Embedding of Binomial Trees in Hypercubes with Link Faults

Jie Wu, Eduardo B. Fernandez, and Yingqiu Luo
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

Abstract

We study the embedding of binomial trees with variable roots in n-dimensional hypercubes (n-cubes) with faulty links. A simple embedding algorithm is first proposed that can embed an n-level binomial tree in an n-cube with up to $n-1$ faulty links in $\log(n-1)$ steps. We then extend the result to show that spanning binomial trees exist in a connected n-cube with up to $\lceil \frac{3(n-1)}{2} \rceil - 1$ faulty links. Our results reveal the fault tolerance property of hypercubes and they can be used to predict the performance of broadcasting and reduction operations, where the binomial tree structure is commonly used.

1 Introduction

The *binomial tree* [5] is one of the most frequently used spanning tree structures for parallel applications in various systems, especially in hypercube systems. Lo et al. [4] have identified the binomial tree as an ideal computation structure for parallel divide-and-conquer algorithms. Hsu [3] showed the use of the binomial tree in prefix computation. The binomial tree structure also has been widely used in performing data accumulation (also called reduction) and data broadcasting.

As the number of processors in a computer system increases, the probability of processor failure also increases. In a spanning binomial tree of a hypercube, if any link used to connect two nodes becomes faulty, the tree will be disconnected. This might jeopardize applications that use this tree. The challenge is to identify a spanning binomial tree that connects all the nodes in the system using only healthy links.

The above problem resembles an embedding problem that deals with mapping a host graph (the binomial tree) into a target graph (the hypercube). There are two types of embedding problems [2]: *specified root embedding* and *variable root embedding*. In the specified root embedding problem the root of the binomial tree must be mapped to a specified node in the hypercube, while in the variable root embedding problem the root of the binomial tree can be mapped to any node in the hypercube.

Algorithms for embedding binary trees into healthy hypercubes have been developed by Wu [6] and Bhatt and Ipson [1]. Clearly, algorithms for embedding binomial trees into a specified root of hypercubes with faulty links might not exist even when there is only one faulty link. This is because any faulty links that are adjacent to the root node will destroy all the possible binomial trees originated from that root node. Among variable root embedding algorithms, Chan [2] studied embedding binary trees into hypercubes with faulty nodes, but relatively little work has been done on embedding of binomial trees in faulty hypercubes. Wu [7] proposed an embedding of incomplete spanning binomial trees into hypercubes with faulty nodes.

In this paper, we focus on finding spanning binomial trees in faulty hypercubes with faulty links only. First, a simple embedding algorithm is given that can tolerate $n-1$ faulty links in an n-cube. We then determine a lower bound on the degree of fault tolerance in hypercubes with faulty links. The bound is $\lceil \frac{3(n-1)}{2} \rceil - 1$ in a connected n-cube. We also provide a embedding scheme based on two simple construction schemes, *extending scheme* and *connecting scheme* (proposed in this paper), to identify spanning binomial trees in the given n-cube. Due to the space limitation, all the proofs of the theorems are omitted, they can be found in [8].

2 Preliminaries

The n-dimensional hypercube (n-cube), Q_n, is a graph having 2^n nodes labeled from 0 to $2^n - 1$. Two nodes are joined by an edge if their addresses, as binary integers, differ in exactly one bit. More specifically, every node a has a bit sequence $a_n a_{n-1} \cdots a_d \cdots a_1$ and a_d is called the d-th bit (also called the d-th dimension) of the address. Let node

