A Secure Message Transmission Scheme in an Extended Network of Crossed Cubes

Bao-lei CHENG[1,2,a], Ting PAN[1,b], Jian-xi FAN[1,c], Jing-ya ZHOU[1,d] and Zhao LIU[1,e]

[1] School of Computer Science and Technology, Soochow University, Suzhou 215006, China

[2] Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, China

Email: [a] chengbaolei@suda.edu.cn, [b] 1549577258@qq.com, [c] jxfan@suda.edu.cn, [d] jy_zhou@suda.edu.cn, [e]liuzhao@suda.edu.cn

Keywords: Extended network of crossed cubes, Secure message transmission scheme, Independent spanning trees, Vertex-disjoint paths.

Abstract. Secure message transmission schemes in interconnection networks can be achieved by constructing multiple vertex-disjoint paths or independent spanning trees. The extended network of crossed cube (ECC for short) has the advantages such as easy to be deployed and low ratio between number of vertices and diameter. In this paper, we have studied a secure message transmission scheme in ECC by constructing n independent spanning trees in S_n and proposed a constructive algorithm with the time complexity $O(N)$, where N is the number of S_n. Furthermore, the maximum length of the vertex-disjoint paths between any two vertices is shown to be $2n+3$. In addition, simulation of vertex-disjoint paths based on JUNG has also been discussed.

Introduction

Multiple vertex-disjoint paths between arbitrary two vertices in interconnection networks can be used to transmit different parts of a message. Every part of the message can use its own encryption methods in its corresponding path with safety. Such vertex-disjoint paths can be constructed based on the multiple independent spanning trees (ISTs) [1, 2]. Towards the conjecture that for any n-connected graph $G(n \geq 1)$, there are n ISTs rooted at an arbitrary vertex on G [1, 2], it was only solved for $n \leq 4$ [1, 2, 3, 4], but remains open for $n \geq 5$. Thus, the results on special graphs are still the focus of researchers and many results have been obtained, such as hypercubes [5, 6], crossed cubes [7], even networks [8], odd networks [9], folded hyper-stars [10], multidimensional torus networks [11], recursive circulant graphs [12], Gaussian networks [13], 2-chordal rings [14], and so on.

In comparison with the hypercube, the crossed cube has good properties such as lower vertex degree and diameter, higher connectivity, symmetry, and etc [15, 16]. But in the real world, it is difficult to build interconnection networks based on crossed cubes, because according to the structure of crossed cubes, a computer is needed to connect many neighbor computers. However, to install two Ethernet cards in a computer is feasible and as a consequence the extended crossed cube ECC is proposed [17], which is easy to be deployed and has low ratio between number of vertices and diameter.

In this paper, we study a secure message transmission scheme in the ECC by constructing multiple ISTs rooted at an arbitrary vertex. Firstly, algorithms to construct spanning trees and ISTs in S_n are proposed. Then, n vertex-disjoint paths between any two vertices are discussed, the maximum length of which is shown to be no more than $2n+3$. Finally, we have simulated the vertex-disjoint paths in S_4 based on JUNG.

Preliminaries

A binary string v of length n will be written as $v_{n-1}v_{n-2}...v_0$, where v_{n-1} and v_0 are the most significant bit and the least significant bit, respectively. The bit $v_i \in \{0,1\}$ is called the ith bit of v for integer i with $0 \le i \le n-1$. The n-dimensional crossed cube, denoted as CQ_n, has 2^n vertices. Each vertex in CQ_n is represented by a unique binary string of length n. We adopt the extended network of crossed cube ECC, denoted as S_n, from [17].

