Dynamic Reconfiguration of Complete Binary Trees in Faulty Hypercubes
在缺失的超立方體中動態重建完全二元樹

Chui-Cheng Chen
Southern Taiwan University of Technology
Department of Information Management
ccchen@mail.stut.edu.tw

Abstract

In this paper we present how to reconfigure dynamically a complete binary tree in faulty hypercubes. First, we use a dynamic algorithm to reconfigure a complete binary tree of height h (h≥0) in an (h+1)-dimensional faulty hypercube. If there is a faulty node in the hypercube, both the *dilation* and *congestion* are 2 after reconfiguration. If there are two faulty nodes in the hypercube, both the *dilation* and *congestion* are 3 after reconfiguration. If there are more than two faulty nodes in the hypercube, we impose a constraint on the type of the faulty nodes, both the *dilation* and *congestion* are 3 after reconfiguration. Then we reconfigure a complete binary tree of height h in an (h+2)-dimensional hypercube with at most 2^{h+1}-1 nodes, and the *dilation* and *congestion* are, respectively, 2 and 1 after reconfiguration. The number of the affected nodes are minimized after reconfiguration.

Keyworks: Reconfiguration, Complete binary tree, Hypercube, Embedding.

1. Introduction

The hypercube is one of the most effective as well as popular network architectures of parallel machines. The hypercube offers a rich interconnection topology, a recursive structure, and a low diameter. The structure of the hypercube can simulate many computational structures with only small constant factor slowdown, such as arrays, binary trees and mesh of trees [1].

Over the years, tree topology has been designed to describe many computations, for example, searching, sorting, and merging problems [2, 3]. Particularly interesting among trees is the complete binary tree, which is a natural computational structure for parallel algorithms, such as "divide-and-conquer" type [4].

Many researches have discussed the embedding of binary trees into hypercubes [1, 5-10]. In [1, 5], it has been proven that a double-rooted complete binary tree of height h (h≥0), denoted by DT_h which is a complete binary tree with the root replaced by a path of length two, is a subgraph of an (h+1)-dimensional hypercube, denoted by H_{h+1}. [6, 7] has shown a complete binary tree of height h, T_h, which has 2^{h+1}-1 nodes, can be embedded into H_{h+2} so that the adjacency of T_h is preserved. There exists no one-to-one node embedding of T_h into H_{h+1} and the adjacency of T_h is preserved [6]. Wagner [8] described the embedding of a binary tree of height h into H_h; the binary tree was complete for the first h-2 levels. Wu *et al.* [9] presented the embedding of binomial trees in hypercubes with link faults. [10-13] has addressed how to reconfigure binary trees in faulty hypercubes.

The purpose of this paper is to present how to reconfigure dynamically a complete binary tree in a hypercube with faulty nodes. First, we

discuss how to reconfigure T_h in an (h+1)-dimensional hypercube with faulty nodes, then discuss how to reconfigure T_h in an (h+2)-dimensional hypercube with at most 2^{h+1}-1 faulty nodes. It is considered that the number of the affected nodes are minimized after faults recovery.

The remaining sections are organized as follows. Section 2 gives the notations and definitions of this paper. In Section 3, we present how to reconfigure T_h in H_{h+1} with one or more faulty nodes. A free node of the hypercube is assigned to recover one faulty node, and the leaf nodes of T_h are embedded into other faulty nodes. In Section 4, we give an algorithm to reconfigure T_h in H_{h+2} with at most 2^{h+1}-1 faulty nodes, the *dilation* and *congestion* of reconfigurable embedding being respectively 2 and 1, and only the faulty nodes suffer the influence after faults recovery. Finally, the conclusion is given in Section 5.

2. Preliminaries

The root of the complete binary tree of height h, T_h, is in level 0, two nodes in level 1, four nodes in level 2, 2^i nodes in level i, *etc.*, and the total number of T_h is 2^{h+1}-1, where $h \geq 0$. The n-dimensional hypercube, H_n, has 2^n nodes. These nodes of H_n are labeled $\{0, 1, 2, \ldots, 2^{n-1}\}$ with binary numbers. Two nodes in the hypercube are linked with an edge if and only if their binary numbers differ by a single bit. The *Hamming distance* is the number of different bits between two nodes. To conveniently describe the embedding, we use two colors, black and white, to correspond to the binary number. If the node has even number of 1's, it is colored black. Otherwise it is colored white. Since the hypercube has a perfect matching, H_n has 2^{n-1} black nodes and 2^{n-1} white nodes.