0190-3918/97 $10.00 © 1997 IEEE

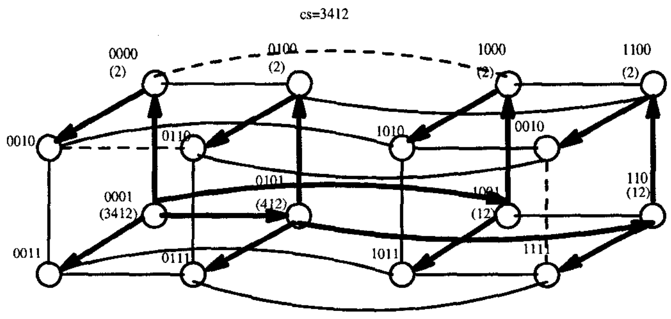

Figure 1: A Q_4 with three faulty links

a^d be the neighbor of node a along dimension d. Every m-dimensional subcube Q_m (m-subcube, or simply m-cube) has a unique address $q_n q_{n-1} \cdots q_d \cdots q_1$ with $q_d \in \{0,1,*\}$, where exactly m bits take the value $*$, a don't-care symbol representing either 0 or 1. For example, $10**$ denotes a 2-cube with four nodes: 1000, 1001, 1011, 1010. Sequence $a_n a_{n-1} \cdots a_{d+1} - a_{d-1} \cdots a_1$ represents a dimension d link connecting two nodes that differ in the d-th bit. Figure 1 shows a Q_4 with three faulty links -000, $0-10$, and $111-$.

A partition of Q_n along the d-th dimension generates two $(n-1)$-cubes, denoted by $Q_{n-1}^{(d)}$ and $Q_{n-1}^{'(d)}$. In our later discussion, we will omit d in both cubes if it does not cause confusion.

The *spanning binomial tree* is a special spanning tree in a hypercube. A 0-level binomial tree (B_0) has one node. An n-level (B_n) is constructed out of two $(n-1)$-level binomial trees by adding one edge between the roots of the two trees and by making either root the new root. A B_n can also be constructed from $B_{n-1}, B_{n-2}, ..., B_1, B_0$ by using a node s (the root node) to connect the root nodes of these trees. To embed a spanning binomial tree in an n-cube, each B_i should be a spanning binomial tree of an i-cube. More specifically, $B_{n-1}, B_{n-2}, ..., B_1, B_0$ are spanning binomial trees in subcubes $Q'_{n-1}, Q'_{n-2}, ..., Q'_1, Q'_0$, respectively, where these subcubes constitute a partition of Q_n (excluding the root node s). The above partition can be generated using the following procedure: Q_n is split into two $(n-1)$-cubes, $Q_{n-1}(s \in Q_{n-1})$ and $Q'_{n-1}(s \notin Q'_{n-1})$, along the d_1-th dimension. Q_{n-1} is further divided into two $(n-2)-$cubes along the d_2-th dimension. This process continues until Q_1 is divided into two 0-cubes, Q'_0 and $Q_0 = s$, along the d_n-th dimension. The dimension sequence $d_1 d_2 ... d_n$ that determines the partition is called the *splitting sequence* (ss) associated with the node s performing the partition. The above process is also called a *splitting process*. The set $\{Q'_{n-1}, Q'_{n-2}, \ldots, Q'_0, Q_0 = s\}$ is a partition of Q_n. The splitting sequence is also called *coordinate sequence* (cs) which determines a spanning binomial tree. Each node in the cube has its own coordinate sequence (which can be the same cs). The coordinate sequence at a node with respect to a specific root node is a subsequence of the associated cs and this subsequence includes only the dimensions of the largest subcube Q'_{n-i} in which the node is a root node of the spanning binomial tree of Q'_{n-i}. Basically, the coordinate sequence decides how the subcube should be partitioned. In Figure 1, each node has the same $cs = 3412$, the coordinate sequence of each node with respect to the root node 0001 is listed under the node address.

There are two basic constructing schemes for a binomial tree. The *extending scheme* is normally used if there is a fault-free dimension, otherwise the *connecting scheme* is applied.

