Loop

=FOR("position",1,length)

bit=RIGHT(BinaryWord,1)
BinaryWord=LEFT(BinaryWord,5-position)
test=2^(position-1)
bitvalue=bit*test
decimal=Decimal+bitvalue

=NEXT()
• end of for loop

=RETURN()

BinaryWord="11010"

=RUN(ToDecimal)
=MESSAGE(TRUE,Decimal)
=RETURN()

```
CreateBinaryNumbers                          • macro to fill in source list (C1)
=ECHO(FALSE)                                 • with binary numbers

=ACTIVATE("hypercube worksheet")
length=5                                     • length = number of bits
=SELECT(!SourceList)                         • in the binary number
=FOR("row",1,32)                             • will create numbers 0 thru
decimal=row-1                                • 31
=MESSAGE(TRUE,"Making Binary List "&Decimal) • get the decimal number
=SELECT("RC2")                               • decimal number in column 2
=FORMULA(Decimal)

=SELECT("RC1")                               • will start list 2 rows down
=RUN(ToBinary)                               • convert decimal to binary
=FORMULA("""&BinaryWord&""")                 • put the binary # in this cell
=SELECT("R[1]C")                             • (with quotes around it)
=NEXT()                                      • next number
=RETURN()
```

```
RemoveBusySource
=ACTIVATE("hypercube worksheet")

=SELECT(!SourceDecimal)                      • select decimal list of source
=FOR("Source",0,31)                          • for each source address

Decimal=GET.CELL(5)                          • contents of current cell
=IF(INDEX(!busytable,Decimal+1,2),,GOTO(NextSource)) • if table says source decimal
                                             • is busy, clear that source
                                             • row
=MESSAGE(TRUE,"Removing Busy Source for Node "&Decimal)
=SELECT("RC1:RC53")                          • clear row
=CLEAR(3)
=SELECT("RC1")
=FORMULA("BUSY")
=SELECT("RC2")                               • back to decimal column

• NextSource
=SELECT("R[1]C")                             • get next row
=NEXT()                                      • next source

=RETURN()
```

```
Algorithm1
=ECHO(TRUE)


=ACTIVATE("hypercube printsheet")
=SELECT(!Print_Area)
=CLEAR(3)                                   print busy table, to do

                                            =ACTIVATE("hypercube worksheet")
                                            =SELECT(!busytable)
                                            =COPY()
                                            =ACTIVATE("hypercube printsheet")
                                            =SELECT(!Print_Area)
                                            =PASTE()

=SELECT("R1C5")
=FORMULA("Left to Right Algorithm, Failed Bits at End")   • put the algorithm in the print title

                                            =PRINT(1,,,1,FALSE,FALSE,1,FALSE,1)


page_toggle=1

=RUN(CreateBinaryNumbers)                   • create a list of source
=RUN(CopySourceList)                        • addresses
=RUN(RemoveBusySource)
• DestinationLoop

=ACTIVATE("hypercube worksheet")
=FOR("Dest",31,0,-1)                        • for each destination
=IF(INDEX(!busytable,Dest+1,2),GOTO(nextDestination))   • skip this destination if
                                            • busy

=RUN(DestinationBinaryNDecimal)             • make list of binary and decimals
                                            • for destination

=RUN(CopyDestinationList)                   • copy and parse destination

=RUN(LeftToRight)                           • determine paths for
                                            • left to right algorithm
=SELECT(!Page)                              • copy the page from
=COPY()                                     • the worksheet
=ACTIVATE("hypercube printsheet")           • (2 worksheet pages per print page)

=IF(page_toggle=1,GOTO(Page1))
• Page2                                     • paste the 2nd worksheet page
=SELECT(!PastePage2)                        • paste the page in the
=PASTE()                                    • print sheet
=MESSAGE(TRUE,"Printing page")
=PRINT(1,,,1,FALSE,FALSE,1,FALSE,1)         • print the page
page_toggle=1

=GOTO(nextDestination)
• Page1                                     • paste the 1st worksheet page
=SELECT(!PastePage1)
=PASTE()

page_toggle=2

• nextDestination
=ACTIVATE("hypercube worksheet")            • go back to the worksheet
=NEXT()                                     • next destination

=IF(page_toggle=1,GOTO(EndofAlgorithm))     • at end, if only 1/2 page left,
                                            • clear bottom 1/2 page
=ACTIVATE("hypercube printsheet")           • print
=SELECT(!PastePage2)
=CLEAR(3)
=MESSAGE(TRUE,"Printing page")
=PRINT(1,,,1,FALSE,FALSE,1,FALSE,1)

• EndofAlgorithm
=RETURN()
```

```
CopyLists

CopySourceList
=MESSAGE(TRUE,"Copying and Parsing Source List")
=SELECT(!SourceList)                        • copy source column(binary)
=COPY()                                     • to another column
=SELECT(!SourceToParse)
=PASTE()

=PARSE("""[[ ][ ][ ][ ][ ]""")              • parse into separate columns
                                            • for each bit

=SELECT(!SourceDecimal)
=COPY()
=SELECT("R3C18")
=PASTE()
=RETURN()

CopyDestinationList
=MESSAGE(TRUE,"Copying and Parsing Destination List")
=SELECT(!DestinationList)                   • do the same for the
=COPY()                                     • destination column
=SELECT(!DestinationToParse)
=PASTE()

=PARSE("""[[ ][ ][ ][ ][ ]""")

=SELECT(!DestinationDecimal)
=COPY()
=SELECT("R3C11")
=PASTE()
=RETURN()
```