The cost of one-to-one node embedding of a *guest graph* into a *host graph* is measured in terms of *dilation* and *congestion*. The dilation of an edge of the *guest graph* is the length of embedded path of the *host graph*. The *dilation* of an embedding is the maximum dilation over all edges of the *guest graph*. The congestion of an edge of the *host graph* is the number of edges of the *guest graph* that are embedded using the same edge of the *host graph*. The *congestion* of an embedding is the maximum congestion over all edges of the *host graph*. Hence, we have to consider the tradeoff between the *dilation* and *congestion* of an embedding.

The faulty model of the hypercube is defined as follows [14, 15].

(1) The computational part of a faulty node is not utilized, while its links are fault-free.
(2) A free node is not assigned initially; it can be used to recover faults but can not be reused to recover any other faults later.
(3) The task of the faulty node is allowed to migrate to the free node.
(4) Assume the faulty diagnosis mechanisms are fault-free.

Hence, all the free nodes can be used to recover from faults in the hypercube.

3. Reconfiguring T_h in a faulty H_{h+1}

In this section, we discuss how to reconfigure T_h in a faulty H_{h+1}. T_h can be embedded into H_{h+1} with *dilation* 2 and there remains a free node in H_{h+1} [6]. When an arbitrary faulty node occurs in H_{h+1}, we can reconfigure T_h in H_{h+1} since the hypercube is symmetric, and the reconfiguration results in all nonfaulty nodes of T_h to be affected. We have to consider how to reduce the overhead of data communication after faults recovery; that is, the number of the affected nodes has to be as few as possible after reconfiguration. Therefore, we present a dynamic algorithm to reconfigure the complete binary tree in a faulty hypercube.

Theorem 1. T_h can be reconfigured dynamically to embed into H_{h+1} with *dilation* 2 and *congestion* 2 when an arbitrary faulty node

occurs in H_{h+1}.

Proof. First, we construct T_h from DT_h. The *dilation* is 2 and the *congestion* is 1 for such construction of T_h in H_{h+1} (see Fig. 1) [1]. When an arbitrary node becomes faulty in H_{h+1}, there are two cases to be considered as follows.

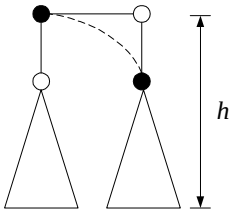


Fig. 1. Construction of T_h from DT_h.

Case 1. If the faulty node is one of both roots of DT_h, we let the nonfaulty root become the root of T_h, and the *dilation* and *congestion* are not altered.

Case 2. When the faulty node occurs in the internal nodes or the leaf nodes of T_h. Without loss of generality, assume the faulty node is in the left subtree of root r_l of DT_h. The path from the faulty node to root r_l has to be modified; that is, let r_r become the root of T_h and the nodes of the path are re-embedded into their parent nodes (see Fig. 2). Hence, each node of the path links its two sons using one edge whose dilation is 2 and the other edge whose dilation is 1, while the parent node of the faulty node uses two edges whose dilation are 2 to link its two sons. The congestion of edges of the path are equal to 2 in H_{h+1}.

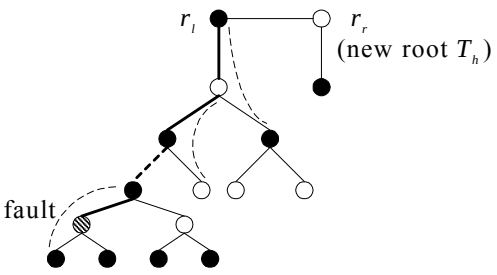


Fig. 2. The path from the faulty node to r_l is described by solid lines.

Therefore, if a faulty node occurs in level i (i>0) of T_h, the number of edges with dilation 2 increases i, and the congestion of edges of the path increases 1 in H_{h+1}. T_h can be reconfigured dynamically in H_{h+1} with *dilation* 2, *congestion* 2, and there are i+1 affected nodes after reconfiguring when an arbitrary faulty node occurs in H_{h+1}. □

Now we consider H_{h+1} with at leat two faulty nodes. For reconfiguring T_h in H_{h+1}, we need the following lemma.

