International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 2, Number 2, pages 234–250, July 2012

Hierarchical Dual-Net: A Flexible Interconnection Network and its Routing Algorithm

Yamin Li
Department of Computer Science
Hosei University
Tokyo 184-8584 Japan

Shietung Peng
Department of Computer Science
Hosei University
Tokyo 184-8584 Japan

Wanming Chu
Department of Computer Hardware
University of Aizu
Aizu-Wakamatsu 965-8580 Japan

Received: January 23, 2012
Revised: May 7, 2012
Accepted: June 19, 2012
Communicated by Yasuaki Ito

Abstract

In this paper, we propose a flexible interconnection network, called hierarchical dual-net (HDN), with low node degree and short diameter for constructing a large scale of supercomputer. The HDN is constructed based on a symmetric product graph (base network). A k-level hierarchical dual-net, HDN(B, k, S), contains $(2N_0)^{2^k}/(2 \times \prod_{i=1}^{k} s_i)$ nodes, where $S = \{G'_1, G'_2, \ldots, G'_k\}$, G'_i is a super-node and $s_i = |G'_i|$ is the number of nodes in the super-node at the level i for $1 \leq i \leq k$, and N_0 is the number of nodes in the base network B. The node degree of HDN(B, k, S) is $d_0 + k$, where d_0 is the node degree of the base network. The HDN structure is better than existing networks such as hypercube and 2D/3D torus with respect to the degree and diameter. Another benefit of the HDN is that we can select suitable super-nodes to control the growing speed of the number of nodes for constructing a supercomputer of the desired scale. We investigate the topological properties of the HDN and compare them to that of other networks and give efficient routing and broadcasting algorithms for the hierarchical dual-net.

Keywords: Interconnection Networks, Routing, Broadcasting, Algorithms

1 Introduction

Recently, because of the advances in computer and networking technologies, supercomputers containing hundreds of thousands of nodes have been built [10]. It was predicted that the parallel systems of the next decade will contain 10 to 100 millions of nodes [2]. The interconnection network plays an important role for achieving high-performance in such ultra-scale parallel systems. The performance of an ultra-scale parallel computers depends largely on the time complexities of communication schemes, and in turn depends on the diameter of the network.

An interconnection network consists of switches with multiple communication ports and cables connecting ports by following certain topologies. For an ultra-scale parallel computer, the traditional interconnection networks may no longer satisfy the requirements for the high-performance computations or efficient communications. For such an ultra-scale parallel computer, the node degree and the diameter will be the critical measures for the effectiveness of the interconnection networks. The node degree is limited by the hardware technologies and the diameter affects all kinds of communication schemes directly. The number of communication ports (node degree) in the network-on-chip (NoC) is typically 4 to 8 in current implementations. The off-chip interconnect switches can have tens of ports, but the cost becomes expensive as the number of ports increases. Other important measures for the effectiveness of the interconnection networks include symmetricity, scalability, and efficient routing algorithms.

The following two categories of interconnection networks have attracted a great research attention and been used in many supercomputers' implementations. One is the hypercube-like family that has the advantage of short diameters for high-performance computing and efficient communications [9]. The other is the 2D/3D mesh or torus family that has the advantage of small and fixed node degrees and easy implementations [1]. Traditionally, most supercomputers including those built by CRAY, IBM, SGI, and Intel use 3D tori or hypercubes.

However, the node degree of the hypercube increases logarithmically as the number of nodes in the systems increases; the diameter of the 2D/3D torus becomes large in an ultra-scale parallel system. To solve these problems, the hierarchical (cluster-based) architectures are proposed in literature [3, 5, 6, 8]. The supercomputers built by IBM recently, Roadrunner, adopt a new approach for the interconnection network [4]. It is a cluster-based architecture: the connection among clusters is fully connected, and the fat-tree is used for the connection inside a cluster.

In this paper, we propose a flexible interconnection network, called *Hierarchical Dual-Net* (HDN). The HDN is symmetric and can connect a large number of nodes with a small node degree, meanwhile keeping the diameter short. The HDN was motivated by recursive dual-net (RDN) [7]. The RDN can be viewed as a special case of HDN. The RDN has merits of low node degree and short diameter. The problem of the RDN is that it grows too fast in size, and there is no mechanism to control the rate of its growth. Different from the RDN, the scale of the HDN can be controlled by setting a set of suitable parameters while generating an expanded network through dual-construction. The HDN also adapts the cluster-based architecture. Compared to the Roadrunner, the HDN is symmetric, uses small number of links, and meanwhile keeps the diameter short. The HDN structure is also better than other popular existing networks such as hypercube and 2D/3D torus with respect to the degree and diameter. We investigate the topological properties of the HDN and show some examples of HDNs with simple base networks of small size. Then we compare them to other networks such as three-dimensional torus used in IBM Blue Gene/L [1], and hypercube [9]. We also establish routing and broadcasting algorithms for the hierarchical dual-net.

The rest of this paper is organized as follows. Section 2 describes the hierarchical dual-net in details. Section 3 discusses the topological properties of the hierarchical dual-net. Section 4 gives the routing algorithm. Section 5 gives the broadcasting algorithm. Section 6 concludes the paper and presents some future research directions.

2 The Hierarchical Dual-Net

We begin with a brief introduction to the recursive dual-net (RDN). The RDN is constructed recursively by a dual-construction. The dual-construction is a way to expand a given symmetric graph G of size n to a new symmetric graph G^* of size $2n^2$. It generates $2n$ copies of G as subgraphs (denoted as clusters) of G^*. Half of them, n clusters, are of class 0 and the others are of class 1. The connection method is described below. Referring to Figure 1, we assign a unique *node number*

i $(0 \leq i < n)$ to each node in G and assign a unique *cluster number* j $(0 \leq j < n)$ to each cluster of class 0 and class 1, respectively. Then, a new link connects the node i in cluster j of class 0 to the node j in cluster i of class 1, for $0 \leq i, j < n$.