Definition 2.1. [17] S_n is defined as follows. S_1 is a complete graph on two vertices 0[0],1[0]. S_2 is a cycle with the vertices set {00[0], 00[1], 01[0], 01[1], 10[0], 10[1], 11[0], 11[1]} and edge set {(00[0], 00[1]), (00[1], 10[1]), (10[0], 10[1]), (10[0], 11[0]), (11[0], 11[1]), (11[1], 01[1]), (01[0], 01[1]), (00[0], 01[0])}. For $n \ge 2$, S_n is the network to extend every vertex in CQ_n into a complete graph on n vertices. The vertex set of S_n is denoted as $V(S_n) = \{u[i] \mid u \in V(CQ_n)$, for $i = 0,1,...,n-1\}$. For any two vertices $u[j] \in V(S_n)$ and $v[k] \in V(S_n)$, where $0 \le k, j \le n-1$, $(u[j], v[k]) \in V(S_n)$ if and only if the following conditions hold:

(1) $j \ne k$ and $u = v$, and

(2) $j = k$ and $(u,v) \in E(CQ_n)$.

Let $V(G)$, $E(G)$, $\delta(G)$, $\lambda(G)$, and $D(G)$ denote the vertex set, edge set, the minimum degree, the edge-connectivity, and the diameter, respectively. As a n-regular graph, S_n has the following properties.

Lemma 2.1. [17] S_n has $n*2^n$ vertices and n^2*2^{n-1} edges.

Lemma 2.2. [17] $\delta(S_n) = n$ for $n \ge 1$.

Lemma 2.3. [17] $\lambda(S_n) = n$ for $n \ge 1$.

Lemma 2.4. [17] $D(S_n) \le n+3$.

Independent Spanning Trees

Broadcasting Tree in S_n

When we want to distribute a message from a vertex to all other vertices in S_n, broadcasting tree can be used to distribute the message in S_n. Now we give an algorithm to construct a spanning trees T. Let $u(k)$ be the root vertex, where $0 \le k \le n-1$ and $u \in V(CQ_n)$. Clearly, u can be written as $u_{n-1}u_{n-2}...u_0$. We use $N_k(u[k])$ to denote the k-neighbor of $u[k]$.

Algorithm SpanningTree
Input: $u[k]$ be the root vertex, where $0 \le k \le n-1$and $u \in V(CQ_n)$;

```
Output: A tree $T$ rooted at $u[k]$ on $S_n$;
Begin
Step 1:
1: Let $V(T)=\{v[m]\mid$ for all $v\in V(CQ_n)$ and all $0\le m\le n-1\}$.
2: Connect $u[i]$ to $u[k]$ for all $0\le i\le k-1$ and $k+1\le i\le n-1$.
Step 2:
3: for $j=0$ to $n-1$ do
4:     for each vertex $v[j]\in V(T)$ do
5:         $E(T)=E(T)\bigcup(v[j],N_j(v[j]))$.
6:         Let $m[j]=N_j(v[j])$.
7:         Connect $m[i]=m[j]$ for all $0\le i\le j-1$ and $j+1\le i\le n-1$.
8:     end for
9: end for
End
```

Figure 1 shows the construction procedures of a spanning tree rooted at 100[2] by **Algorithm** SpanningTree in S_3. Firstly, by **Step 1**, the vertices in T is created. The edges in the set {(100[2], 100[0]), (100[2], 100[1])} are also constructed (See Figure 1(a)). Secondly, Figure 1(b), Figure 1(c), and Figure 1(d) show the three iterations of construction of edges. The edge sets are constructed as {(100[0], 101[0]), (101[0], 101[2]), (101[0], 101[1])},{(101[1], 111[1]), (111[1], 111[2]), (111[1], 111[0]), (100[1], 110[1]), (110[1], 110[2]), (110[1], 110[0])}, {(100[2], 000[2]), (000[2], 000[1]), (000[2], 000[0]), (101[2], 011[2]), (011[2], 011[1]), (011[2], 011[0]), (111[2], 001[2]), (001[2], 001[1]), (001[2], 001[0]), (110[2], 010[2]), (010[2], 010[1]), (010[2], 010[0])}, respectively.