Left to Right Algorithm, Failed Bits at End

DEST	SOURCE	STP 1	STP 2	STP 3	STP 4	DEST
0 1 0 0 0 8						
0 1 0 0 0 8						
0 1 0 0 0 8						
0 1 0 0 0 8						
0 1 0 0 0 8						
0 1 0 0 0 8						
0 1 0 0 0 8						
0 1 0 0 0 8	0 0 1 1 1 7	0 0 1 1 1 7	0 1 1 1 1 15	0 1 0 1 1 11	0 1 0 0 1 9	0 1 0 0 0 8
0 1 0 0 0 8	0 1 0 0 0 8	0 1 0 0 0 8	0 1 0 0 0 8	0 1 0 0 0 8	0 1 0 0 0 8	0 1 0 0 0 8
0 1 0 0 0 8	0 1 0 0 1 9	0 1 0 0 1 9	0 1 0 0 1 9	0 1 0 0 1 9	0 1 0 0 1 9	0 1 0 0 0 8
0 1 0 0 0 8	0 1 0 1 0 10	0 1 0 1 0 10	0 1 0 1 0 10	0 1 0 1 0 10	0 1 0 0 0 8	0 1 0 0 0 8
0 1 0 0 0 8	0 1 0 1 1 11	0 1 0 1 1 11	0 1 0 1 1 11	0 1 0 1 1 11	0 1 0 0 1 9	0 1 0 0 0 8
0 1 0 0 0 8	0 1 1 0 0 12	0 1 1 0 0 12	0 1 1 0 0 12	0 1 0 0 0 8	0 1 0 0 0 8	0 1 0 0 0 8
0 1 0 0 0 8	0 1 1 0 1 13	0 1 1 0 1 13	0 1 1 0 1 13	0 1 0 0 1 9	0 1 0 0 1 9	0 1 0 0 0 8
0 1 0 0 0 8	0 1 1 1 0 14	0 1 1 1 0 14	0 1 1 1 0 14	0 1 0 1 0 10	0 1 0 0 0 8	0 1 0 0 0 8
0 1 0 0 0 8	0 1 1 1 1 15	0 1 1 1 1 15	0 1 1 1 1 15	0 1 0 1 1 11	0 1 0 0 1 9	0 1 0 0 0 8
0 1 0 0 0 8	1 0 0 0 0 16	1 1 0 0 0 24	1 1 0 0 0 24	1 1 0 0 0 24	1 1 0 0 0 24	0 1 0 0 0 8
0 1 0 0 0 8	1 0 0 0 1 17	1 0 0 0 1 17	1 0 0 0 1 17	1 0 0 0 0 16	1 1 0 0 0 24	0 1 0 0 0 8
0 1 0 0 0 8	1 0 0 1 0 18	1 0 0 1 0 18	1 0 0 0 0 16	1 0 0 0 0 16	1 1 0 0 0 24	0 1 0 0 0 8
0 1 0 0 0 8	1 0 0 1 1 19	1 0 0 1 1 19	1 0 0 0 1 17	1 0 0 0 0 16	1 1 0 0 0 24	0 1 0 0 0 8
0 1 0 0 0 8	1 0 1 0 0 20	1 0 0 0 0 16	1 0 0 0 0 16	1 0 0 0 0 16	1 1 0 0 0 24	0 1 0 0 0 8
0 1 0 0 0 8	1 0 1 0 1 21	1 0 0 0 1 17	1 0 0 0 1 17	1 0 0 0 0 16	1 1 0 0 0 24	0 1 0 0 0 8
0 1 0 0 0 8	1 0 1 1 0 22	1 0 0 1 0 18	1 0 0 0 0 16	1 0 0 0 0 16	1 1 0 0 0 24	0 1 0 0 0 8
0 1 0 0 0 8	1 0 1 1 1 23	0 0 1 1 1 7	0 1 1 1 1 15	0 1 0 1 1 11	0 1 0 0 1 9	0 1 0 0 0 8
0 1 0 0 0 8	1 1 0 0 0 24	0 1 0 0 0 8	0 1 0 0 0 8	0 1 0 0 0 8	0 1 0 0 0 8	0 1 0 0 0 8
0 1 0 0 0 8						
0 1 0 0 0 8						
0 1 0 0 0 8						
0 1 0 0 0 8						
0 1 0 0 0 8						
0 1 0 0 0 8						
0 1 0 0 0 8						

