Artificial Intelligence, Social Computing and Wearable Technologies, Vol. 113, 2023, 316–321
https://doi.org/10.54941/ahfe1004204

Design of New Routing Algorithm and Embedding for Hierarchical Hypercube Networks

Woo-Young Kim[1] and Hyeong Ok Lee[2]

[1]Department of Computer Engineering, Sunchon National University; Sunchon 315, Korea
[2]Department of Computer Education, Sunchon National University; Sunchon 315, Korea

ABSTRACT

In this paper, we propose a new interconnection network topology, hierarchical hyper-star network *HHS(Cn,Cn)* is based on hyper-star network. Hierarchical networks based on hypercube have been previously proposed; it has been shown that these networks are superior to the basic networks, in terms of various performance including diameter, network cost, fault tolerance etc. Our results show that the proposed hierarchical hyper-star network performs very competitively in comparison to hyper-star network, *HCN(n,n),* and *HFN(n,n)* have been previously proposed. We also investigate a various topological properties of the hierarchical hyper-star network *HHS(Cn,Cn)* including routing algorithm, diameter, connectivity, broadcasting.

Keywords: Interconnection network, Parallel processing, Routing algorithm, Graph

INTRODUCTION

As the need for developing high-performance parallel computers increases day by day, research on interconnection networks that have an important impact on the performance of these parallel computers has also been actively conducted.

The interconnection network is represented by a non-directional graph representing each processor as a node and a communication channel as an edge between processors. The interconnection network proposed so far can be divided into a mash class expressed as k×n, a hypercube class expressed as 2ˆn, and a star graph class expressed as n. Measures for evaluating interconnection networks include branching, connectivity, symmetry, diameter, average distance, fault diameter, network cost, broadcasting, embedding, etc.

The hypercube network has the advantages of symmetry in nodes and edges, and a simple recursive structure, making it easy to provide the network structure required in various applications. Thus, it is widely used in existing research and commercial systems. Moreover, from the perspective of embedding, there are advantages in that other network structures such as rings, trees, pyramids, and meshes can be efficiently embedded. However, there is a drawback in that the diameter and the average distance between nodes are not short compared to the branching factor. To overcome these

problems, new networks such as Hypernet, HCN (Hierarchical Cubic Network), and HHN (Hierarchical Hypercube Networks) have been proposed, which improve the network of the hypercube from the perspective of network costs by manipulating the adjacency of nodes and edges.

The HNN (Hierarchical Hypercube Networks) is a graph proposed as a new topology for large-scale parallel computers. It is a hierarchical interconnection network based on the hypercube and has a smaller node branching factor and fewer links than a hypercube with a similar number of nodes. Therefore, it is an excellent interconnection network for composing large-scale parallel computers and is a superior network compared to the widely distributed hypercube for large-scale parallel systems. When setting up routing algorithms, it uses simple routing algorithms.

However, it is inefficient because it has many unnecessary routing paths by constructing a path based on the simple routing algorithm.

Therefore, in this paper, we propose a routing algorithm with an improved average distance compared to the simple routing algorithm. Although the worst-case path value is the same, the average distance value has improved as shown in <Table 3>.

Chapter 2 will discuss the structure and characteristics of the HHN network, Chapter 3 will discuss the problems with the HHN's simple routing algorithm, Chapter 4 will present an improved routing algorithm for HHN, and finally, we will draw conclusions.

Related Research

In HHN (Hierarchical Hypercube Networks), the edges that connect nodes within a hypercube, called a cluster, are referred to as internal edges, and the edges that connect between clusters are called external edges. Figure 1 shows the structure of the HHN, from the lowest level of the hypercube to the two-dimensional level of the HHN, in a hierarchical manner.

The definition of HHN(m:n,h) is as follows (m≤n):
m: the number of clusters, 2^m
n: n^-dimensional hypercube
h: dimension (=level)

Each node is composed of $(m + nh)$ binary digits, and the node address is denoted by $(A_h, \cdots, A_1, A_0)$. The number of nodes is 2^{m+nh}, the branching factor is $n + h$, the number of edges is $(n + h)\, 2^{m+nh-1}$, the diameter is $m + hn + h$, and the average distance is $\frac{m+nh}{2} + h - \frac{\frac{m}{2}+1}{2^m} - \frac{(h-1)(\frac{n}{2}+1)}{2^n}$

Internal edges are formed by the definition of the hypercube, and external edges occur according to the following two conditions.

1. For $1 \leq j < h$, if $A_j \neq A_0$, then the node $(\cdots, A_{j+1}, A_j, A_{j-1}, \cdots, A_1, A_0)$ is connected to the node $(\cdots, A_{j+1}, A_0, A_{j-1}, \cdots, A_1, A_j)$ through the jth external edge L_j.

2. If $A_h \neq A_0^L$, then the node $A_h \neq A_0^L$ is connected to the $(A_0^L, A_{h-1}, \cdots, A_1, A_0^H | A_h)$ through the hth external edge L_h.

