Fault Tolerant Networks with Small Degree

Li Zhang[*]
Department of Computer Science
Stanford University
lizhang@cs.stanford.edu

ABSTRACT

In this paper, we study the design of fault tolerant networks for arrays and meshes by adding redundant nodes and edges. For a target graph G (linear array or mesh in this paper), a graph G' is called a k-fault-tolerant graph of G if when we remove any k nodes from G', it still contains a subgraph isomorphic to G. The major quality measures for a fault-tolerant graph are the number of spare nodes it uses and the maximum degree it has. The degree is particularly important in practice as it poses constraints on the scalability of the system. In this paper, we aim at designing fault-tolerant graphs with both small degree and small number of spare nodes. The graphs we obtain have degree $O(1)$ for arrays and $O(\log^3 k)$ for meshes. The number of spare nodes used are $O(k \log^2 k)$ and $O(k^2/\log k)$, respectively. Compared to the previous results, the number of spare nodes used in our construction has one fewer linear factor in k.

1. INTRODUCTION

In many parallel computer systems, the processors are connected by *interconnection networks*. Such networks usually have regular topology and small degree to allow scalability. Popular instances of interconnection networks include arrays, meshes, trees, and hype-cubes [8]. A central issue in the design of interconnection networks is *fault-tolerance* as it is essential for a large parallel system to work properly even when some processors fail. Compared to bus-based systems, the fault-tolerance of interconnection networks is more intrigue as the failure of one processor may affect the communication of other processors; for example, in a linear array, the failure of one processor disconnects the array into two connected components.

In literatures, there are mainly two approaches to achieving fault-tolerance. In one approach, the faulty processors are simulated by non-faulty (or *healthy*) processors [6, 7, 8, 9]. This way, we do not add any hardware redundancy but allow slow-down of the system when failures happen. The other approach is to add hardware redundancy, *spare* nodes or edges, so that the system can be reconfigured to simulate the desired topology to maintain the full performance with the presence of failures [1, 2, 4, 10, 11]. In this paper, we study the design of fault-tolerant networks for arrays and meshes in the second approach.

The latter approach can be conveniently described as a graph problem [10, 11]. Suppose that G is a graph which represents the topology of an interconnection network. We say that another graph G' is a k-fault-tolerant graph (or $\mathrm{FT}(G,k)$) of the *target graph* G if when we delete any k nodes (and incident edges) from G', it still contains a subgraph isomorphic to G. In the following, we use $v(G)$ to denote the number of nodes in G and $d(G)$ the maximum degree of G. The most important quality measures of fault tolerant networks are the number of spare nodes and edges added.

In [5], Erdös *et al.* study the design of fault-tolerant graphs for directed linear arrays. They gave probabilistic construction of directed acyclic graph with $O(n \log n)$ edges to tolerate n faults for a directed linear array with n nodes. Also, they prove a lower bound of $\Omega(n \log n/\operatorname{loglog} n)$ on the number of edges for such graphs. In [2], Alon *et al.* construct fault tolerant graphs for (undirected) linear arrays. In [1], a general method based on probabilistic argument is presented to design fault-tolerant graphs for any graph G. The construction uses k spare nodes, but the degree of the constructed graph is $O(kd(G))$. In the same paper, explicit constructions are given only for small k's. In their constructions, the degree of the graph is linear in k and can be prohibitively high in practice — we usually require the degree of the graphs be small to allow scalability. This problem is addressed in [4] in which the authors consider the design of fault-tolerant graphs with small degree for arrays and meshes. The results of their paper are summarized in Table 1. In the table, L_n denotes the linear array with n nodes, and M_n denotes the $n \times n$ mesh.

Although the fault tolerant networks built in [4] have small degree, the numbers of spare nodes needed are high. In this paper, we show how to improve the number of spare nodes for fault tolerant graphs of arrays and meshes while keeping the degree small. Our results are summarized in Table 1 as well. Compared to the construction in [4], our methods use

[*]Supported in part by National Science Foundation grant CCR–9623851 and by US Army MURI grant DAAH04-96-1-0007.

SPAA 2000, Bar Harbor, Maine USA
© ACM 2000 1-58113-185-2/00/07...$5.00

	$\mathrm{FT}(L_n, k)$		$\mathrm{FT}(M_n, k)$	
	spare nodes	degree	spare nodes	degree
[4]	$O(k^2)$	$O(1)$	$O(k^3)$	$O(1)$
			$O(k^{5/2})$	$O(\sqrt{k})$
This paper	$O(k \log^2 k)$	$O(1)$	$O(k^2/\log k)$	$O(\log^3 k)$

Table 1. Summary of the results.

smaller number of spare nodes.

