Fault-Tolerant Computing on Trees

Ajit Agrawal

Brown University
Dept. of Computer Science
Providence, Rhode Island 02912

Technical Report No. CS-90-05
March 1990

Fault-Tolerant Computing on Trees

Ajit Agrawal

Abstract

Much attention has been given recently to the resilience of different networks to faults. Hypercubes have been shown to be robust in presence of a constant fraction of random node and edge failures [HLN89]. Constant degree networks like the meshes have been analyzed with a view to wafer-scale integration in presence of faults [LC82]. Hypercube derivative networks have been shown to be robust in the presence of node failures [Ann89], though the more difficult case of edge failures remains unsolved.

Here we analyze the fault tolerant computing issues on a complete binary tree. Binary trees are well known networks, and their structure makes them susceptible to edge faults. Analysis of binary trees would be important in its own right as well as in analyzing many other tree derivative networks like the fat-trees, tree of meshes, pyramids etc.

Given a complete binary tree with faulty components, we show how to simulate the fault-free complete binary tree under different models of faults.

If nodes of an N-node complete binary tree fail randomly with constant probability $p < 1/2$, then the faulty tree can simulate the fault-free tree with $O(\log \log N)$ slowdown, with high probability.

If both edges and nodes fail with independent probability $p < 1/2$, we show that $\Omega\left((\frac{N}{\log N})^{\log(\frac{1}{1-p})}\right)$ is a lower bound on the simulation time of the fault-free tree by the faulty tree. We also show an almost strict upper bound on simulation time which is within a $\log^2 N$ factor of the lower bound for $p < 1/3$. The above results hold with high probability. The technique of bounding the size of the largest connected component in presence of edge faults, presented in this paper, is new and can be extended to other related networks.

1 Introduction

With the increase in the density of VLSI chips, multiple processors of a parallel machine are being implemented on the same chip. A positive fraction of such pro-

cessors are likely to have faults. Such faults have been modeled as independent probabilities of failure for each of the processors. Much attention has been given in the literature to the tolerance of networks with respect to these faults. [HLN89] have shown the hypercube to be a robust network in presence of a constant fraction of independent processor and edge faults. An N-node faulty hypercube has been shown to simulate a fault-free hypercube of the same size with a constant slowdown. [Ann89] has shown that N-node hypercube derivative networks can simulate their fault-free counterpart networks with $O(\log \log N)$ slowdown in the presence of a constant fraction of processor faults. His analysis does not take into account the more difficult case of edge failures. [Rag89] showed that routing on a mesh can be accomplished in $O\left(\sqrt{N} \log N\right)$ time with high probability even if a constant fraction of its edges and nodes are faulty.

The key parameters that characterize the mapping of a guest graph G to a host graph H are load, edge dilation, edge congestion, and path congestion [KLM+89]. Edges in G are mapped to paths in H, while the nodes in G are mapped to nodes in H. The load is the maximum number of nodes of G mapped to any node in H. The edge dilation is the maximum length of the path in H simulating an edge in G. The edge congestion is the largest number of such paths crossing an edge in H. The path congestion is the largest number of such paths that intersect any such path in H.

In this paper we analyze the tolerance of complete binary trees under different fault models. We show that an N-node faulty tree can simulate the N-node fault-free tree with $O(\log \log N)$ slowdown, if the probability of node failure is $p < 1/2$, and the edges are assumed to be fault-free. If the edges themselves could fail with independent probability p, then the tree itself may be disconnected. In this fault model, we present lower and upper bounds on the time for simulating the complete binary tree on the faulty tree. The analysis hinges on showing lower and upper bounds on the size of the largest connected components in the tree with faults. All the results hold with high probability.

2 Node failures

In this section we analyze simulating a fault-free complete binary tree on a faulty binary tree, where the nodes of the faulty tree could fail with probability $p < 1/2$. It is assumed that the communication hardware including the edges incident to a faulty processor are fully functional. Hence two nodes could communicate via edges of failed nodes. This fault model is easier to analyze than the model of both node and edge failures presented later. We show an upper bound of $O(\log \log N)$

simulation time. We show a lower bound of $\Omega(\log\log N)$, for using all the live processors.

2.1 Lower bound on simulation time

Since the edge dilation is a lower bound on the simulation time, it suffices to show that the edge dilation in connecting all the live nodes is large.

Lemma 1: In presence of node failures, the edge dilation in connecting all the live nodes is $\Omega(\log\log N)$, with high probability.

*Proof:*Consider the disjoint subtrees formed by the lowest $2k$ levels of the tree. In each of the subtrees, there is one center node with all the nodes in its k-neighborhood within the subtree. The number of such nodes is at most 3^k, and the probability that they are all dead is at least p^{3^k}. In a given subtree, consider the even A that the center node is alive and all its k-neighbors are dead. $Pr[A]$ is at least $(1-p)p^{3^k}$. Hence the probability that the event A does not occur in each of the subtrees, is less than $(1-(1-p)p^{3^k})^{\frac{N}{2^{2k}}}$, which is less than $e^{-(1-p)p^{3^k}\frac{N}{2^{2k}}}$. This probability is less than $\frac{1}{N}$ for $k = c\log\log N$, and the proper choice of the constant c. Hence, with high probability, there is a live node with all its $\Omega(\log\log N)$ neighbors dead. Hence to connect this live node to other live nodes must require a path length of $\Omega(\log\log N)$. Q.E.D.

2.2 Simulating fault-free tree on a faulty tree

Lemma 2: In presence of node failures, the faulty tree can simulate the functional tree with $O(\log\log N)$ slowdown with high probability.

To prove the above lemma, we start by defining a mapping of the nodes of the fault-free tree to the live nodes of the faulty tree. The dilation and edge congestion for the mapping will be shown to be $O(\log\log N)$. We will then show how to route on-line any set of paths in the tree with maximum edge congestion c and maximum path length d in $O(c+d)$ time. That would complete the proof.

We use different algorithms for mapping the nodes in the tree at height greater than $(\log\log N + c_1)$ (call them *high* nodes), and the the rest of the nodes (call them *low* nodes), where c_1 is a constant and will be specified later.

All the live high nodes are mapped to themselves. The dead high nodes look through their descendants at depth exactly $\log\log N + c_1$ in some order (say left to right) and map themselves to the first live node encountered. With the proper choice of the constant c_1, each dead node will find a live descendant to map to, with high probability. Moreover, each live node has exactly one dead ancestor

that may try to map itself to the node. Hence, the maximum load due to the mapping of high nodes is 2.

All the low nodes belonging to the subtree rooted at a node at height $\log\log N + c_1$ are mapped to at least half the live nodes within the subtree. Each such subtree has at least a constant fraction (say $b < (1-p)$) of its nodes live with high probability. The constant b will be specified later. Let m be the largest integer such that $(2^m - 1)$ is smaller than $b2^{\log\log N + c_1}$. The nodes in the top m levels of the subtree ($2^m - 1$ in number) rooted at height $\log\log N + c_1$, are mapped one to one, to $(2^m - 1)$ live nodes within the subtree. To do the mapping, we postorder the live nodes in the subtree. We also separately postorder the nodes in the top m levels of the subtree. The node with postorder number i in the fault-free subtree is mapped to the live node with postorder number i in the faulty subtree. Any node not in the top m levels of the fault-free subtree is mapped to the node to which its ancestor in the mth level from the top, is mapped (fig 1). The number of such nodes is $2^{\log\log N + c_1 - m + 1}$, which is no more than $4/b$, as $m \simeq \log\log N + c_1 - \lfloor \log \frac{1}{b} \rfloor$.

