ON A PERFORMANCE OF INTERCONNECTION NETWORK

By

PRAVEEN KUMAR - 021224
INDU BHUSHAN KUMAR - 021210

विद्या तत्व ज्योतिसमः

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

MAY - 2006

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY – WAKNAGHAT

CERTIFICATE

This is to certify that the work entitled , "On a performance of Interconnection network" submitted by Praveen Kumar (021224) and Indu Bhushan Kumar (021210) in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science Engineering of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Nitin
13th MAY 2006
Mr. NITIN

ACKNOWLEDGMENT

Completion of B.Tech. dissertation is a marathon errand with various aspects to the project work. One has to view the project from many different angles and many different permutation and combination have to be worked out to make this project a success. In the project many new things were worked out, Understanding interconnection network, a very new thing was a bit complex. To this complacency, real life problems and worries added a new factor.

All these things made completion of the project as a sort of mission. Now as this mission is completed and as we look back retrospect, we can see those helping hands, which have helped us in successful completion of this mission. It's our heart felt to acknowledge them, here and right now.

We are fortunate to have *Mr. NITIN* as our project guide. We are indebted to him for the immense help and for the valuable aspects and different queries. All this has helped us in gaining a deeper insight into the system with a lot of confidence. At same time, he kept us on our toes by his valuable criticism. We are very thankful to all who have helped us in any way.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

- **IN** - Interconnection network
- **MIN** – Multistage interconnection network
- **ABN** - Augmented Baseline Network
- **FDOT** – Fault Tolerant Double tree
- **MDOT**- Modified Double Tree
- **MTTF** – Mean Time To Failure
- **SE** – Switching element

ABSTRACT

In this project, the study of different regular and irregular multipath hybrid multistage interconnection networks (MINs) named as Fault-tolerant Augmented Baseline Networks (ABN), Fault-tolerant double-tree Networks (FDOT), and Modified double-tree Networks (MDOT) have been carried out.

Two algorithms for calculating the path-length and the routing-tag algorithm of ABN networks have been proposed. The existing reliability equations (in terms of upper and lower bound of MTTF) of ABN and FDOT have been automated using certain assumptions. In addition to this, the implementation of the path-length and routing tag algorithm of FDOT and MDOT networks have also been carried out. Moreover, the complexities of path-length and routing tag algorithm for ABN, FDOT and MDOT networks are calculated and compared. All experimental and simulation results for the reliability analysis of network size starting from 4 x 4 to 1024 x 1024 are provided. The reliability comparison results that FDOT is better in comparison to ABN as the network size increases. Regarding the comparison on complexities, the result shows that complexity of MDOT is low as compared to complexities of ABN and FDOT. However, MDOT is not a fault-tolerant network and regular ABN and irregular FDOT have the same complexities.

PROBLEM DEFINITION

1. To develop the path-length and routing tag algorithm for Fault-tolerant Augmented Baseline Networks and automate its existing reliability equation for both upper and lower bound.

2. To automate existing path-length, routing tag algorithm and reliability for Fault-tolerant FDOT Networks and MDOT Networks.

3. To analyze the characteristics of the two networks viz ABN and FDOT based on reliability graphs and corresponding complexities

Chapter 1

INTRODUCTION

With the present state of technology, building multiprocessor systems with hundreds of processor is feasible. A vital component of these systems is the interconnection network (IN) that enables the processors to communicate among themselves or with memory units. Any processor in a multiprocessor system should be able to directly address every shared memory module through the IN. As a result the performance of a multiprocessor system rests primarily on the design of its IN [1].

A number of techniques have been proposed to increase the reliability of MINs. The modest cost of unique-path MINs makes them attractive for large processor systems, but their lack of fault-tolerance is major drawback. To mitigate this problem, three hardware options are available: replicate the entire network, add extra stages, and/or additional links.

The general goals for the design of fault-tolerant MINs are high reliability, good performance even in the presence of faults, low cost. However fault tolerant MINs cannot achieve all of these goals at the same time. Some of the networks fail to tolerate faults in the first and/or last stages. Some others can tolerate faults at any stage but they are, in general, too costly [2].

1.1 Description of ABN

This study involves MINs with redundant paths between every source-destination pair. **ABN** (Augmented baseline network) is a network with N sources and N destinations. We form two identical groups of N/2 sources and N/2 destinations. Each group consists of a multiple path modified baseline network of size N/2. The modified baseline network is a network with one less stage and feature links among switches belonging to the same stage and forming several loops of switches. In the figure on the next page, shown is 16 x 16 ABN networks [2].