For arrays, we first use a recursive construction to obtain fault-tolerant graphs with $O(k \log k)$ spare nodes and $O(\log k)$ degree. Following the same idea as in [4], we can reduce the degree to $O(1)$ by increasing the number of spare nodes to $O(k \log^2 k)$ by using a gadget called *Moon graph*. A crucial property of the Moon graphs is that there is a Hamiltonian path between any two distinct nodes in the graph. The fault-tolerant graphs for arrays gives us a straight-forward way to construct fault-tolerant meshes with $O(k \log k)$ spare nodes and maximum degree $O(k \log k)$. By using Moon graphs, we can reduce the degree of the graph to $O(\sqrt{k} \log k)$ while using $O(k^{3/2} \log n)$ spare nodes. To reduce the degree further, we design a new gadget called *tower graphs*, with the structure similar to the classical *finger trees*. By exploiting and utilizing certain properties of tower graphs, we are able to reduce the degree of the fault-tolerant graphs for meshes to $O(\log^3 k)$ by using $O(k^2/\log k)$ spare nodes.

In all the previous work, it is assumed that the number of faults is bounded, and the number of nodes of the target network is also fixed. In this approach, the spare nodes are really "spare" as the size of simulated network is fixed regardless of the number of faults to tolerate. It is however more appropriate to assume that the number of nodes in the original graph is fixed and to ask what is the largest size of graphs with wanted topology can be realized in the graph with node failures. An implication of our construction is that we can construct directed acyclic graphs with size n and degree $O(\log k)$ so that if any f ($f = O(k/\log k)$) nodes fail, there always exists a directed path with size $n - O(f \log f)$. This is related to the work on fault-tolerant data structures [3]. In this paper, we always perform worst-case analysis, i.e. the graphs constructed are guaranteed to contain the target graph if at most k nodes fail. There are also researches ([4, 12, 13]) on considering randomly distributed faults, in which case it suffices to guarantee that a target graph exists with high probability. Under such model, it is usually possible to obtain graphs with smaller number of spare nodes or edges.

The paper is organized as follows. In Section 2, we will introduce some notations and useful facts. We then present the construction for linear arrays in Section 3 and for meshes in Section 4. The main results are stated in Theorem 3.3 and 4.3.

2. PRELIMINARIES

For a graph G, let $v(G), d(G)$ denote the number of vertices and maximum degree of G, respectively. A graph G' is a k-fault-tolerant (or $\mathrm{FT}(G, K)$) graph of G if when we remove any k nodes from G', it still contains an isomorphic copy of G. The isomorphic copy of G is called a *realization* of G.

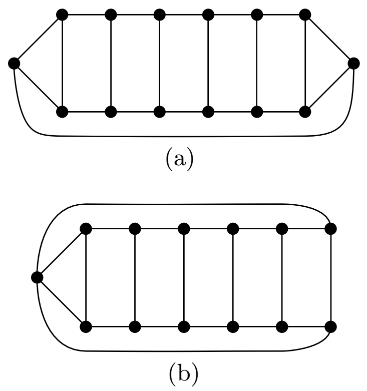


Figure 1. The Moon graphs of (a) even and (b) odd number of vertices.

When G is clear from context, we simply say that G' tolerates k faults. We call those nodes that have to be removed *faulty* nodes and others *healthy* nodes.

In the following, we denote by L_n and $\hat{L}_n$ the undirected and directed linear arrays with n nodes, respectively, and by $M_{r,c}$ the $r \times c$ mesh. When $r = c$, it is simply written as M_r.

Following the notations in [4], for a natural number n and an integer set T, define $C(n, T)$ to be the undirected graph with the vertices indexed $\{0, 1, \cdots, n-1\}$ and the edge set $\{(i, i+s) \mid s \in T \text{ and } 0 \leq i, i+s < n\}$. The directed graph $\hat{C}(n, T)$ is defined similarly except that each undirected edge $(i, i+s)$ is replaced by the directed edge $\langle i, i+s \rangle$.

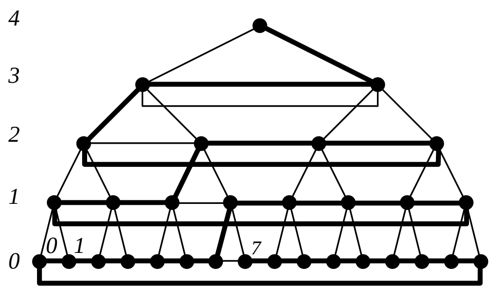
As mentioned in the introduction, we will use some gadgets to reduce the graph degree in our constructions. The gadgets we use are the *Moon graphs* and *tower graphs*.