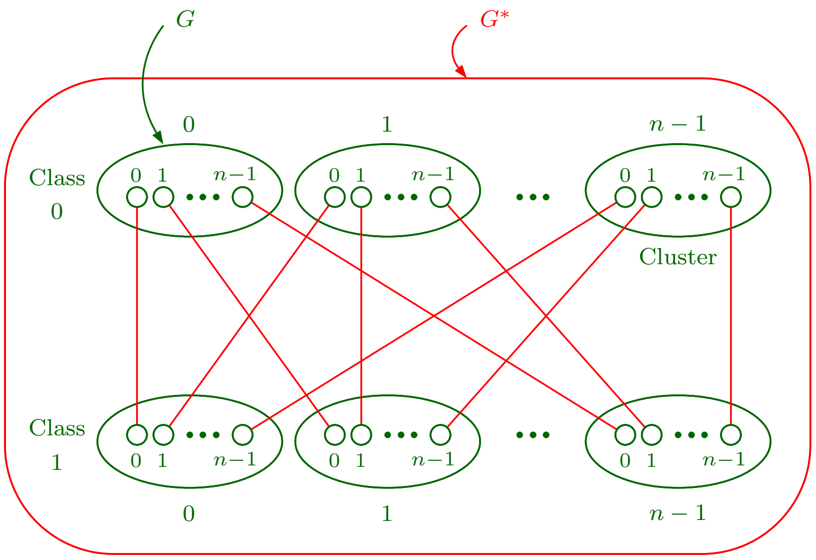

Figure 1: Dual-construction from G to G^*

If G is symmetric then the expanded graph G^* is unique and symmetric. Therefore, the dual-construction can be applied recursively from a symmetric network (the base network). $RDN(m, k)$ denotes an RDN generated from a base network of size m by applying dual-construction k times. The problem about an RDN is that its growth rate is super-exponential $((2m)^{2^k})$. There is very little space for selection of the size of an RDN. For example, let the base network be a 3-cube, then the sizes of $RDN(8, k)$ will be 2^7, 2^{15}, and 2^{31} for $k = 1, 2$, and 3, respectively. In HDN, we provide a mechanism to control the growth rate through its expansion from a base network. This new interconnection network has a very flexible way for adjusting its size.

The *hierarchical dual-net*, HDN(B, k, S), contains three sets of parameters: B is a *symmetric product graph*, we call it *base network*; k is an integer that indicates the *level* of the HDN (the number of *dual-constructions* applied); and $S = \{G'_1, G'_2, \ldots, G'_k\}$, where G'_i is a *sub-graph* of HDN$(B, k-1, S)$ and $s_i = |G'_i|$ is the number of nodes in a *super-node* at the level i for $1 \leq i \leq k$. All these terminologies will be defined in the following paragraphs.

Given r graphs $G_i = (V_i, E_i)$, $1 \leq i \leq r$, their product graph $G = G_1 \times G_2 \times \ldots \times G_r$ is defined as the graph $G = (V, E)$, where $V = \{(v_1, v_2, \ldots, v_r) | v_i \in V_i,\ 1 \leq i \leq r\}$ and $E = \{[(u_1, u_2, \ldots, u_r), (v_1, v_2, \ldots, v_r)] |$ for some j, $(u_j, v_j) \in E_j$ and for $i \neq j, u_i = v_i\}$.

In other words, the nodes of the product graph G are labeled with r-tuples, where the ith element of the r-tuples is chosen from the node set of the ith component graph. The edges of the product graph connect pairs of nodes whose labels are identical in all but the jth element, and the two nodes corresponding to the jth elements in the jth component graph are connected by an edge. Three product graph examples are shown in Figure 2.

Meshes/tori or hypercubes are typical examples of product graphs. See Figure 2(b), the 2D $p \times q$ torus is $C_p \times C_q$, where C_p and C_q are rings with p and q nodes, respectively. Any node in the torus can be represented by an ordered pair (u, v), where $u \in C_p$ and $v \in C_q$. Note that the product graph $G = G_1 \times G_2$ can be viewed as being constructed from $|V_1|$ copies of G_2 or $|V_2|$ copies of G_1. Similarly, as shown in Figure 2(c), an r-cube is a product of r numbers of K_2 (complete graph of two nodes represented by 0 and 1). So nodes in r-cube can be represented by an r-bit binary number which is an r-tuple of 0 and 1, and two nodes are connected iff they differ in exactly 1 bit.

Given a product graph $G = G_1 \times G_2 \times \ldots \times G_r$, we define a *quotient graph* Q as $Q = G/G'$ where G' is a sub-product graph of G such that $G = G' \times Q$. A node in a product graph $G =$

