REVIEW OF FAULT TOLERANT TECHNIQUES FOR DIFFERENT TYPES OF GRAPHS

BY-

HATEM NASSRAT

TARAK SHINGNE

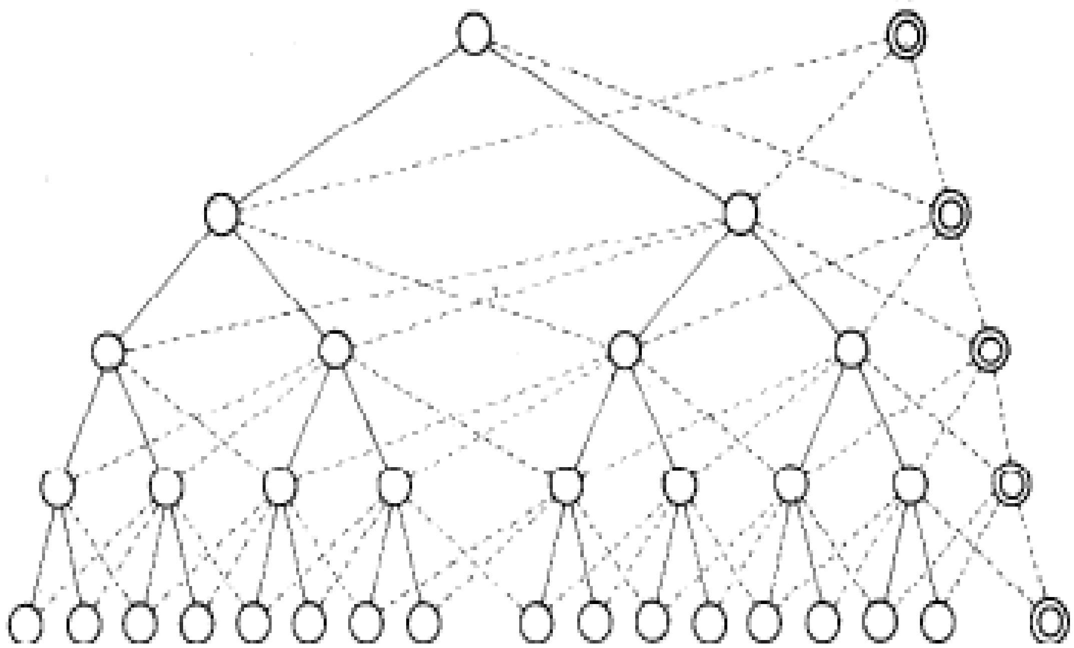
Outline

- General view of Fault Tolerance
- Ft-Design approaches
  - Trees
  - Meshes & Hypercubes
- conclusion

Introduction

- **Fault tolerance:** It is the property that enables a system to continue operating properly in the event of the failure of some of its components. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naively –designed system in which even a failure can cause total breakdown.
- **Fault tolerant design:** It refers to a method for designing a system so it will continue to operate ,possibly at a reduced level ,rather than failing completely ,when some of the parts of the system fails.

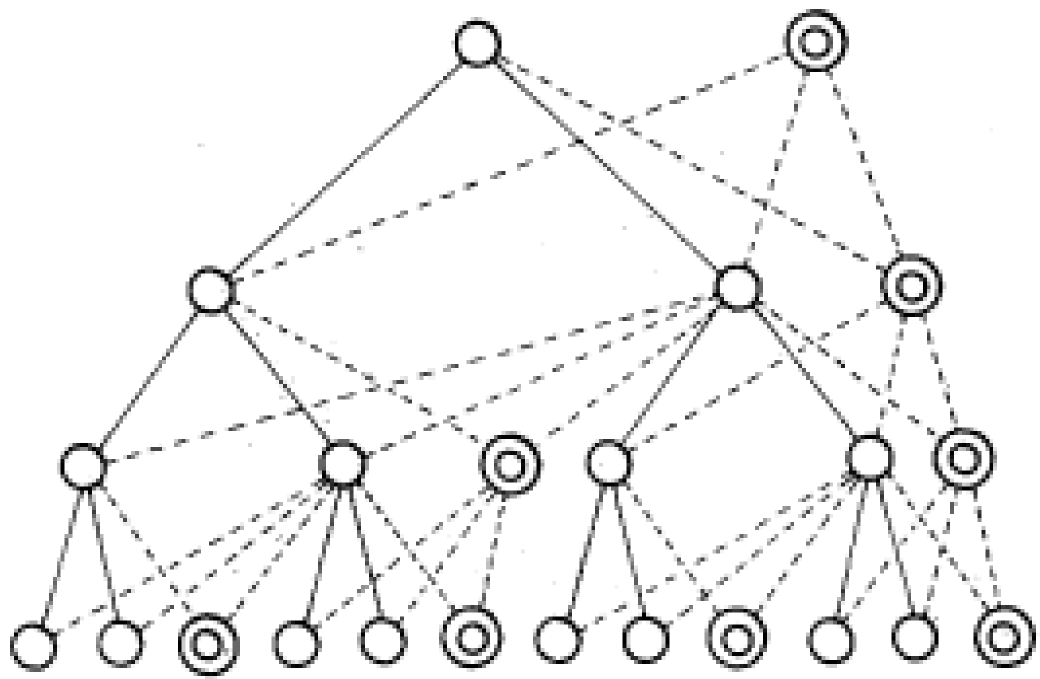
Scheme with spares [1]



Scheme with spares [1]

- There is a spare node for each level in the tree there are redundant links indicated by dashed lines
- As it is very evident from the figure, single failure in each level can be tolerated
- In the case of a node failure, reconfiguration is done to maintain the logical structure of a tree
- This scheme tolerates several failures if they are in different levels of the tree
- Additional spare nodes can be used at lower levels of the tree where the number of nodes increases rapidly