The Moon graph [4] with s nodes, denoted by K_s, is a constant degree graph so that for any pair of distinct nodes in K_s, there exists a Hamiltonian path connecting those two nodes. The constructions are shown in Figure 1. It is easy to verify that for any two nodes in K_s, there is a Hamiltonian path ending at them.

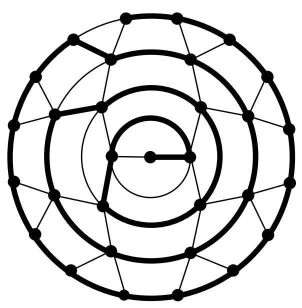
The tower graph Q_s, where $s = 2^t$ for some non-negative integer t, is a perfectly balanced binary tree on s leaves with all the nodes on the same level connected[1] (Figure 2 (a)). Clearly, $v(Q_s) = 2s - 1$ and $d(Q_s) = 5$. Figure 2 (b) shows a planar layout of Q_s.

Each node in the tower graph can be indexed by (l, p) where l is its level and p is its position in that level. The levels are indexed starting from 0 at the leaf level and increase bottom up. On each level, the position of nodes are indexed from 0 to $s/2^l - 1$ and increase from left to right (Figure 2(a)). In the tower graph, we can form a Hamiltonian path $H(i)$ from each leaf node $(0, i)$ to the root level by level. On each level, we trace the loop until we reach the node that precedes the beginning node. Then, we follow the tree edge to go to the upper level and repeat the process until getting to the root

[1]In general, such graphs are called finger trees.



(a)



(b)

Figure 2. The tower graph Q_{16} drawn as (a) a finger tree, and (b) a planar layout. The Hamiltonian paths from a leaf node to the root are shown in thickened lines. In (a), we also show the indexing of the levels and some nodes.

(Figure 2). Define $H^j(i)$ to be the position of the first level-j node encountered on $H(i)$, and $H_m(i)$ to be the m-th node in the path $H(i)$. According to the formation of the path, we have that $H^0(i) = i$, and $H^{j+1}(i) = \lfloor (H^j(i) - 1)/2 \rfloor$ when $H^j(i) > 0$, or $s/2^{j+1} - 1$ when $H^j(i) = 0$. For any two leaf nodes i_1, i_2, $H_m(i_1)$ and $H_m(i_2)$ must be on the same level. An important property of the tower graph is that we can bound the difference between the indices of $H_m(i_1)$ and $H_m(i_2)$ according to the difference between i_1 and i_2. This property is made precise by the following lemma.

Lemma 2.1. *$|(H^j(i_2) - H^j(i_1)) \bmod (s/2^j) - (i_2 - i_1)/2^j| \leq 2$ for $i_2 > i_1$.*

Proof. (By induction) When $j = 0$, it is obviously true as $H^0(i) = i$.

Suppose that it is true for level j. We shall prove that it is also true for level $j + 1$. Without loss of generality, assume that $H^j(i_1) \leq H^j(i_2)$. When $H^j(i_1) > 0$, we have that:

$$\begin{aligned}
& H^{j+1}(i_2) - H^{j+1}(i_1) \\
= \; & \lfloor (H^j(i_2) - 1)/2 \rfloor - \lfloor (H^j(i_1) - 1)/2 \rfloor \\
\leq \; & (H^j(i_2) - 1)/2 - (H^j(i_1) - 1)/2 + 1 \\
= \; & (H^j(i_2) - H^j(i_1))/2 + 1 \\
\leq \; & ((i_2 - i_1)/2^j + 2)/2 + 1 \\
& \text{(by the induction hypothesis)} \\
= \; & (i_2 - i_1)/2^{j+1} + 2\,.
\end{aligned}$$

When $H^j(i_1) = 0$, it is easy to verify that the same argument holds when taking modular arithmetic.

The same argument proves that $H^{j+1}(i_2) - H^{j+1}(i_1) \geq (i_2 - i_1)/2^{j+1} - 2$. □

3. FAULT TOLERANT GRAPHS FOR LINEAR ARRAYS

In this section, we show the construction of fault-tolerant graphs for linear arrays. What we actually obtain are fault-tolerant graphs for *directed* linear arrays, and the fault-tolerant graphs are *directed acyclic graphs* (DAG). Or more intuitively, we can number the nodes in the original graph G' from 0 to $v(G') - 1$, and the ordering is maintained in the realization of the array when there are faults.