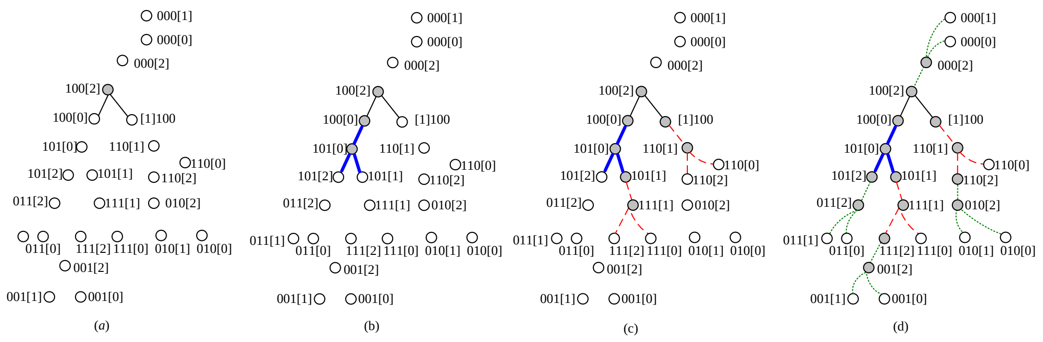


Figure 1. Demonstration of different steps of edges constructed by **Algorithm** SpanningTree in S_3: (a) **Step 1**; (b) First iteration **Step 2**; (c) Second iteration of **Step 2**; (d) Third iteration of **Step 2**.

Here, we also use line types such as solid line, bold solid line, dashed lines, dotted lines to denote the above edges in the four sets, respectively. By **Algorithm** SpanningTree, $V(T)=\{v[m]\mid$ for all $v\in V(CQ_n)$ and all $0\le m\le n-1\}$. Based on **Definition 2.1**, it is clear that T contains all vertices in S_n, which is extended from CQ_n from the aspect of vertices, thus it also has potential recursive construction. Using the recursive relation, we can verify that T is an undirected graph in which any two vertices are connected by exactly one path. Thus, the following lemma holds. By computing the relation of edges and iteration times, the height of T is $2n+1$. Then, the following theorem holds.

Theorem 3.1. Given the input $u[k]$ as the root, where $0 \le k \le n-1$ and $u \in V(CQ_n)$, T constructed by **Algorithm** SpanningTree is a spanning tree on S_n with the height $2n+1$.

In **Algorithm** SpanningTree, line 4 " **for** each vertex $v[j] \in V(T)$ **do** " can be parallized, thus the broadcasting tree can be constructed by n steps with the time complexity $O(n)$. The unicast path between any two vertices and the multicast tree rooted at an arbitrary vertex can also be obtained. By **Algorithm** SpanningTree, the length of the unicast path and the height of the multicast tree are no more than $2n+1$.

Independent Spanning Trees in S_n

In the above section, the root vertex of the spanning trees T constructed by **Algorithm** SpanningTree has multiple children. But according to the definition of independent spanning trees (ISTs) [1, 2], if we want to obtain n ISTs in S_n, then the root of every IST can have only one child.