DEST	SOURCE	STP 1	STP 2	STP 3	STP 4	DEST
0 0 1 1 1 7						
0 0 1 1 1 7						
0 0 1 1 1 7						
0 0 1 1 1 7						
0 0 1 1 1 7						
0 0 1 1 1 7						
0 0 1 1 1 7						
0 0 1 1 1 7	0 0 1 1 1 7	0 0 1 1 1 7	0 0 1 1 1 7	0 0 1 1 1 7	0 0 1 1 1 7	0 0 1 1 1 7
0 0 1 1 1 7	0 1 0 0 0 8	0 1 0 0 0 8	0 1 1 0 0 12	0 1 1 1 0 14	0 1 1 1 1 15	0 0 1 1 1 7
0 0 1 1 1 7	0 1 0 0 1 9	0 1 0 0 1 9	0 1 1 0 1 13	0 1 1 1 1 15	0 1 1 1 1 15	0 0 1 1 1 7
0 0 1 1 1 7	0 1 0 1 0 10	0 1 0 1 0 10	0 1 1 1 0 14	0 1 1 1 0 14	0 1 1 1 1 15	0 0 1 1 1 7
0 0 1 1 1 7	0 1 0 1 1 11	0 1 0 1 1 11	0 1 1 1 1 15	0 1 1 1 1 15	0 1 1 1 1 15	0 0 1 1 1 7
0 0 1 1 1 7	0 1 1 0 0 12	0 1 1 0 0 12	0 1 1 0 0 12	0 1 1 1 0 14	0 1 1 1 1 15	0 0 1 1 1 7
0 0 1 1 1 7	0 1 1 0 1 13	0 1 1 0 1 13	0 1 1 0 1 13	0 1 1 1 1 15	0 1 1 1 1 15	0 0 1 1 1 7
0 0 1 1 1 7	0 1 1 1 0 14	0 1 1 1 0 14	0 1 1 1 0 14	0 1 1 1 0 14	0 1 1 1 1 15	0 0 1 1 1 7
0 0 1 1 1 7	0 1 1 1 1 15	0 1 1 1 1 15	0 0 1 1 1 7	0 0 1 1 1 7	0 0 1 1 1 7	0 0 1 1 1 7
0 0 1 1 1 7	1 0 0 0 0 16	1 0 0 0 0 16	1 0 1 0 0 20	1 0 1 1 0 22	1 0 1 1 1 23	0 0 1 1 1 7
0 0 1 1 1 7	1 0 0 0 1 17	1 0 0 0 1 17	1 0 1 0 1 21	1 0 1 1 1 23	1 0 1 1 1 23	0 0 1 1 1 7
0 0 1 1 1 7	1 0 0 1 0 18	1 0 0 1 0 18	1 0 1 1 0 22	1 0 1 1 0 22	1 0 1 1 1 23	0 0 1 1 1 7
0 0 1 1 1 7	1 0 0 1 1 19	1 0 0 1 1 19	1 0 1 1 1 23	1 0 1 1 1 23	1 0 1 1 1 23	0 0 1 1 1 7
0 0 1 1 1 7	1 0 1 0 0 20	1 0 1 0 0 20	1 0 1 0 0 20	1 0 1 1 0 22	1 0 1 1 1 23	0 0 1 1 1 7
0 0 1 1 1 7	1 0 1 0 1 21	1 0 1 0 1 21	1 0 1 0 1 21	1 0 1 1 1 23	1 0 1 1 1 23	0 0 1 1 1 7
0 0 1 1 1 7	1 0 1 1 0 22	1 0 1 1 0 22	1 0 1 1 0 22	1 0 1 1 0 22	1 0 1 1 1 23	0 0 1 1 1 7
0 0 1 1 1 7	1 0 1 1 1 23	0 0 1 1 1 7	0 0 1 1 1 7	0 0 1 1 1 7	0 0 1 1 1 7	0 0 1 1 1 7
0 0 1 1 1 7	1 1 0 0 0 24	0 1 0 0 0 8	0 1 1 0 0 12	0 1 1 1 0 14	0 1 1 1 1 15	0 0 1 1 1 7
0 0 1 1 1 7						
0 0 1 1 1 7						
0 0 1 1 1 7						
0 0 1 1 1 7						
0 0 1 1 1 7						
0 0 1 1 1 7						
0 0 1 1 1 7						