Lemma 1. DT_{h-1} (h≥1) can be embedded into H_h as each leaf node of DT_{h-1} is linked to a certain internal node of DT_{h-1} via an edge in H_h.

Proof. We color the nodes of DT_{h-1} with black or white in H_h. Suppose the leaf nodes of left subtree of the roots are black and the leaf nodes of right subtree of the roots are white. We prove the lemma by induction on h.

Hypothesis: DT_{h-2} can be embedded into H_{h-1} as each leaf node DT_{h-2} is linked to a certain internal node of DT_{h-2} via an edge in H_{h-1}.

Basis step: When h=1, 2, it is trivial. DT_2 can be embedded into H_3 as shown in Fig. 3. The figure shows the links between leaf nodes and internal nodes in H_3; for example, leaf nodes *n3*, *n4*, *n7* and *n8* link to internal nodes *n2*, *n5*, *n1* and *n6*, respectively.

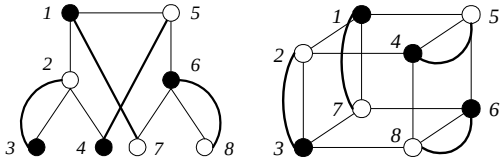


Fig. 3. Linking leaf nodes to certain internal nodes of DT_2. The solid lines represent the links in H_3.

Induction step: We partition H_h into two H_{h-1}'s by the most significant bit. Since DT_{h-2} is embedded into H_{h-1}, we can merge two DT_{h-2}'s to

DT_{h-1} as shown in Fig. 4 [1]. The links between leaf nodes and certain internal nodes in DT_{h-1} are the same as the hypothesis above describes. Therefore, the lemma is proved. □

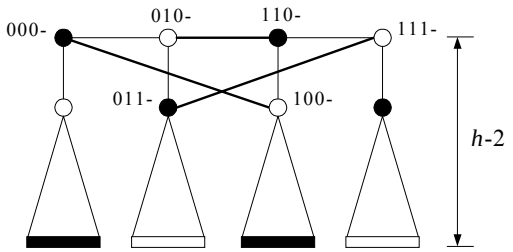


Fig. 4. Construction of DT_{h-1} from two DT_{h-2}'s. The added edges are shown in solid lines. Nodes 010- and 110- are both roots of DT_{h-1}.

Theorem 2. T_h can be reconfigured dynamically to embed into H_{h+1} with *dilation* 3 and *congestion* 3, and a leaf node of T_h is embedded into one faulty node of H_{h+1} when two arbitrary faulty nodes occur in H_{h+1}.

Proof. Likewise, We construct T_h from DT_h in H_{h+1} (see Fig. 1). When two arbitrary faulty nodes u and v occur in H_{h+1}, there are two cases to be considered as follows.

Case 1. If the two faulty nodes u and v are not adjacent in H_{h+1}, and u is in level i and v is in level j, where $i \le j$ (see Fig. 5). Faulty node u is reconfigured as Theorem 1 describes, and the other faulty node v has to be reconfigured to embedded into a leaf node of T_h to reduce the influence of the structure of T_h. Since internal node v of DT_h has an edge to link a certain leaf node, w, according to Lemma 1, faulty node v can be re-embedded into leaf node w. Hence, the dilation of edge which links the parent node of v and w is 2, and the dilation of edges which link w and two sons of v are 2. The congestion of edge (v, w) of H_{h+1} is 3.

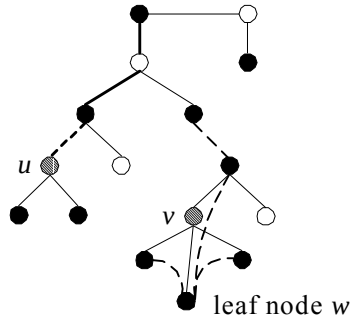
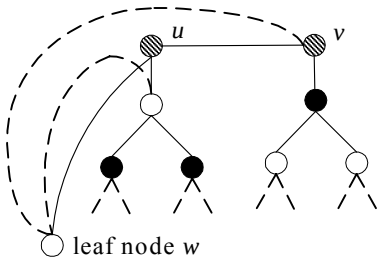


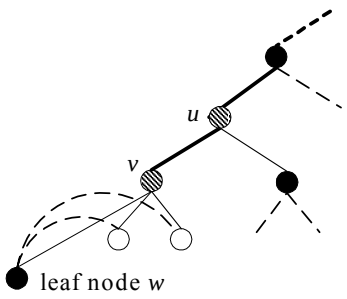
Fig. 5. Two faulty nodes u and v are not adjacent in H_{h+1}.