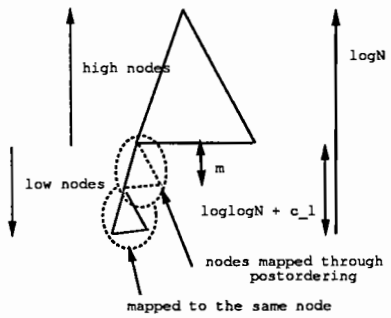

Fig. 1

2.3 Analysis

We call a dead high node *unmapped* if it is not able to find a live descendant at depth $\log\log N + c_1$ from itself. The probability q that a dead high node will be unmapped is the probability that all such descendants are dead. Since the number of such descendants is $2^{c_1}\log N$, the value of q is $p^{2^{c_1}\log N} = \frac{1}{N^{2^{c_1}\log(1/p)}}$. The probability that there exists an unmapped high node is no more than Nq. This probability can be made smaller than $1/N^c$, for any c, by the proper choice of c_1.

Now we show that every subtree consisting only of low nodes has at least a constant fraction b ($< 1-p$) of live nodes with high probability . The probability (q') that a given set of $2^{c_1}\log N$ nodes has less than a fraction b live nodes is

the same as the probability that more than the fraction $(1-b)$ of the nodes are dead. This probability can be bounded above by the Bernoulli bound given by $[B((1-b)2^{c_1}\log N, 2^{c_1}\log N, 1-p)]$. Let $\beta = \frac{1-b}{1-p}$. Using Chernoff bound, we get

$$\begin{aligned} q' &\leq e^{-\frac{1}{2}\beta^2(1-p)2^{c_1}\log N} \\ &= e^{-\frac{1}{2}\frac{(1-b)^2}{(1-p)}2^{c_1}\log N} \end{aligned}$$

We can choose constants b and c_1 such that the probability q' is $O(1/N^2)$. Hence the probability that there exists a subtree rooted at depth $\log N - \log\log N - c_1 + 1$ with less than a fraction b live nodes is $O(1/N)$.

2.4 Load

Mapping of the high nodes gives rise to a maximum load of 2. Mapping of the low nodes gives rise to a maximum load of $2/b$. Hence the maximum load for the mapping for any node is $(2 + \frac{4}{b})$ which is $O(1)$.

2.5 Dilation

A high node is mapped to a distance exactly $(\log\log N + c_1)$ from its original position, hence any edge between the high nodes in the fault-free tree has a maximum dilation of $2(\log\log N + c_1)$. Since all the low nodes are mapped within a subtree of height $(\log\log N + c_1)$, any edge between the low nodes has maximum dilation $2(\log\log N + c_1)$. The same bound holds for edges between the high and low nodes.

2.6 Edge congestion

We show that no edge in the faulty tree has more than O $(\log\log N)$ paths passing through it.

Let (u, v) be a tree edge in the fault-free tree. Let u and v be mapped to nodes u' and v' respectively. For high nodes u, v, the path between the nodes u' and v' can be simulated by the following segments: the path $(u' - u)$, the edge (u, v), and the path $(v - v')$. In bounding the edge congestion, therefore, it suffices to consider just the edge congestion caused by the paths $(u' - u)$ between the faulty nodes and the nodes they are mapped to.

Let (x, y) be an edge in the faulty-tree and let x be the father of y. For any path $(u' - u)$, where u' is the live counterpart to a dead node u, to use an edge (x, y), u must be an ancestor of y. Since the maximum distance between u and u'

is only $(\log\log N + c_1)$, the maximum number of ancestors for y, and hence the maximum edge congestion for (x, y) is also the same. This proves that the edge congestion for the mapping of high nodes is $O(\log\log N)$.

We now give the edge congestion due to mapping of the low nodes. The postorder numbering of any binary tree of height h has the property that any set of consecutively numbered nodes in the tree has no more than $2h$ edges leaving the set. This implies that the postorder mapping used for the low nodes will lead to an edge congestion of no more than $2(\log\log N + c_1)$. Hence the total edge congestion on any edge due to the mapping of both the high and low nodes is no more than $4(\log\log N + c_1)$.

2.7 On-line $O(c + d)$ routing on a tree

In this section we show how to route on-line a set of paths on a binary tree with maximum edge congestion c and maximum distance d in $O(c + d)$ time. [LMR88] shows that for any network, an $O(c + d)$ schedule for the paths exist, though no efficient algorithms are presented. The algorithm presented here will be used to simulate the edges of the fault-free tree by their corresponding paths in the faulty tree in $O(\log\log N)$ time.

The on-line routing algorithm is simple. The priority of a message is given by the height of the least common ancestor (lca) of its source and destination nodes. (We will also use the term lca of a message to be the lca of its source and destination nodes.) Of all the messages waiting at an edge, the one with the maximum priority gets to leave first. Ties are broken arbitrarily. Consider the path of any message from u to v. Let z be the lca of u and v, and z', z'' the children of z on the paths $(u - z)$ and $(z - v)$ respectively (fig. 2). On its path from u to z, this message can only be delayed by another message whose lca is either z or an ancestor of z. All such messages must use the edge (z', z) and hence there are at most c such messages. Similarly, the message can be delayed a maximum of c on its path from z to v, as all the messages that can delay it must use the edge (z, z''). This shows that no message will be delayed any more than $2c$ steps and hence the routing can be done on-line in time $O(c + d)$ time with $O(c)$ queue size at each node.

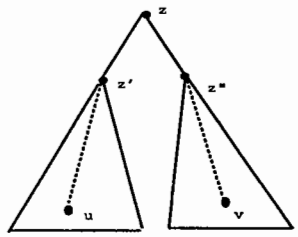

Fig 2

3 Edge failures

In this section we analyze the case where the nodes and edges may both fail with independent probability p. In this failure model, the edge failures could disconnect the graph into multiple connected components. The fault-free tree must hence be simulated on one of the connected components. We present upper bounds on the size of the largest connected component in the tree in presence of edge failures. This puts a lower bound on the simulation time. We also present lower bounds on the size of the largest connected component and show how to simulate the fault-free tree on the faulty tree, thus proving upper bounds on the simulation time. The bounds are close to tight, being within a $\log^2 N$ factor of each other. The analysis extends to the failure model in which the node fails with probability p, and all the edges incident to the failed node are assumed to fail too. All results hold with high probability.

3.1 Upper bound on the size of the largest connected component

Lemma 3: If the edge failure probability is $p < 1/2$, then with high probability, the largest connected component in an N-node complete binary tree is $S = O\left(N(\frac{N}{\log N})^{-\log \frac{1}{1-p}}\right)$.

*Proof:*We will show that the probability of having a connected component larger than S, rooted at a given node v is less than $1/N^2$. Hence the probability that there exists a connected component of size greater than S rooted at any of the nodes v is no more than $1/N$. This will complete the proof.