A_0^H represents the upper $(n - m)$ bits,

A_0^L represents the lower m bits of A_0,
$A_0^H|A_h$ is the concatenation of the two numbers A_0^HH and A_h

Simple Routing Algorithm and Problems of HHN

In HHN $(m : n, h)$, let's assume two arbitrary nodes as u=$(A_h, \cdots, A_1, A_0)$ and v=$(B_h, \cdots, B_1, B_0)$. The routing from node u to node v is as follows:

If $j < h$, the path is

$$(\cdots, A_{j+1}, A_j, A_{j-1}, \cdots, A_1, A_0)\,(\cdots, A_{j+1}, A_0, A_{j-1}, \cdots, A_1, A_j)$$
$$\mathrm{A}_{j+1},\ \mathrm{B}_j,\ \mathrm{A}_{j-1}, \cdots, A_1, A_j)$$

If $j = h$, the path is $(A_h, A_{h-1}, \cdots, A_1, A_0)\,(A_h, A_{h-1}, \cdots, A_1, A_0^H|B_h) \rightarrow (A_h, \mathrm{A}_{h-1}, \cdots, A_1, A_0^H|A_h)$

Here, $\rightarrow$ represents the internal routing path within the cluster, and $\rightarrow$ represents the external routing path between clusters.

For instance, consider the routing path from $u = (0010, 0100, 0011, 1001)$ to $v = (0110, 0011, 1000, 0111)$ within $HHN(4; 4, 3)$. The routing path based on the simple routing algorithm is as follows (the number next to $\rightarrow$ represents the number of bits during internal routing):

(0010, 0100, 0011, 1001) $\rightarrow$ 4 (0010, 0100, 0011. 0110) $\rightarrow$ (0110, 0100, 0011, 0010) $\rightarrow$ 1(0110, 0100, 0011, 0011) $\rightarrow$ (0110, 0011, 0011, 0100) $\rightarrow$ 2(0110. 0011, 0011, 1000) $\rightarrow$ (0110, 0011, 1000, 0011) $\rightarrow$ 1(0110, 0011, 1000, 0111).

So, examining the routing values, it involves eight internal routings and three external routings, making the total number of routings eleven. That is, there are eleven routing paths from node u to node v.

As seen in the above example, it's suboptimal but simple routing algorithms were proposed that are adjusted according to the order of descending levels from the top-level (B_h) to the bottom level of the destination node v. Due to this simple routing algorithm, unnecessary routing path values are generated, and it fails to obtain optimal routing path values proportional to the smaller number of links compared to the hypercube. Therefore, an improved routing algorithm is proposed in the next chapter.

Improved Routing Algorithm for HHN

Let's assume two arbitrary $u = (A_h, \cdots, A_1, A_0), v = (B_h, \cdots, B_L, B_0)$. In this context, $\Rightarrow$ denotes the routing within the hypercube, $\rightarrow$ denotes the external link routing, and tuples represent each element constituting nodes u and v.

The improved routing algorithm determines the path according to the following three steps:

Step 1: XOR($\oplus$) the corresponding values in the row and column, each made up of tuples, from nodes u and v in a matrix. Use the number of 1's as an element of the matrix.

Step 2: Pair selection

- Pair: Tuple values of the row and column of the selected element
- p: The number of $\oplus$ 1s of the tuple values paired as matrix elements.

- D (diagonal): The constituents of the matrix are diagonally as follows:

$$D = (a_h b_h,\ a_{h-1} b_{h-1},\ \cdots,\ a_1 b_1,\ a_0 b_0)$$

- A_k: The tuple value at the highest level among the elements with the same value.

1. Find p = 0 among the constituents of D and select it as a pair. Remove the column and row containing the selected element.
2. Find p = 0 in all areas excluding D and select it as a pair. Remove the column and row containing the selected element. However, if there is a value of p = 0, select the value of A_k.
3. Add 1 to each value located on the diagonal D.
4. Select the smallest remaining value among the B_0 column and remove its column and row. However, if there are the same values, select the value of A_k.
5. Select the smallest values in all areas other than the paired values, pair them in order, and remove their columns and rows, from column B_0 to B_h. If there are the same values among the given column values, select the value of A_k.
6. After all are selected as pairs, subtract 1 added to the diagonal D, and start routing as in step 3.

Step 3: Routing

- Start routing from A_0 of the start node u in the order of pairing as follows:

if A_0 is paired in the order as follows, it routes.

else if p = 0 for the value of diagonal D, it does not route because it's the same position.

else if A_0 is paired with P = 0 and is not diagonal D, it routes internally(⇒) to the given value of column B_x paired with A_0 and externally(→) to the same level value, and swaps it. However, if that value is the lowest level value (B_0) of node v, it exchanges it with the upper level value of A_0, which has not been routed among node u, externally(→).

The process continues to repeat until it reaches the destination node v.

For example, when considering the routing path from u=(0010, 0100, 0011, 1001) to v=(010, 0011, 1000, 0111) in HHN(4;4,3), the process from step 1 to step 2 by the improved routing algorithm proposed in this paper is shown in Table 1. In addition, the part marked with the point (.) relatively shows the state selected between nodes by a simple routing algorithm

The process of the routing path by step 3 is as follows. (The number next to ⇒ refers to the number of bits for internal routing.).