Now we give a revised algorithm to construct independent spanning trees in S_n based on **Algorithm** SpanningTree.

```
Algorithm IST
Input: $u[k]$ be the root vertex, where $0 \le k \le n-1$ and $u \in V(CQ_n)$;
Output: $n$ trees $T_0$, $T_1$, . . . , $T_{n-1}$ rooted at $u[k]$ on $S_n$;
Begin
1: for $l = 0$ to $n-1$ do in parallel
2:         Let $V(T_l) = \{v[m] \mid$ for all $v \in V(CQ_n)$ and all $0 \le m \le n-1\}$.
3:         Let $w_l = N_l(u[l])$.
5:         Connect $w[i]$ to $w[l]$ for all $0 \le i \le k-1$ and $k+1 \le i \le n-1$.
6:     for $j = 0$ to $n-1$ do
7:         for each vertex $v[j] \in V(T)$ do
8:             $E(T_l) = E(T_l) \cup (v[j], N_{(j+1) \bmod n}(v[(j+1) \bmod n]))$.
9:             Let $m[(j+1) \bmod n] = N_{(j+1) \bmod n}(v[(j+1) \bmod n])$.
10:            Connect $m[i]$ to $m[j]$ for all $0 \le i \le (j-1) \bmod n$ and
                   $(j+l) \bmod n \le i \le n-1$.
11:        end for
12:    end for
13:    if $(l = k)$ then
14:        Adjust vertex $u[i]$ as the children of vertex $N_i(u[i])$
               for $i = 0,1,\ldots,k-1,k+1,k+2,\ldots,n-1$.
15:    end if
16: end for
End
```

Figure 2 shows the three ISTs rooted at 000[1] in S_3 constructed by **Algorithm** IST. The construction of T_0 and T_2 is similar to the tree constructed by **Algorithm** SpanningTree(See Figure 2(a) and Figure 2(c)). Every tree can be constructed by several iterations. For tree T_1,based on lines 13-15 of **Algorithm** SpanningTree, we have deleted the edges (000[1], 000[0]) and (000[1], 000[2]). Two new edges (100[2], 000[2]) and (001[0], 000[0]) are appended(See Figure 2(b)). Since the n trees of **Algorithm** IST can be constructed in parallel, based on **Definition 2.1** and the ISTs results in CQ_n, we can prove that the following theorem holds.

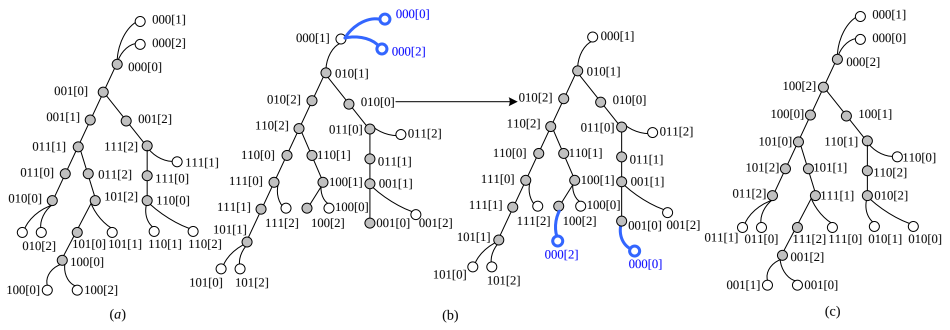


Figure 2. Three ISTs rooted at 000[1] in S_3 :(a) T_0; (b) T_1;(c) T_2

Lemma 3.1. Algorithm IST constructs n trees rooted at vertex $u[k]$ in $O(N)$ time in S_n, where $N = n*2^n$ is the vertex number of S_n.

Theorem 3.2. For an integer n with $n \geq 1$ and an arbitrary vertex $u[k]$ on S_n, T_0, T_1, . . . , and T_{n-1} constructed by **Algorithm** IST are n ISTs rooted at vertex $u[k]$ in S_n.

Discussion of Vertex-disjoint Paths in S_n

Based on **Algorithm** IST, any vertex can be the root vertex of ISTs is arbitrary, thus we can construct vertex-disjoint paths between any two vertices. The three vertex-disjoint paths between vertices 000[1] and 101[1] are as follows.

000[1]→ 000[0]→ 001[0]→ 001[1]→ 011[1] → 011[2] → 101[2]→ 101[1]
000[1]→ 010[1]→ 010[2]→ 110[2]→ 110[0] → 111[0] → 111[1] →101[1]
000[1]→ 000[2]→ 100[2]→ 100[0]→ 101[0] → 101[1]

The iterative sequence in **Algorithm** IST is 0, 1, ..., $n-1$. Since the root vertex of **Algorithm** IST is arbitrary, if we change the sequence, we can obtain another set of three vertex-disjoint paths between any two vertices. For example, using the sequence 0, 2, 1, we can obtain another three vertex-disjoint paths between 000[1] and 110[0].