Let Y be the random variable denoting the number of children of a node connected by live edges. Let μ_0 and V_0 respectively be the expectation and variance of Y. For the given probability p of edge failure, $\mu_0 = 2(1-p)$, and $V_0 = 2p(1-p)$. Let X_i be the random variable denoting the number of nodes at level i of the tree

that are connected to the root v via live edges. It can be easily derived that $E_i = E(X_i) = {\mu_0}^i$, and $V_i = V(X_i) = V_0{\mu_0}^{i-1}\frac{{\mu_0}^i-1}{\mu_0-1}$.

We aim for a loose upper bound by assuming that all the edges in the first $O(\log\log N)$ levels from the root v of the tree are alive. We say that a level i is crowded if X_i is greater than m_i, where m_i is defined below (here α is a constant greater than 1):

$$m_i = \begin{cases} 2^i & \text{if } i \leq \log\log N + \alpha \\ \mu_0 m_{i-1}(1+\beta_{i-1}) & \text{otherwise} \end{cases}$$

where,

$$\beta_i = \sqrt{\frac{c\log N}{\mu_0 m_i}}$$

Let A_i be the event that at least one of the levels up to the level i is crowded. Since all the edges in the top $(\log\log N + \alpha)$ levels are assumed to be alive, the probability of A_i is 0, for i no greater than $\log\log N + \alpha$. We shall derive the value of $Pr[A_i]$ for all other levels. Let C_i be the event that the ith level is crowded. We now write the equation for $Pr[A_i]$.

$$\begin{aligned} Pr[A_i] &= Pr[A_{i-1} \bigcup C_i] \\ &= Pr[A_{i-1}] + Pr[C_i|\overline{A_{i-1}}] \end{aligned}$$

Since the number of nodes at a level depends only upon the number of nodes at the previous level, the probability of a level being crowded depends only upon the previous level. Hence, $Pr[C_i|\overline{A_{i-1}}]$ is equal to $Pr[C_i|\overline{C_{i-1}}]$.

Suppose the value of X_{i-1} is x_{i-1}, where $x_{i-1} \leq m_{i-1}$. Then X_i is binomially distributed with mean $\mu_0 x_{i-1}$ and probability of success $(1-p)$. Furthermore, $Pr[X_i > \mu_0 m_{i-1}(1+\beta_{i-1})]$ is no more than $Pr[X_i > \mu_0 x_{i-1}(1+\sqrt{\frac{c\log N}{\mu_0 x_{i-1}}})]$, because $x_{i-1} \leq m_{i-1}$, which by Chernoff's bound is no more than $e^{-\frac{c\log N}{2}}$. This leads to the following recurrence for $Pr[A_i]$:

$$Pr[A_i] \leq Pr[A_{i-1}] + e^{-c\log N/2}$$

The probability that any of the levels of the tree is crowded is given by $A_{\log N}$ the value of which can be bounded using the above recurrence.

$$Pr[A_{\log N}] \leq e^{-c\log N/2}(\log N - \log\log N - \alpha)$$

$Pr[A_{\log N}]$ is $O\left(\frac{1}{N^2}\right)$ for the proper choice of the constant c. Hence the probability that there exists a connected component of size at least $\sum_{i<\log N} m_i$ rooted at any node of the faulty tree is no more than $N * O(1/N^2)$ which is $O\left(\frac{1}{N}\right)$.

To bound the value of m_i, we first bound the value of β_i, which is $\sqrt{\frac{c \log N}{\mu_0 m_i}}$. Noting that $m_i = \mu_0 m_{i-1}(1+\beta_{i-1})$, we get

$$\begin{aligned} \beta_i &= \frac{\beta_{i-1}}{\sqrt{\mu_0(1+\beta_{i-1})}} \\ &< \frac{\beta_{i-1}}{\sqrt{\mu_0}} \\ &= x\beta_{i-1} \end{aligned}$$

where

$$x = \sqrt{\frac{1}{\mu_0}}$$

Note that x is less than 1 for $p < 1/2$. Now, for $i > \log\log N + \alpha$,

$$\begin{aligned} m_i &= \mu_0 m_{i-1}(1+\beta_{i-1}) \\ &= m_{\log\log N+\alpha} \prod_{j>\log\log N+\alpha}^{i-1} [\mu_0(1+\beta_j)] \end{aligned}$$

Using $(1+\beta_j) \leq e^{\beta_j}$, $\prod_j(1+\beta_j)$ is no more than $e^{\sum_j \beta_j}$. Moreover, $\sum_j \beta_j$ is no more than $\frac{1}{1-x}\beta_{\log\log N+\alpha}$, where $\beta_{\log\log N+\alpha}$ is given by $(1+\sqrt{\frac{c\log N}{\mu_0 \log N 2^\alpha}})$ and is $O(1)$ for every choice of c and α. Hence

$$m_i \leq m_{\log\log N+\alpha} e^{O(1)} \mu_0^{i-\log\log N-\alpha}$$

Now we can bound the size of the largest connected component :

$$\begin{aligned} \sum_{i=0}^{\log N} m_i &\leq \sum_{i=0}^{\log\log N+\alpha} 2^i + \sum_{i=\log\log N+\alpha+1}^{\log N} m_i \\ &= O\left(N\left(\frac{N}{\log N}\right)^{-\log\frac{1}{1-p}}\right) \end{aligned}$$

Since the largest connected component is no more than $O\left(N(\frac{N}{\log N})^{-\log\frac{1}{1-p}}\right)$, the time to simulate a complete binary tree of N nodes on the faulty tree is at least $\Omega\left((\frac{N}{\log N})^{\log\frac{1}{1-p}}\right)$. This proves lemma 3.

3.2 Lower bound on the size of the largest connected component

Lemma 4: If the edge failure probability is $p < 1/3$, then with high probability, the largest connected component in an N-node complete binary tree is $\Omega\left(\frac{N}{\log N}(\frac{N}{\log^2 N})^{-\log\frac{1}{1-p}}\right)$.

*Proof:*The proof will proceed in two steps. We first show that there is a subtree of small depth ($O(\log\log N)$) rooted at level $O(\log\log N)$ with $O(\log N)$ of its leaves connected by live edges. We will then show that given such a subtree, it will have a connected component of the appropriate size with high probability. That will complete the proof.

Let v be a node in the tree. Let X_l be the random variable denoting the number of descendents of v at the lth level of the tree that are connected to v via live edges. Recall that the expected value and the standard deviation of the random variable X_l (called E_l and σ_l) are given by ${\mu_0}^l$ and $\sigma_0\sqrt{{\mu_0}^{l-1}}\sqrt{\frac{{\mu_0}^l-1}{\mu_0-1}}$ respectively. Also recall that $\mu_0 = 2(1-p)$ and $\sigma_0 = \sqrt{V_0} = \sqrt{2p(1-p)}$. Using Chebyshev's Inequality, $Pr(X_l < E_l - k\sigma_l) \leq 1/k^2$, for any $k > 1$. Now,

$$\begin{aligned} E_l - k\sigma_l &= {\mu_0}^l(1 - \frac{k\sigma_0}{\sqrt{\mu_0(\mu_0-1)}}\sqrt{\frac{\mu_0^l-1}{\mu_0^l}}) \\ &\geq {\mu_0}^l(1 - \frac{k\sigma_0}{\sqrt{\mu_0(\mu_0-1)}}) \\ &= {\mu_0}^l(1-f) \end{aligned}$$

where

$$f = k\sqrt{\frac{p}{1-2p}}$$

For $p < 1/3$, we can always find a $k > 1$, such that $f < 1$. By choosing $l = (\frac{1}{1-\log\frac{1}{1-p}})(\log\log N + \log\frac{D}{1-f})$, we get $Pr(X_l < D\log N) < 1/k^2$, for any constant D. Consider the trees rooted at the nodes of the the dth level, where $d = \log\log N + \gamma$. Call a subtree unpopulated if has less than $D\log N$ nodes connected by live edges at the lth level of the subtree, where l is as defined above. The probability that all the subtrees rooted at the dth level are unpopulated is no more than $(1/k^2)^{2^\gamma \log N}$. For any choice of D, the proper choice of the constant γ will ensure that this probability is less than $1/N^c$, for some constant $c \geq 1$.