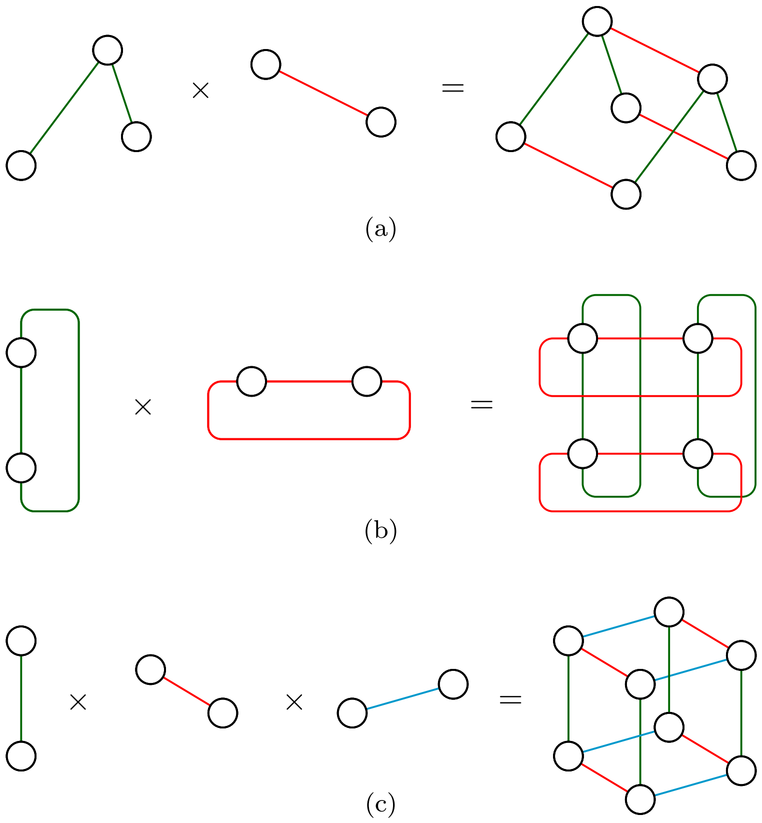

Figure 2: Product graphs

$G_1 \times \ldots \times G_i \times \ldots \times G_r$ can be represented by $(a_1, \ldots, a_i, \ldots, a_r)$ with $0 \le a_i \le |G_i| - 1$. We define a sub-graph G' as $G' = G_1^{''} \times \ldots \times G_j^{''} \times \ldots \times G_q^{''}$ with $G_j^{''} = G_i$ for $1 \le j \le q \le r$ and $1 \le i \le r$, $G_j^{''} \ne G_k^{''}$ if $j \ne k$ for $1 \le j, k \le q$. Then a node in the sub-graph G' can be represented by $(b_1, \ldots, b_i, \ldots, b_q)$ with $0 \le b_i \le |G_i^{''}| - 1$. We can consider a quotient graph Q as a reduced graph of G with G' being mapped into a single node (a super-node).

A graph G is symmetric (node-symmetric) if all its nodes looks alike. A product graph is symmetric if all its component graphs are symmetric. Both the graphs in Figures 2(b) and 2(c) are symmetric product graphs but the graph in Figure 2(a) is not. We use the symmetric product graph as the base network for generating a hierarchical dual-net through dual-constructions. We denote the base network as $B = B_1 \times B_2 \times \ldots \times B_r$ where all the B_i, $1 \le i \le r$, are symmetric. We define a super-node of B, denoted as SN as a sub-product graph of B. That is, $SN = B_{i_1} \times B_{i_2} \times \ldots \times B_{i_q}$, where $i_j, 1 \le j \le q$, are distinct and $q \le r$.

Let $|B_i| = b_i$ be the number of nodes in B_i for $1 \le i \le r$. The HDN$(B, 0, S) = B$ is the base network. For $i > 0$, the HDN(B, i, S) is generated from HDN$(B, i-1, S)$ by a construction to be explained below. Note that $S = \{G'_1, G'_2, \ldots, G'_k\}$, where G'_i is a sub-graph of HDN$(B, k-1, S)$ and $s_i = |G'_i|$ is the number of nodes in a super-node at the level i for $1 \le i \le k$. First, we define a super-node of level i, denoted as SN^i, to be a sub-product graph G'_i of size s_i in B. Then, we define graph Q^i as the quotient graph HDN$(B, i-1, S)/SN^i$. Suppose that there are N_{i-1} nodes in the HDN$(B, i-1, S)$, then the number of nodes n_i in Q^i is N_{i-1}/s_i. The s_i can be 1 or $\prod_{j=1}^{q} |B_{i_j}|$, where $1 \le i_j \le r$ and $q \le r$. That is, s_i can be a product of any number of integers in $\{b_1, b_2 \ldots, b_r\}$. For example, if $r = 3$, $b_1 = 2$, $b_2 = 3$, and $b_3 = 5$, the possible s_i can be 1, 2, 3, 5, 2×3, 2×5, 3×5, or $2 \times 3 \times 5$.

The construction of HDN(B, i, S), $1 \le i \le k$, can be defined by a two-step process: First, we perform a dual-construction on the quotient graph $Q^{i-1} =$ HDN$(B, i-1, S)/SN^i$ (HDN$(B, 0, S)$=

B). Let the graph generated by the dual-construction be Q^i, and the subgraph of two nodes that is connected by a cross-edge of level i be K_2. Second, to get the HDN(B, i, S), we replace every K_2 in Q^i by a product graph $K_2 \times SN$. We call HDN$(B, i-1, S)$ *cluster* of HDN(B, i, S).

Referring to Figure 3, an HDN(B, i, S) consists of $2n_i$ clusters which are divided into two *classes*: class 0 and class 1 with each class containing n_i clusters. That is, the number of clusters in each class is equal to the number of super-nodes in a cluster. At level i, each super-node in a cluster has s_i new links to a super-node in a distinct cluster of the other class. Because there are s_i nodes in a super-node, one node contributes a new link. The dual-construction of an RDN is a special case of the construction of an HDN with $s_i = 1$ for $1 \le i \le k$.

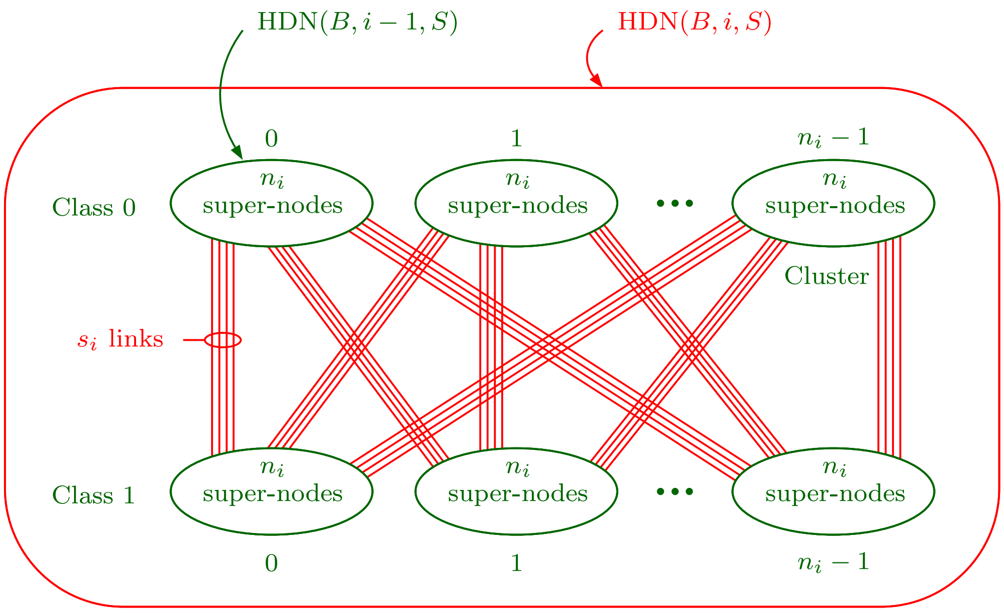

Figure 3: Build an HDN(B, i, S) from HDN$(B, i-1, S)$

The indexes of the nodes in HDN(B, k, S) can be defined as follows. Let SN^k_{id} be a *super-node_id* in a cluster of HDN(B, k, S) and N^k_{id} be a *node_id* in a super-node, then a node in the HDN(B, k, S) can be represented by $(C^k, U^k_{id}, SN^k_{id}, N^k_{id})$ where C^k is the *class_id* (0 or 1) and U^k_{id} is the *cluster_id*. A cross-edge at level k connects node $(C^k, U^k_{id}, SN^k_{id}, N^k_{id})$ and node $(\overline{C^k}, SN^k_{id}, U^k_{id}, N^k_{id})$.