In the following, we assume that $k = 2^t$ for $t \geq 0$. The construction is done recursively. When $t = 0$, $\hat{C}(n + 1, \{1, 2\})$ is an $\mathrm{FT}(\hat{L}_n, 1)$ graph [4]. Now we show how to construct graphs that tolerate $2k$ faults based on the graphs that tolerate k faults. We take two copies of $\mathrm{FT}(\hat{L}_{\lceil n/2 \rceil + k + 1}, k)$ graphs G_1, G_2 and create the graph G as follows. First, we re-number the nodes in G_1, G_2. For the node with index m in G_1, it is indexed $2m$ in G, and for the node m in G_2, its index is $2m + 1$ in G. In addition to the edges in G_1 and G_2, we also create the directed edges $\langle a, a + 1 \rangle$, for $0 \leq a < 2v(G_1)$ to stitch the two copies together. Now, we claim that:

Lemma 3.1. *The graph G is a directed acyclic graph and an $\mathrm{FT}(\hat{L}_n, 2k)$ graph.*

Proof. Clearly, G is a directed acyclic graph since G_1 and G_2 are acyclic graphs, and the additional edges added are consistent with the linear ordering of G_1 and G_2.

When there are $2k$ faulty nodes in G, either G_1 or G_2 contains no more than k faulty nodes. Without loss of generality, suppose that it is G_1. According to the induction hypothesis, G_1 is an $\mathrm{FT}(\hat{L}_{\lceil n/2 \rceil + k + 1}, k)$ graph. Therefore, there is a directed path with length $\lceil n/2 \rceil + k + 1$ in G_1. Suppose the directed chain thus obtained is

$$\hat{L} = i_1 \rightarrow i_2 \rightarrow i_3 \rightarrow \cdots \rightarrow i_{\lceil n/2 \rceil + k + 1}\,,$$

where $i_1 < i_2 < \cdots < i_{\lceil n/2 \rceil + k + 1}$. For each node i_j where $1 \leq j \leq \lceil n/2 \rceil + k$, consider the path P_j: $i_j \rightarrow i_j + 1 \rightarrow i_j + 2 \cdots \rightarrow i_{j+1} - 1 \rightarrow i_{j+1}$. Since i_j's are in increasing order, all the P_j's are node disjoint except at the end points of the paths. Call P_j *healthy* if all the nodes in P_j are healthy. Since there are at most $2k$ faulty nodes, and P_j's do not share interior nodes, at least $\lceil n/2 \rceil + k - 2k = \lceil n/2 \rceil - k$ paths are healthy. For each healthy path P_j, we replace the corresponding edge $\langle i_j, i_{j+1} \rangle$ in $\hat{L}$ with P_j. For each such replacement, we are able to add at least one node to the original path. Thus, we obtain a directed list with length at least $\lceil n/2 \rceil + k + 1 + (\lceil n/2 \rceil - k) \geq n$. This proves that the above graph is indeed an $\mathrm{FT}(\hat{L}_n, 2k)$ graph. □

Now, we analyze the complexity of the graph. For simplicity, denote by $v(n, k), d(n, k)$ to be $v(\mathrm{FT}(C_n, k))$ and

$d(\mathrm{FT}(C_n, k))$, respectively. We then have the following recurrences according to the above construction:

$$\begin{cases} v(n, 2k) \le 2v(\lceil n/2 \rceil + k + 1, k),\ v(n, 1) = n + 1\,; \\ d(n, 2k) \le d(\lceil n/2 \rceil + k + 1, k) + 2,\ d(n, 1) = 4\,. \end{cases}$$

These recurrences solve to $v(n, k) = n + O(k \log k)$ and $d(n, k) = O(\log k)$. Thus, we have that:

Lemma 3.2. *For any $n, k > 0$, there are $\mathrm{FT}(\hat{L}_n, k)$ graphs with $n + O(k \log k)$ nodes and $O(\log k)$ degree.*

There is a direct way to describe the graphs constructed in the above recursive procedure. For any $k > 0$, define T_k to be the set $\{2^j \mid 0 \le j \le \lceil \log k \rceil + 1\}$. Then, the graph constructed is exactly the graph $\hat{C}(n + c_1 k \log k, T_k)$ for some constant $c_1 > 0$.