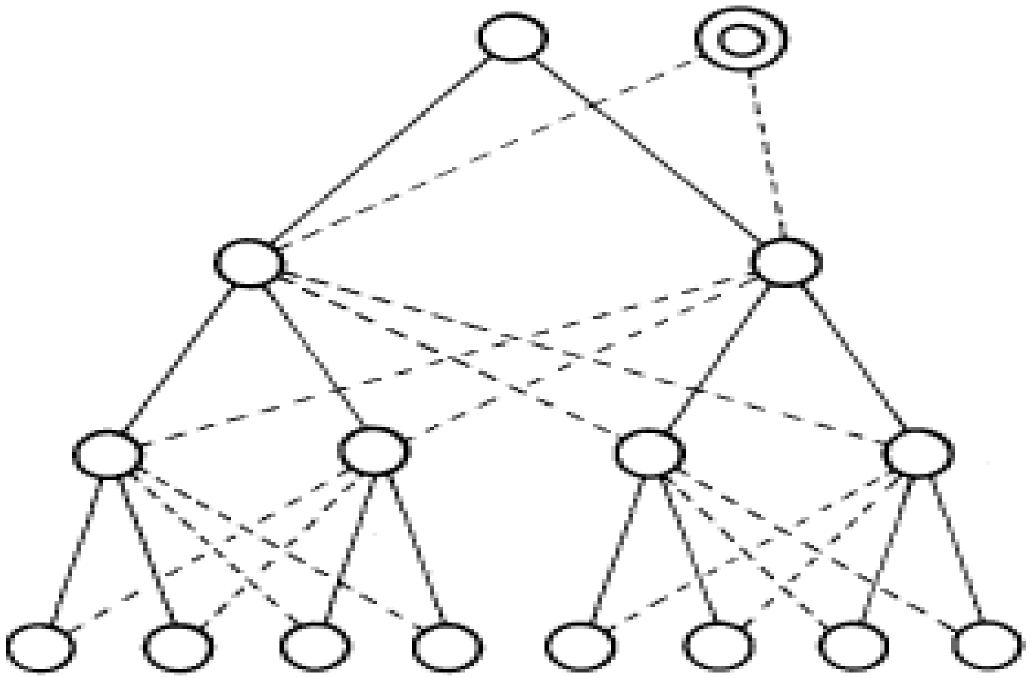
Extensions to the scheme with spares [1]



Extensions to the scheme with spares [1]

- The scheme with spares can be extended by increasing the number of spares as the nodes per level of tree increases
- The technique is to provide 1 spare for every $k=2^j$, for some value of j
- Variety of arrangements is possible depending on the value of j

Scheme with performance degradation [1]



Scheme with performance degradation [1]

- As the name implies, this scheme operates with performance degradation when the node fails
- Only one spare node for root
- Rest of the nodes are covered by extra links from each node
- Neighbor will have to take care of the computations in case of failure, so performance get affected
- Failures of one out of two can be tolerated
- Multiple failures can be tolerated if they are non-adjacent
- Suitable design where processors are very powerful in computation and load sharing

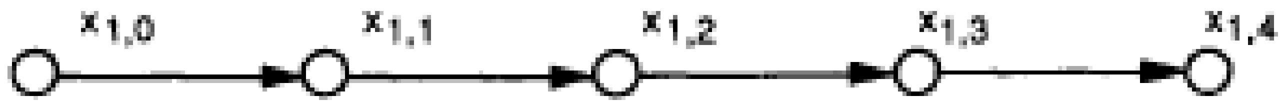
1-ft design for trees [2]

- A super graph G, of a given graph H, is a k-fault tolerant realization of H if for any set F of k nodes in G, the graph induced by V(G)-F contains a subgraph isomorphic to H.

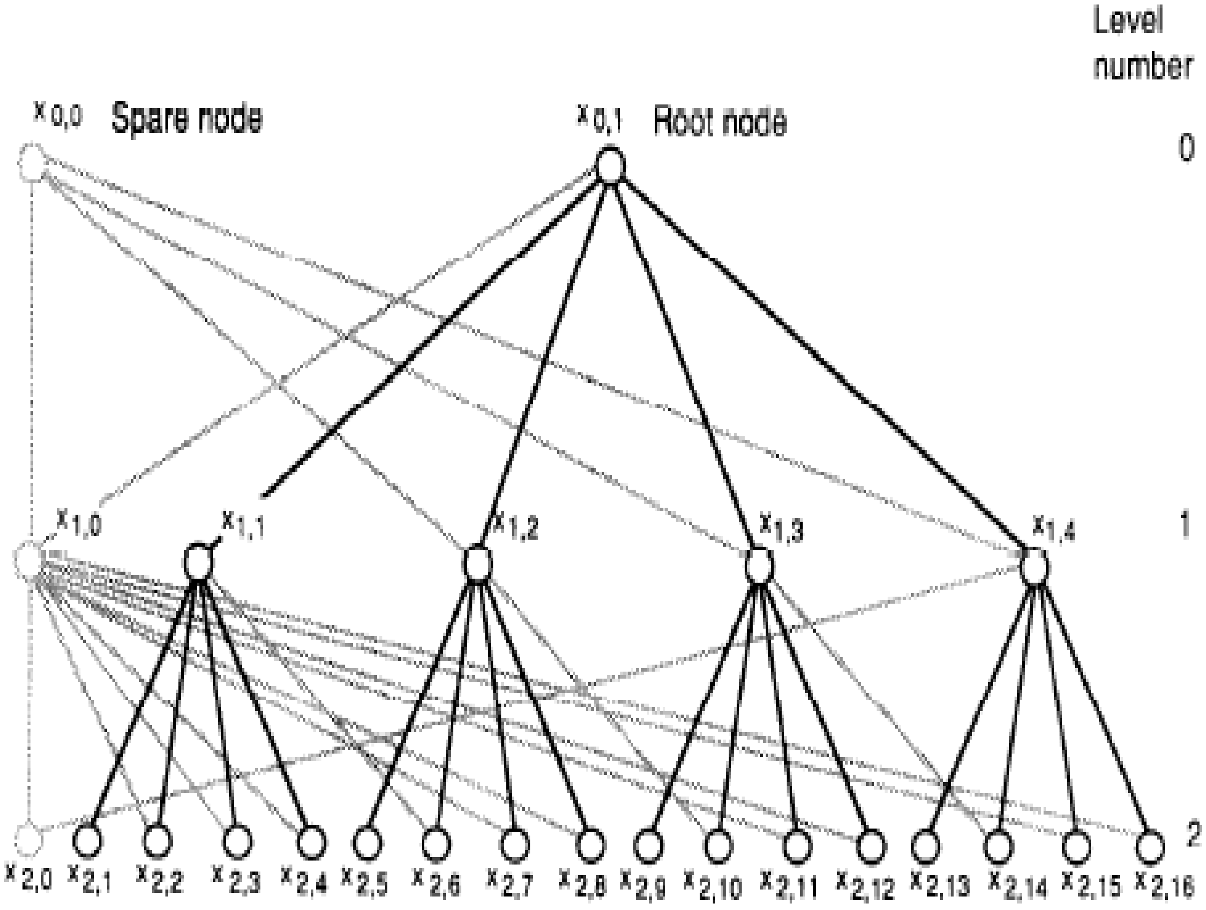
- Important factors for design for fault tolerance:
  - Number of spare nodes
  - Number of spare edges
  - Node degree
  - Reconfiguration time