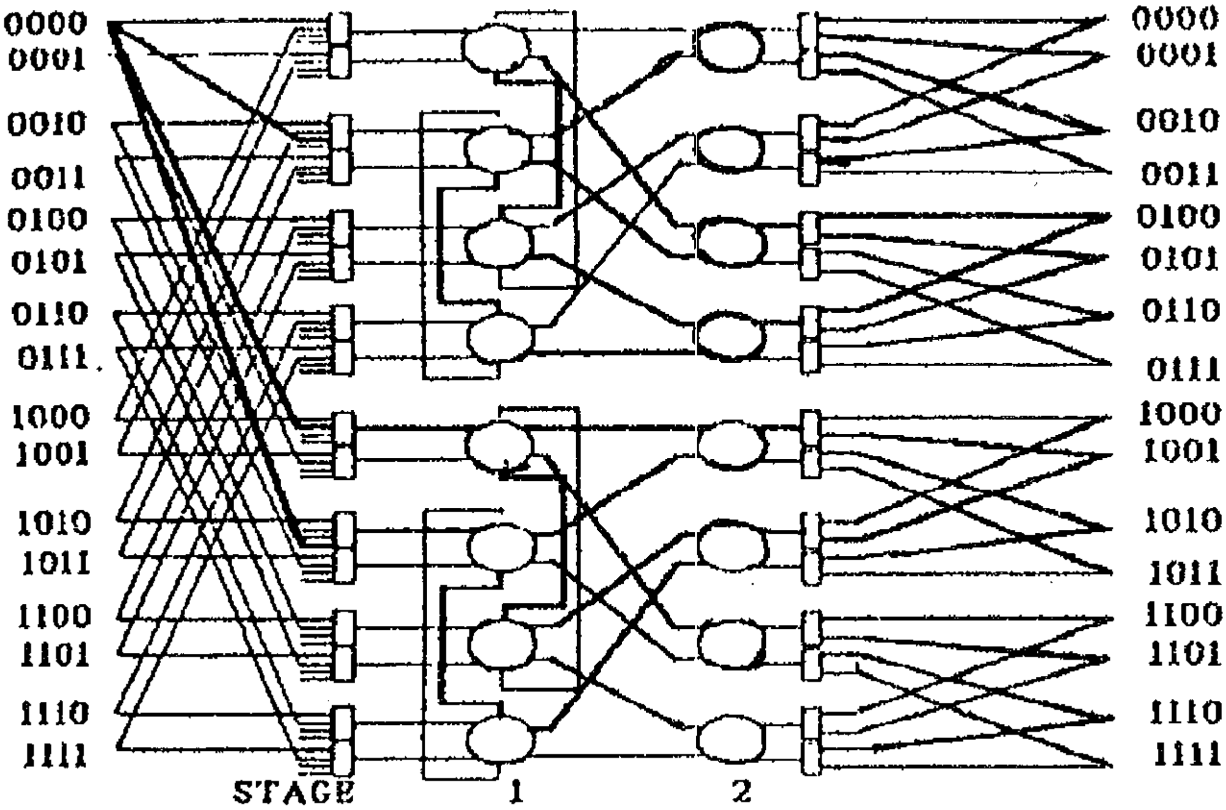


Fig.1.1 - The 16 x 16 ABN Network

1.2 Description of FDOT

FDOT (Fault tolerant double tree network) is an irregular type of MIN. It consists of a right half and a left half. Each half of the network resembles a binary tree. The left and right trees are mirror images of each other. A dot network of size 2^n x 2^n has 2^n input and 2^n output terminals and (2n-1) number of stages. Further, it has 2^{n+1} -3 switching elements (SEs).

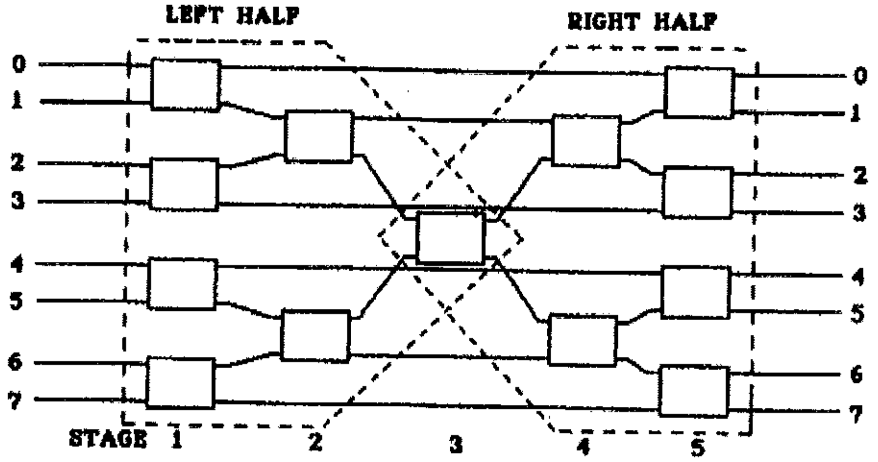


Fig.1.2 -The 2^3 x 2^3 FDOT NETWORK

1.2.1 Existing algorithm for Path-length of FDOT

For a given source-destination pair, there are multiple paths of different path-lengths in a FDOT-k network. The number of possible paths between a given source-destination pair varies from (k+1) to n x (k-1) for an N x N network, depending upon the addresses of the source and destination terminals. The algorithm for allocation of path-length gives the information about different possible paths between a source-destination pair. The minimum path-length between a source-destination pair depends upon the N/k bits of the source and the destination respectively i.e. d_{n-1}.......d_1, d_0 and sn-1...s1,s0 in any subnetwork

The source S and destination D are represented as:

$$S = S_{i,j} = S_n, (S_{n-1}, \ldots S_1, S_0)_2$$

$$D = D_{i,j} = d_n, (d_{n-1}, \ldots d_1, d_0)_2$$

Where s_n and d_n represents digits in a radix k number system.

In any subnetwork the possible path length algorithm is:

If

$[(s_{n-1} \oplus d_{n-1}) + (s_{n-2} \oplus d_{n-2}) + \ldots\ldots + (s_1 \oplus d_1)$ is zero

Then

Minimum path length is 2 and all paths of different lengths are possible i.e. Paths of length 2, 4, 6... (2n-2), (2n-1)

Else

If

$[(s_{n-1} \oplus d_{n-1}) + (s_{n-2} \oplus d_{n-2}) + \ldots\ldots + (s_2 \oplus d_2)$ is zero

Then

All paths of length equal to or greater than 4 are possible

Else

- .
- .

If

$[(s_{n-1} \oplus d_{n-1}) + (s_{n-2} \oplus d_{n-2}) + \ldots\ldots + (s_j \oplus d_j)$ is zero
Where $(1 <= j <= (n-1))$

Then

All paths of length equal to or greater than 2j are possible

Else

Path of length 2n-1 (i.e. longest path) is possible only[2].

1.2.2 Routing Tag algorithm

Routing algorithm for FDOT-k network gives the information about the distributed routing control tag required to establish a path between any source-destination terminal pair for a given path length (if it exists).

If

$2 <= x <= (2n-1)$

Then

Routing tag =

$S_{n-1}.\ (1.1\ldots\ldots1)_{(\lfloor x/2 \rfloor -1)}\ .0.\ (d_{(\lfloor x/2 \rfloor -1)} \ldots\ldots d_0)\ .\ d_n$

(Where x is the path length which varies in step of 2)

Else

If

$x = (2n-1)$

Then

Routing tag =

$s_n\ (1.1\ldots\ldots1)_{(n-1)}\ .(d_{(n-1)} \ldots\ldots\ldots\ldots d_0)\ .\ d_0$

Else

No tag is possible[1].