By applying the method in [4], we can reduce the degree to constant by increasing the number of spare nodes to $O(k \log^2 k)$. The idea is to first construct an $\mathrm{FT}(\hat{L}_{n/s}, k)$ graph G and then use it as a "template" to construct another graph G' as described below. For each node i of G, we create a super-node A_i in G'. The super-node is a copy of the Moon graph K_s. We divide the nodes in A_i arbitrarily into two roughly equal-sized disjoint groups A_i^0 and A_i^1. To connect those super-nodes together, we add the following edges: for a directed edge $\langle i, j \rangle$ in G, we connect a node in A_i^1 to a node in A_j^0, with the only requirement being that all the edges incident to a super-node be evenly distributed among the nodes it contains. This way, we obtain the graph G'.

First, we shall show that G' tolerates k faults. The super-node A_i in G' is called *healthy* if all the nodes in A_i are healthy. When there are k faults, there are at most k super-nodes in G' are not healthy. Since G is an $\mathrm{FT}(\hat{L}_{n/s}, k)$ graph, there exists a directed path L with length n/s in G so that all the nodes in L correspond to healthy super-nodes in G'. We can then construct a path with length n in G' according to L. For a node i in G, suppose that $\langle j_1, i \rangle$ and $\langle i, j_2 \rangle$ in L are the (directed) edges incident to i. According to the construction, there are corresponding edges in G' between super-nodes A_{j_1}, A_i and A_{j_2}, A_i. Suppose those edges are incident to nodes i_1 and i_2, respectively, in A_i. Then, i_1 must be in A_i^0, and i_2 in A_i^1, i.e. i_1 and i_2 are different nodes. By the property of Moon graph, there exists a Hamiltonian path between i_1 and i_2. We can then unroll the super-node A_i to the Hamiltonian path between i_1 and i_2. By unrolling all the super-nodes corresponding to the nodes in L, we thus obtain a path with length $s \times n/s = n$ in G'.

The size of G' is clearly $s \times (n/s + O(k \log k)) = n + sk \log k$ and the degree of the graph is $O(\log k/s)$ since the edges are distributed evenly in a super-node. By taking $s = \log k$, we then have that

Theorem 3.3. *For any $n, k > 0$, there are $\mathrm{FT}(L_n, k)$ graphs with $n + O(k \log^2 k)$ nodes and $O(1)$ degree.*

Compared with the construction given in [4], our construction uses $O(k \log^2 k)$ spare nodes while theirs uses $O(k^2)$ spare nodes. The graphs we construct have another property as the longest array that it contains depends on the number of faulty nodes in the graph.

Corollary 3.4. *For some constant $c_2 > 0$ and for any $n, k > 0$, we can construct graph G' with n nodes and degree $O(\log k)$, so that for any $f < k/\log k$, if there are f faulty nodes in G', it contains a directed path with length at least $n - c_2 f \log f$.*

Proof. Consider the graph $C(n, T_k)$. By induction on k, we can prove it by following the same argument as in the proof of Lemma 3.1. The details are omitted. $\square$

In the following section, we will see how to apply the result obtained in this section, with the help of more subtle gadgets, to obtain improved fault-tolerant graphs for meshes.

4. FAULT TOLERANT GRAPHS FOR MESHES

In [4], constant degree fault-tolerant graphs are constructed for meshes by using $O(k^3)$ spare nodes. In this section, we shall show the construction of k-fault-tolerant graphs for meshes with $O(k^2/\log k)$ spare nodes and with $O(\log^3 k)$ maximum degree.

In the following, we first construct fault-tolerant graphs for meshes with $O(k \log k)$ spare nodes and $O(k \log k)$ degree. For any $r, c, k > 0$, let G be the $\mathrm{FT}(\hat{L}_{rc}, k)$ graph as in the previous section. According to Theorem 3.3, $v(G) = rc + c_1 k \log k$ for some constant $c_1 > 0$. In addition, for each node i in G, we create edges $\langle i, i+j \rangle$ for $c \le j \le c + c_1 k \log k$. Let G' be the resulted graph. We claim that:

Lemma 4.1. *The graph G' is an $\mathrm{FT}(M_{r,c}, k)$ graph with $O(k \log k)$ spare nodes and with $O(k \log k)$ maximum degree.*

Proof. Suppose that there are (up to) k faulty nodes. Since G' contains an $\mathrm{FT}(\hat{L}_{rc}, k)$ graph as a subgraph, there exists a path with length rc in the graph. Suppose that the nodes on the path are $i_1 < i_2 < \cdots < i_{rc}$. Because there are $c_1 k \log k$ spare nodes, we know that $j_2 - j_1 \le i_{j_2} - i_{j_1} \le (j_2 - j_1) + c_1 k \log k$ for $0 < j_1 < j_2 \le rc$. Thus, $c \le i_{j+c} - i_j \le c + c_1 k \log k$ for all j. According to our construction of G', there is an edge between the nodes i_j and i_{j+c}. Thus, we obtain a $r \times c$ mesh, where the j-th row consists of the nodes $i_{jc+1}, i_{jc+2}, \cdots, i_{jc+c}$.