Graph covering concept [2]

- Definition: A node X i,u is said to (completely) cover X i,v if Xi,u has edges to all of the Childs of X i,v ,provided X i,v has a set of Childs. In this case X i,v is called ***dependent*** on X i,u
- For example:



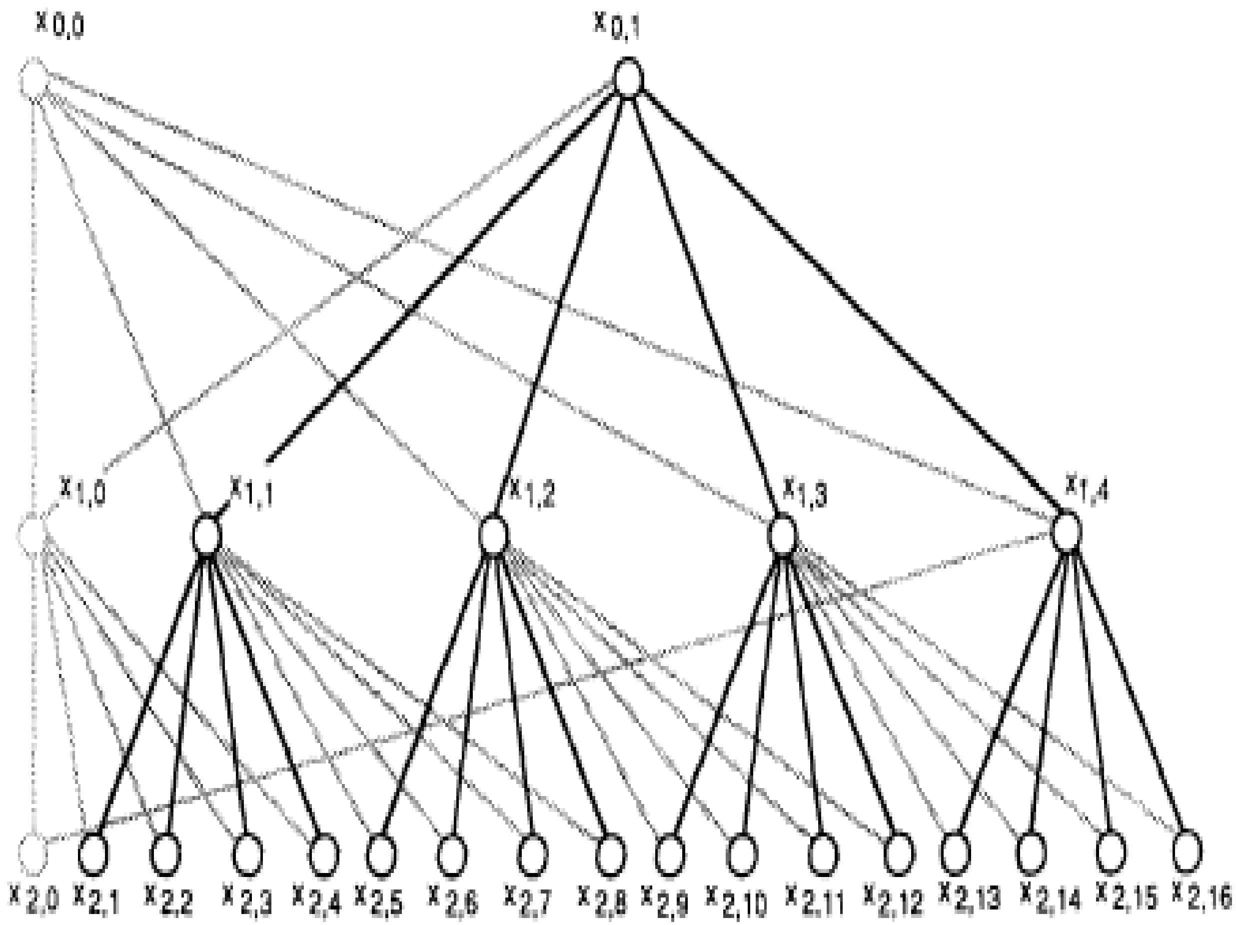
A design for 1-ft [2]



Drawbacks [3]

- There is a severe imbalance of node degrees. Nodes of high degree are costly to implement
- When a node X fails ,reconfiguration has to take place in levels i down l-1, thus disrupting normal processing of the nonfaulty nodes
- Only one faulty node is tolerated as it is evident from the figure
- The node utilization is not 100 %