Three HDN examples are shown in Figures 4, 5, and 6, where the base network is a 2-cube. Figure 4 shows an HDN$(B, 1, S)$ with $s_1 = 2$. There are 2 super-nodes (SN 0 and SN 1) in a cluster and each contains 2 nodes: node 0 and node 1. Each class has 2 clusters (the number of clusters in a class is equal to the number of super-nodes in a cluster). Figure 5 in the next page shows an HDN$(B, 2, S)$ with $s_2 = 2$, based on HDN$(B, 1, S)$. Figure 6 shows an HDN$(B, 2, S)$ with $s_2 = 4$, also based on HDN$(B, 1, S)$.

3 Topological Properties of HDN

If we use a $2 \times 3 \times 5$ torus (Figure 7) as the base network, Table 1 lists the number of nodes in HDN$(B, 1, S)$ and HDN$(B, 2, S)$ under the different configurations of S. The node degrees are 7 and 8 for HDN$(B, 1, S)$ and HDN$(B, 2, S)$, respectively, because the node degree of B is 6. From the table, we can see that the HDN covers the nodes range from several hundreds to several millions.

Suppose that the node degree of the base network B is d_0, the node degree of the HDN(B, k, S) is $d_0 + k$. Let N_{i-1} be the number of nodes in the HDN$(B, i-1, S)$. There are $N_i = 2 \times (N_{i-1}/s_i) \times N_{i-1} = 2N^2_{i-1}/s_i$ nodes in the HDN(B, i, S) for $1 \le i \le k$, where N_{i-1}/s_i is the number of clusters in each class. That is, the number of nodes in the HDN(B, k, S) is $(2N_0)^{2^k}/(2 \times \prod_{i=1}^{k} s_i)$, where N_0

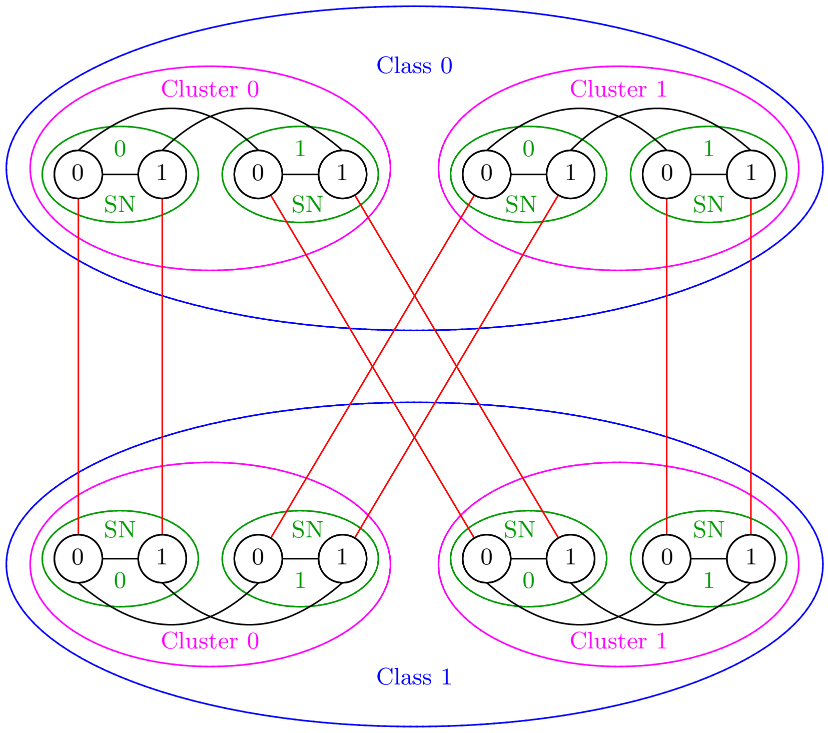

Figure 4: An HDN$(B, 1, S)$ with $s_1 = 2$

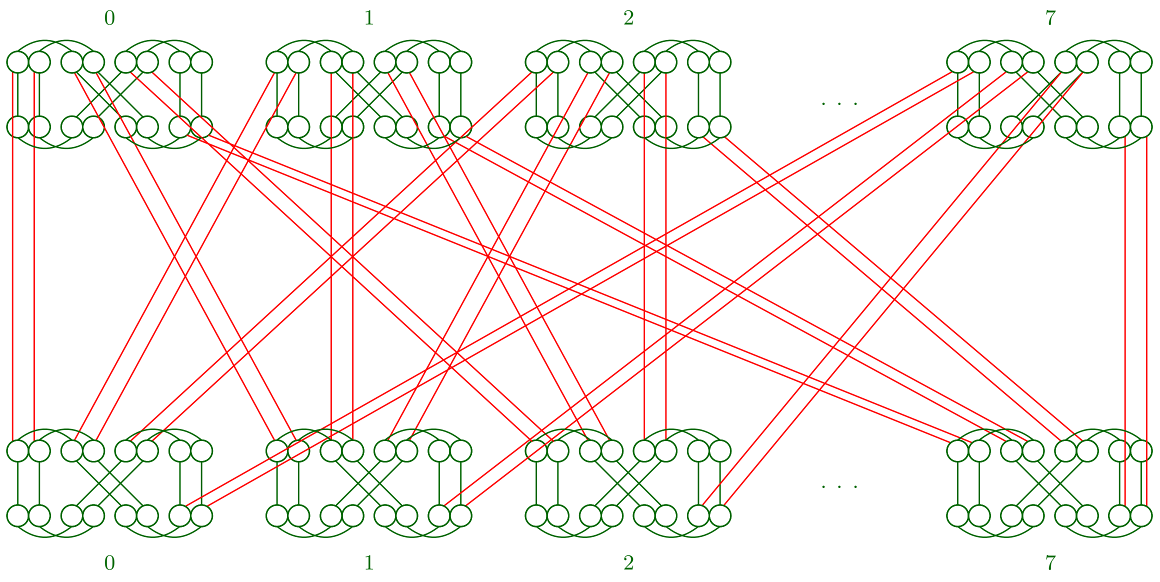

Figure 5: An HDN$(B, 2, S)$ with $s_1 = 2$ and $s_2 = 2$

is the number of nodes in the base network.