Clearly, the number of vertices of G' is just the same as that of G's, and the degree of G' is $O(k \log k)$. $\square$

While G' uses small number of spare nodes, its degree is high. In the following, we shall show how to reduce the degree by using Moon graphs and tower graphs.

The edges in G' can be classified into two types, the *horizontal edges*, which are in G, and the *vertical edges*, which are added between $\langle i, i+s\rangle$ for $c \leq s \leq c + c_1 k \log k$. The horizontal and vertical edges correspond respectively to the row and column edges in the target mesh.

To reduce the degree, we follow the same approach in Section 3 by adding more spare nodes. In the following, we will first show how to reduce the degree to $O(\sqrt{k}\log k)$ by using Moon graphs. Then, we show how to reduce the degree further to $O(\log^3 k)$ by using tower graphs. In what follows, we only consider for $n \times n$ meshes. For general meshes, the same results hold.

To reduce the degree to $O(\sqrt{k}\log k)$, we first build a graph $G \in \mathrm{FT}(M_{n/s,n}, k)$ by Lemma 4.1 and then construct another graph G' using G as a template, just as in Section 3. For each node i in G, we form a super-node A_i, which is again a copy of Moon graph K_s and consists of two groups A_i^0 and A_i^1. For each horizontal edge $\langle i, j\rangle$ in G, we connect each node in A_i to all the nodes in A_j. For each vertical edge $\langle i, j\rangle$, we add an edge between a node in A_i^1 and a node in A_j^0 in such a way so that all the vertical edges added are distributed evenly in a super-node. We first claim that G' is a k-fault-tolerant graph for the $n \times n$ mesh. When there are faulty nodes, a super-node is said healthy if all of the nodes in it are healthy. There can be at most k super-nodes that are not healthy. Thus, there is an $n/s \times n$ mesh M in G where each node in M corresponds to a healthy super-node in G'. For each node i in M, there are at most two vertical edges incident to it, one to the upper row and the other to the lower row. Those two vertical edges have corresponding edges between super-nodes in G'. We can unroll the super-node A_i into a Hamiltonian path with the end points being the nodes in A_i that are incident to the corresponding vertical edges. This way, we obtain n columns, each with length n. What about rows? Since in our construction, every pair of nodes are connected whenever there is a horizontal edge between the two corresponding super-nodes, we can weave those n columns together and obtain an $n \times n$ mesh.

The number of nodes in G' is $s(n^2/s + O(k \log k)) = n^2 + O(sk \log k)$, and the degree is $O(k \log k/s + s \log k)$, where $k \log k/s$ accounts for the vertical edges, and $s \log k$ for the horizontal edges. Taking $s = \sqrt{k}$, we have that

Corollary 4.2. *For any n, k, there are $\mathrm{FT}(M_n, k)$ graphs with $O(k^{3/2}\log k)$ spare nodes and $O(\sqrt{k}\log k)$ degree.*

In the above construction, whenever there is a horizontal edge between two super-nodes, we add an edge between each pair of the nodes in those two super-nodes. While this ensures that we can weave the columns in the correct order to form a mesh, it also add unnecessarily many edges. The reason that we have to add edges between every pair of nodes between two super-nodes is that we distribute the vertical edges between super-nodes arbitrarily. This makes the Hamiltonian paths unrolled in each super-node arbitrary, and thus we have to add a horizontal edge between each pair of nodes in the corresponding super-nodes to make sure that there are edges between two Hamiltonian paths in the correct order.

As shown in Lemma 2.1, the tower graph has the property that the index difference of the corresponding nodes in the Hamiltonian paths can be bounded by the index difference between the starting nodes. Further, we observe that in the construction of the fault-tolerant graphs for arrays, two nodes are connected only if their index difference is in the set T_k, whose size is $O(\log k)$. Thus, by using tower graphs as the super-nodes, if we distribute vertical edges more carefully, we may not need to create pairwise horizontal edges between two super-nodes. Indeed, this is exactly the intuition behind our method to reduce the degree by using tower graphs.