000[1]→ 000[0] → 001[0]→ 001[2]→ 111[2]→ 111[1]→ 101[1]
000[1]→ 010[1]→ 010[0]→ 011[0]→ 011[2]→ 101[2]→ 101[1]
000[1]→ 000[2]→ 100[2]→ 100[1]→ 101[1]

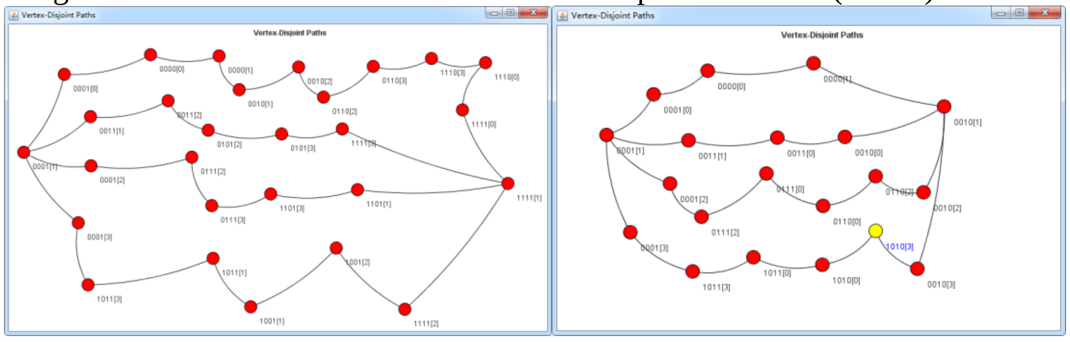
Observing the above two set of vertex-disjoint paths, the lengths of the former are 7, 7, and 5, respectively. The total length of the paths is 19. The lengths of the latter are 6, 6, and 4, respectively, and the total length of which is 16, which is lower than the former. The maximum lengths of the paths in two sets are 7 and 6, respectively, both of which are lower than the height of any of the IST.

Now we discuss the maximum length of the vertex-disjoint paths. By **Definition 2.1**, for any two vertices $v[i]$ and $v[j]$ in S_n, where $0 \le i < j \le n-1$, $v[i]$ is connected with $v[j]$. By **Algorithm** IST, the number of iterations is n, the root vertex is the child of the input vertex. Thus, for any tree, the number of the vertices in the path from the root vertex to the farthest leaf vertex is $2*(n+2)-1 = 2n+3$. Thus, we have the following theorem.

Theorem 3.3. For any two vertices $u[k]$ and $v[i]$ $(v[i] \ne u[k])$ in S_n, the maximum length of the n vertex-disjoint paths between the two vertices is $2n + 3$.

Simulation of Vertex-disjoint Paths

Now we simulate the construction of vertex-disjoint paths between two vertices based on Algorithm IST and Java Universal Network/Graph Framework (JUNG).



(a) (b)

Figure 3. Vertex-disjoint paths between vertices (a) 0000[1] and 1111[1]; (b) 0000[1] and 0010[1]

Figure 3(a) shows the four vertex-disjoint paths between vertices 0001[1] and 1111[1]. We can see that the maximum length of the four paths is 11 and the total length of the four paths is 30. In another case, let 0010[1] be the destination vertex, the maximum length of the four paths becomes 7 and the total length of the four paths is 22, which is shown in Figure 3(b).

Here, if we transmit secure data in the latter four paths, the efficiency is higher than the former four paths. In addition, in a large network of computers based on S_n. If we want to choose different computers to backup data, the maximal length of the path is lower, then the efficiency may be higher. Thus, we can choose the proper vertices based on our real needs.