- **Extending Scheme**: If Q_n can be divided into two $(n-1)$-cubes Q_{n-1} and Q'_{n-1} along a fault-free dimension d, we can construct an $(n-1)$-level binomial tree B_{n-1} in one of these two subcubes, say Q_{n-1}. By extending along dimension d, we can construct an n-level binomial tree B_n from B_{n-1}. The root node remains unchanged and all the leaf nodes in B_{n-1} are connected to the corresponding nodes in Q'_{n-1}. That is, each connection is a link along dimension d.
- **Connecting Scheme**: Suppose Q_n is divided into two $(n-1)$-cubes Q_{n-1} and Q'_{n-1}. with two $(n-1)$-level binomial trees B_{n-1} and B'_{n-1} in Q_{n-1} and Q'_{n-1}, respectively, and their roots are connected by a healthy link in dimension d (this dimension may or may not be fault-free). We can construct an n-level binomial tree B_n by randomly choosing one of two roots as the new root of B_n and connecting roots of B_{n-1} and B'_{n-1} by that healthy link in dimension d.

Note that in the extending scheme, cs is the reverse of the splitting sequence, while the splitting sequence derived by recursively applying the connecting scheme is cs. Figure 1 shows a spanning binomial tree with root node 0001. This spanning tree can be interpreted in different ways: (1) It is constructed using the extending scheme by expanding a spanning binomial tree in $**0*$ (with root 0001) to include $**1*$. (2) The tree is generated by combining two spanning binomial trees in $*0**$ (with root 0001) and $*1**$ (with root 0101) through the connecting scheme, i.e., by connecting roots 0001 and 0101 with 0001 being the new root.

3 Embedding in n-cubes with up to $n-1$ Faulty Links

In an n-cube only 2^n-1 out of $n\cdot 2^{n-1}$ links are used for embedding an n-level binomial tree. This provides many ways of selecting a spanning binomial tree in n-cubes even in the presence of faulty links.

Theorem 1: *An n-level binomial tree can be embedded using at most $log(n-1)$ subcube splits in an n-cube Q_n in the presence of up to $n-1$ faulty links.*

The extending scheme can be used to construct a spanning binomial tree in an n-cube with up to $n-1$ faults. The coordinate sequence of a potential root node a will be a randomly-selected splitting sequence of one of the fault-free subcubes that contains node a concatenating the reverse of a splitting sequence, where the splitting sequence defines the order of dimensions along which the n-cube is split into small cubes until the subcube that contains node a is fault-free.

```
procedure BT($f_k, Q_k, rss$) return (a root and its $cs$)
 /* $f_k$ is the set of faulty links in $Q_k$ */
 /* $rss$ is the reverse of a $ss$ before reaching $Q_k$ */
 if $Q_k$ is fault-free then
   { randomly choose a node in $Q_k$ as the root;
     select $ss$ as a permutation of dimensions in $Q_k$;
     $cs := ss \,||\, rss$ /* $cs$ at the root node */ }
 else
   { $Q_k = Q_{k-1} + Q'_{k-1}$ along a fault-free dimension $d$;
     $f_k = f_{k-1} + f'_{k-1}$, where $f_{k-1} \in Q_{k-1}$, $f'_{k-1} \in Q'_{k-1}$;
     $rss = d \,||\, rss$; /* insert $d$ to the front of $rss$ */
     if $|f_{k-1}| \leq |f'_{k-1}|$
     then BT($f_{k-1}, Q_{k-1}, rss$)
     else BT($f'_{k-1}, Q'_{k-1}, rss$) }
```

The root and its cs for a binomial tree can be located by a call BT(f, Q_n, ϕ), where f is the fault set in the target cube Q_n and ϕ is the empty set. Note that BT generates only one root node and a global cs (in the sense that each node in the cube has the same cs). The BT algorithm can be easily extended to a general one, where each node in a fault-free Q_k can be a root node and selects different splitting sequences of dimensions in Q_k, and then the reverse splitting sequence is attached to obtain different cs's for different root nodes. The time complexity of BT is $\Theta(n)$

We use the example in Figure 1 to explain how the BT algorithm works. As dimension 2 is fault-free, we split Q_4 into $**0*$ and $**1*$, and each node will carry a reverse splitting sequence 2. Because $**0*$ contains fewer faults than $**1*$, the BT algorithm chooses $**0*$ for further splitting. Both dimensions 1 and 3 are fault free in $**0*$ and we assume that dimension 1 is chosen. Thus $**0*$ is split into $**00$ and $**01$. The reverse splitting sequence of each node in $**0*$ is updated into 12. As $**01$ is fault-free, we can randomly choose a node, say 0001, to be the root node and construct a 2-level binomial tree in $**01$ by randomly choosing splitting sequence 34. We then extend it into a 4-level binomial tree by combining the splitting sequence and the reverse splitting sequence into a coordinate sequence $3412 = 34||12$.