Example

Let the data be routed from S = 0000 to the various destinations of a $2^4 \times 2^4$ FT network. The Path-lengths are calculated for sets of destinations and are summarized in the table.

S	D	Path length(s) available(x)
0000	0000 0001 1000 1001	2,4,5
	0010 0011 1010 1011	4,5
	0100 0101 0110 0111 1100 1101 1110 1111	5

1.2.3 Complexity of FDOT [8, 10]

Statement:	Frequency	Total Steps
//b = find_n(N);	0	O(0)
for (i = n - 1; i >= 0; i--)	n	O(n)
{	0	O(0)
if (((s >> i) & j) ^ ((d >> i) & j))	1	O(1)
break;	1	O(1)
}		
if (i < n – 1)	1	O(1)
printf ("\n\the min_path is %d\n\n", k1 = (i + 1) * 2);	1	O(1)
else	0	O(0)
printf("\n\the min_path is %d\n\n", k1 = 2 * n - 1);	1	O(1)
for (; k1 <= 2 * n - 1; k1++)	2n	O(n)
{		
if (k1 >= 2 && k1 < 2*n - 1)	1	O(1)
{		
r [0] = s >> n;	1	O(1)
for (i = 0; i < floor (k1/2) - 1 ; i++)	n*n	O(n^2)
{		
r[i + 1] = 1;	1	O(1)
}		
k = i + 1;	1	O(1)

Statement	Frequency	Total Steps
r[k++] = 0;	1	O(1)
for(i = floor(k1/2) - 1; i >= 0; i--) {	n/2	O(n/2)
r[k++] = (d >> i) & 1; }	1	O(1)
r[k] = d >> n;	1	O(1)
printf("the routing tag for possible path = %d is :", k1);	1	O(1)

The Complexity of this program will be the highest power of n.

Complexity of FDOT = O (n^2).

Where 'O' is called "Big Oh" Notation.

1.2.4 Reliability of FDOT Upper-bound

$R_{FDOT\_UB}(t) = [1-(1-e^{-\lambda_m t})^2]^{N/2} \cdot [1-(1-e^{-\lambda_3 t})^2]^{N/4+N/4+N/8+\ldots+1} \cdot [1-(1-e^{-\lambda_{2d} t})^2]^{N/4}$

$MTTF_{FDOT\_UB}(t) = {}_0\int^{\infty} R_{FDOT\_UB}(t).dt$ [2]

1.2.5 Reliability of FDOT Lower-bound

$R_{FDOT\_LB}(t) = [1-(1-e^{-\lambda_{3m} t})^2]^{N/4} \cdot [1-(1-e^{-\lambda_3 t})^2]^{N/4+N/8+\ldots+1(n-3)} \cdot [1-(1-e^{-\lambda_{2d} t})^2]^{N/4}$

$MTTF_{FDOT\_LB}(t) = {}_0\int^{\infty} R_{FDOT\_LB}(t).dt$

1.3 Description of MDOT

MDOT is an irregular fault tolerant MIN also known as four-tree network. MDOT is similar to FDOT, the slight change between the two is regarding the connections of MDOT. It's basically the modified form of FDOT. A FT network of size $2^n \times 2^n$ is subdivided in two identical groups, each consisting of a MDOT network of size $2^{n-1} \times$
2^{n-1}, which are arranged one above the other. The two groups are formed based on the most significant bit (MSB) of the source-destination terminals.

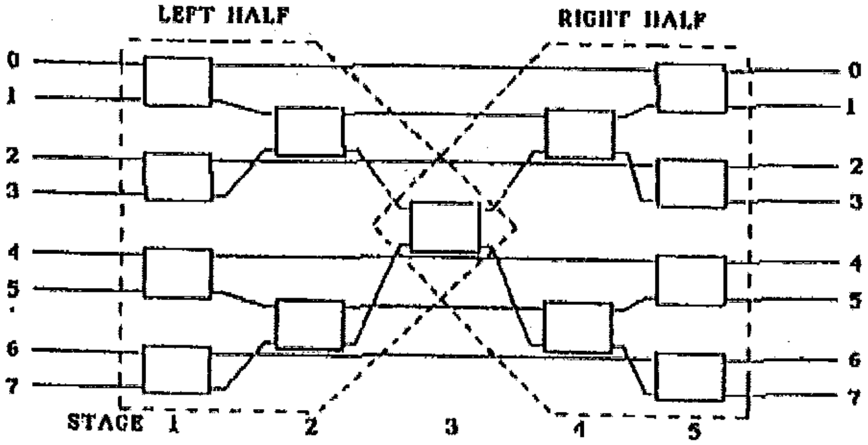


Fig.1.3-The $2^3 \times 2^3$ **MDOT NETWORK**

1.3.1 Existing algorithm for Path-length of MDOT

The possible path lengths between a particular source-destination pair vary from 2 to 2m-1 for a $2^n \times 2^n$ network. The path-length algorithm follows:

If

$[(s_{n-2} \oplus d_{n-2}) + (s_{n-3} \oplus d_{n-3}) + \ldots\ldots\ldots + (s_1 \oplus d_1)]$ is zero

($\oplus$ Represents an exclusive - OR and '+' represents an OR operation)

Then

Minimum path length is 2 and all paths of different lengths are possible i.e. Path of length 2, 4, 6 … (2m – 2), (2m – 1).

Else

If

$[(s_{n-2} \oplus d_{n-2}) + (s_{n-3} \oplus d_{n-3}) + \ldots\ldots\ldots + (s_2 \oplus d_2]$ is zero

Then

All path of length equal to or greater than 4 are possible

Else

.

.

If

$[(s_{n-2} \oplus d_{n-2}) + (s_{n-3} \oplus d_{n-3}) + \ldots\ldots\ldots + (s_j \oplus dj]$ is zero

{Where 1<=j<= (n-2)}

Then

All path of length equal to or greater than 2j are possible

Else

Path of length 2m-1 (i.e. longest path) is possible only [3].