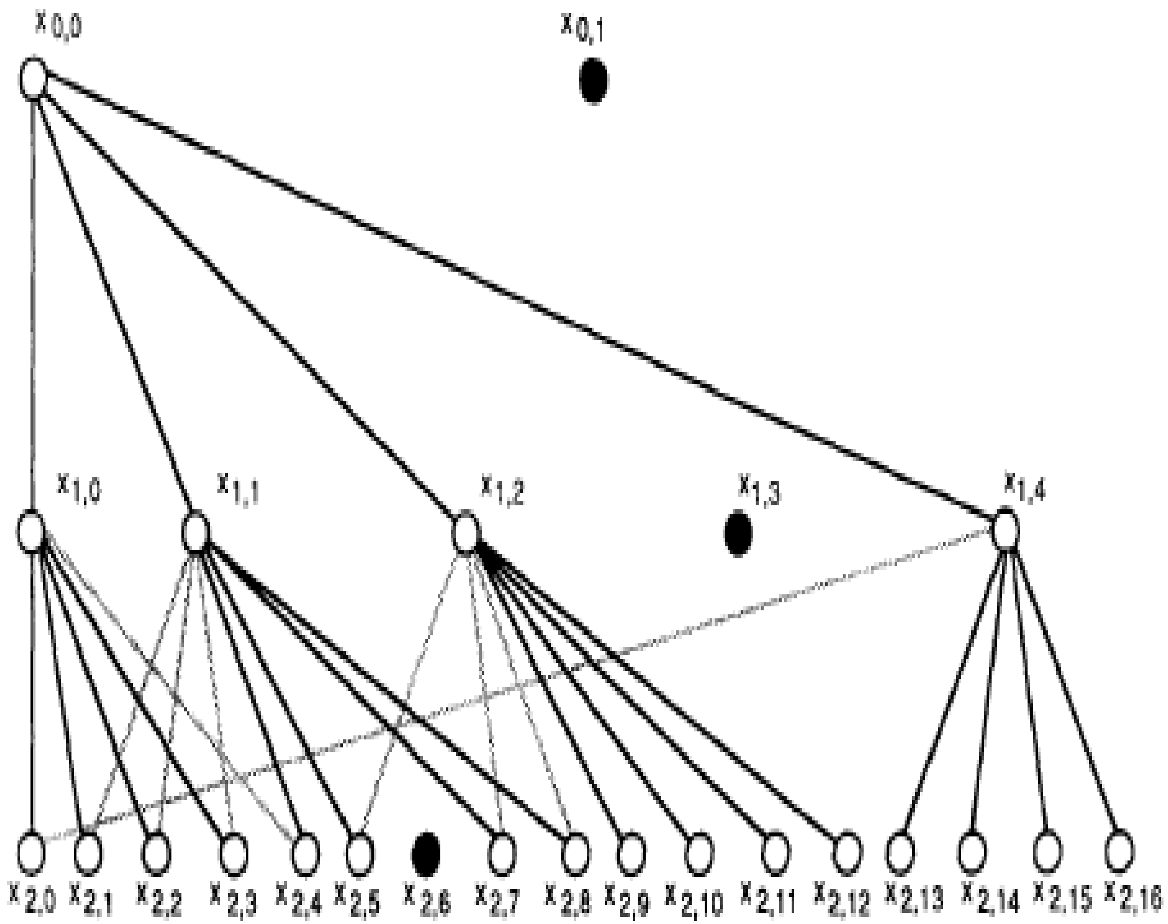
Improved 1-ft design for trees [2]



Advantages compared to previous design [2]

- The node degree is much better balanced as compared to the previous design as it is evident from the figure
- For any fault in level i, the reconfiguration is confined to levels i-1, i, and i+1
- One faulty node is tolerated at **each level**
- The node utilization is 100%.

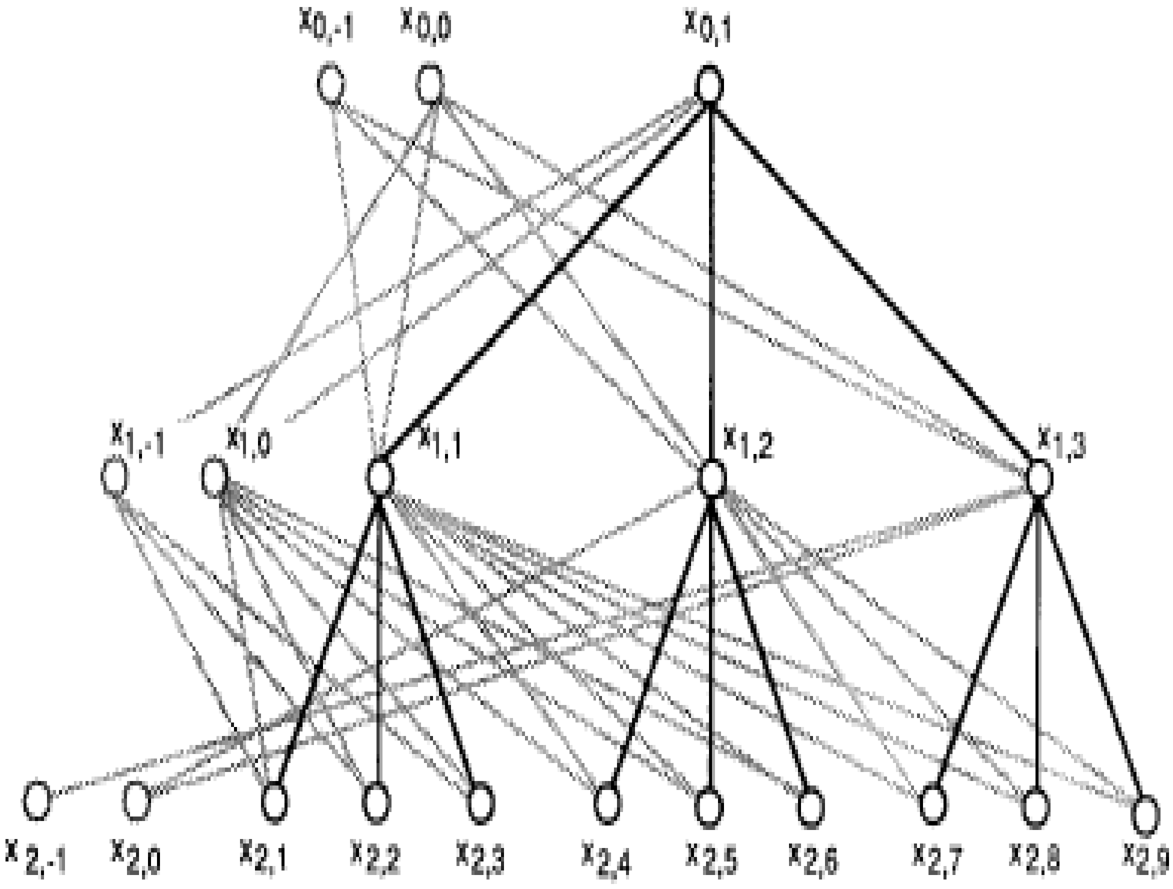
Reconfiguration [2]