Let the diameter of the HDN$(B, i-1, S)$ be D_{i-1} and the diameter of the super-node (SN) be $D(SN_i)$. Then, if we map a super-node into a single node, the diameter of the quotient graph Q^{i-1} is $D(Q^{i-1}) = D_{i-1} - D(SN^i)$.

Referring to Figure 8, to route a node u in a cluster of class 0 (or 1) to a node v in a different cluster of the same class, we can route u along with a direct link of level i to a node u' in a cluster

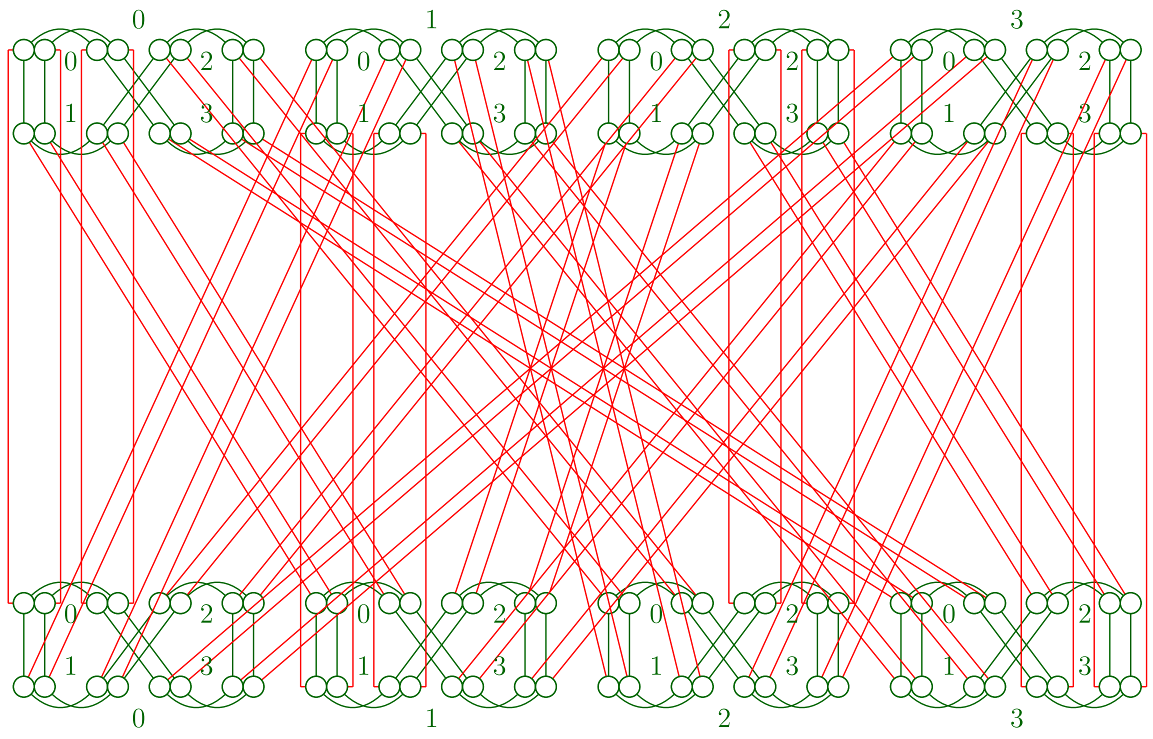

Figure 6: An HDN$(B, 2, S)$ with $s_1 = 2$ and $s_2 = 4$

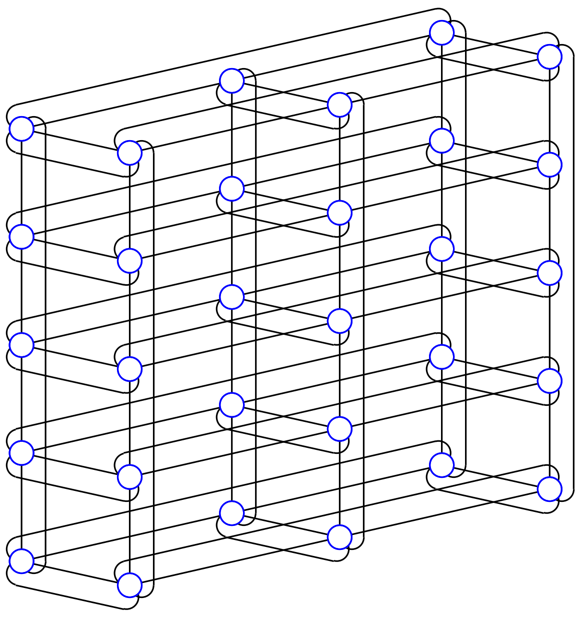

Figure 7: A base network: $2 \times 3 \times 5$ torus

of class 1 (or 0). This takes one step. Then, we route u' inside the cluster to a node w' that can reach a node w in the same cluster of node v along with direct link of level i. The longest distance between nodes u' and w' is $D(Q^{i-1})$.

Similarly, we can route node w' to a node w (by one step) and then to a node v' which is in the same super-node of v (by $D(Q^{i-1})$ steps). Finally, we route v' to node v, this takes $D(SN^i)$ steps.