1.3.2 Routing Tag algorithm

The routing tag algorithm gives the information about the distributed routing control required to establish a path between any source – destination terminal pair for a given path length.

If

$2 <= x < (2m - 1)$

Then

Routing tag =

$S_{n-1} . (1.1\ldots\ldots1)_{(\lfloor x/2 \rfloor -1)} .0. (d_{(\lfloor x/2 \rfloor -1)} \ldots\ldots d_0) . d_{n-1}$

(Where x is the path length which varies in step of 2)

Else

If

$x = (2m - 1)$

Then

Routing tag =

$S_{n-1} . (1.1\ldots\ldots1)_{(\lfloor x/2 \rfloor)} (d_{(\lfloor x/2 \rfloor)} \ldots\ldots d_0) . d_{n-1}$

Else

No tag is possible [3].

1.3.3 Complexity of MDOT [8,10]

Statement	Frequency	Total Steps
//b = find_n(N);	0	O(0)
for(i = n - 1; i > 0; i--)	n	O(n)
{	0	O(1)
if(((s >> i) & j) ^ ((d >> i) & j)) break;	1 1	O(1) O(1)
}	0	O(0)
if(i < n - 1)	1	O(1)
{	0	O(1)
printf("\n\n the min_path is %d\n\n", (i + 1) * 2);	1	O(1)
printf("other possible paths are :");	1	O(1)

Statement	Frequency	Total Steps
for(j = (i + 1) * 2; j < 2 * n - 1; j += 2)	2n	O(2n)
printf("%d ,",j);	1	O(1)
printf("%d", 2 * n - 1);	1	O(1)
}	0	O(1)
else	1	O(1)
printf("\n\nthe min_path is %d\n\n", 2 * n - 1);	1	O(1)

The Complexity of this program will be the highest power of n.

Complexity of MDOT = O (n).

Where 'O' is called "Big Oh" Notation.

Chapter 2

WORKING WITH ABNs

There was no algorithm as such for calculating path-length. However, there was routing scheme provided for the network but still no algorithm was available for its implementation and that is what we have tried here to develop.

2.1 Path length and routing algorithm for ABN network [2]

Binary function

```
Input--a, n, b []
Output--binary

Initialize i, j
For i <- (n-1) to i<- 0
        do b[i] <-a%2

If a/2=1
Then b [i-1] =a/2
        For j<-(i-2) to j <- 0
        do b[j] <- 0
        break

Else
        a <- a/2
```

Log function

```
Input number
Output log value

Initialize b, count
b <- 2
count <- 1

While true
        do b<-(2*b)
        count<-(count+1)
        If b=a
        Then break

Return count-2
```

Path-length and hence routing

```
Input n
Output path length
```

```
Initialize i , count

For i<-0 to n
Do count<-count+1
        If (i! = (n-1))
        Then print.

Print minimum path length=count
```

Example

Let the data be routed from S = 0000 to the various destinations of a 16 x 16 ABN network. The Path-lengths are calculated for sets of destinations and are summarized in the table.

S	D	Path length(s) available(x)
0000	0000 0001 1000 1001	2,4
	0010 0011 1010 1011	4
	0100 0101 0110 0111 1100 1101 1110 1111	4

2.2 Complexity of ABN network [8,10]

Statement	Frequency	Total Steps
void binary(int a,int n)	1	O(1)
{	0	O(0)
int i,j;	1	O(1)
for(i=(n-1);i>=0;i--)	n	O(n)
{	0	O(0)
b[i]=a%2;	1	O(1)
if(a/2==1)	1	O(1)

Statement	Frequency	Total Steps
{	0	O(0)
b[i-1]=a/2;	1	O(1)
for(j=i-2;j>=0;j--)	n*n	$O(n^2)$
b[j]=0;	1	O(1)
break;	1	O(1)
}	0	O(0)
else	1	O(1)
a=a/2;	1	O(1)
}	0	O(0)
}	0	O(0)
int log_base2(int a)	1	O(1)
{	0	O(0)
int b=2;	1	O(1)

int count=1;	1	O(1)
while(1)	2n	O(2n)
{	0	O(0)
b=2*b;	1	O(1)
count++;	1	O(1)
if(b==a)	1	O(1)
break;	1	O(1)
}	0	O(1)

Statement	**Frequency**	**Total Steps**
return (count-2);	1	O(1)
}	0	O(0)
void routing(int n)	1	O(1)
{	0	O(0)
int i,count=0;	1	O(1)
printf("\n\n\n THE ROUTING-TAG FOR THE PARTICULAR \nCOMBINATION\nOF SOURCE AND DESTINATION IS");	1	O(1)
for(i=0;i<n;i++)	n	O(n)
{	0	O(0)
printf("%d",b[i]);	1	O(1)
count++;	1	O(1)
if(i!=n-1)	1	O(1)
printf(" . ");	1	O(1)
}	0	O(0)

printf("\n\nTHE MINIMUM PATH LENGTH IS : %d",count);	1	O(1)
}	0	O(0)

The Complexity of this program will be the highest power of n.

Complexity of ABN = O (n^2).

Where 'O' is called "Big Oh" Notation.

2.3 Reliability analysis

The reliability of ABN in terms of Mean Time to Failure (MTTF) is analyzed. The assumptions used are:

1) Switch failures occurs independently in a network with a failure rate of λ for 2x2 crossbar switches ($\lambda = 10^{-6}$ per hr).

2) Failures of multiplexers and demultiplexers also occur independently with a failure rates of λ_m and λ_d respectively , which can be different from λ based on the gate counts, we can assume $\lambda_m = \lambda_m/4$.

We consider the 2x2 switch and its associated DEMUX as a single component (SE_{2d}) , so $\lambda_{2d} = 2\lambda$ can be assigned to this group of elements also let λ_3 be the failure rate for the 3x3 switch (SE_3) ,then based on gate count $\lambda_3 = 2.25\lambda$ [3].