Case 2. If the two faulty nodes u and v are adjacent in H_{h+1}, u and v are in level 0, or u is in level i and v is in level j, where j-i=1 (see Fig. 6). When nodes u and v are in level 0 (see Fig. 6(a)), we let leaf node w, which has an edge to link faulty node u, become the root of T_h. The dilation of two edges linking w and its two sons are respectively 3 and 2. The congestion of edge (w, u) of H_{h+1} is 2.

When faulty node u is in level i and v is in level j, where j-i=1 (see Fig. 6(b)). Faulty node u is reconfigured to be the same as in Theorem 1. Assume leaf node w, which has an edge to link faulty node v, is used to recover faulty node v. The dilation of edge which links the parent node u and w is 3 after re-embedding, and the dilation and congestion of remaining edges are the same as in case 1.



(a)



(b)

Fig. 6. Two faulty nodes *u* and *v* are adjacent in H_{h+1}.

Therefore, T_h can be reconfigured dynamically to embed into H_{h+1} with two arbitrary faulty nodes, and a leaf node of T_h is embedded into one faulty node. Both the *dilation* and *congestion* are 3, and there are at most h+2 affected nodes after reconfiguring. □

When there are more than two faulty node appearing in H_{h+1}, we impose a constraint on the number and the type of faulty nodes as follows.

Constraint: When an internal node of the double-rooted complete binary tree in a hypercube occurs to be faulty, the nodes which are adjacent with the faulty node have to be nonfaulty. The leaf nodes, which have edges to link the faulty node and its adjacent nonfaulty nodes, have to be at least two nonfaulty nodes.

Now we consider how to reconfigure T_h with this constraint.

Theorem 3. With the above constraint, T_h can be reconfigured dynamically to embed into H_{h+1} with *dilation* 3 and *congestion* 3, and leaf nodes of T_h are embedded into the faulty nodes of H_{h+1}.

Proof. First, we construct T_h from DT_h in H_{h+1} (see Fig. 1). Each internal node has an edge to link a certain leaf node by Lemma 1; such linking edges are described by dashed lines as shown in Fig. 7. Without loss of generality, we consider probable cases as follows. Assume node *n2* is faulty and node *a* is nonfaulty (see Fig. 7), node *n2* is re-embedded into node *a*; hence, the dilation of edges (*n1*, *a*), (*a*, *n3*) and (*a*, *n4*) are respectively 2, and the congestion of edge (*n2*, *a*) of H_{h+1} is 3.

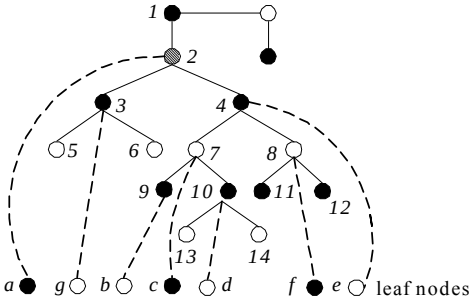


Fig. 7. The dashed lines describe the linking between the internal nodes and the leaf nodes.

Moreover, assume node *a* is also faulty, while at least one of the two leaf nodes *g* and *e*, which link respectively to nodes *n3* and *n4*, is nonfaulty. Without loss of generality, let node *e* be nonfaulty node, then node *n2* be re-embedded into node *n4* and node *n4* be re-embedded into node *e*. Hence, the dilation of edges (*n1, n4*), (*n4, n3*), (*e, n7*) and (*e, n8*) are 2, the dilation of edge (*n4, e*) is 1, and the congestion of edge (*n4, e*) of H_{h+1} is 3.