This proves that with high probability there is a set of $D \log N$ nodes at level s, connected by live edges to some ancestor node in the tree. Note that $s = (l + d)$, and is less than $2 \log \log N + \mathrm{O}(1)$. This completes the first part of the proof.

Consider the subtree that has at least $D \log N$ nodes at level s. We can now put a lower bound on the total number of nodes connected by live edges in this subtree. The analysis is similar to that in the previous section. Note that if there are m_i nodes at the ith level that are connected by live edges, then the probability that more than $m_i 2p(1 + \beta_i)$ of its children edges are dead is less than $e^{-\frac{\beta_i^2}{2} 2pm_i}$. As before, for the choice of $\beta_i = \sqrt{\frac{c \log N}{2pm_i}}$, this probability is $\mathrm{O}(1/N^c)$. Hence the number of nodes m_{i+1} at the $(i+1)$th level connected by live edges to one of the nodes in m_i, is with high probability is at least $2m_i(1-p(1+\beta_i))$. Define m_i for all levels $i > s$ as $m_i = 2(1-p(1+\beta_{i-1}))m_{i-1}$, where $\beta_i = \sqrt{\frac{c \log N}{2pm_i}}$, for some constant c. As we just showed, $Pr(x_i < m_i) \leq e^{-c \log N/2}$. Analogously to the definitions in the previous section, call a level i unpopulated if $x_i < m_i$. As before, one can show that with high proabability no levels in this tree are unpopulated. Hence the size of the set of nodes connected by live edges in this subtree is $(\sum_{i>S} m_i + \mathrm{O}(\log N))$. In a similar manner to the previous section, β_i is less than $x\beta_{i-1}$, where $x = \frac{1}{\sqrt{2p}}$. We shall now estimate m_i.

$$\begin{aligned} m_i &= 2(1 - p(1 + \beta_{i-1}))m_{i-1} \\ &= 2(1-p)(1 - \frac{p}{1-p}\beta_{i-1})m_{i-1} \\ &= [\prod_{j>s}^{i} 2(1-p)(1 - \frac{p}{1-p}\beta_{j-1})]m_s \end{aligned}$$

Note that $\frac{p}{1-p}$ is less than 1 for $p < 1/2$, and that $\prod_j (1-\beta_j)$ is less than $(1-\sum_j \beta_j)$. Hence, we get the following :

$$\begin{aligned} m_i &\geq (2(1-p))^{i-s}(1 - \sum_{j>s}^{i} \beta_j)m_s \\ &\geq (2(1-p))^{i-s}(1 - \frac{\beta_s}{1-x})m_s \end{aligned}$$

Now we can bound the size of the largest connected component given by $(\sum_{i>s}^{\log N} m_i + \mathrm{O}(\log N))$, using the fact that m_s is at least $D \log N$.

$$\sum_{i>s}^{\log N} m_i = \Omega\left((2(1-p))^{\log N - 2\log\log N - \mathrm{O}(1)} D \log N\right)$$

$$= \Omega\left(\frac{N}{\log N}\left(\frac{N}{\log^2 N}\right)^{-\log\frac{1}{1-p}}\right)$$

This proves lemma 4. Note that the bounds on the size of the largest connected component are almost strict, being within $\log^2 N$ factor of each other. We shall show in the next section that the upper bound on the simulation time using this result is $\mathrm{O}\left(\log N(\frac{N}{\log^2 N})^{\log\frac{1}{1-p}}\right)$.

3.3 Simulating a complete binary tree on the faulty tree

We show how to simulate an N node complete binary tree T_G on another binary tree T_F of size M and height h in $\frac{N}{M} + h + \log N$ time. This is optimal for $M = \mathrm{O}(N/\log N)$.

The algorithm is similar to one described earlier. We map each of the nodes in the top $\lfloor \log \frac{N}{M} \rfloor$ levels of T_G into a different node of T_F, by the method described below. Any node in another level of T_G is mapped to where its ancestor node at level $\lfloor \log \frac{N}{M} \rfloor$ is mapped in T_F. This leads to a maximum load of $2\frac{N}{M}$ in a processor.

To perform the mapping of the nodes in top $\lfloor \log \frac{N}{M} \rfloor$ levels of T_G, we do a post-order labeling of the nodes in these levels of the tree. We also do a post order labeling of the nodes in T_F. Processor with postnumber i in T_G is mapped to the processor with postnumber i in T_F. An edge of the complete tree by such a mapping is simulated by a path of length $O(h)$ in the binary tree. Each edge in the binary tree has congestion no more than $O(\log N)$ due to the paths, by the same argument as presented in section 2.5. As shown in section 2.7, the communications represented by the paths can be done by an on-line algorithm in $O(h + \log N)$ time. Thus the total simulation time would be $\mathrm{O}\left(\frac{N}{M} + h + \log N\right)$. Note that for the faulty tree $M = \theta(N^{1-\alpha})$ with high probability, and $h = \mathrm{O}(\log N)$, which yields an optimal simulation time of $\mathrm{O}(N^{\alpha})$.

4 Other fault models

It should be noted that the above analysis and bounds also hold for other fault models. The probability of failure of an edge (u, v) in the above discussions, where u is the father of v, can be considered conditioned on the event that node u is alive. Hence, under the fault model where the edges of a failed processor are considered failed, and the node failures are independent, the above analysis also holds. Even under the fault model of independent edge and node failures, the

analysis will hold by scaling the edge survival probability by the node survival probability.

5 Conclusions

We analyzed the simulation of a fault-free complete binary tree on a faulty one under different fault models. We showed that the tree with processor failures can simulate the fault free tree with $O(\log \log N)$ slowdown. The edge failures are much harder to deal with as they disconnect the tree. We presented almost tight lower and upper bounds on simulation time even with edge failures.

It would be interesting to adapt this analysis for analyzing fault-tolerance in the fat-trees, the tree of meshes, or related networks.

6 Acknowledgments

We would like to thank Tom Leighton for suggesting the proof of lemma 3, and Philip Klein for his suggestions and encouragement. We would also like to thank Sandeep Bhatt for helpful discussions.

References

[Ann89] F. Annexstein. Fault tolerance in hypercube derivative networks. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 179–188, June 1989.

[HLN89] J. Hastad, T. Leighton, and M. Newman. Fast computation using faulty hypercubes. In *21st Annual ACM Symp. on Theory of Comput.*, pages 251–263, 1989.