2.3.1 Upper bound:

Expression for the upper bound of the ABN reliability is:

$$\mathbf{R_{ABN\_UB}\ (t)} = [1- (1 - e^{-\lambda_m t})^2]^{N/2} \ .\ [1- (1 - e^{-\lambda_3 t})^2]^{(N/4)(n-3)} \ .\ [1- (1 - e^{-\lambda_{2d} t})^2]^{N/4}$$

$$\mathbf{MTTF_{ABN\_UB}\ (t)} = {}_0\int^{\infty} R_{ABN\_UB}\ (t).dt$$

2.3.2 Lower bound:

Expression for the lower bound of the ABN reliability is:

$$\mathbf{R_{ABN\_LB}\ (t)} = [1- (1 - e^{-2\lambda_{3m} t})^2]^{N/8} \ .\ [1- (1 - e^{-2\lambda_3 t})^2]^{(N/8)(n-4)} \ .\ [1- (1 - e^{-\lambda_{2d} t})^2]^{N/4}$$

$$\mathbf{MTTF_{ABN\_LB}\ (t)} = {}_0\int^{\infty} R_{ABN\_LB}\ (t).dt$$

Chapter 3

COMPARISON AND ANALYSIS

In this chapter, we will draw different graphs in terms of reliability and Path-length and draw conclusion based on the graphs.

3.1 Points drawn out from ABN Reliability upper bound

Keeping t and λm (order of 1e – 6) fix, we can have points for MTTF and n.
At t = 5 seconds and switch failure rate, λm = 0.000045

N	MTTF
4	5.847650
8	4.513927
16	3.373369
32	2.049566
64	1.200479
128	0.736502
256	0.466385
512	0.301469
1024	0.197636

3.2 Points for ABN Reliability Lower-bound

AT t = 5 seconds and switch failure rate, λm = 0.000045

N	MTTF
4	5.041469
8	4.430859
16	3.048297
32	1.054849
64	0.926867
128	0.555431
256	0.346488
512	0.221705
1024	0.144294

3.3 Points drawn out for FDOT Reliability upper bound

Keeping t and λm (order of 1e – 6) fix, we can have points for MTTF and n.
At t = 5 seconds and switch failure rate, λm = 0.000045

N	MTTF
4	3.621916
8	2.939288
16	2.169978
32	1.517644
64	1.053646
128	0.734022
256	0.513109
512	0.359676
1024	0.252644

3.4 Points for FDOT Reliability Lower-bound

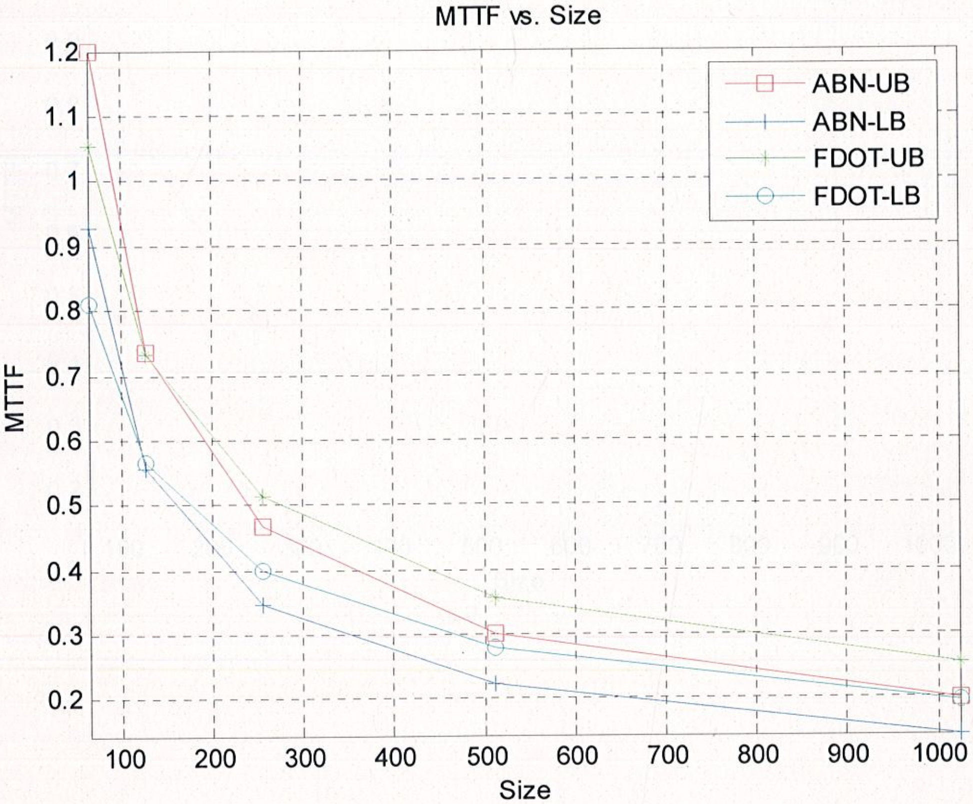
AT t = 5 seconds and switch failure rate, λm = 0.000045