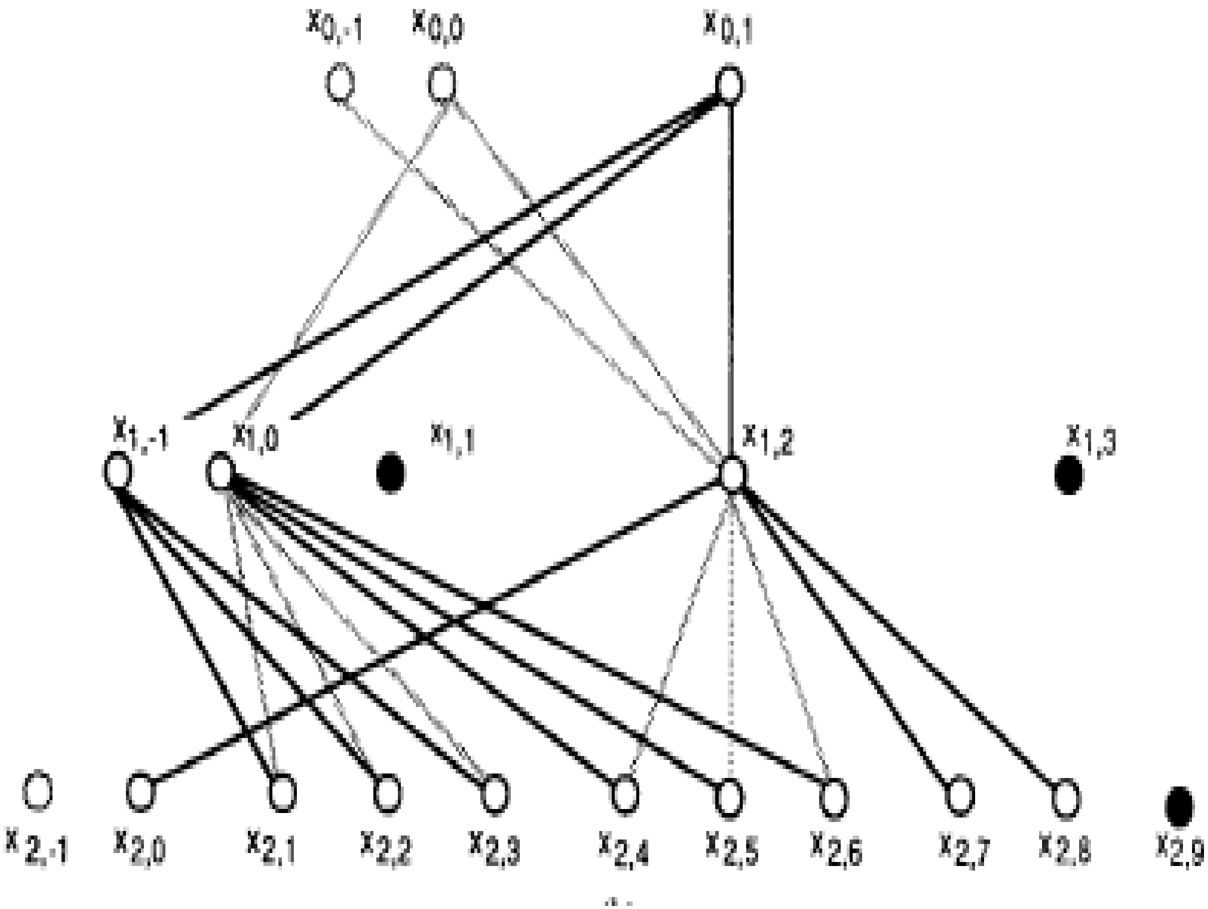
K-FT Design (k<d) [2]

- Theorem-1:In any K-FT NST $G[k,T_N(d,l)]$, every set of k/d+1 nodes in original graph O_i has to be covered by at least k-k/d other nodes in Xi for reconfiguration around any k or fewer faults
- Theorem -2:If each node v in original graph of $G[k,T_N(d,l)]$ is covered by at least k other nodes and the covering graph is acyclic, then there exists a covering sequence for any set of k or fewer faults in X_i
- Lemma-1: At least k(k+1)/2 edges are required between X_i and S_{i+1} in $G[k,T_N(d,l)]$

K-FT Design (k<d) [2]



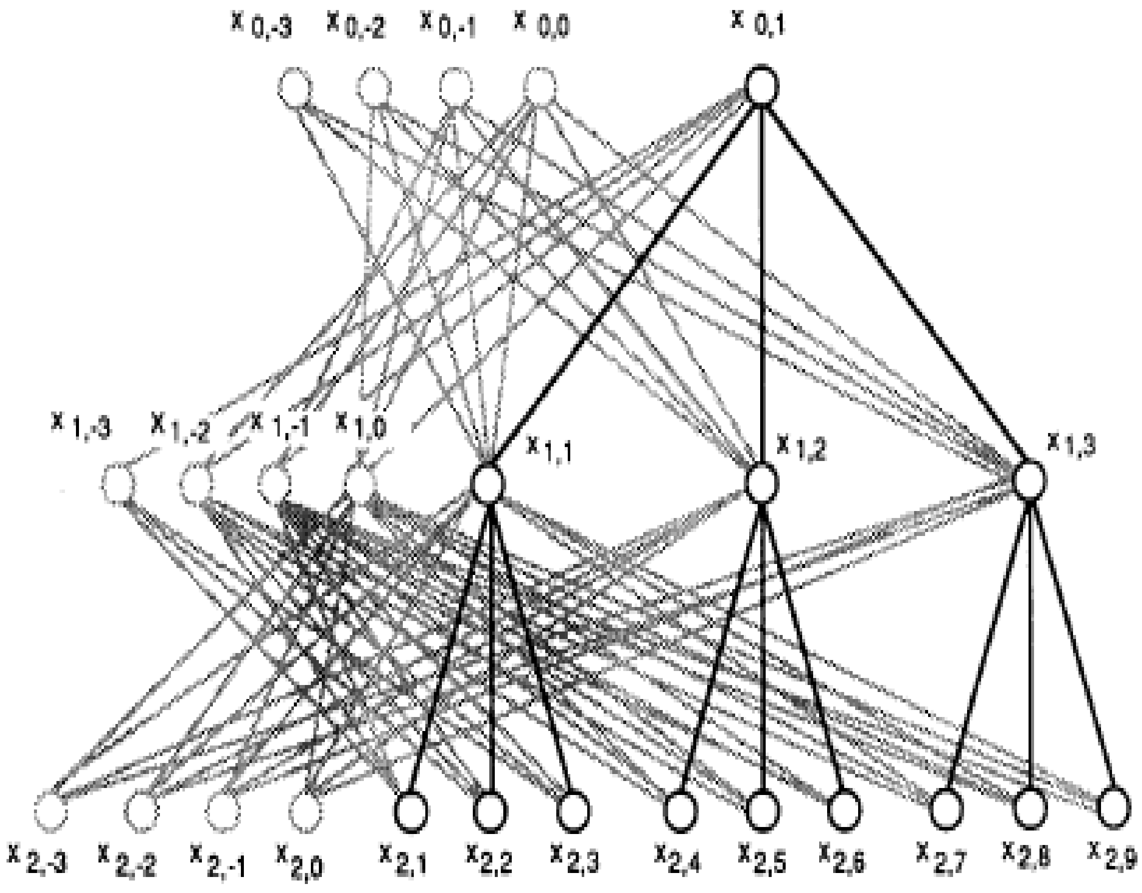
K-FT Design (k<d) [2]