[KLM+89] R. Koch, T. Leighton, B. Maggs, S. Rao, and A. Rosenberg. Work-preserving emulations of fixed-interconnection networks. In *?? Annual ACM Symp. on Theory of Comput.*, pages 0–0, 1989.

[LC82] T. Leighton and Leiserson C.E. Wafer-scale integration of systolic arrays. *IEEE Trans. on Computers*, c-34:448–461, 1982.

[LMR88] F.T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *29th Annual IEEE Symp. on Found. of Comp. Sc.*, pages 256–269, 1988.

[Rag89] P. Raghavan. Robust algorithms for packet routing on a mesh. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 344–350, June 1989.

Fault-Tolerant Computing on Trees

Ajit Agrawal

Department of Computer Science
Brown University
Providence, Rhode Island 02912

Technical Report No. CS-90-05
Revised Version
September 1990

Fault-Tolerant Computing on Trees

Ajit Agrawal*

Abstract

We analyze the fault-tolerance of complete binary tree networks. We show how an N-node tree subject to random failures can simulate a perfect, fault-free tree under different fault models.

If edges fail with independent probability $p < \frac{1}{3}$, we show both a lower bound and an approximately tight upper bound on the time required to simulate a perfect tree on the faulty tree. If edges are assumed to be robust and the nodes have a failure probability of $p < \frac{1}{2}$, then the faulty tree can simulate the fault-free tree with $O(\log \log N)$ slowdown. We also show a matching lower bound for using all non-faulty processors. The above results are combined to analyze other fault models. All the above results hold with high probability.

1 Introduction

Wafer scale integration technology makes it feasible to implement large networks on a chip. However, due to fabrication errors, chips are expected to have some faults. The faults occur either in the processors or in their interconnections. Such faults alter both the size and the structure of the useful network on the chip. Faulty chips can be put to use by working around the faults and simulating the fault-free network on the faulty one. It is thus of interest to analyze how well a network in the presence of faults can simulate the fault-free network. A network robust to faults will have low simulation time.

The "goodness" of a simulation can be characterized in terms of three key parameters - edge dilation, edge congestion, and load [LMR88]. Let G represent the original network. Let the network achieved after deleting the faulty nodes and edges in G be G'. In a simulation of G by G', the nodes in G are mapped to nodes in G', and the edges in G are mapped to paths in G'. *Edge congestion* is the maximum number of such paths passing through an edge in G'. *Edge dilation* is the maximum path-length in G' corresponding to an edge in G. *Load* is the maximum number of nodes in G that are mapped to a node in G'. In simulating a network G'' on G', if the mapping from the nodes in G'' to nodes in G' is one to one, and the edges are mapped to disjoint physical paths, then the network G' is said to be *reconfigured.*

We consider the random failure model of components. A component prone to failure is modeled with a constant probability of failure, and the failure events of the components are assumed independent. Under different fault models, different failing components are considered. In the node failure model, the processors could be faulty while the edges are assumed robust, while in the edge failure model the assumptions are the opposite. In the edge-node failure model, either the processors and the edges are considered independent components, or a component consists of a processor with all its associated edges.

Researchers have also looked at fault-tolerance issues in other well known networks under similar failure models. Leiserson et. al.

*Dept. of Computer Science, Box 1910, Brown University, Providence, RI 02912. Email: aka@cs.brown.edu.

[LL82] and Gamal et. al. [GG84] have presented lower and upper bounds for many reconfiguration problems in meshes. Hastad et. al. [HLN89] have shown the hypercube to be a robust network in presence of a constant fraction of independent processor and edge failures. An N-node faulty hypercube has been shown to simulate a fault-free hypercube of the same size with a constant slowdown. Annexstein [Ann89] has shown that N-node hypercube derivative networks can simulate their fault-free counterpart networks with $O(\log\log N)$ slowdown in the presence of a constant fraction of processor failures. His analysis does not take into account the more difficult case of edge failures. Raghavan [Rag89] showed that routing on a mesh can be accomplished in $O\left(\sqrt{N}\log N\right)$ time even if a constant fraction of its edges and nodes are faulty. All these results hold with high probability.

We present the fault analysis for complete binary trees under different failure criteria for its components. We give almost strict lower and upper bounds on simulation time for each failure criteria. Let N be the size of the complete binary tree being fabricated. If both the processors and edges have independent probability of failure of $p < \frac{1}{3}$, then the size of the largest remaining connected component is with high probability bound within polylog multiplicative factors of $N^{1-\alpha}$, where α is $\log(\frac{1}{1-p})$. This provides a lower bound on the simulation time of the fault-free tree of size N by the faulty tree. We also show an upper bound on simulation time which is within a $\log^3 N$ factor of the lower bound, and has edge dilation and edge congestion of $O(\log N)$. The technique of bounding the size of the largest connected component is new and can be extended to other related networks.

If the edges are assumed to be robust, and only the processors may fail, then we show that for $p < \frac{1}{2}$, the faulty tree can simulate the fault-free tree with $O(\log\log N)$ slowdown. We prove a matching lower bound if every live processor is to be used in the simulation. For a tree, we also present an on-line $O(c+d)$ time routing algorithm for a set of messages with maximum distance d and maximum edge congestion c due to the message paths. All our simulation algorithms use this result. A schedule with the same time bound is shown to exist for any network in [LMR88], but no efficient algorithms were presented.

In Section 2 we give the on-line routing algorithm for a tree network. In Sections 3 and 4 we analyze the cases of edge failures and node failures respectively. We show how the analysis extends to other component models in Section 5, and present conclusions in Section 6.

2 On-line $O(c+d)$ routing on a tree

In this section, we show how to route on-line on a binary tree, a set of messages with associated paths such that the maximum path length is d and the maximum edge congestion due to these paths on an edge is c. Processors are assumed to be the sources or the destinations of no more than a constant number of such messages. Leighton et. al. [LMR88] show that for any network, an $O(c+d)$ schedule for the paths exist, and can be found through large offline computations.

The on-line routing algorithm is simple. The priority of a message is given by the height of the least common ancestor (*lca*) of its source and destination nodes. (We will also use the term lca of a message to denote the lca of the source and destination nodes of the message.) Of all the messages waiting for an edge, the one with the maximum priority leaves first. Ties are broken arbitrarily. Using this algorithm, we claim the following lemma.

Lemma 2.1 *Routing on a binary tree can be done in $O(c+d)$ time.*

Proof: Consider a message M from u to v with lca z. Let α be the last edge on its path from u to z. Any message that delays M must have

either z or an ancestor of z as its lca, and hence must also pass through the edge α. Since the maximum edge congestion is c, a maximum of c such messages can delay M on its path from u to z. A similar argument holds for M's path from z to v. Hence M will be routed in at most $2c + d$ steps. □

3 Edge failures

In this section we analyze the case where the edges may fail but the processors are assumed to be robust. We show in later sections, that the results for this failure model carry over to other component models. In this failure model, the edge failures could disconnect the graph into multiple connected components. The fault-free tree must hence be simulated by one of the connected components. We present upper bounds on the size of the largest connected component, thus providing a lower bound on the simulation time. We also present lower bounds on the size of the largest connected component and show how to simulate the fault-free tree on the faulty tree. The lower and upper bounds on simulation time are close to tight, being within a $\log^3 N$ factor of each other.

We state some useful lemmas without proof that will be used in the analysis. Let X be a random variable with mean μ and standard deviation σ.