OLD EXPLAN. OF ALGOR.

PERMUTAT 6/4/90 Page 1

Permutation Tables

PERM[120][5] PERM4[5][5] PERM5[72][5]

PERM - all different ways to rearrange "01234"

1st index	[0]	[1]	[2]	[3]	[4] permutation
[0]	0	1	2	3	4
[1]	0	1	2	4	3
[2]	0	1	3	4	2
[3]	0	1	3	2	4
[4]	0	1	4	2	3
[5]	0	1	4	3	2
[6]	0	2	3	4	1
...					
[24]	1	2	3	4	0
...					
[119]	4	3	2	1	0

PERM table is used to test possible paths for:
5-step direct paths (no extra dimensions added)
4-step direct paths (no extra dimensions added)

For 4-step paths, only PERM[0] thru PERM[23] is needed
and then only PERM[x][1] thru PERM[x][4], that
gives all different ways to rearrange "1234"

facts used to increase speed and avoid testing
path that will be busy:
[0] digit changes every 24 elements
[1] digit changes every 6 elements
[2] digit changes every even element
[3] digit changes every odd element
[4] digit changes every element

PERM5 - all different ways to rearrange "01234" without "0" and "4" next to each other

1st index	[0]	[1]	[2]	[3]	[4] permutation
[0]	1	2	3	1	4
[1]	1	2	3	4	1
[17]	3	1	2	1	4
[18]	3	1	2	4	1
[53]	3	2	1	1	4
...					
[71]	4	3	2	1	0

PERM5 table is used to test possible paths for:
5-step paths, with a dimension added

Number_Of_Steps from source to destination was 3,
e.g. bit_sequence 123

Dimensions_Not_Taken are 0 and 3

Added an unused dimension to beginning and end of bit_sequence

e.g. bit_sequence 01240
31243

When traveling an extra dimension, it must first be taken, and later retraced, in order to reach destination. It is not taken and then retraced in the next step, so the bits in bit_sequence should not be rearraged so that first and last digits are together.

The elements of PERM5 are tested one after the other.

PERM4 - all different ways to rearrange "0123" without "0" and "3" next to each other

1st index	[0]	[1]	[2]	[3] permutation
[0]	0	1	2	3
[1]	0	1	3	2
[2]	0	2	3	1
[3]	0	2	1	3
[4]	1	3	2	0
[5]	1	0	2	3
[6]	2	3	1	0

PERM4 table is used to test possible paths for:
4-step paths, with a dimension added

Number_Of_Steps from source to destination was 2,
e.g. bit_sequence 03

Dimensions_Not_Taken are 1, 2 and 4

Added an unused dimension to beginning and end of bit_sequence
e.g. bit_sequence 1031
2032
4034

When traveling an extra dimension, it must first be taken, and later retraced, in order to reach destination. It is not taken and then retraced in the next step, so the bits in bit_sequence should not be rearraged so that first and last digits are together.

The elements of PERM4 are tested one after the other.