K-FT Design ($k \geq d$) [2]

- Theorem-3:When each node only has complete covers, $G[k,T_N(d,l)]$ is an optimal K-FT graph for $T_N(d,l)$ with respect to minimizing number of spare nodes and edges
- Theorem-4:In $G[k,T_N(d,l)]$, for any $f=k-2k/d+2h \leq k$ faults in Xi, there exists a covering sequence for at least $k-2k/d+h$ faults, if $h \geq 1$,for all f faults otherwise

K-FT Design (k$\geq$d) [2]



K-FT Design (k$\geq$d) [2]

- X1,2 and X1,3 which are in level 1 do not cover any node
- X1,1 covers X1,3 and X 1,0 covers X1,2 and X1,3
- X1,-2 covers two nodes X1,1 and X1,2 while X1,-3 covers X1,1
- X1,-1 covers three nodes

Conclusion for K-FT trees

- Designing of K-FT trees should consider important factors such as number of nodes, number of edges, node degree, reconfiguration time
- Designing should be done based on the application requirements
- Node covering provides unifying concept for implementing K-FT versions of various types of trees and tree like systems

Fault tolerance and reconfiguration of circulant graphs

with application in meshes and hypercubes

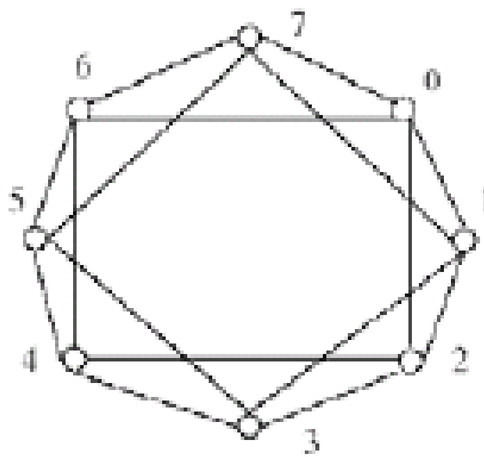
Important Definitions

Circulant Graph

"An n-node circulant graph is defined by a set of nodes numbered {0, 1, ..., n-1} and a set of integers, called offsets, denoted A = {a_l, a_2, ..., a_i,}. Two nodes x and y are joined by an edge iff there is an offset ai such that x-y=h (modn)."

[4]

Example: An 8-node circulant graph with offsets 1,2 noted as G[1,2:8]



Important Definitions

Theorem 2.1 [4]

- an n-node circulant graph G with a set of offsets $A=\{a_1, a_2, ..., a_i,\}$ has a k-ft extension H, with n+k nodes and offsets $\{a_1, a_1+1, ..., a_1+k\} \cup \{a_2, a_2+1, ..., a_2+k\} \cup ... \cup \{a_i, a_i+1, ..., a_i+k\}$.

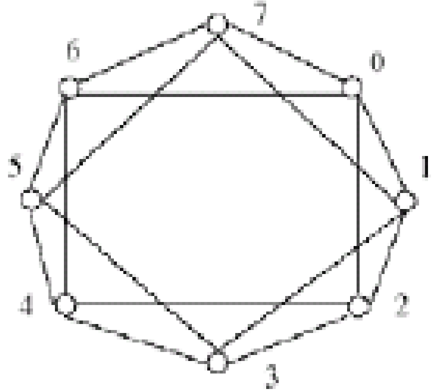


Fig. 2.1 A graph G[1,2:8].

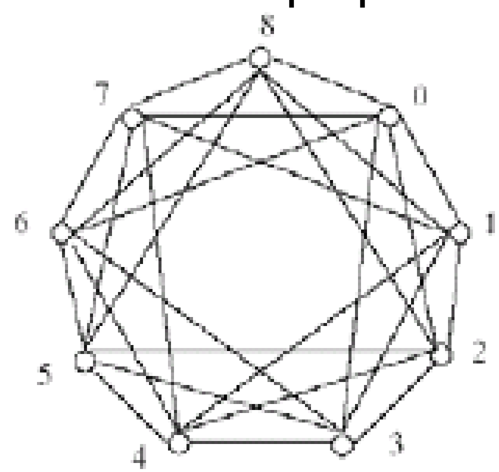


Fig. 2.2 A 1–ft of G[1,2:8].

[5]

Important Definitions

Partitioning sequences

"Let n and m be any pair of integers such that gcd(n,m)=1 and n > m > 0. We define an ordered sequence, based on n and m, denoted S(n,m)= <s_1, s_2,.., $s_{\lfloor n/2 \rfloor}$> where the i-th element in this sequence is computed as follows: if [i $*$ m (mod n)] ≤ $\lfloor n/2 \rfloor$, then s_i = [i $*$ m (mod n)]; otherwise, s_i = n - [i $*$ m (mod n)].

For instances, for n= 7 and m= 3, S(7,3)= <3,1,2>, and for n= 14 and m= 5, S(14,5)=<5,4,1,6,3,2,7>. "

[5]

Important Definitions

m-distance subsets

Let G be an n-node circulant graph with offsets A, and

Let $m \in \mathbb{N}\,; gcd(n,m) = 1$

Then $\text{m-distance subset} \subseteq A\,;$ all the offsets in the subset appear in consequtive order in S(n,m) (the corresponding m-partitioning sequence).

The following example illustrates m-distance subsets:

a 14-node circulant graph G with offsets A={1,4,6,7}, to look for the 5-distance sunsets. Compute S(14,5) = {5,4,1,6,3,2,7}. We get the following m-distance subsets, {4}, {1}, {6}, {7}, {1,4}, {4,6}, {1,4,6}.