Fact 3.1 (Chebyshev's Inequality) $Pr(|X - \mu| \geq c\sigma) \leq \frac{1}{c^2}$.

Fact 3.2 (Chernoff's Bound) *If we have N independent Bernoulli trials, each with probability p of success, and if $m = Np(1+\beta)$, where $0 \leq \beta \leq 1$, then the probability of at least m successes is at most $e^{-\frac{1}{3}\beta^2 Np}$.*

Consider the tree rooted at some node v, such that all the nodes in the tree are connected to the root v through paths of live edges. Let X_i be the random variable denoting the number of nodes at depth i of this tree. The root is considered to be at depth 0.

We shall denote the expectation and variance of a random variable X by $E(X)$ and $V(X)$ respectively. For ease of notation, we refer to $E(X_i)$ and $V(X_i)$ by μ_i, and V_i respectively. For the probability of failure of an edge being p, the following hold.

Fact 3.3 $\mu_1 = 2(1-p)$, *and* $V_1 = 2p(1-p)$.

Fact 3.4 $E(X_i|X_{i-1} = x) = \mu_1 x$, *and* $V(X_i|X_{i-1} = x) = V_1 x$.

Fact 3.5 ([Hai81]) $\mu_i = {\mu_1}^i$, *and* $V_i = V_1 {\mu_1}^{i-1} \frac{{\mu_1}^i - 1}{\mu_1 - 1}$.

Lemma 3.1 *For any $c \geq x \geq x'$, $Pr[X_{i+1} > c|X_i = x] \geq Pr[X_{i+1} > c|X_i = x']$.*

Proof: Follows from the binomial distribution of X_{i+1}, with means and variance defined in Fact 3.4. Details omitted for brevity. □

Lemma 3.2 $Pr[X_{i+1} > c|X_i \leq m] \leq Pr[X_{i+1} > c|X_i = m]$, *for any m.*

Proof: $Pr[(X_i \leq m) \bigcap (X_{i+1} > c)]$ can be written as $Pr[\bigcup_{j=1}^{m} \{(X_i = j) \bigcap (X_{i+1} > c)\}]$ and hence is at most $\sum_{j=1}^{m} Pr[X_i = j] Pr[X_{i+1} > c|X_i = j]$. Using Lemma 3.1, this is at most $\sum_{j=1}^{m} Pr[X_i = j] Pr[X_{i+1} > c|X_i = m]$, and hence at most $Pr[X_i \leq m] Pr[X_{i+1} > c|X_i = m]$. Since $Pr[(X_i \leq m) \bigcap (X_{i+1} > c)]$ also equals $Pr[X_i \leq m] Pr[X_{i+1} > c|X_i \leq m]$, the claim follows. □

3.1 Largest connected component - an upper bound

In this section, we show an upper bound on the size of the the largest tree that is connected by a set of non-faulty edges. We show that the probability of having a "large" connected component rooted at a given node is $O\left(\frac{1}{N^2}\right)$, and hence with high probability there is no large connected component in the faulty tree. We make precise what we mean by "large" below.

Consider the maximal live tree rooted at a given node v. Let the maximum possible height

of this tree be h, where h can be at most $\log N$. We say that a level i is crowded if X_i is greater than m_i, for m_i as defined below. We call the tree *large* if its size is greater than $\sum_{i=0}^{h} m_i$. We show that no level in the tree is crowded, and hence the tree rooted at v cannot be large.

We define m_i in the following manner. α and c are constants specified later.

$$m_i = \begin{cases} 2^i & ; \text{if } i \leq \log\log N + \alpha \\ \mu_1 m_{i-1}(1 + \sqrt{\frac{c \log N}{\mu_1 m_{i-1}}}) & ; \text{o.w.} \end{cases}$$

m_i is thus the same as the number of nodes in the complete binary tree until the level $\log\log N + \alpha$, and then it is no more than a fraction of the value in the previous level. α is the smallest integer constant such that $m_{\log\log N+\alpha+1}$ is smaller than $2^{\log\log N+\alpha+1}$, and can easily be shown to be $\lceil \log(\frac{c\mu_1}{(2-\mu_1)^2}) \rceil$. The choice of m_i, α, and c will be clear from the discussion below.

We have the following lemma.

Lemma 3.3 *The probability that any of the levels is crowded is* $O\left(\frac{1}{N^2}\right)$.

Proof: By the choice of m_i, none of the levels until the level $\log\log N + \alpha$ can be crowded, since m_i is the maximum number of nodes at level i in a complete binary tree.

Let B_i denote the event that the level i is crowded given that none of the levels until $i-1$ are crowded. Since the number of nodes at a level depends only upon the number of nodes in the previous level, the probability of a level being crowded given that all the previous levels are not crowded is the same as the probability of the level being crowded given its previous level is not crowded. Hence $Pr[B_i]$ is the same as $Pr[X_i > m_i | X_{i-1} \leq m_{i-1}]$, which by Lemma 3.2 is no more than $Pr[X_i > m_i | X_{i-1} = m_{i-1}]$. By Fact 3.2, this probability is no more than $e^{-\frac{c}{2}\log N}$, for any i.

The probability that at lease one of the levels is crowded is given by $\sum_{i=0}^{h} Pr[B_i]$, which from above is no more than $he^{-\frac{c}{2}\log N}$. Since $h \leq \log N$, this probability can be made smaller than $\frac{1}{N^2}$, by the proper choice of the constant c. □

This shows that for a given node, with high probability no level is crowded in its tree. Hence the probability that there is a large tree rooted at v is also $O\left(\frac{1}{N^2}\right)$. This implies that the probability of a large tree rooted at any of the nodes is $O\left(\frac{1}{N}\right)$.

We now bound the value of $\sum_{i=0}^{\log N} m_i$. Let β_i denote the value $\sqrt{\frac{c \log N}{\mu_1 m_i}}$. The following lemma bounds the value of β_i.

Lemma 3.4 $\beta_i \leq \sqrt{\frac{1}{\mu_1}}\beta_{i-1}$.

Proof: Since $m_i = \mu_1 m_{i-1}(1+\beta_{i-1})$, we have $\beta_i = \sqrt{\frac{c \log N}{\mu_1{}^2 m_{i-1}(1+\beta_{i-1})}} = \frac{\beta_{i-1}}{\sqrt{\mu_1(1+\beta_{i-1})}}$, which is no more than $\sqrt{\frac{1}{\mu_1}}\beta_{i-1}$. □

The following lemma bounds the size of m_i.

Lemma 3.5 $m_i \leq c_2 m_k \mu_1{}^{i-k}$, *for* $i > k$, *and some constant* c_2, *where* $k = \log\log N + \alpha$.

Proof: Since $m_i = \mu_1 m_{i-1}(1+\beta_{i-1})$, using the inequality $(1+y) \leq e^y$ for positive y, m_i is no more than $\mu_1 m_{i-1} e^{\beta_{i-1}}$, and hence no more than $\mu_1{}^{i-k} m_k e^{\sum_{j>k}^{i} \beta_j}$. Let $x = \sqrt{\frac{1}{\mu_1}}$. x is less than 1 for $p < \frac{1}{2}$ by Fact 3.3. Hence $\sum_{j>k}^{i} \beta_j$ is no more than $\frac{\beta_k}{1-x}$. By choosing $c_2 = e^{\frac{\beta_k}{1-x}}$, the claim follows. □

We now state the main result of this section.