GIVEN

	e.g.	address	
source node	3	00011	
destination node	28	11100	
busy nodes	0	00000	
	2	00010	
	27	11011	
NUMBER_OF_BITS	5	01234	size of hypercube # of dimensions # of digits in address of node number bits starting with 0

msb 0 travels left and right between subcubes
1 travels up and down between subcubes
2 travels forward and back within a subcube
3 travels left and right within a subcube
lsb 4 travels up and down within a subcube

3581
Feb

GIVE

Algor1[0] = 0 for no direct path to destination
1 there is at least one direct path to destination
-1 did not test this source-to-destination (e.g. source was busy)

Algor1[1] thru
Algor1[x] bit_sequence used for first successful path
If Number_Of_Steps = 1, only Algor1[1] used.
If Number_Of_Steps = 2, Algor1[1] - Algor1[2] used.
If Number_Of_Steps = 3, Algor1[1] - Algor1[3] used.
If Number_Of_Steps = 4, Algor1[1] - Algor1[4] used.
If Number_Of_Steps = 5, Algor1[1] - Algor1[5] used.

Path1[0] number of paths tried until successful one found

Path1[1] thru
Path1[x] nodes taken in successful path
"x" is 1 thru 5 depending on Number_Of_Steps (see Algor1).

Path1[6] index of Perm table containing bit_sequence for first successful path

All unused dimensions will be tested if Number_Of_Steps is 1, 2 or 3.
Length_of_sequence after dimension added is Number_Of_Steps + 2 (3, 4, or 5).
Dim1 and Dim1Path calculated for Number_Of_Steps 1, 2, and 3.
Dim2 and Dim2Path calculated for Number_Of_Steps 1, 2, and 3.
Dim3 and Dim3Path calculated for Number_Of_Steps 1 and 2.
Dim4 and Dim4Path calculated for Number_Of_Steps 1.

Dim1[0] = 0 for no path thru the first unused dimension to destination
1 there is at least one path thru the first unused dimension
-1 not tested (e.g. Number_Of_Steps > 3)
Dim1 calculated for Number_Of_Steps 1, 2, and 3.

Dim1[1] thru
Dim1[x] bit_sequence used for first successful path
If Length_of_sequence = 3, Dim1[1] - Dim1[3] used.
If Length_of_sequence = 4, Dim1[1] - Dim1[4] used.
If Length_of_sequence = 5, Dim1[1] - Dim1[5] used.

Dim1Path[0] number of paths tried until successful one found

Dim1Path[1] thru
 Dim1Path[x] nodes taken in sucessful path
             "x" is 1 thru 5 depending on Length_of_sequence (see Dim1).

Dim1Path[6] index of Perm4 or Perm5 table containing bit_sequence for
             first successful path.
             Perm4 used for Length_of_sequence 3 and 4.
             Perm5 used for Length_of_sequence 5.

Dim2 and Dim2Path
Dim3 and Dim3Path... see Dim1 and Dim1Path
Dim4 and Dim4Path

example of output table for

source	3	00011
destination	28	11100
xor/BinaryWord		11111
bitsequence		01234
Number_Of_Steps		5
Busy_Nodes		0, 2, 27

table

	Algor1		Dim1	Dim2	Dim3	Dim4
index		Path1	Dim1Path	Dim2Path	Dim3Path	Dim4Path
[0]	1	2	-1	-1	-1	-1
[1]	0	19				
[2]	2	23				
[3]	3	21				
[4]	4	20				
[5]	1	28				
[6]		6				

example of output table for

source	3	00011
destination	20	10100
xor/BinaryWord		10111
bitsequence		0 234
Number_Of_Steps		4
Busy_Nodes		0, 2, 22

table

	Algor1		Dim1	Dim2	Dim3	Dim4
index		Path1	Dim1Path	Dim2Path	Dim3Path	Dim4Path
[0]	1	2	-1	-1	-1	-1
[1]	0	19				
[2]	1	23				
[3]	3	21				
[4]	2	20				
[5]						
[6]		2				

example of output table for

```
 source          3   00011
 destination 16      10000
xor/BinaryWord       10111
bitsequence          0   34
Number_Of_Steps      3
Busy_Nodes           1, 2, 19, 20
```

table

Algor1		Dim1		Dim2		Dim3		Dim4	
index	Path1	Dim1Path		Dim2Path		Dim3Path		Dim4Path	
[0]	0	1	1	1	2	-1		-1	
[1]		1	11	2	7				
[2]		0	27	0	23				
[3]		3	26	3	21				
[4]		4	24	2	17				
[5]		1	16	4	16				
[6]			0		2				