The **maximal m-distance subsets** are defined as the m-distance subsets that are not contained within any other subsets. In the above example they would be the sets {1,4,6} and {7}.

Important Definitions

m-distance partition P(A,n,m):

- P(A,n,m) is defined as the set

$$\{x\,;\, x \in \text{set of maximal m-distance subsets for a given A,n,m}\}$$

- Example: P({1,4,6,7}, 14, 5) = {{1,4,6},{7}}

- Algorithm to partition A, O(|A| n). To run for all valid m's it would have a loose upper bound of O(|A|n^2).

Procedure Partition(A,n,m) {

1. Construct the sequence S(n,m)= $<s_1, s_2,.., s_{\lfloor n/2 \rfloor}>$ as explained before;
2. For every element s_i in S(n,m), if s_i appears as an offset in A, we keep it; otherwise, we replace s_i in S(n,m) by a special separation symbol, say "&";
3. For every maximal subsequence in S(n,m) that does not include the symbol "&", form an m-distance subset corresponding to this subsequence;
4. Return a partition P(A,n,m) consisting of all m-distance subsets formed above;

}

[5]

Important Definitions

This is an example of the different m-distance partitions that can be formed for different values of m, for a 36-node circulant graph with the offsets shown bellow.

m	S(36,m)	Partition of {1, 4, 6, 9, 11, 15, 16}
1	<1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18>	{1}, {4}, {6}, {9}, {11}, {15,16}
5	<5,10,15,16,11,6,1,4,9,14,17,12,7,2,3,8,13,18>	{15,16,11,6,1,4,9}
7	<7,14,15,8,1,6,13,16,9,2,5,12,17,10,3,4,11,18>	{1,6}, {4, 11}, {16,9}, {15}
11	<11,14,3,8,17,6,5,16,9,2,13,12,1,10,15,4,7,18>	{1}, {6}, {11}, {16,9}, {15,4}
13	<13,10,3,16,7,6,17,4,9,14,1,12,11,2,15,8,5,18>	{1}, {6}, {4,9}, {11}, {15}, {16}
17	<17,2,15,4,13,6,11,8,9,10,7,12,5,14,3,16,1,18>	{16,1}, {15,4}, {6,11}, {9}

Important Definitions

Block Graph $BL(G(n,m_i,P_{mi}))$

- Formed by multiplying the inverse of m_i (mod n) by each of the maximal m-distance subsets (in P_{mi})
- Example: $BL(G(23,5,P_5 = \{\{3,8,10\}\}))$ = a 23-node circulant graph with offsets {2,3,4}
- Since the transformation is bi-directional, n ft-extension of the block graph is also an Ft-extension of the original graph.
- The original theorem can now be used to effeciently construct an optimal k-ft extension.

Fault Tolerance in Circulant Graphs

Algorithm Fault-tolerance(G,k) {

1. Generate all integers {m1,m2,..,mj} such that for all mj, we have gcd(n,mj)=1 and $1 \leq mj < (n/2)$;
2. For each mj generated above, find the corresponding partition of the offsets using Procedure Partition(A, n, mj) given before. The graph corresponding to this partition is denoted $G_{mj}[a_1,a_2,..,a_i: n]$;
3. For each graph $G_{mj}[a_1,a_2,..,a_i: n]$ generated above, construct its corresponding block graph $BL(G_{mj}[a_1,a_2,..,a_i: n])$ as described earlier;
4. For each block graph $BL(G_{mj}[a_1,a_2,..,a_i: n])$, use Theorem 2.1 to construct its k-ft solution;
5. Compare all k-ft solutions constructed in (4), and select the one with the least node-degree;

}

time complexity upper bound $O(n^2 \log |A| + n\, k\, |A|)$

Mesh Applicable

- n*n mesh can be embed into an n^2-node circulant graph with offsets {1,n}
- k-ft extension for a circulant graph embedding an n*n mesh would have at most k+2 offsets
- n*n*n mesh can be embed into an n^3-node circulant graph with offsets $\{1,n,n^2\}$
- k-ft extension with at most, 2k+3 offsets if $k \le n-2$ and n+k+1 offsets if $k > n-2$

[4]

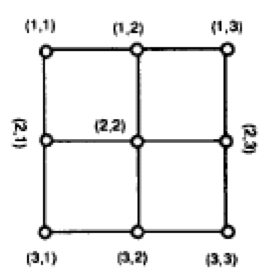


Fig. 5.1 A mesh M[3,3].

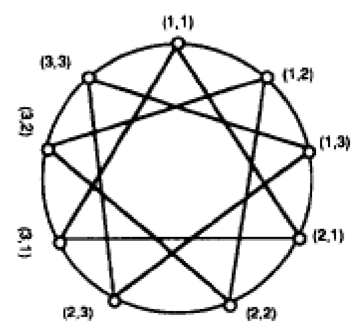


Fig. 5.2 A circulant graph embedding the mesh M[3,3].

Hypercube applicable

- a q ($q \geq 2$) dimension hypercube can be embed into a circulant graph $G[1, 2^1, 2^2, ..., 2^{q-2}; 2^q]$.

q ?	k=4	k=8	k=16	k=32
The costs of k-ft for a q-dimensional hypercube				
q=5	(09,12)	(13,17)	(17,24)	(25,32)
q=6	(13,15)	(17,22)	(25,33)	(33,48)
q=7	(18,18)	(25,27)	(33,42)	(49,65)
q=8	(23,21)	(33,32)	(49,51)	(55,82)
q=9	(28,24)	(43,37)	(66,60)	(97,99)

Approach compared to the one to be discussed in the following slides [5]

Reconfiguring Circulant Graphs & Hypercubes

- The graphs produced via the algorithm can be reconfigured with an upper bound time complexity of O((n+k) |A| log |A|).
- In the hypercube & mesh reconfiguration, the mapping from original structure to the circulant graph has to be reversed.