4 Embedding in Connected n-cubes with up to $\lceil\frac{3(n-1)}{2}\rceil - 1$ Faulty Links

When the number of faulty links is more than n, a spanning binomial tree may not exist if there is an isolated node. Even for a connected n-cube, if a splitting dimension is not carefully selected, the subcubes can be disconnected. Therefore it is rather complex to construct an n-level binomial tree in a connected n-cube when the faulty links are more than n. The following results show that by carefully choosing root nodes and the coordinate sequence at each node, we can still find spanning binomial trees in n-cubes in the presence of up to $\lceil\frac{3(n-1)}{2}\rceil - 1$ faulty links.

Theorem 2: *Given $n \geq 4$ and $\delta = |f| - n \geq 0$, there exist at least $2^{n-\delta} - 2^{\delta+1} - 2^{\delta}$ nodes that can be chosen as the root of an n-level binomial tree in a connected n-cube Q_n with $|f|$ link faults, where $n \leq |f| < \lceil\frac{3(n-1)}{2}\rceil$.*

Theorem 3: *Given $n \geq 4$, there exist n-level binomial trees in a connected n-cube Q_n in the presence of up to $\lceil\frac{3(n-1)}{2}\rceil - 1$ faulty links.*

In order to obtain the correct coordinate sequence for each node (instead of one global cs as in BT), we consider separately the splittings along fault-free dimensions and the ones along faulty dimensions. In BT1 we use ss_h to record the splitting on fault-free dimensions and ss_f to record the splitting on faulty dimensions. In ss_h, a new splitting dimension is inserted as the first element of the sequence, while in ss_f it is appended as the last element of the sequence. The final splitting sequence cs is obtained by concatenating ss_f and ss_h, i.e., $cs = ss_f||ss_h$.

```
procedure BT1($f_k$, $Q_k$, $ss_f$'s and $ss_h$'s in $Q_k$)
return($root_{Q_k}$)
/* $f$ is the set of faulty links in $Q_k$ */
/* $ss_h$'s and $ss_f$'s are $ss$'s before partitioning $Q_k$ */
if $|f_k| < \frac{3(k-1)}{2}$ then
{ if there exist fault-free dimensions then
  { $Q_k = Q_{k-1} + Q'_{k-1}$ along fault-free dimension $d$;
    $f_k = f_{k-1} + f'_{k-1}$, where $f_{k-1} \in Q_{k-1}$, $f'_{k-1} \in Q'_{k-1}$;
    $ss_h = d \,||\, ss_h$ }
  else
  { $Q_k = Q_{k-1} + Q'_{k-1}$ along a dimension $d'$ such
    that each subcube has at least one link fault
    or dimension $d'$ has at least two link faults ;
    assume $f_{k-1} \in Q_{k-1}$ and $f'_{k-1} \in Q'_{k-1}$;
    $ss_f = ss_f \,||\, d'$ /* $ss_f$ is for each node except for
    the ones with an adjacent faulty link along $d'$ */ }
  if $Q_{k-1}$ is fault-free then
  { $root_{Q_k}$ = all the nodes in $Q_{k-1}$;
    $ss_h$= a splitting sequence of $Q_{k-1} \,||\, ss_h$
    /* nodes in $Q_k$ may have different $ss_h$'s
        using different splitting sequences */ }
  else
    $root_{Q_{k-1}}$ =BT1($f_{k-1}$, $Q_{k-1}$, $ss_f$'s and $ss_h$'s in $Q_{k-1}$);
  if $Q'_{k-1}$ is fault-free then
  { $root_{Q_k}$ = all the nodes in $Q'_{k-1}$;
    $ss_h$= a splitting sequence of $Q'_{k-1} \,||\, ss_h$}
  else
    $root_{Q'_{k-1}}$ =BT1($f'_{k-1}$, $Q'_{k-1}$, $ss_f$'s and $ss_h$'s in $Q'_{k-1}$);
  if there exist fault-free dimensions then
    $root_{Q_k} = root_{Q_{k-1}} \cup root_{Q'_{k-1}}$
  else
    $root_{Q_k} = \{ a, b | (a,b)$ is a healthy link, where
              $a \in root_{Q_{k-1}}$ and $b \in root_{Q'_{k-1}} \}$ }
else $root_{Q_k} = \phi$
```

