Approaches to form fault tolerant graphs for Trees, Meshes and Hypercubes

Abstract:

Graphs form the basis of many topologies used in Compute Science and many of them require Fault tolerant extensions for many applications. This paper aims to show different approaches to creating fault tolerant trees, meshes and hypercubes. Circulant graphs are explained to aid in some of the approaches along with the mention of diagonal graphs. The similar approaches are compared for complexity of the solution and the path to the solution.

Hatem Nassrat B00393388

Tarak Shingne B00467792

Introduction

Fault tolerant describes a computer system or component designed so that, in the event that a component fails, a backup component or procedure can immediately takes its place with no loss of service. As for example a real time system is a system in which computations must be completed within specified deadlines. In hard real time systems, missing a deadline can be catastrophic. Such kind of systems may be based on a single processor or multiple processors. A multiple processor system may be tightly coupled i.e. the processors are connected together by a high-speed bus, possibly sharing memory and typically physically located within a few racks or cabinets. A failure in one or more processors can degrade the performance of the system and can lead to the whole operation of a system to massive failure. In order to keep running the system intact, even in the presence of failure, backup processors should be used. The designing of the fault tolerant system should be in such a way that failure does not hinder the operation of the system and it keeps running with the same efficiency or slightly degrading efficiency. The design techniques should take consideration of the cost associated with every additional component into a system. Also, a system should be self-recovering from the failure.

We have covered different techniques for the fault tolerance of graphs like trees, hypercubes and meshes. Also reconfiguration is the vital part in design of such systems; we have not covered reconfiguration in detail. But the details of the reconfiguration can be found from the reference list.

Fault-Tolerant Tree Architecture

A tree is commonly used interconnection structure in computing systems. The nodes of the tree corresponds to computing facilities (computers, processors, or I/O devices) while the edges represent the communication links between the facilities. For some application like real time application it is very necessary that tree structure be maintained in the presence of faults. Spare nodes and edges should be added to the basic tree structure in order to recover from the fault if any occurs. In this way nonfaulty nodes can be reconfigured into a graph isomorphic to the original tree. Some of the definitions and notations which can be useful in understanding the terms are as follows:[1]

Symmetric tree: A symmetric tree is a rooted tree in which every interior node has the same number of children and all leaf nodes are equidistant from the root [2].

Nonhomogeneous tree: A nonhomogeneous tree is one in which nodes in different levels are of different types i.e. they represent different kinds of computing facilities, while all nodes within a same level are of the same type [2].

Homogeneous tree: A homogeneous tree is one in which all the nodes are of the same type.
Node covering: A node x_{iu} is said to completely cover x_{iv} if all the child of x_{iv} are covered by x_{iu}. That is there are edges from x_{iu} to al the children of x_{iv} (explained later) [2].

V (G) will denote the vertex or node of a graph G, and E (G) denotes the set of edges of G if G is an undirected graph. Also, A (G) denotes a set of edges (arcs) of G if G is a directed graph.
A graph that contains a given graph H as a subgraph is denoted by G (k,H). G [k,H] is a k-fault tolerant realization of H if, for any set o F of k nodes in G, the graph induced by V(G)-F contains a subgraph isomorphic to H. A d-ary, l-level NST (Non-homogeneous Symmetric Trees) is denoted by $T_N(d,l)$, where d,l>1.The root node of $T_N(d,l)$ is at level 0, and the leaf node at level l-1. The number of nodes at level i of $T_N(d,l)$ is $n_i = d^n$. The set of original nodes in level I of $T_N(d,l)$ is denoted by O_i. In G [k, $T_N(d,l)$] there is a set S_l of spare nodes for each level I of $T_N(d,l)$ and
$X_i \cup O_i$ .The nodes in S_i are indexed from 0 to (k_i -1) where k_i = mod of S_l and the nodes in O_i are indexed from 1 to n_i. For a tree T, the non faulty subgraph serving as the tree isomorphic to T in G [k,T] is called current tree and the nodes of the current tree are called active nodes. Also the edges of the current tree are called active edges. The nodes and the edges which are not in the current tree are considered as (redundant) spares [2].

We have analyzed the approach of designing of K- fault tolerant trees from the beginning. First we have considered the basic paper of designing of 1-ft and then we after analyzing it with respect to the number of factors such as number of required spare nodes, number of redundant edges and drawbacks we switched to the alternative design of the 1-ft which removes the said drawbacks. And after the review of the improved design of 1-ft we have focused n the generalization of k-ft design.