(0010, 0100, 0011, 1001) ⇒1 (0010, 0100, 0011, 1000) → (0010, 0100, 1000, 0011) → (0010, 0011, 1000, 0100) ⇒1 (0010, 0011, 1000, 0110) → (0110, 0011, 1000, 0010) ⇒2 (0110, 0011, 1000, 0111)

Therefore, if you look at the route value, the total number of internal routing 4 times and external routing 3 times is 7. In other words, the number of routing path values from the u node to the v node was reduced by 4 times, and the routing path value of about 36% was improved.

Table 1. Process of steps1 and 2 of improved ¡.

⊕		b_h	b_2	b_1	b_0
	0110	0011	1000	0111	
a_h	0010	1_2	1.	2	2
a_2	0100	1	3_4	2.	2
a_1	0011	2	0	3_4	1.
a_0	1001	4.	2	1	3_4

Table 2 shows the comparison of the routing result values of the two algorithms for two nodes u=(0100, 0100, 0011, 1001) and v=(010, 0011, 1000, 0111).

Table 2. Comparison of path values of two algorithms in a given example¡.

Sortation		Simple Routing Algorithm	Improved routing algorithms
ex	Internal Routing Value	8	4
	External Routing Value	3	3
	Total routing path value	11	7(36%)

In the worst case, for example, when routing to nodes u = (1111, 1111, 1111, 1111) and v = (0000, 0000, 0000, 0000, 0000), both routing algorithms show a total of 19 routing times, with 16 internal routes and 3 external routes also, HHN (2;2;2), HHN (3;3;2), HHN (4;4;3), Table 3 shows the results of comparing the average path value of the improved routing algorithm to the path value of the simple routing algorithm for any two nodes in HHN (5;5;4).

However, HHN (2;2;2) and HHN (3;3;2) referenced all values of node v after fixing one value of node u due to too many cases, and HHN (5;5;4) was extracted from one value of node u by 1/100 of the number of node v, 355,554).

Table 3. Comparison of path values of algorithms in HHN (2;2;2), HHN (3;3;2), HHN (4;4;3), and HHN (5;5;4)¡.

Sortation	Simple Routing Algorithm	Improved routing algorithms	
	average distance	average distance	Improvement value
HHN(2;2;2)	4	2.5	37.5%
HHN(3;3;2)	5.875	4	32%
HHN(4;4;3)	10.438	8	23%
HHN(5;5;4)	16.063	12	25%

Therefore, as shown in Table 3, comparing the routing path values of the simple routing algorithm and the improved routing algorithm shows that the

improved routing algorithm of this paper improves the average distance value by 32% in the case of HHN(3;3;2) compared to the simple routing algorithm.

CONCLUSION

The HHN(m;n;h) graph is a hierarchical interconnection network based on a hypercube, and it is an interconnection network that constitutes a large-scale parallel computer because it has smaller node branches and smaller links than a similar hypercube. Therefore, in this paper, a new routing algorithm is proposed as an improvement over a simple routing algorithm.

When comparing the improved routing algorithm and the simple routing algorithm proposed in this paper, the average distance value of the simple routing algorithm is the same in the worst case, but in the case of HHN(3;3;2), the average distance value of the improved routing algorithm is improved by 32% to 4.

A future research challenge is to find the optimal routing algorithm.

ACKNOWLEDGMENT

This research was funded by National Research Foundation of KOREA(NRF) grant funded by the Korea government(MSIT) (No. 2020R1A2C1012363).

MIST: Ministry of Science and ICT (Corresponding Author: Hyeong Ok Lee).

REFERENCES

K. Ghose and K. R. Desai "Hierarchical Cubic Networks." IEEE Trans. Parallel Distributed syst, Vol. 6 No. 4, pp. 427–435, 1995.

K. Hwang, and J. Ghosh, "Hypernet: A communication efficient architecture for constructing massively parallel computers." IEEE Trans. Comput. C-36 12 (Dec. 1987), 1450–1466.

F. T. Leighton, Introduction to Parallel Algorithms and Architectures; Arrays, Hypercubes, Morgan Kaufmann Publishers, 1992.

V. E. Mendia nad D. Sarkar, "Optimal Broadcasting on the Star Graph," IEEE Trans, Parallel Distributed syst, Vol. 3, No. 4, pp. 389–396, 1992.

Sang Kyun Yun and Kye Ho Park, Hierarchical Hypercube Networks (HHN) for Massively parallel Computers. Journal of Parallel and Distributed Computin 37, 194_199(1996) article no. 0119.

H. Lee, M. Kang, D. Kim, D. Seo and Y. Li, "Epidemic Vulnerability Index for Effective Vaccine Distribution against Pandemic," IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2022.

H. Lee and D. Seo, "FedLC: Optimizing Federated Learning in Non-IID Data via Label-Wise Clustering," IEEE Access, Vol. 11, pp. 42082–42095, 2023.