Furthermore, assume node *n7* or *n8* is faulty. Now we consider node *n7* only. At least one of the three leaf nodes *b*, *c* and *d*, which link respectively to internal nodes *n9*, *n7* and *n10*, is nonfaulty. Let node *d* be nonfaulty, node *n7* be re-embedded into node *n10* and node *n10* be re-embedded into node *d*. Hence, the dilation of edges (*e, n10*), (*n10, n9*), (*n10, d*), (*d, n13*) and (*d, n14*) are respectively 3, 2, 1, 2 and 2, and the congestion of edge (*n10, d*) of H_{h+1} is 3. Similarly, if nodes *n8* or *n13* or *n14*, *etc.*, are faulty, these faulty nodes can be reconfigured with dilation at most 3 and congestion at most 3.

Therefore, T_h with the constraint can be reconfigured dynamically to embed into H_{h+1} with *dilation* 3 and *congestion* 3 if there are

more than two faulty nodes appearing in H_{h+1}, and there are at most three affected nodes for reconfiguring each faulty node. □

4. Reconfiguring T_h in H_{h+2} with Faults

T_h has been shown to be able to be embedded into H_{h+2} so that the adjacency of T_h is preserved [6, 7]. There remains $2^{h+1}+1$ free nodes for embedding T_h into H_{h+2}, hence, H_{h+2} provides more fault tolerance. While T_h has to be reconfigured whenever a faulty node occurs in T_h, there are $2^{h+1}-1$ nodes which may be affected for each reconfiguring in H_{h+2}. The performance of the system will then suffer much influence for recovering the faults.

In this section, we present a reconfigurable algorithm to embed T_h into H_{h+2} with faults. The algorithm allows at most $2^{h+1}-1$ faulty nodes in T_h and only the faulty nodes are affected for the reconfiguration, and the dilation and congestion of reconfigurable embedding are respectively 2 and 1.

First, we give a definition and a lemma before reconfiguring T_h in H_{h+2}.

Definition 1. Let LT_h (h≥0) denote a graph which has two complete binary trees of the same height h and there is an augmented edge to link two nodes on the same position between two complete binary trees (see Fig.8 for LT_2).

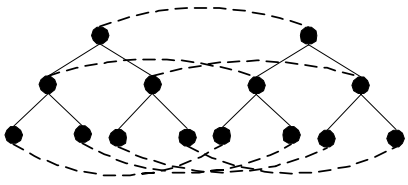


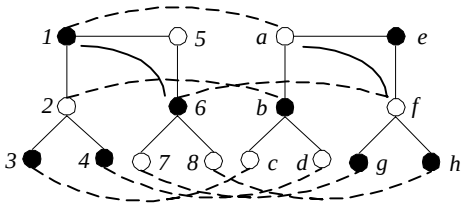
Fig. 8. LT_2 (The augmented edges are described by dashed lines).

Lemma 2. LT_h (h≥0) can be embedded into H_{h+2} with *dilation* 2 and *congestion* 1.

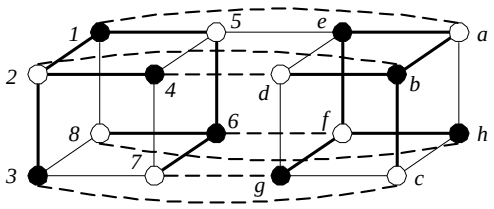
Proof. We prove the theorem by induction on h.

Hypothesis: Embedding LT_{h-1} into H_{h+1} with *dilation* 2 and *congestion* 1 is true.

Basis step: When h=0, 1 and 2, it is trivial. The embedding of LT_2 into H_4 is shown in Fig. 9. It is one-to-one node embedding, the dilation of edges (*n1, n6*) and (*a, f*) of LT_2 are respectively 2, and the *congestion* is 1.



(a)

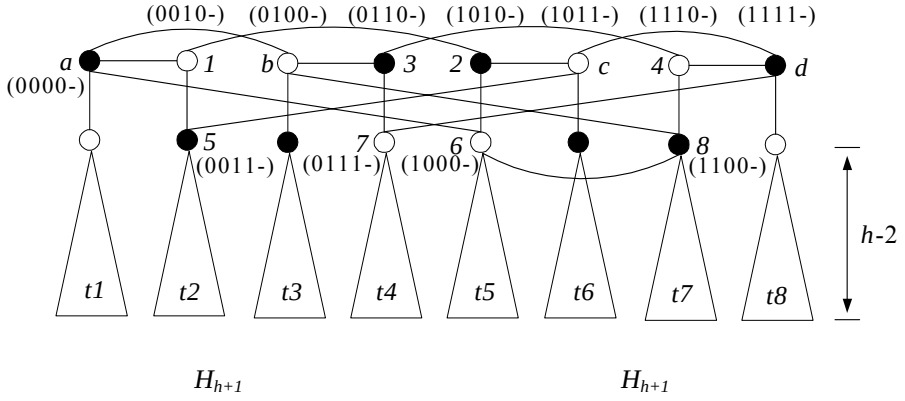


(b)