To form a super-node Q'_s, we take two copies Q_s^0 and Q_s^1 of Q_s, the tower graph with s nodes, and connect their root nodes. Note that $v(Q'_s) = 2v(Q_s) = 4s - 2$ and $d(Q'_s) = 5$. Define $T_{k,s} = \{(2^j - 2^i) \bmod s \,|\, 0 \leq i, j \leq \log k\}$. Now, we construct an $\mathrm{FT}(M_{n/(4s-2),n}, k)$ graph G and use it as a template to construct another graph G'. Again, for each node i in G, we form a super-node A_i which is a copy of Q'_s. Each super-node A_i consists of two copies, A_i^0 and A_i^1, of tower graph. According to the edges in G, we create the following edges in G':

- For a horizontal edge $\langle i, j\rangle$ in G, for each $t \in T_{k,s}$, connect each node (l, a) in A_i^0 to nodes $(l, (a+m) \bmod s/2^l)$ in A_j^0, where $t/2^l - 2 \leq m \leq t/2^l + 2$. Repeat the same procedure for A_i^1 and A_j^1. This way, we increase the degree of each node in A_i, A_j by $O(|T_{k,s}|)$ for each horizontal edge $\langle i, j\rangle$.

- For a vertical edge $\langle i, j\rangle$, connect the leaf node $(0, (j - i - n) \bmod s)$ in A_i^1 to the leaf node $(0, (j - i - n) \bmod s)$ in A_j^0. Since the vertical edges connect the node i to the nodes $i+n, i+n+1, \cdots, i+n+c_1 k \log k$, this way, the vertical edges are distributed evenly to all the leaf nodes. Thus, the degree of each node is increased by $O(k \log k/s)$ by this procedure.

Now, we claim that G' tolerates k faults. Suppose that there are up to k faulty nodes. Again, we call a super-node healthy if it does not contain any faulty node. Then, there are at most k unhealthy nodes. Therefore, there is an $n/(4s-2) \times n$ mesh M that consists of nodes corresponding to healthy super-nodes. We then unroll each super-node into a path with length $4s - 2$ and weave them to form an $n \times n$ mesh as follows. For any node i_1 in M, it is incident to two vertical edges (i_0, i_1) and (i_1, i_2). According to the construction of G', we know there are corresponding edges between $A_{i_0}^1$ and $A_{i_1}^0$, and $A_{i_1}^1$ and $A_{i_2}^0$. Suppose that these two edges incident to the node a_i^0 in A_i^0 and a_i^1 in A_i^1, respectively. We can then unroll A_i into a Hamiltonian path by connecting the Hamiltonian path $H(a_i^0)$ in A_i^0 and $H(a_i^1)$ in A_i^1 at their root nodes. This way, we obtain a path with length $(4s-2) \times n/(4s-2) = n$ for each column. In the following, we argue that there are edges connecting those paths into an $n \times n$ mesh.

Let us consider four nodes i_1, i_2, j_1, j_2 in G where i_1, i_2 and j_1, j_2 are horizontal neighbors, and i_1, j_1 and i_2, j_2 are vertical neighbors in M, as shown as follows:

$$\begin{tikzcd} & \vdots & \vdots & \\ \cdots & i_1 \arrow[r, dash] \arrow[d, dash] & i_2 \arrow[d, dash] & \cdots \\ \cdots & j_1 \arrow[r, dash] & j_2 & \cdots \\ & \vdots & \vdots & \end{tikzcd}$$

According to the construction, we know that for (i_1, i_2) to be neighbors, it must be the case that $i_2 - i_1 = 2^m$, for some $0 \leq m \leq \lceil \log k \rceil + 1$. Therefore we have that $(j_2 - j_1) - (i_2 - i_1) = 2^{m_1} - 2^{m_2}$. That is to say that $t = ((j_2 - i_2 - n) - (i_2 - i_1 - n)) \bmod s \in T_{k,s}$. Or in other words, $(a^1_{i_2} - a^1_{i_1}) \bmod s = t \in T_{k,s}$. By Lemma 2.1, $t/2^l - 2 \leq (H^l(a^1_{i_2}) - H^l(a^1_{i_1})) \bmod s/2^l \leq t/2^l + 2$. Thus, by our construction, there is an edge between $H^l(a^1_{i_1})$ to $H^l(a^1_{i_2})$ for each l. According to the way that $H(i)$ is formed, we know that there is an edge between $H_p(a^1_{i_1})$ and $H_p(a^1_{i_2})$ for each p. Similarly, there must exist an edge between $H_p(a^0_{j_1})$ and $H_p(a^0_{j_2})$ for each p as well. Therefore, those edges weave all the columns together to form an $n \times n$ mesh, i.e. G' is an $\mathrm{FT}(M_n, k)$.