The coordinate sequence cs of each root node is included in the return messages $root_{Q_k}$ through a call **BT1**(f, Q_n, ss_f's = ϕ and ss_f's = ϕ). For fault-free dimensions, we use the extending scheme to calculate ss_h and root nodes. For faulty dimensions, we use the connecting scheme to calculate ss_f and root nodes. The time complexity of BT1 is $\Theta(n^3)$. Once a specific node is chosen as the root of the binomial tree, we need to calculate coordinate sequences for all the nodes with respect to this root node. The following procedure CS (coordinate sequence calculation) produces a coordinate sequence of each node which is a subsequence of the associated cs. At each node, it basically deletes some dimensions from the associated cs which have already been used during the splitting process.

```
Procedure CS($r_k, Q_k$) /* $r_k$ is the root of $Q_k$ */
if $k \geq 1$ then
 { choose the first element $d$ of $cs$ associated with $r_k$,
   which has not been chosen; delete $d$ from all $cs$'s
   in $Q_k$, except the one associated with $r_k$;
   split $Q_k$ into $Q_{k-1}$ and $Q'_{k-1}$ along dimension $d$;
   CS($r_k, Q_{k-1}$);
   CS($r_k^d, Q'_{k-1}$) }
```

5 Conclusion

We have determined a lower bound on the number of link faults that can be tolerated to ensure the existence of spanning binomial trees in a connected hypercube. Our bound is $\lceil \frac{3(n-1)}{2} \rceil - 1$ faults in a connected n-cube. Note that the actual bound can be higher. However, with more faults included the probability of generating a connected hypercube will be reduced, making the assumption of the bound unrealistically restrictive. Therefore, a better bound can be only of theoretically interest. Two embedding schemes have been proposed based on the number of faulty links in the given n-cube.

References

[1] S. N. Bhatt and I. C. F. Ipsen. How to embed trees in hypercubes. Yale University Res. Rep. RR-443, Dec. 1985.

[2] M. Y. Chan, F. Y. L. Chin, and C. K. Poon. Optimal simulation of full binary trees on faulty hypercubes. Technical Report, Dept. of Computer Science, University of Hong Kong, 1991.

[3] W. J. Hsu, C. V. Page, and J. S. Liu. Computing prefixes on a large family of interconnection topologies. *Proc. of the 1992 International Conference on Parallel Processing.* Aug. 1992, III, 153-159.

[4] V. M. Lo, S. Rajopadhye, S. Gupta, D. Keldsen, M. A. Mohamed, and J. Telle. Mapping divide-and-conquer algorithms to parallel architectures. *Proc. of the 1990 International Conference on Parallel Processing.* Aug. 1990, III, 128-135.

[5] H. Sullivan, T. Bashkow, and D. Klappholz. A large scale, homogeneous, fully distributed parallel machine. *Proc. of the 4th Annual Symposium on Computer Architecture.* March 1977, 105-124.

[6] A. Y. Wu. Embedding of tree networks into hypercubes. *Journal of Parallel and Distributed Computing.* 2, (3), August 1993, 238-249.

[7] J. Wu. Tight bounds on the number of l-nodes in a faulty hypercube. *Parallel Processing Letters.* 5, (2), 1995, 321-328.

[8] J. Wu, E. B. Fernandez, and Y. Luo. Embedding of binomial trees in hypercube multiprocessors with link faults. Tech. Rep. TR-CSE-96-60, Florida Atlantic University, 1996.