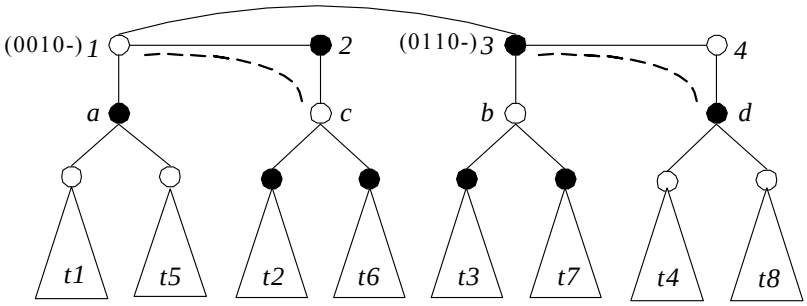
Fig. 9. (a) LT_2 in two DT_2's. (b) The embedding of LT_2 into H_4 (The augmented edges are described by dashed lines).

Induction step: First, we decompose H_{h+2} into two H_{h+1}'s. Then we can construct respectively two LT_{h-1}'s in both H_{h+1}'s. Let both node *a* and *b* denote the roots of LT_{h-1} in one H_{h+1}, and both node *c* and *d* denote the roots of LT_{h-1} in the other H_{h+1} as shown in Fig. 10(a). The nodes labeled with binary number are different at the most significant bit between two H_{h+1}'s. Since any hypercube is symmetric, we let nodes *a*, *n1*, *n5*, *b*, *n3* and *n7* link to nodes *n6*, *n2*, *c*, *n8*, *n4* and *d*, respectively. There are eight T_{h-2}'s in two H_{h+1}'s, which are denote *t1, t2, t3, t4, t5, t6, t7* and *t8* from left to right as shown in Fig. 10(a),

respectively.



(a)



(b)

Fig. 10. Construction of LT_h from LT_{h-1}'s in both H_{h+1}'s. The binary numbers of the leftmost four bits of the nodes are written aside the nodes.

In such linking, we can construct LT_h. The links among nodes on the same positions are the same as the hypothesis describes except for two new roots: *n1*, *n3*, while there is an unused edge to link *n1* and *n3* (see Fig. 10(b)). The *dilation* and *congestion* are the same as in the hypothesis. Therefore, this theorem is true for any dimension of hypercube according to the induction. □

Theorem 4. T_h can be reconfigured dynamically to embed into H_{h+2} with *dilation* 2 and *congestion* 1, and only the faulty nodes suffer the influence of reconfiguration when then there are at most 2^{h+1}-1 faulty nodes in T_h.

Proof. We can embed LT_h into H_{h+2} with *congestion* 1, the dilation of two edges are 2 and those of the others are 1 according to Lemma 2. Each node of one T_h, denoted by T_l, of LT_h has an edge to link a node (on the same position) of the other T_h, denoted by T_r, of LT_h.

Assume T_h is embedded initially into T_l with *dilation* 2 and *congestion* 1. When node *n1* (or *n3*) is faulty in T_l (see Fig. 11), we let node *n1* (or *n3*) be re-embedded into node *n8*. Moreover, if nodes *n1* and *n3* are faulty in T_l, we let both faulty nodes be re-embedded into nodes *a* and *c* of T_r, respectively. Similarly, if there are other arbitrary faulty nodes appearing in T_l, we re-embed the faulty nodes into the free nodes (on the same position) of T_r and link to two sons of the free nodes, then return to two sons of faulty node of T_l. The dilation of edges which link the free nodes of T_r to its parent and sons is at most 2, and the congestion of edges of H_{h+2} is 1.