1-FT design [1]
The following figure is the design of the 1-FT which is 3 level designs and as we can see from the figure it is 4 degree design. Each node has 4 degree.

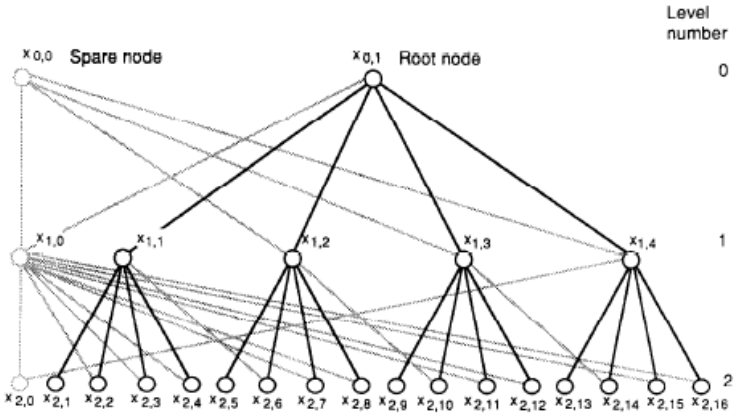

Figure 1[1]

In this design, the redundant edges are placed as follows. The single spare node S_i where i indicates, the level has an edge to each of the rightmost d-2 children of each of the n_i nodes in O_i as well as to the nodes in S_{i+1}. Each node j in X_i, $0 \leq j < n_i$, has an edge to the second child of node j+1, while n_i has an edge to the node in S_{i+1}. The above mentioned design achieves the strict optimality but it also has following drawbacks.

1) In the above design, there is a severe imbalance of node degrees and this can be considered as a vital problem as nodes are costly to implement.
2) Also another problem is when a node $x_{i\,j}$ fails then reconfiguration has to take place in

levels i down to l-1, thus disrupting normal processing in non faulty levels (We have not covered the concept of reconfiguration in our paper, but the proof can be found from the reference).

3) The problem with this design is also that it can only tolerant one faulty node, thus node utilization id less than 100 %.

New improved 1-FT design [1]

The solution of the above drawbacks is considered in this design, but the strict optimality in the number of redundant edges for a balanced node degree is sacrificed. This design also gives the simpler reconfiguration process. The following figure gives the improved design of 1-FT.

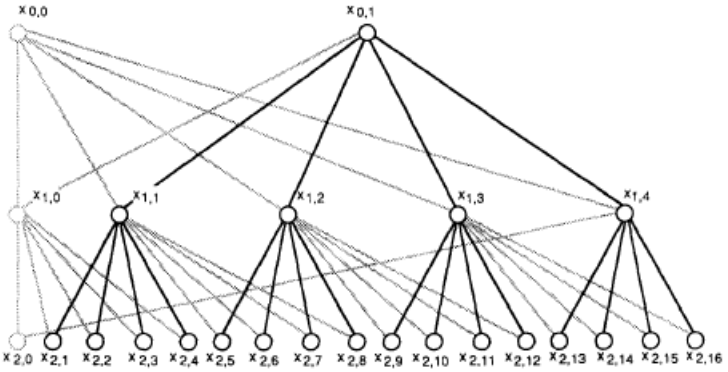

Figure 2[1]

From the above figure it is observed that each node j in X_i has redundant edges to all d children of node j+1,that is j covers j+1, for $0 \leq j < n_l$. Node n_i has an edge to the spare node in X_{i+1} and spares in neighboring levels have edges between them. Reconfiguration here requires each node $X_{i\ h}$ to be connected to the d children of node $X_{i\ h+1}$ for h<j, when node $X_{i\ j}$ fails. This improved design avoids the shortcomings of the previous 1-FT design in the following way:

1) Here, it can be seen from the figure, node degree here is much better balanced.
2) For any fault in any level i, the reconfiguration is confined to levels i-1, i and i+1.
3) The node utilization is 100% as one faulty node is tolerated in each level.

The paper considers the design of non homogeneous symmetric trees. Some of the factors like number of spare nodes, number of spare edges, node degree plays an important role in many applications and their associated costs increases as they grows in number. So in order to design K-FT solution, the above factors should be considered and design is considered near optimal if the minimum number of edges, node degrees and spare nodes are used.