Theorem 3.1 *The largest connected component in the tree is no more than* $O\left(N(\frac{N}{\log N})^{-\log\frac{1}{1-p}}\right)$. *Hence the time to simulate a complete binary tree of* N *nodes on the faulty tree is at least* $\Omega\left((\frac{N}{\log N})^{\log\frac{1}{1-p}}\right)$.

Sketch Proof: The size of the largest connected component is at most $\sum_{i=0}^{\log N} m_i$. Substituting the proper values of μ_1, and that of m_i from its definition and Lemma 3.5, the claim follows. □

3.2 Largest connected component - a lower bound

In this section we show a lower bound on the size of the largest connected component in the faulty tree. To show this bound, we prove that of all the nodes at a depth of $\log\log N+\gamma$ from the root, for some constant γ specified later, at least one of the nodes is the root of a "big" component. This is proved in two parts. We first show that at least one such node has at least $C\log N$ nodes at depth $\alpha\log\log N+\beta$, connected to the node through live edges, for a given constant C for the proper choice of α, β, and γ. We then show in a manner similar to the previous section that the tree rooted at this node is guaranteed to be big.

Consider the subtree rooted at a node v, such that every node is connected to v through live paths. Let X_i^v be the random variable denoting the number of nodes at level i in this tree. We then claim the following lemma.

Lemma 3.6 *If $p<\frac{1}{3}$, then for any choice of C, we can choose constants α,β and γ such that there exists a node v at depth $\log\log N+\gamma$ with $X^v_{\alpha\log\log N+\beta}\geq C\log N$, with high probability.*

Proof: Let σ_l denote the standard deviation of the random variable X_l. By Fact 3.1, we know that $Pr[X_l^v<E_l-k\sigma_l]\leq\frac{1}{k^2}$. By setting l to be $(\frac{1}{1-\log\frac{1}{1-p}})(\log\log N+\log\frac{C}{1-kf})$, where $f=\sqrt{\frac{V_1}{\mu_1(\mu_1-1)}}$, $E_l-k\sigma_l$ is at least $C\log N$. Then we get $Pr[X_l^v<C\log N]\leq\frac{1}{k^2}$. For any choice of $p<\frac{1}{3}$, we can choose some $k>1$ such that kf is less than 1. For $p<\frac{1}{3}$, l is smaller than $3\log\log N+\mathrm{O}(1)$.

Call a subtree *weak* if it does not have $C\log N$ nodes at level l. Consider all subtrees rooted at nodes at depth $\log\log N+\gamma$. There are $2^\gamma\log N$ subtrees. The probability that each of them is weak is no more than $(\frac{1}{k^2})^{2^\gamma\log N}$, which can be made smaller than $\frac{1}{N^2}$ for any $k>1$ by the proper choice of γ. □

Consider the node v at depth $\log\log N+\gamma$ that has at least $C\log N$ nodes in its subtree at level l. Let $s=l+\log\log N+\gamma$. By the choice of l, s equals $4\log\log N+\mathrm{O}(1)$. The expected number of dead edges for a node is $2p$. We call a level $i\geq s$ in this subtree *unpopulated* if it has fewer than m_i nodes at level i, where m_i is defined below, in a manner similar to before.

$$m_i = \begin{cases} C\log N & ;\text{ if } i=s \\ 2m_{i-1}-2m_{i-1}(1+\sqrt{\frac{c\log N}{2pm_{i-1}}}); & \text{o.w.}\end{cases}$$

We call a tree *big* if it has at least $\sum_{i>s}^{\log N} m_i$ nodes connected through live edges. We now claim the following lemmas.

Lemma 3.7 *With high probability, no level in the subtree rooted at v is unpopulated.*

Proof: The proof is similar to the proof for Lemma 3.3, and is omitted for brevity. □

Let $\beta_i=\sqrt{\frac{c\log N}{2pm_{i-1}}}$. Then the following holds.

Lemma 3.8 *$\beta_i\leq x\beta_{i-1}$, where $x=\sqrt{\frac{1}{2p}}$.*

Proof: Similar to the proof for Lemma 3.4. □

Lemma 3.9
$m_i\geq(2(1-p))^{i-s}(1-\frac{\beta_s}{1-x})C\log N$, for $i>s$.

Proof: Easily derived from the definition of m_i. We use the inequality $(1-x)\leq e^{-x}$, and that $\frac{p}{1-p}<1$, for $p<\frac{1}{2}$. □

We now claim the main result of this section.

Theorem 3.2 *With high probability, there exists a subtree of size $\Omega\left(\frac{N}{\log^3 N}(\frac{N}{\log^4 N})^{-\log\frac{1}{1-p}}\right)$.*

Proof: The size of the largest connected component is at least $\sum_{i>s}^{\log N} m_i+\mathrm{O}(\log N)$, which by using Lemma 3.9 is at least $\Omega\left((2(1-p))^{\log N-s}C\log N\right)$. The claim then follows. □

Note that the upper and lower bounds on the size of the largest component is within poly-log multiplicative factor of each other. Having shown that there is a big connected component in the tree with high probability, we present an algorithm for the optimal simulation of the fault-free tree on this connected component.

3.3 Simulating a complete binary tree on the faulty tree

We show how to simulate an N node complete binary tree T_G on another binary tree T_F of size M and height h in $\frac{N}{M} + h + \log N$ time. This is optimal for $M = \mathrm{O}(N/\log N)$.

We map each of the nodes in the top $\lfloor \log \frac{N}{M} \rfloor$ levels of T_G into a different node of T_F, by the method described below. Any node in another level of T_G is mapped to where its ancestor node at level $\lfloor \log \frac{N}{M} \rfloor$ is mapped in T_F. This leads to a maximum load of $2\frac{N}{M}$ in a processor.

To perform the mapping of the nodes in top $\lfloor \log \frac{N}{M} \rfloor$ levels of T_G, we do a post-order labeling of the nodes in these levels of the tree. We also do a post order labeling of the nodes in T_F. Processor with postnumber i in T_G is mapped to the processor with postnumber i in T_F.

We claim without proof the following property of postorder numbering.

Lemma 3.10 *The postorder numbering of any binary tree of height h has the property that any set of consecutively numbered nodes in the tree has no more than 2h edges leaving the set.*

An edge of the complete tree by such a mapping is simulated by a path of length $O(h)$ in the faulty binary tree. Consider an edge (u, v) in the faulty tree, where u is the parent of v. The set of paths going through (u, v) must have an endpoint in the subtree rooted at v. Moreover, the nodes in this subtree are numbered consecutively, and hence by Lemma 3.10, the number of such paths is no more than $O(\log N)$. By Lemma 2.1, the communications represented by these paths can be done by an on-line algorithm in $O(h + \log N)$ time. Thus the total simulation time will be $\mathrm{O}\left(\frac{N}{M} + h + \log N\right)$. Note that for the faulty tree M is $\theta(N^{1-\alpha})$ with high probability, and $h = \mathrm{O}(\log N)$, which yields an optimal simulation time of $\mathrm{O}(N^{\alpha})$.