Conclusions

In this paper, we have discussed a secure message distribution scheme in an extended network of crossed cubes ECC by constructing independent spanning trees. Based on the set of independent spanning trees constructed by **Algorithm** IST, we can also obtain the n vertex-disjoint paths between any two vertices, the maximum length of which is $2n + 3$.

Acknowledgment

This work is supported by National Natural Science Foundation of China (No. 61572337 and No. 61502328), China Postdoctoral Science Foundation Funded Project (No. 2015M581858), the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (No. 14KJB520034 and No. 15KJB520032), and the Jiangsu Planned Projects for Postdoctoral Research Funds (No. 1501089B) and the eighteenth batch of college students extracurricular academic research fund projects.

References

[1] A. Itai and M. Rodeh, The multi-tree approach to reliability in distributed networks, Inform. Comput. 79(1) (1988) 43-59.

[2] A. Zehavi and A. Itai, Three tree-paths, J. Graph Theor. 13(2) (1989) 175-188.

[3] J. Cheriyan and S. N. Maheshwari, Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs, J. Algorithm. 9(4) (1988) 507-537.

[4] S. Curran, O. Lee, and X. Yu, Finding four independent trees, SIAM J. Comput. 35(5) (2006) 1023-1058.

[5] K. Obokata, Y. Iwasaki, F. Bao, and Y. Igarashi, Independent spanning trees of product graphs and their construction, IEICE T. Fund. Electr. E79-A(11)(1996) 1894-1903.

[6] S.-M. Tang, Y.-L. Wang, and Y.-H. Leu, Optimal independent spanning trees on hypercubes, J. Inf. Sci. Eng. 20(1) (2004) 143-155.

[7] B. Cheng, J. Fan, X. Jia, and J. Wang, Dimension-adjacent trees and parallel construction of independent spanning trees on crossed cubes, J. Parallel Distr. Com. 73(5) (2013) 641-652.

[8] J.-S. Kim, H.-O. Lee, E. Cheng, and L. Lipták, Independent spanning trees on even networks, Inf. Sci. 181(13) 2892-2905.

[9] J.-S. Kim, H.-O. Lee, E. Cheng, and L. Lipták, Optimal independent spanning trees on odd graphs, J. Supercomput. 56(2) (2011) 212-225.

[10] J.-S. Yang and J.-M. Chang, Independent spanning trees on folded hyper-stars, Networks 56(4) (2010) 44-53.

[11] S.-M. Tang, J.-S. Yang, Y.-L. Wang, and J.-M. Chang, Independent spanning trees on multidimensional torus networks, IEEE Trans. Comput. 59(1)(2010) 93-102.

[12] J.-S. Yang, J.-M. Chang, S.-M. Tang, and Y.-L. Wang, On the independent spanning trees of recursive circulant graphs G(cd^m, d) with d > 2," Theoret. Comput. Sci. 410 (21-23) (2009) 2001-2010.

[13] Z. Hussain, B. AlBdaiwi, and A. Cerny, Node-independent spanning trees in Gaussian networks, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2016, 24-29.

[14] Y. Hamada, Independent spanning trees of 2-chordal rings, IEICE T. Fund. Electr. 99(1) (2016) 355-362.

[15] C.-P. Chang, T.-Y. Sung, and L.-H. Hsu, Edge congestion and topological properties of crossed cubes, IEEE T. Parall. Distr. 11(1) (2000) 64-80.

[16] K. Efe, A variation on the hypercube with lower diameter, IEEE T. Comput. 40(11) (1991) 1312-1316.

[17] B.-L. Cheng, J.-X. Fan, J.-W. Yang, Z. Liu, and J.-Y. Zhou, An extended network of crossed cubes, Proc. 2016 International Conference on Computer Science and Information Security (CSIS 2016), 590-596 April 16-17, 2016, Nanjing, China.