Table 1: Number of nodes in HDN(B, k, S) where B is a $2 \times 3 \times 5$ torus

$k = 1$	$s_1 = 1$	$s_1 = 2$	$s_1 = 3$	$s_1 = 5$	$s_1 = 6$	$s_1 = 10$	$s_1 = 15$	$s_1 = 30$
	1,800	900	600	360	300	180	120	60

$k = 2$	$s_2 = 1$	$s_2 = 2$	$s_2 = 3$	$s_2 = 5$	$s_2 = 6$	$s_2 = 10$	$s_2 = 15$	$s_2 = 30$
$s_1 = 1$	6,480,000	3,240,000	2,160,000	1,296,000	1,080,000	648,000	432,000	216,000
$s_1 = 2$	1,620,000	810,000	540,000	324,000	270,000	162,000	108,000	54,000
$s_1 = 3$	720,000	360,000	240,000	144,000	120,000	72,000	48,000	24,000
$s_1 = 5$	259,200	129,600	86,400	51,840	43,200	25,920	17,280	8,640
$s_1 = 6$	180,000	90,000	60,000	36,000	30,000	18,000	12,000	6,000
$s_1 = 10$	64,800	32,400	21,600	12,960	10,800	6,480	4,320	2,160
$s_1 = 15$	28,800	14,400	9,600	5,760	4,800	2,880	1,920	960
$s_1 = 30$	7,200	3,600	2,400	1,440	1,200	720	480	240

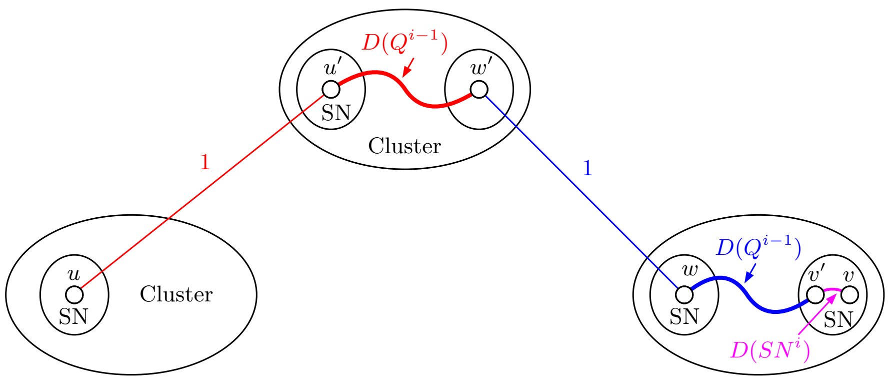

Figure 8: The diameter of HDN(B, i, S)

Therefore, we have the following recurrence:

$$\begin{aligned} D_i &= 2 \times (1 + D(Q^{i-1})) + D(SN^i) \\ &= 2D_{i-1} - D(SN^i) + 2 \end{aligned}$$

Solving the above recurrence, we get the diameter D_k of HDN(B, k, S) as below:

$$D_k = 2^k D(B) - \sum_{j=0}^{k-1} 2^j D(SN^{k-j}) + 2^{k+1} - 2$$

where $D(B)$ and $D(SN^i), 1 \leq i \leq k$, are the diameters of the base network and the super-nodes, respectively. The results of the analysis in this section are summarized in the following theorem.

The bisection width is defined as the minimum number of links that must be removed to partition the network into two equal halves. From the definition of the dual-construction, we know that there is no link between the clusters of level k that are of the same type. Referring to Figure 9, if we divide the clusters of the same type into two parts: left-part and right-part, then half super-nodes in the left-part of class 0 are connected to the super-nodes in the right-part of class 1 (the other half super-nodes in the left-part of class 0 are connected to the super-nodes in the left-part of class 1). Similarly, half super-nodes in the right-part of class 0 are connected to the super-nodes in the left-part of class 1 (the other half super-nodes in the right-part of class 0 are connected to the super-nodes in the right-part of class 1). Therefore, if the number of super-nodes in left-part is equal to the number of super-nodes in right-part, the bisection width is $N/2/2/2 + N/2/2/2 = N/4$, where N is the number of nodes in HDN(B, k, S). If the numbers are not equal, the expression of the bisection width is little bit complex, we leave it as an open question.

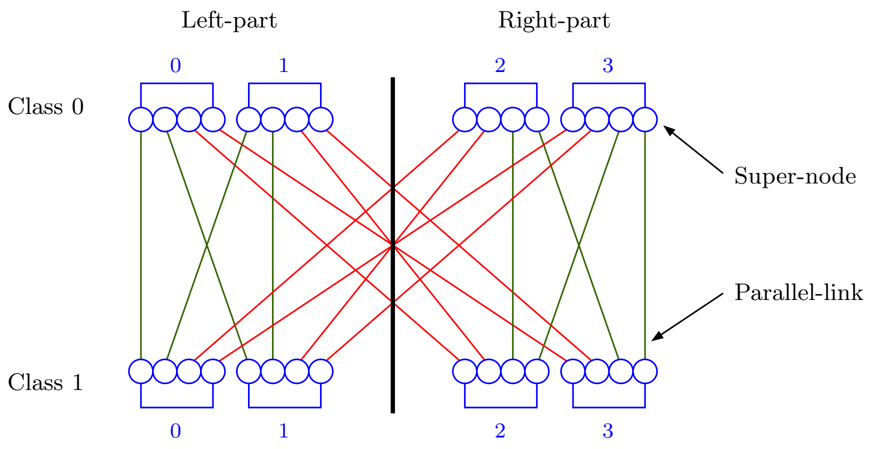

Figure 9: The bisection width of HDN(B, k, S)

Theorem 1 *Assume that the base network B is a symmetric, product graph and $SN^i, 1 \leq i \leq k$, are sub-product graphs of B with $|SN^i| = s_i$. Let the number of nodes, the node-degree, and the diameter of B be N_0, d_0, and D_0, respectively. Let the diameters of $SN^i, 1 \leq i \leq k$, be $D(SN^i)$. Let $S = \{G'_1, G'_2, \ldots, G'_k\}$, where G'_i is a sub-graph of HDN$(B, k-1, S)$ and $s_i = |G'_i|$ is the number of nodes in a super-node at the level i for $1 \leq i \leq k$. Then, the number of nodes of HDN(B, k, S) is $(2N_0)^{2^k}/(2\prod_{i=1}^{k} s_i)$, the node-degree is $d_0 + k$, and the diameter is $D_k = 2^k D(B) - \sum_{j=0}^{k-1} 2^j D(SN^{k-j}) + 2^{k+1} - 2$, where N is the number of nodes in HDN(B, k, S).*

Table 2 lists the topological properties of the torus, n-cube, CCC [8], Dual-Cube [6], RDN, and HDN. The CCC (cube-connected cycles) is obtained by replacing a node in an n-cube with an n-node cycle. The Dual-Cube is a special case of RDN with $k = 1$ and a base network of an n-cube.