For designing K-FT solution, concept of node covering plays n important role. A covering graph structure is one in which each spare node covers al the nodes in O_i . Another possibility is to have each spare node cover a small number of nodes in O_i, which in turn cover a small number other nodes in O_i, and so on until all the nodes in O_i are covered .Two different designs for constructing K-FT are given and they are as follows:

1) K-FT design (k<d)
2) K-FT design (k≥d)

The above designs are based on theorems and lemmas which are stated and provided with proof in the original paper. We are just mentioning the statement of those theorems and lemmas.

The first theorem concerns with the minimum number of nodes necessary to cover any set of nodes of a specific cardinality to ensure k- fault tolerance.
Theorem-1: In any K-FT NST G [k,T_N(d,l)], every set of $\lfloor$**k/d**$\rfloor$ **+1** nodes in O_i has to be covered by at least **k-** $\lfloor$**k/d**$\rfloor$ other nodes in X_i for reconfiguration around any k or fewer faults.
Theorem-2: If each node v in O_i 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 .
Note that the concept of covering sequence is given in [1].

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)].

The design of K-FT (k<d) is based on Theorem 1, Theorem 2 and Lemma 1.The following figure summarizes the design.

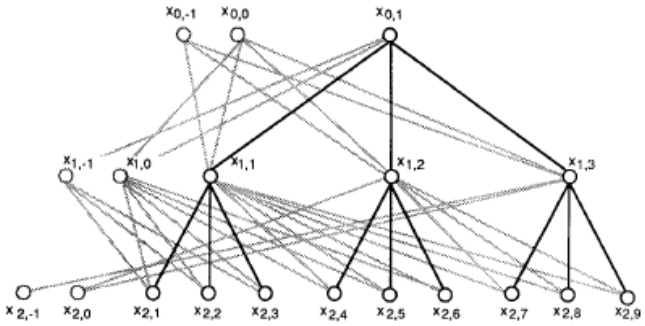

Figure 3[1]

From the above figure it is clear that each node in O_i is covered by exactly k nodes and the covering graph for each level is acyclic-note the concept of covering is covered in paper[1] in detail. The number of redundant edges incident on the nodes of X_i for i>0 is n_{i-1}kd+ n_i kd+2(k(k+1)/2= n_ik(d+1)+k(k+1).The number of non redundant edges incident on X_i is n_i+ n_{i+1}= n_i(d+1).The edge redundancy defined as the ratio of redundant to non redundant edges for X_i is thus (n_ik(d+1)+k(k+1)/ n_i(d+1))=K+K(K+1)/ n_i(d+1) which is equivalent to k. Those nodes in X_i that cover k nodes, and whose parents in X_{i-1} are covered by k nodes, will have a maximum node degree for redundant edges of k (d+1). Also, since there are n_i +k nodes in X_i, if the redundant edges incident on X_i are evenly distributed among all the nodes, node degree of (n_ik(d+1)+k(k+1)/(n_i+k) is obtained. For n>> k, this node degree is approximately k (d+1).Therefore, the maximum node degree in X_i asymptotically approaches the least possible maximum node degree as i increases. So, this design is optimal with respect to the number of spare nodes and edges, and is asymptotically optimal with respect to the maximum node degree.

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 the number of spare nodes and edges.[1]

The design of K-FT (k≥d) is based on the above Theorem 3. The following figure summarizes

the design.

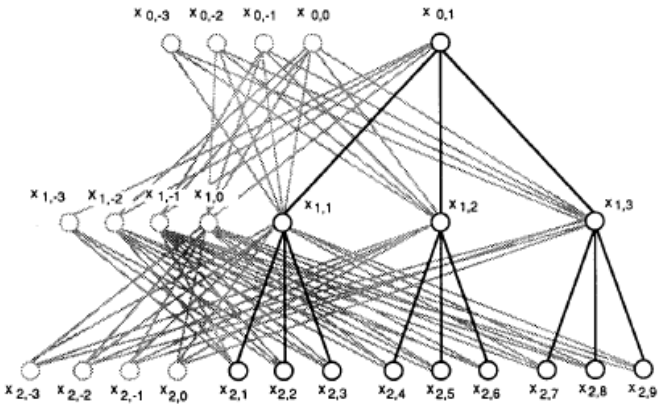

Figure 4[1]

In the above design every node O_i is covered by exactly **k-2 ⌊k/d⌋+1** other nodes in X_i. The following properties can be deduced from this design.