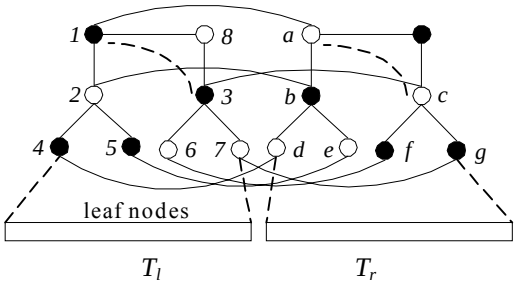


Fig. 11. T_h is reconfigured in LT_h.

Therefore, T_h can be reconfigured dynamically to embed into H_{h+2} with *dilation* 2, *congestion* 1, and the number of the affected nodes are minimized after reconfiguration when there are at most 2^{h+1}-1 faulty nodes in T_h. □

5. Conclusion

This paper has presented simple but effective algorithms to reconfigure dynamically complete binary trees in hypercubes. If there is an arbitrary faulty node in H_{h+1}, both the *dilation* and *congestion* are 2 after reconfiguring T_h. If there are two arbitrary faulty nodes in H_{h+1}, both the *dilation* and *congestion* are 3 after reconfiguration. There are at most h+2 affected nodes after reconfiguring T_h. Moreover, if there are more than two faulty nodes in H_{h+1}, we discuss how to reconfigure dynamically T_h with the constraint in H_{h+1}, both the *dilation* and *congestion* are 3 after recovery, and there are at most three affected nodes for reconfiguring each faulty nodes.

In addition, we present an algorithm to reconfigure dynamically T_h with at most 2^{h+1}-1 faulty nodes in H_{h+2}, the *dilation* and *congestion* being respectively 2 and 1, and the number of the affected nodes are minimized after reconfiguration.

References

[1] T. Leighton, "Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes", Morgan Kaufmann, Reading, MA, 1992.

[2] J. Bentley and H.T. Kung, "A tree machine for searching problems", Proc. Int'l Conf. Parallel Processing, pp. 257-266, 1979.

[3] Q. F. Stout, Sorting, "Merging, selection, and filtering on tree and pyramid machine", Proc. Int'l Conf. Parallel Processing, pp. 214-221, 1983.

[4] E. Horowitz and A. Zorat, "Divide-and-conquer for parallel processing", IEEE Trans. Computers, vol. 32, pp. 582-585, 1983.

[5] L. Nebesky, "On cubes and dichotomic tree", Časopis Pest. Mat., vol. 99, pp. 164-167, 1974.

[6] A. Y. Wu, "Embedding of tree networks into hypercubes", J. Parallel and Distributed Computing, vol. 2, pp. 238-249, 1985.

[7] E. L. Leiss and H. N. Reddy, "Embedding complete binary trees into hypercubes", Information Processing Letters, vol. 38, pp. 197-199, 1991.

[8] A. S. Wagner, "Embedding the complete tree in the hypercube", J. Parallel and Distributed Computing, vol. 20, pp. 241-247, 1994.

[9] J. Wu, E. B. Fernandez and Y. Luo, "Embedding of binomial trees in hypercubes with link faults", J. Parallel and Distributed Computing, vol. 54, pp. 59-74, 1998.

[10] M. Y. Chan and S. J. Lee, "Fault-tolerant embedding of complete binary trees in hypercubes", IEEE Trans. on Parallel and Distributed Systems, vol. 4, no. 3, pp. 277-288, 1993.

[11] B. M. Y. Chan, F. Y. L. Chin and C. K. Poon, "Optimal simulation of full binary trees on faulty hypercubes", IEEE Trans. on Parallel and Distributed Systems, vol. 6, no. 3, pp. 269-, 1995.

[12] P. J. Yang and C. S. Raghavendra, "Embedding and reconfiguration of binary trees in faulty hypercubes", IEEE Trans. on Parallel and Distributed Systems, vol. 7, no. 3, pp. 237-245, 1996.

[13] A. Wang, R. Cypher and E. Mayr, "Embedding complete binary trees in faulty hypercubes", Proc. of the third IEEE symposium on Parallel and Distributed

Processing, pp. 112-119, 1991.

[14] J. Hastad, T. Leighton and M. Newman, “Reconfiguring a hypercube in the presence of faults”, Proc. 19th Annu. ACM Symp. Theory Comput., pp. 274-284, 1987.

[15] N. Krishnakumar, V. Hegde and S. S. Iyengar, “Fault tolerant based embeddings of quadtrees into hypercubes, Proc. Int’l Conf. Parallel Processing, pp. 244-249, 1991.