In [7], we introduced the CR (cost ratio) for measuring the combined effects of the hardware cost (node degree) and the software efficiency (diameter) of an interconnection network. Instead of CR, this paper uses a more general measure, namely *weighted cost ratio* $CR_w(G)$, for the evaluation. The *weighted cost ratio* $CR_w(G)$ is defined as below. Let $|(G)|$, $d(G)$, and $D(G)$ be the number of nodes, the node degree, and the diameter of G, respectively. We define $CR_w(G)$ as

$$CR_w(G) = \frac{w_1 \times d(G) + w_2 \times D(G)}{\log_2 |(G)|}$$

where w_1 and w_2 are weights for node degree and diameter, respectively. We have $w_1 + w_2 = 100\%$.

The weighted cost ratio CR_w of an n-cube is always 1 regardless of its size and weights. The CR_w for some HDN(B, k, S) is shown in Table 3 where B is a $2 \times 3 \times 5$ torus and we assume $w_1 = w_2 = 50\%$. For simplicity, we use the number of nodes in super-nodes to represent S, instead of sub-graphs. From the table, we can see that the HDNs are more effective than hypercubes and tori measured by the weighted cost ratio although as the s_i increases, the CR_w becomes larger. The minimum CR_w shown in the list is 0.69. Unfortunately, we do not know the theoretical or experimental optimal value of CR_w up to the date we wrote this paper and it can be an open question for the future.

Table 2: Comparison of topological properties

Network	# of nodes	Degree
3D Torus	$x * y * z$	6
n-cube	2^n	n
CCC(n)	$n * 2^n$	3
Dual-Cube(n)	2^{2n-1}	n
RDN(m, k)	$(2m)^{2^k}/2$	$d_0 + k$
HDN(B, k, S)	$(2\|B\|)^{2^k}/(2\prod_{i=1}^{k} s_i)$	$d_0 + k$

Network	Diameter
3D Torus	$(x + y + z)/2$
n-cube	n
CCC(n)	$2n$
Dual-Cube(n)	$2n$
RDN(m, k)	$2^k * D_0 + 2^{k+1} - 2$
HDN(B, k, S)	$2^k(D(B) - \sum_{j=0}^{k-1} 2^j(D(SN^{k-j})) + 2^{k+1} - 2$

Table 3: CR_w with $w_1 = w_2 = 50\%$ for some HDN(B, k, S)

Network	n	d	D	CR
10-cube	1,024	10	10	1.00
3D-Tori(10)	1,000	6	15	1.05
HDN($B, 1, (1)$)	1,800	7	10	0.79
HDN($B, 1, (2)$)	900	7	9	0.82
HDN($B, 1, (3)$)	600	7	9	0.87
19-cube	524,288	19	19	1.00
3D-Tori(80)	512,000	6	120	3.32
HDN($B, 2, (2, 2)$)	810,000	8	19	0.69
HDN($B, 2, (2, 5)$)	324,000	8	18	0.71
HDN($B, 2, (5, 2)$)	129,600	8	17	0.74

4 Routing on HDN

In this section, we describe a basic node-to-node routing on HDN. We first introduce some notations needed in the proposed routing algorithm. In Section 2, we defined the product and quotient graphs. Now, we define the *difference graph* as follows. Let SN_1 and SN_2 are two super-nodes in base network B, the difference graph $SN_1 - SN_2$ is the sub-product graph of B such that $B_i, 1 \leq i \leq r$, is in $SN_1 - SN_2$ if and only if $B_i \subset SN_1$ and $B_i \not\subset SN_2$. For example, if $B = C_2 \times C_3 \times C_5$, $SN_1 = C_2 \times C_3$, and $SN_2 = C_3 \times C_5$ then $SN_1 - SN_2 = C_2$.

We also need a re-indexing process of nodes in the cluster, which is an HDN($B, i-1, S$), for routing via cross-edges of level i since the indexes of nodes in HDN$(B, i-1, S)$ is based on SN^{i-1} and the cross-edge of level i is defined based on SN^i. The index of a node in HDN$(B, i-1, S)$ contains four parts $(C^{i-1}, U^{i-1}_{id}, SN^{i-1}_{id}, N^{i-1}_{id})$ as explained in the previous section.

At the construction of the ith level, HDN$(B, i-1, S)$ becomes a cluster containing only two parts, SN^i_{id} and N^i_{id}, of the node index in HDN(B, i, S). The other two parts, C^i and U^i_{id}, are generated from the construction at the ith level. The re-indexing process that generates an 1-to-1 mapping between $(C^{i-1}, U^{i-1}_{id}, SN^{i-1}_{id}, N^{i-1}_{id})$ and (SN^i_{id}, N^i_{id}) on an HDN$(B, i-1, S)$ is necessary for the proposed routing algorithm.

Since the number of super-nodes SN^i in HDN$(B, i-1, S)$ equals to N_{i-1}/s_i, the range of SN^i_{id} is $2 \times |U^{i-1}/(SN^i - SN^{i-1})| \times |(SN^{i-1} - SN^i)|$. If $s_{i-1} = s_i$ then the re-indexing is simple: 1-1 mapping between SN^i_{id} and the 3-tuple $(C^{i-1}_{id}, U^{i-1}_{id}, SN^{i-1}_{id})$. However, when $s_{i-1} \neq s_i$, the re-indexing is a little complicate and is explained below.