As for the size of the graph, the number of nodes is $(n^2/(4s-2)+O(k \log k)) \times v(Q'_s) = n^2+O(sk \log k)$, and the maximum degree is $O(k \log k/s + |T_{k,s}| \log k) = O(k \log k/s + \log^3 k)$, where the first term accounts for those vertical edges and the second term for the horizontal edges. By setting $s \approx k/\log^2 k$, we have that:

Theorem 4.3. *For any $n, k > 0$, there are $\mathrm{FT}(M_n, k)$ graphs with $O(k^2/\log k)$ spare nodes and maximum degree $O(\log^3 k)$.*

5. CONCLUSION

In this paper, we have shown construction of fault tolerant graphs with small degree for linear arrays and meshes. For linear arrays, we obtained constructions that use $O(k \log^2 k)$ spare nodes and have constant degree. For meshes, our constructions use $O(k^2/\log k)$ spare nodes and have maximum degree $O(\log^3 k)$. Compared to the former constructions, our construction uses smaller number of spare nodes while keeping the degree small.

There are still several interesting open questions in this area.

1. Is it possible to construct explicit fault tolerant graphs for directed linear arrays with $O(k \log k)$ spare nodes and constant maximum degree? or fault tolerant graphs for undirected linear arrays with $O(k)$ spare nodes and constant maximum degree?

2. Is it possible to improve the degree of $\mathrm{FT}(M_n, k)$ to constant while using about k^2 spare nodes? and is it possible to reduce the number of spare nodes further?

The technique of using super-nodes to reduce degree seems to maintain the product between the number of spare nodes and the maximum degree. It would be interesting to know if this is just a technical artifact or an intrinsic trade-off.

Acknowledgment

The author would like to thank Leo Guibas for his support and helpful discussion and Nick Pippenger for pointers to some references.

6. REFERENCES

[1] M. Ajtai, N. Alon, J. Bruck, R. Cypher, C. T. Ho, M. Naor, and E. Szemerédi. Fault tolerant graphs, perfect hash functions and disjoint paths. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 693–702, 1992.

[2] N. Alon and F. Chung. Explicit construction of linear sized tolerant networks. *Discrete Mathematics*, 72:15–19, 1988.

[3] Y. Aumann and M. A. Bender. Fault tolerant data structures. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 580–589, 1996.

[4] J. Bruck, R. Cypher, and C. Ho. Fault-tolerant meshes with small degree. *SIAM J. Comput.*, 26(6):1764–1784, Dec. 1997.

[5] P. Erdös, R. Graham, and E. Szemerédi. On sparse graphs with dense long paths. *Computers and Mathematics with Applications*, (1):365–369, 1975.

[6] J. Hastad, T. Leighton, and M. Newman. Fast computation using faulty hypercubes (extended abstract). In *Proc. ACM Symp. on Theory of Computing*, pages 251–263, 1989.

[7] C. Kaklamanis, A. R. Karlin, F. T. Leighton, V. Milenkovic, P. Raghavan, S. Rao, C. Thomborson, and A. Tsantilas. Asymptotically tight bounds for computing with faulty arrays of processors (extended abstract). In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 285–296, 1990.

[8] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan-Kaufmann, San Mateo, CA, 1992.

[9] T. Leighton, B. Maggs, and R. Sitaraman. On the fault tolerance of some popular bounded-degree networks. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 542–552, 1992.

[10] A. L. Rosenberg. The Diogenes approach to testable fault-tolerant arrays of processors. *IEEE Trans. Comput.*, C-32:480–489, 1983.

[11] A. L. Rosenberg. Fault-tolerant interconnection networks: a graph-theoretic approach. In *Proc. 9th Workshop on Graph-Theoretic Concepts in Computer Science*, pages 286–297, 1983.

[12] H. Tamaki. Construction of the mesh and the torus tolerating a large number of faults. *Journal of Computer and System Sciences*, 53(3):371–379, Dec. 1996.

[13] H. Tamaki. Efficient self-embedding of butterfly networks with random faults. *SIAM Journal of Computing*, 27(3):614–636, June 1998.