N	MTTF
4	6.651063
8	4.522378
16	2.173333
32	1.097199
64	0.630571
128	0.384727
256	0.242635
512	0.116251
1024	0.102053

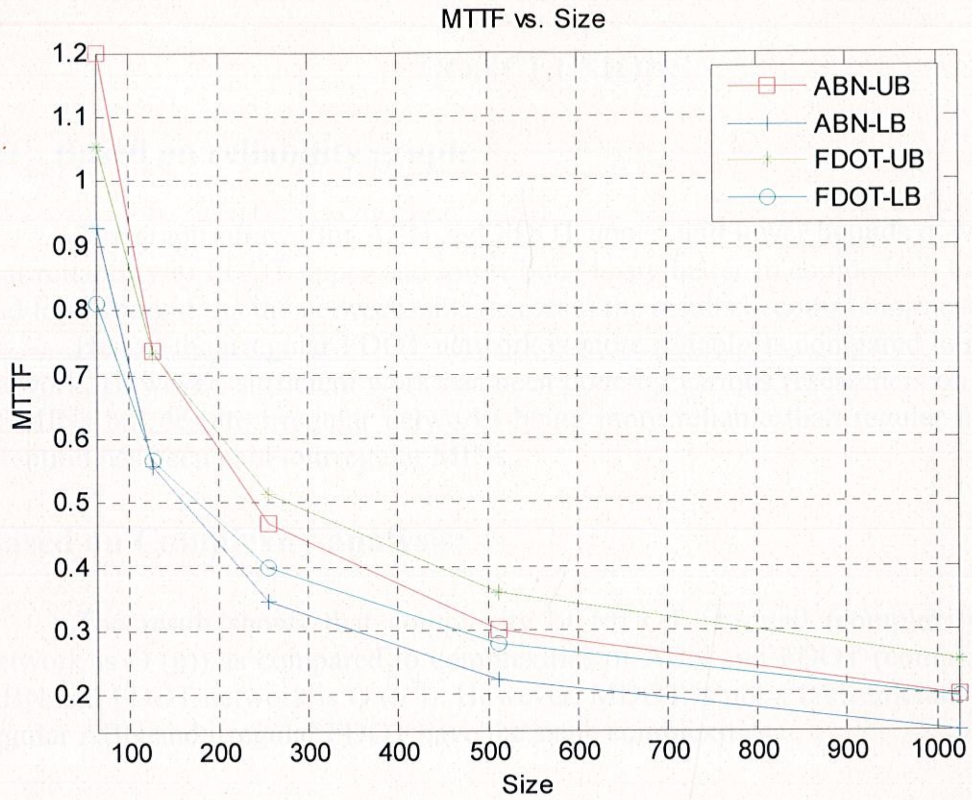
3.5 Graph of Reliability

Here we draw the graphs of reliability for ABN and FDOT network for their upper and lower bound respectively [4, 5].

3.5.1 CASE 1: For t = 0 to t = 5s



3.5.2 CASE 2: For t=0 to t=1000s



Chapter 4

CONCLUSIONS

4.1 Based on reliability graph:

The graphs drawn for ABN and FDOT upper, and lower bounds of MTTF results that reliability of FDOT upper and lower bounds are better in comparison to ABN upper and lower bound. As the network, size increases the results becomes more exciting.

Hence, the irregular FDOT network is more reliable as compared to regular ABN network. However, sufficient work has been done by various researchers on regular type of MINs but despite irregular networks being more reliable than regular network little attention has been paid to irregular MINs.

Based on Complexity analysis:

The result shows that complexity of MDOT is small (complexity of MDOT network is O (n)) as compared to complexities of ABN and FDOT (complexity of both ABN and FDOT network is O (n^2)). However, MDOT is not a fault-tolerant network and regular ABN and irregular FDOT have the same complexities.

4.2 Future scopes

1. Search for more new topological designs and analysis of static and dynamic, regular and irregular MINs with increased improvement in performance and reliability.
2. Combining multiple layers of sub networks to improve reliability and performance.
3. The use of MINs in ATM applications.
4. Generating Systems of Equations for Performance Evaluation of Multistage Interconnection Networks needs more exploration.

APPENDIX A: ABN

A.1 ABN Path-Length, Routing Tag Source Code

```
#include<stdio.h>
#include<conio.h>


int b [10];
void binary (int, int);
int log_base2 (int);
void routing (int);

void main ()
	{

	int N, src, dst, n, i;
	clrscr ();
	printf ("ENTER THE NO OF SOURCE OR DESTINATION N<N=2^x>: ");
	scanf ("%d", &N);
	printf ("\n\n\nENTER THE VALUE FOR SOURCE (0 to %d):", N-1);
	scanf ("%d", &src);
	printf ("\n\n\nENTER THE VALUE FOR DESTINATION (0 to %d):", N-1);
	scanf ("%d", &dst);
	n=log_base2 (N);
	binary (dst,(n+2));
	routing (n+2);
	getch ();

	}

Void binary (int a, int n)
	{
	int i,j;
	for(i=(n-1);i>=0;i--)
		{
		b[i]=a%2;
		if(a/2==1)
			{
			b[i-1]=a/2;
			for(j=i-2;j>=0;j--)
				b[j]=0;
			break;
			}
		else
			a=a/2;
```

```
		}
	}

int log_base2(int a)
	{
	int b=2;
	int count=1;
	while(1)
			{
			b=2*b;
			count++;
			if(b==a)
				break;
			}
	return (count-2);
	}

void routing(int n)
	{
	int i,count=0;
	printf("\n\n\nTHE ROUTING-TAG FOR THE PARTICULAR
COMBINATION\nOF SOURCE AND DESTINATION IS \n\n\t\t\t\t\t: ");
	for(i=0;i<n;i++)
		{
		printf ("%d",b[i]);
		count++;
		if(i!=n-1)
			printf(" . ");
		}
	printf("\n\nTHE MINIMUM PATH LENGTH IS :	%d",count);

	}
```

A.2 Reliability source code
Upper bound:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

int log_2(int);

int main(void)
{
  int N,                        //size of the network
      n;                        //log of N to the base 2
  unsigned long int Tm,         //total time in milli-seconds
                    t;
  double T,                     //upper time limit
         f,                     //failure rate of 2*2 crossbar switch
         fm,                    //failure rate of a multiplexer
         f2,                    //failure rate of a 2*2 switch
         f3,                    //failure rate of a 3*3 switch
         y, y1, y2, y3;
  long double integral = 0;

  clrscr();
  printf("\n\n enter the size of network(N), upper limit of time in seconds(T),"
         "\n failure rate of 2*2 crossbars switches (f)");
  scanf ("%d%lf%lf", &N, &T, &f);

  n = log_2(N);
  fm = f / 4;
  f2 = 2 * f;
  f3 = 2.25 * f;
  Tm = (unsigned long int) T * 1000;