1) The rightmost **2 ⌊k/d⌋** nodes in X_i do not cover any other node in X_i as shown in figure x_{12} and x_{13} in level 1 do not cover any node.
2) Each node (n_i-h) where $2\lfloor k/d \rfloor \le h < k$, in X_i covers exactly $h-2\lfloor k/d \rfloor+1$ other nodes. As shown in figure x_{11} covers x_{13} and x_{10} covers x_{12} and x_{13}.
3) Each node $-(j-1)$ where $j>2\lfloor k/d \rfloor$, covers $k-j+1$ other nodes. From the figure, we can see that x_{1-2} covers two nodes x_{11} and x_{12}, while x_{1-3} covers one node x_{11}.
4) All the other nodes in X_i cover exactly $k-2\lfloor k/d \rfloor+1$ node. Thus in the above figure, x_{1-1} covers three nodes.

Edge redundancy for each level i is approximately k, and that the maximum node degree as n_i becomes large. For the above design, following lemma specifies the minimum number of covers for any subset of O_i.

Lemma 2: In $G[k,T_N(d,l)]$, any subset F of f nodes on O_i is covered by at least k-2 □k/d⌋+f other nodes in X_i when $1 \le f \le \lfloor k/d \rfloor$ and by at least k- ⌊k/⌋d other nodes when f>⌊k/d.⌋

Fault tolerance and reconfiguration of circulant graphs with application in meshes and hypercubes

Circulant graphs have certain properties that allow for good fault tolerant schemes. Circulant graphs also allow for embedding other graph types allowing them to also use the same approach in finding their fault tolerant extensions. This segment of the paper aims to discuss a fault tolerant approach proposed in a paper published in 1995 by Dr. Farrag [4] with its application in meshes [4] and hypercubes [5]. Also to look at a fault tolerant approach discussed in Bruck et al. [6] for both meshes and hypercubes.

Fault tolerant approach for circulant graphs

To explain the fault tolerant approach, certain concepts have to be defined:

Definition 1: **Circulant Graphs**

> "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]

Thereom 2.1 as stated in [4], explains that an n-node circulant graph G with a set of offsets A={a_l, a_2, ..., a_i,} has a k-ft extension H, with n+k nodes and offsets {a_l, a_l+1, ..., a_l+k} ∪ {a_l, a_2+1, ..., a_2+k} ∪ ... ∪ {a_i, a_i+1, ..., a_i+k}. Using this theorem one can create an efficient k-ft extension when all offsets in an arbitrary graph G are in consecutive order. When the offsets are spaced apart, it is apparent that the size of the set of offsets A, denoted as |A| would be least optimal. The following definitions will aid in forming a procedure that can transform an arbitrary circulant graph G into one with the largest set of consecutive offsets.

Definition 2: **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]

Definition 3: **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 an m-distance subset is defined as follows:

> 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}.

The maximal m-distance subsets explained in the above example are used to create

an **m-distance partition** written as P(A,n,m).

Definition 4: **P(A,n,m)** is defined as $\{x\,;\, x \in$ set of maximal m-distance subsets for a given A,n,m $\}$

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

Illustrated in [4] and [5] is the algorithm shown in figure A1. This procedure can be performed in O(|A| n). Recognizing that m is an integer relatively prime to n, and that the sequences S(n,m) and S(n,n-m) are the same, all the m-distance partitions for applicable values of m can be calculated in polynomial time with a loose upper bound 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;
}

Figure A1: Procedure partition [5]

Figure A2 shows the sequences and partitions for a 36-node circulant graph with offsets {1, 4, 6, 9, 11, 15, 16}. To specify one of these instances, the definition of the notion G(n,m_i,P_{mi}) for a given n-node circulant graph G with m_i belonging to a set of possible m's and P_{mi}=the corresponding offset partition. This graph can be transformed to its block form noted as **BL(G(n,m_i,P_{mi}))**.

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}

Fig A2: S(36,m) and corresponding m-partitions for different m's [4]

This block graph is formed by multiplying the inverse of m_i (mod n) by each of the maximal m-distance subsets (in P_{mi}) to transform them into a block of consecutive integers. 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, an Ft-extension of the block graph is also an Ft-extension of the original graph. The theorem explained previously (Theorem 2.1) can now be used to find a superiorly optimal Ft-extension. The full algorithm explained in [4] & [5] can be seen in Fig A3. And has a time complexity upper bound O(n^2 log |A| + n k |A|).