4 Node failures

In this section we analyze for the case of faulty processors but robust edges. It is assumed that the communication hardware including the edges incident to a faulty processor are fully functional. Hence two processors could communicate via edges of failed nodes. For a failure probability of $p < \frac{1}{2}$, we consider simulating an N-node complete binary tree on a same sized network with faulty processors. We show that for using all the live processors, the simulation must take $\Omega(\log\log N)$ time. We provide a matching upper bound on the simulation time.

4.1 Lower bound on the simulation time

Since the edge dilation is a lower bound on the simulation time, it suffices to show that the edge dilation in connecting all the live nodes is large.

Lemma 4.1 *In presence of node failures, the edge dilation in connecting all the live nodes is* $\Omega(\log\log N)$.

Sketch Proof: We choose $k = c\log\log N$, and consider some set of $\frac{N}{2^{2k}}$ disjoint subtrees of height $2k$. For the proper choice of the constant c, we can show that for at least one of the subtrees, with high probability, the center node is alive and all its k-neighbors are dead. The path connecting this node to any other live node must hence be of length at least $\Omega(\log\log N)$. □

4.2 Simulating the fault-free tree on a faulty tree

We provide an algorithm for mapping the fault-free tree on the faulty tree. We show that the algorithm leads to constant load, and $\mathrm{O}(\log\log N)$ edge congestion and edge dilation. Thus by Lemma 2.1, the simulation can be done in $\mathrm{O}(\log\log N)$ time.

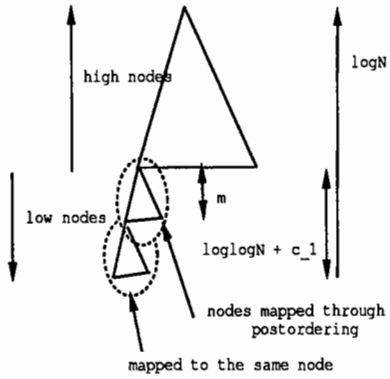

Figure 1: Mapping for node-failure model.

4.2.1 The mapping algorithm

We divide the nodes in the fault-free tree in two sections - the *high nodes* and the *low nodes*. Let the height of a node be the shortest distance of the node from any leaf. The nodes at height greater than $(\log\log N + c_1)$ are the high nodes. The rest are the low nodes (fig. 1). The constant c_1 will be specified later. We use different algorithms for mapping the high nodes and the low nodes.

All the live high nodes are mapped to themselves. To map a dead high node, we search through its descendants at distance $\log\log N + c_1$ from the dead node in some order, and map the dead node to the first live node encountered. We shall claim that for the proper choice of the constant c_1, all the high nodes are mapped with high probability.

The low nodes are mapped differently. We call the subtrees rooted at nodes at height $\log\log N + c_1$ the *low subtrees*. The nodes in each low subtree are mapped to the live nodes within the subtree using the mapping algorithm presented in Section 3.3.

4.2.2 Analysis

Lemma 4.2 *With high probability, every high node is mapped to a live node.*

Proof: Proof omitted for brevity. □

Lemma 4.3 *There is a constant b such that each low subtree has at least a constant fraction b of its nodes alive.*

Proof: Each subtree under consideration has $2^{c_1}\log N - 1$ nodes, and the number of such subtrees is less than N. The probability (q') that a given set of $2^{c_1}\log N$ nodes has less than a fraction b live nodes is the same as the probability that more than the fraction $(1-b)$ of the nodes are dead. This probability can be bounded using Lemma 3.2 to be no more than $e^{-\frac{1}{2}\frac{(1-b)^2}{(1-p)}2^{c_1}\log N}$. We can choose constants b and c_1 such that the probability q' is $O(1/N^2)$. Hence the probability that there exists a subtree rooted at height $\log\log N + c_1$ with less than a fraction b live nodes is less than Nq', which is $O(1/N)$. □

4.2.3 Load, Dilation, and Congestion

The maximum load at a node due to the mapping of the high nodes is 2 and due to the mapping of low nodes is $2/b$, hence the maximum load for any node is $O(1)$.

The edge dilation due to the mapping of both the high nodes and the low nodes is at most $2(\log\log N + c_1)$. The same holds for edges between the high and low nodes.

We will show that the edge congestion due to the mapping of the high nodes is at most $2(\log\log N + c_1)$. For the low nodes, an edge congestion of at most $2(\log\log N + c_1)$ follows from the argument in Section 3.3. Thus the total edge congestion for the mapping is $O(\log\log N)$.

We state the following lemma without proof.

Lemma 4.4 *Let the high node u be mapped to its descendant node u'. The edge congestion due to all such paths $(u'-u)$ is at most $(\log\log N + c_1)$.*

Let (u, v) be a tree edge in the fault-free tree. Let u and v be mapped to nodes u' and v' respectively. For high nodes u, v, the path between the nodes u' and v' can be simulated

by the following segments: the path $(u' - u)$, the edge (u, v), and the path $(v - v')$. By Lemma 4.4, the edge congestion for each of the path segments $(u' - u)$ and $(v - v')$ is at most $(\log\log N + c_1)$. Hence the edge congestion for the paths $(u' - v')$ is at most $2(\log\log N + c_1)$.

5 Other fault models

The results of previous sections also carry over to other component models. In one such model of practical importance, the edges to and from a failed processor are also assumed to have failed. All the analysis and bounds from Section 3 also hold for this model if p is considered to be the failure probability of each of the children of a node.

The analysis of Section 3 also applies to the model in which each of the edges and nodes are assumed to have independent failure probabilities of p_1 and p_2 respectively. The size of the largest component connected by live edges is still bound by earlier analysis assuming $p = p_1$. Moreover, since the component under discussion is large, it can be shown using Fact 3.2 that a constant fraction of the nodes are alive. Hence all the bounds on simulation time also hold to within a constant factor.

6 Conclusions

We presented the simulation of a fault-free complete binary tree on a faulty one under different fault models. We showed almost tight lower and upper bounds on simulation time for different fault models. The results show that the tree network can sustain node failures but degrades considerably in presence of edge failures.

It would be interesting to adapt this analysis for analyzing fault-tolerance in the tree derivative networks, i.e. fat-tree, the tree of meshes, or related networks.

7 Acknowledgments

We would like to thank Tom Leighton for his suggestions in proving Theorem 3.1, and Philip Klein for his suggestions and encouragement. We would also like to thank Sandeep Bhatt and David Greenberg for helpful discussions.

References

[Ann89] F. Annexstein. Fault tolerance in hypercube derivative networks. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 179–188, June 1989.

[GG84] J.W. Greene and A. El Gamal. Configuration of vlsi arrays in presence of defects. *Journal of ACM*, 31:694–717, 1984.

[Hai81] F. A. Haight. *Applied Probability*, pages 119–123. Plenum Press, 1981.

[HLN89] J. Hastad, T. Leighton, and M. Newman. Fast computation using faulty hypercubes. In *21st Annual ACM Symp. on Theory of Comput.*, pages 251–263, 1989.

[LL82] T. Leighton and C.E. Leiserson. Wafer-scale integration of systolic arrays. *IEEE Trans. on Computers*, c-34:448–461, 1982.

[LMR88] F.T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *29th Annual IEEE Symp. on Found. of Comp. Sc.*, pages 256–269, 1988.

[Rag89] P. Raghavan. Robust algorithms for packet routing on a mesh. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 344–350, June 1989.