  for(t = 1; t < Tm; t++)
  {
     y = pow(1 - pow(1 - exp(-fm*t), 2), N/2)*              //in calculation of
         pow(1 - pow(1 - exp(-f3*t), 2), N/4*(n - 3))*      //y we have assumed
         pow(1 - pow(1 - exp(-f2*t), 2), N/4);              //an interval of
                                                            //1ms each.
     integral += y;
  }

  printf("\n\n\n the value of MTTF is : %Lf" , integral/1000);

  getch();
```

```
    return 0;
}

int log_2(int j)
{
    int i , n = 0;


    for(i = 1; i < j; i += i)
    n++;

    return n;
}
```

Lower bound:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

int log_2(int);

int main(void)
{
    int N,                          //size of the network
          n;                        //log of N to the base 2
    unsigned long int Tm,           //total time in milli-seconds
                        t;
    double T,                       //upper time limit
            f,                      //failure rate of 2*2 crossbar switch
            f2,                     //failure rate of a 2*2 switch
            f3,                     //failure rate of a 3*3 switch
            y, y1, y2, y3;
    long double integral = 0;

    clrscr();
    printf("\n\n enter the size of network(N), upper limit of time in seconds(T),"
            "\n failure rate of 2*2 crossbar switches(f)");
    scanf("%d%lf%lf", &N, &T, &f);

    n = log_2(N);
    f2 = 2 * f;
    f3 = 2.25 * f;
    Tm = (unsigned long int) T * 1000;

    for(t = 1; t < Tm; t++)
```

```
    {
        y =  pow(1 - pow(1 - exp(-2*f3*t), 2), N/8)*
             pow(1 - pow(1 - exp(-2*f3*t), 2), N*(n - 4)/8)*
             pow(1 - pow(1 - exp(-f2*t), 2), N/4);

        integral += y;
    }

    printf ("\n\n\n the value of MTTF is : %Lf" , integral/1000);

    getch ();
    return 0;
}

int log_2(int j)
{
    int i , n = 0;


    for(i = 1; i < j; i += i)
    n++;

    return n;
}
```

APPENDIX B: FDOT

B.1 FDOT Path-length, Routing Tag source code

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

int log_2(int);

int main(void)
{
  int N, k, n, s, d, i, j = 1,k1, r[20];

  clrscr();
  printf("\n enter the value for N for the N*N  FDOT network\n(N should be power of 2)
\n");
  scanf("%d", &N);
  printf("\n enter the value of an FDOT - k network\n(k should be a power of 2) \n");
  scanf("%d", &k);

  n = log_2(N/k);
  //printf("\n\n%d\n\n" , n);

  printf("\n enter the value for source \n( the value must lie between 0 & %d)", N - 1);
  scanf("%d", &s);
  printf("\n enter the value for destination \n( the value must lie between 0 & %d)", N -
1);
  scanf("%d", &d);

  //b = find_n(N);

  for (i = n - 1; i >= 0; i--)
  {
      if(((s >> i) & j) ^ (( d >> i) & j))
      break;
  }

  if(i < n - 1)
  printf("\n\nthe min_path is %d\n\n", k1 = (i + 1) * 2);
  else
  printf("\n\nthe min_path is %d\n\n", k1 = 2 * n - 1);

  for(; k1 <= 2 * n - 1; k1++)
  {
      if(k1 >= 2 && k1 < 2*n - 1)
      {
```

```
    r[0] = s >> n;
    for(i = 0; i < floor(k1/2) - 1 ; i++)
    {
        r[i + 1] = 1;
    }

    k = i + 1;
    r[k++] = 0;

    for(i = floor(k1/2) - 1; i >= 0; i--)
    {
        r[k++] = (d >> i) & 1;
    }

    r[k] = d >> n;

    printf("the routing tag for possible path = %d is :", k1);

    for(i = 0; i <= k; i++)
    {
        printf("%d%c", r[i], '.');
    }

}
else
{
    r[0] = s >> n;

    for(i = 0; i < n - 1; i++)
    {
        r[i + 1] = 1;
    }

    k = i + 1;

    for(i = n - 1; i >= 0; i--)
    {
        r[k++] = (d >> i) & 1;
    }

    r[k] = d >> n;

    printf("\n\nthe routing tag for possible path = %d is :", k1);

    for(i = 0; i <= k; i++)
    {
```

```
            printf ("%d%c", r[i], '.');
        }
        printf("\n\n");
      }
  }

  getch();
  return 0;
}

int log_2(int j)
{
  int i , n = 0;


  for(i = 1; i < j; i += i)
  n++;

  return n;
}
```

B.2 FDOT Reliability source code
Upper-bound:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

int log_2(int);

int main(void)
{
  int N;                      //size of the network
  unsigned long int Tm,         //total time in milli-seconds
                    t;
  double T,                     //upper time limit
         f,                      //failure rate of 2*2 crossbar switch
         fm,                    //failure rate of a multiplexer
         f2,                   //failure rate of a 2*2 switch
         f3,                  //failure rate of a 3*3 switch
         y, y1, y2, y3;
  long double integral = 0;

  clrscr();
  printf("\n\n enter the size of network(N), upper limit of time in seconds(T),"
         "\n failure rate of 2*2 crossbar switches(f)");
```

```
  scanf("%d%lf%lf", &N, &T, &f);

  fm = f / 4;
  f2 = 2 * f;
  f3 = 2.25 * f;
  Tm = (unsigned long int) T * 1000;

  for(t = 1; t < Tm; t++)
  {
      y = pow(1 - pow(1 - exp(-fm*t), 2), N/2)*        //in calculation of
         pow(1 - pow(1 - exp(-f3*t), 2), N + 1)* //y we have assumed
         pow(1 - pow(1 - exp(-f2*t), 2), N/4);        //an interval of
                                                     //1ms each.
      integral += y;
  }

  printf("\n\n\n the value of MTTF is : %Lf" , integral/1000);

  getch();
  return 0;
}
```