Algorithm Reconfigure_C2G (C, G, m, S) {

1. Exclude every faulty node and its adjacent edges from C;
2. Renumber the nodes in the same order along the cycle as before reconfiguration, starting at any healthy node as 0 and skipping every faulty node, i.e., after renumbering, the healthy nodes will be labeled as 0,1,.., n-1;
3. Exclude every edge (X,Y) as non-relevant if it satisfies the following condition: $X-Y \neq \pm a_i * m^{-1}$ (mod n) for all offsets a_i in G;
4. Form a subgraph equal to G by multiplying every healthy node's number by m (mod n);
5. Return the subgraph formed above;

}

Algorithm Reconfigure-C2H (C,G,H,m,S) {

1. Q = Reconfigure-C2G (C,G,m,S); /* return a subgraph equal to G */
2. For every node X in Q, renumber X (using hypercube notation) as follows:
   - 2.1 If $X < 2^{q-2}$, assign X a q-bit binary number equal to X;
   - 2.2 If $2^{q-2} \leq X < 2^{q-1}$, assign X a q-bit binary number equal to X;
   - 2.3 If $2^{q-1} \leq X < 2^{q-2}+2^{q-1}$, assign X a q-bit binary number equal to $X+2^{q-2}$;
   - 2.4 If $2^{q-2}+2^{q-1} \leq X < 2^{q}$ assign X a q-bit binary number equal to $X-2^{q-2}$;
3. Exclude (or ignore) every edge (X,Y) if X and Y differs in more that one corresponding bit; /* the rest of edges are those of hypercube */
4. Return the subgraph formed above;

}

Another approach to Fault Tolerant Meshes and Hypercubes

- Use same basic theorem to create FT-extensions
- In the mesh case, analyze mesh relabeling approaches.

[6]

0	1	2	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	35	36	37	38	39

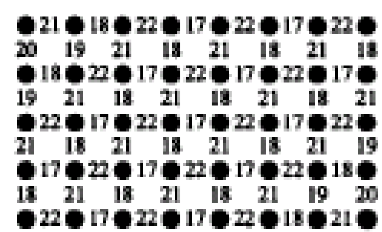
(a)

0	9	18	27	36	5	14	23
8	17	26	35	4	13	22	31
16	25	34	3	12	21	30	39
24	33	2	11	20	29	38	7
32	1	10	19	28	37	6	15

(b)

0	21	3	25	8	30	13	35
20	2	24	7	29	12	34	17
1	23	6	28	11	33	16	38
22	5	27	10	32	15	37	19
4	26	9	31	14	36	18	39

(c)



(d)

Fig. 2. Three orderings of mesh nodes.

Mesh Construction 1 [6]

- The first method uses the basic approach for embedding a mesh in a circulant graph to achieve the least optimal node degree. This node degree is proven to be at most 4k+4 (or 2k+2 offsets).

0	1	2	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	35	36	37	38	39

Mesh Construction 2 [6]

- anti diagonal numbering of the mesh is used to lead to a circulant graph with 2 offsets that are consecutive
- Achieving an ft-extension with at most 2k+4 node-degree (equivalent to the k+2 offsets achieved using an m-distance partition = A with a block graph translation)

0	9	18	27	36	5	14	23
8	17	26	35	4	13	22	31
16	25	34	3	12	21	30	39
24	33	2	11	20	29	38	7
32	1	10	19	28	37	6	15

Mesh Construction 3 [6]

- Interleaved anti-diagonal major ordering
- leads to a circulant graph with offsets clustered around the value rc/2 for an r*c mesh

Lemma:

$Let\ r, c \in \mathbb{N}$

$Let\ a = \frac{rc}{2} - \frac{r}{2} \qquad Let\ b = \frac{rc}{2} + \frac{r}{2}$

$Let\ S = \mathbb{N} \cap [a, b]$

The mesh $M_{r,c}$ is a subgraph of the r*c node circulant graph with offsets S.

Mesh Construction 4 [6]

- The final approach discussed in [6] mixes both approaches two and three

Upper bounds for node degree using this approach

Then the circulant graph $C_{rc+k,T}$ is $(k, M_{r,c})$-tolerant and has degree at most

$$d(k,r,c) = \begin{cases} 2k+4, & r \text{ is odd and } k \leq c-3, \\ c+k+1, & r \text{ is odd and } k > c-3, \\ 2k+4, & r \bmod 4 = 0 \text{ and } k \leq 2c-3, \\ 2c+k+1, & r \bmod 4 = 0 \text{ and } k > 2c-3, \\ 2k+4, & r \bmod 4 = 2 \text{ and } k \leq 4c-3, \\ 4c+k+1, & r \bmod 4 = 2 \text{ and } k > 4c-3. \end{cases}$$

Ft in d-dimensional meshes and hypercubes

- Embed into diagonal graphs
- Use a formula to transform diagonal graphs to Ft-extended circulant graph
- Ft-extension of a d-dimentional mesh has a degree at most (k+2)d if k is even and at most (k+1)d if k is odd
- Specific Cube example:

  1-Ft d-dimension hypercube has 2^{d+1}-node circulant graph with offsets $\{1, 2, 4, ..., 2^{d-1}\}$

Conclusion

Several approaches to fault tolerance have been discussed in the paper. The approaches use different techniques to achieve a semi-optimal ft-extension for their graphs. The paper corresponding to this talk includes more detail on the workings of the algorithms.

Thank You

Questions?

References:

[1] C. S. Raghvendra, A. Avizienis, and M. D. Ercegovac, “Fault tolerance in binary tree architectures,” IEEE Trans. Comput., pp.568-572,June 1984

[2] S. Dutt and J. P. Hayes, “On Designing and Reconfiguring k-Fault Tolerant Tree Architectures,” IEEE transactions on computers, Vol.39. No. 4. April 1990

[3] S. Dutt and J. P. Hayes, “On Designing and Reconfiguring strategies for Near -Optimal k-Fault Tolerant Tree Architectures,” IEEE Proceedings of FTCS-25, Vol-III 1996

[4] A. Farrag, "Algorithm for Constructing Fault-Tolerant Solutions of the Circulant Graph Configuration", *in Proc. of 5th IEEE Symp. On Frontiers of Massively Parallel Computations*, Virginia, Feb. 1995, pp. 514-520.

[5] A. Farrag, S. Lou, “Designing and Reconfiguring Fault-Tolerant Hypercubes”, HPCS'06.

[6] J. Bruck, R. Cypher and C. Ho, "Fault-Tolerant Meshes and Hypercubes with Minimal Number of Spares", *IEEE Trans on Comp*, Sept 1992, V. 42, N. 9, pp. 1089-1103.