Algorithm Fault-tolerance(G,k) {
1. Generate all integers {m1,m2,..,mj} such that for all mj, we have gcd(n,mj)=1 and $1 \le mj < (n/2)$;
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;
}

Fig A3: Ft-Algorithm using above notions [5]

Application in meshes:

Shown in [4], an n*n mesh can be embed into an n^2-node circulant graph with offsets {1,n}. Similarly an n*n*n mesh can be embed into an n^3-node circulant graph with offsets {1,n,n^2}. After embedding into the circulant graph, one can use the previous algorithm and apply it to the mesh. It is proven that the k-ft extension for a circulant graph embedding an n*n mesh would have at most k+2 offsets. Similarly a graph embedding an n*n*n mesh has a k-ft extension (found using the algorithm) with at most 2k+3 offsets if k≤n-2 and n+k+1 offsets if k>n-2.

Application in Hypercubes:

Hypercubes like meshes can also be embed into circulant graphs. Doing so allows us to use the previous algorithm in finding the k-ft extension of the hypercube. In [5] it is proven that a q (q≥2) dimension hypercube can be embed into a circulant graph G[1, 2^1, 2^2, ..., 2^{q-2}: 2^q]. This approach has been compared to the approach used in [6]. The table in Fig A4 compares the optimality of the solutions found using both methods. From the table the method described in [5] produces more optimal solutions than the method used in [6].

q ?	The costs of k-ft for a q-dimensional hypercube k=4	k=8	k=16	k=32
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)

Fig A4: Entries if the form (x,y). Where x is the number of offsets in the solution of [5] and y belongs to [6]

Circulant Graph Reconfiguration:

A graph produced by the Ft-approach in [4] & [5] can be reconfigured using the algorithm described in Fig A5. This algorithm has an upper bound time complexity of O((n+k) |A| log |A|). To reconfigure a graph mapped from a hypercube, one has to note the transformation that occurred in mapping the hypercube to the circulant graph. This will allow mapping back to the hypercube after the circulant graph reconfiguration algorithm is run. This approach is illustrated in the algorithm seen in Fig A6 and can also be adapted to reconfigure meshes.

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: $X-Y \neq \pm a_i * m^{-1} \pmod n$ 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;

}

Fig A5: Circulant graph reconfiguration algorithm [5]

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;

}

Fig A6: Algo to reconfigure hypercubes [5]

Another approach to fault tolerant Meshes and Hypercubes

The paper [6] compared to [5] in the previous section uses a different approach for creating Ft-extensions for meshes and hypercubes. In this paper the authors use the basic formula for creating Ft-extensions however do not renumber the nodes in the circulant graph (as the block graphs do in [5]) but modify the numbering order of the mesh. Bellow are 4 different methods to renumber meshes along with an Ft-approach for creating fault tolerant hypercubes that are discussed in the paper [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).

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

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

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

Fig A7: Mesh upper bound using 4th approach [6]

The second approach requires 2 additional edges per node for each additional fault that is tolerated. The third approach only requires one additional edge per node, However requires a larger initial node degree. This construction is achieved by applying an interleaved anti-diagonal major ordering to the mesh. This ordering (described in [6]) leads to a circulant graph with offsets clustered around the value rc/2 for an r*c mesh. The lemma shown bellow is shown and proven in [6]. This approach allows for an Ft-extension of the mesh that has degree at most K+r+1 when r is odd and c is even, and at most k+r otherwise.

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.

The final approach discussed in [6] is a hybrid approach that mixes both approach two and three. Using this approach the upper bound illustrated in Fig A7 can be achieved.

Bruck et al. also discussed the fault tolerant application of d-dimensional meshes and hypercubes through embedding them into diagonal graphs, then using a theorem that finds an Ft-extension of the diagonal graph. This theorem transforms the diagonal graph G into an Ft-extension H that is a circulant graph. This theorem, shown in the figure bellow, is proven in [6] and states that the 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. Through this theorem a specific example for a d-dimensional hypercube and 1-Ft extension is shown. This grah is a 2^{d+1}-node circulant graph with offsets {1, 2, 4, ..., 2^{d-1}}. The general approach has been compared to the approach proposed in [5] within [5] and is shown to provide less optimal solutions for larger hypercubes.

Conclusion

Several approaches to fault tolerance have been discussed in this paper. The approaches use different techniques to achieve a semi-optimal ft-extension for their graphs. These techniques are difficult to compare, and were compared according to the upper bound time complexity of their algorithms, with focus on the optimality of the solutions returned by the algorithms.

References:

[1] 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

[2] 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.