Lower-bound:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

int log_2(int);

int main(void)
{
  int N,                    //size of the network
      n;                    //log of N to the base 2
  unsigned long int Tm,      //total time in milli-seconds
                    t;
  double T,                 //upper time limit
         f,                 //failure rate of 2*2 crossbar switch
         fm,                //failure rate of a multiplexer
         f2,                //failure rate of a 2*2 switch
         f3,                //failure rate of a 3*3 switch
         y, y1, y2, y3;
  long double integral = 0;

  clrscr();
  printf("\n\n enter the size of network(N), upper limit of time in seconds(T),"
```

```
        "\n failure rate of 2*2 crossbar switches(f)");
    scanf("%d%lf%lf", &N, &T, &f);

    n = log_2(N);
    fm = f / 4;
    f2 = 2 * f;
    f3 = 2.25 * f;
    Tm = (unsigned long int) T * 1000;

    for(t = 1; t < Tm; t++)
    {
        y = pow(1 - pow(1 - exp(-fm*t), 2), N/4)*        //in calculation of
            pow(1 - pow(1 - exp(-f3*t), 2), (N + 1)*(n - 3))* //y we have assumed
            pow(1 - pow(1 - exp(-f2*t), 2), N/4);        //an interval of
                                                         //1ms each.
        integral += y;
    }

    printf("\n\n\n the value of MTTF is : %Lf" , integral/1000);

    getch();
    return 0;
}

int log_2(int j)
{
    int i , n = 0;


    for(i = 1; i < j; i += i)
    n++;

    return n;
}
```

APPENDIX C: MDOT

C.1 MDOT Path-length, Routing Tag source code

```
#include<stdio.h>
#include<conio.h>

int log_2(int);

int main(void)
{
   int N, k, n, s, d, i, j = 1;

   clrscr();
   printf("enter the value for N for the N*N  MDOT network(N should be power of 2) \n");
   scanf("%d", &N);
   printf("\n enter the value of an MDOT - k network\n(k should be a power of 2) \n");
   scanf("%d", &k);

   n = log_2(N/k);
   //printf("\n\n%d\n\n" , n);

   printf("\n enter the value for source \n( the value must lie between 0 & %d): ", N - 1);
   scanf("%d", &s);
   printf("\n enter the value for destination \n( the value must lie between 0 & %d): ", N -
1);
   scanf("%d", &d);

   //b = find_n(N);

   for(i = n - 1; i > 0; i--)
   {
        if(((s >> i) & j) ^ (( d >> i) & j))
        break;
   }

   if(i < n - 1)
   {
        printf("\n\nthe min_path is %d\n\n", (i + 1) * 2);
        printf("other possible paths are :");
        for(j = (i + 1) * 2; j < 2 * n - 1; j += 2)
        printf("%d ,",j);
        printf("%d", 2 * n - 1);
   }
   else
   printf("\n\nthe min_path is %d\n\n", 2 * n - 1);
   getch();
```

```
    return 0;
}

int log_2(int j)
{
    int i , n = 0;


    for(i = 1; i < j; i += i)
    n++;

    return n;
}
```

APPENDIX D: GRAPH

D.1 Corresponding Matlab 7.0.1 Program code

/* Matlab 7.0.1 program

For t = 0 to t = 5s

```
hold on

x =[64 128 256 512 1024];

a = [1.200479    0.736502    0.466385    0.301469    0.197636]
b = [0.926867    0.555431    0.346488    0.221705    0.144294]
c = [1.053646    0.734022    0.513109    0.356978    0.252644]
d = [0.810830    0.566806    0.397547    0.279342    0.196508]

plot(x, a,'-rs');
plot(x, b,'-b+');
plot(x, c,'-g*');
plot(x, d,'-co');

box on
axis tight
grid on

title('MTTF vs. Size')
xlabel('Size')
ylabel('MTTF')

legend('ABN-UB','ABN-LB','FDOT-UB','FDOT-LB')
```

D.2 Corresponding Matlab 7.0.1 Program code

/* Matlab 7.0.1 program
for t = 0 to t = 1000

```
hold on

x =[64 128 256 512 1024];

a = [1.200486   0.736502   0.466385   0.301469   0.197636]
b = [0.926868   0.555431   0.346488   0.221705   0.144294]
c = [1.053646   0.734022   0.513109   0.359676   0.252644]
d = [0.810830   0.566806   0.397547   0.279342   0.196508]
plot(x, a,'-rs');
plot(x, b,'-b+');
plot(x, c,'-g*');
plot(x, d,'-co');

box on
axis tight
grid on

title('MTTF vs. Size')
xlabel('Size')
ylabel('MTTF')

legend('ABN-UB','ABN-LB','FDOT-UB','FDOT-LB')
```

REFERENCES

Research Papers

[1] Bansal, P. K., Singh K., and Joshi, R. C., "Quad tree: a cost-effective fault-tolerant multi-stage interconnection network," IEEE INFOCOM, 20, pp. 860-866, May 4-8, 1992.

[2] Bansal, P. K., Joshi, R. C., Singh K., and Siroha, G. P., "Fault-tolerant augmented baseline multi-stage interconnection network," IEEE TENCON, Vol.2, pp. 200-204, August 28 - 30, 1994.

[3] Bansal, P. K., Singh K., and Joshi, R. C., "Routing and path length algorithm for a cost-effective four-tree multi-stage interconnection network," International Journal of Electronics, Vol. 73, No. 1, pp. 107-115, 1992.

[4] Patel, J. H., "Performance of processor-memory interconnection for multiprocessors," IEEE Transaction on Computers, Vol. C-30, pp. 771-780, October 1981.

[5] Shooman, M. L., "Reliability of computer systems and networks: fault-tolerance, analysis, and design," John Wiley & Sons, Inc., New York 2002.

Books

[6] Jose Duato, Sudhakarar Yalamanchili and Lionel NI, "Interconnection networks: An engineering approach", page 1-39,139-204.

[7] William James Dally, Brian Towles, "Principals and practices of interconnection networks", page 1-24

[8] Gregory L Heileman, "Data structure, algorithms and object-oriented programming", page 5-20

Websites

[9] http://www.llnl.gov/computing/tutorials/parallel_comp

[10] http://www.cs.wisc.edu/~hasti/cs367-common/notes/COMPLEXITY