Let the q-tuple, $(b_{i_1}, \ldots, b_{i_q})$ be the index of a node in a super-node SN, where $b_{i_1} \times \ldots \times b_{i_q} = |SN|$. Then the re-indexing from $(C^{i-1}, U^{i-1}_{id}, SN^{i-1}_{id}, N^{i-1}_{id})$ to (SN^i_{id}, N^i_{id}) moves the indexes of those $B_j \subset SN^i - SN^{i-1}$ into N^i_{id} and the indexes of those $B_j \subset SN^{i-1} - SN^i$ into SN^i_{id}. For example, let $B = C_2 \times C_3 \times C_5$, $s_1 = |C_2| \times |C_3| = 6$, and $s_2 = |C_3| \times |C_5| = 15$, then, the nodes in HDN$(B, 1, S)$ can be represented by $(C^1, U^1_{id}, SN^1_{id}, N^1_{id})$, where $C^1 = 0$ or 1, $0 \leq U^1_{id} < 5$, $0 \leq SN^1_{id} < 5$, and $0 \leq N^1_{id} < 6$. For the indexes of the nodes in HDN$(B, 2, S)$, we perform re-indexing of nodes in HDN$(B, 1, S)$, which is a cluster of HDN$(B, 2, S)$, to get (SN^2_{id}, N^2_{id}), where $0 \leq SN^2_{id} < 2 \times 5 \times 2 = 20$, and $0 \leq N^2_{id} < 3 \times 5 = 15$, obtained by swapping $|B_1|$ and $|B_3|$. That is, $|SN^2| = |C^1| \times |U^1| \times |B_1| = 2 \times 5 \times 2 = 20$, and $|N^2| = |B_2| \times |B_3| = 15$.

Table 4 shows four examples of re-indexing in detail for a cluster in the HDN$(B, 2, S)$ with $B = C_2 \times C_3 \times C_5$, $s_1 = 2 \times 3 = 6$, and $s_2 = 3 \times 5 = 15$. In the HDN$(B, 1, S)$, the node representation $(C^1, U^1_{id}, SN^1_{id}, N^1_{id})$ can be converted to a serial number i by $i = C^1 \times (|B|/s_1)^2 \times s_1 + U^1_{id} \times (|B|/s_1)^1 \times s_1 + SN^1_{id} \times s_1 + N^1_{id} = C^1 \times 150 + U^1_{id} \times 30 + SN^1_{id} \times 6 + N^1_{id}$. Similarly, the (SN^2_{id}, N^2_{id}) can be converted to a number $SN^2_{id} \times s_2 + N^2_{id} = SN^2_{id} \times 15 + N^2_{id}$.

Table 4: Re-indexing examples

Index in HDN$(B, 1, S)$		Index in HDN$(B, 2, S)$	
$(C^1, U^1_{id}, SN^1_{id}, N^1_{id})$	Serial number	(SN^2_{id}, N^2_{id})	Serial number
$(0, 0, 0, 0)$	$0 \times 150 + 0 \times 30 + 0 \times 6 + 0 = 0$	$(0, 0)$	$0 \times 15 + 0 = 0$
$(1, 4, 2, 3)$	$1 \times 150 + 4 \times 30 + 2 \times 6 + 3 = 285$	$(19, 0)$	$19 \times 15 + 0 = 285$
$(0, 0, 2, 2)$	$0 \times 150 + 0 \times 30 + 2 \times 6 + 2 = 14$	$(0, 14)$	$0 \times 15 + 14 = 14$
$(1, 4, 4, 5)$	$1 \times 150 + 4 \times 30 + 4 \times 6 + 5 = 299$	$(19, 14)$	$19 \times 15 + 14 = 299$

Assume that the point-to-point routing algorithm in the base network is available. The proposed algorithm for routing node u to node v in HDN(B, k, S) works as follows. We first perform re-indexing of u and v if $k > 1$. Then, there are three cases: the two nodes are in the same cluster (Case 1), in the distinct clusters of the same class (Case 2), and in the distinct clusters of distinct classes (Case 3). Case 1 is trivial. Case 3 can be reduced to Case 2 by routing u via a cross-edge of level k. Therefore, we explain only the Case 2: The two nodes are in the distinct clusters with the same class. We first identify the super-nodes, denoted as $SN^k_{u'}$ and $SN^k_{v'}$, in the two Q^{k-1}s containing u and v, respectively, such that $SN^k_{u'}$ and $SN^k_{v'}$ are connected by a unique cross-edge of level k in Q^k from the dual-construction. Then, we route node u to node u', and node v to node v' inside the clusters of level k, respectively. Notice that, u' and v' are not unique although $SN^k_{u'}$ and $SN^k_{v'}$ are unique. The algorithm finds the u' and v' that leave u^k_3 and v^k_3 unchanged if possible. And then, the routing from u to v is done by routing u' to $u'' \in SN^k_{v'}$ via a cross-edge of level k in HDN(B, k, S) and routing from u'' to v' inside $SN^k_{v'}$. The algorithm is formally presented as Algorithm 1. The correctness of the algorithm and its time complexity are given in Theorem 2.

```
Algorithm 1: HDN_routing(HDN($B, k, S$), $u$, $v$)
        input: HDN($B, k, S$);
        input: node $u = (u_0^k, u_1^k, u_2^k, u_3^k)$ (the node representation of level $k$);
        input: node $v = (v_0^k, v_1^k, v_2^k, v_3^k)$ (the node representation of level $k$);
        output: a path $u \Rightarrow v$;
begin
        if $k = 0$ then
                Base_routing($B, u, v$);
        else
                if $k > 1$ then                                        /* perform re-indexing */
                        $(u_0^{k-1}, u_1^{k-1}, u_2^{k-1}, u_3^{k-1}) \leftarrow (u_2^k, u_3^k)$;
                        $(v_0^{k-1}, v_1^{k-1}, v_2^{k-1}, v_3^{k-1}) \leftarrow (v_2^k, v_3^k)$;
                endif
                Case 1: $u_0^k = v_0^k$ and $u_1^k = v_1^k$                 /* $u$ and $v$ are in the same cluster */
                        if $k > 1$ then
                                HDN_routing(HDN($B, k-1, S$), $u$, $v$);
                        else
                                Base_routing($B, u, v$);
                        endif
                Case 2: $u_0^k \neq v_0^k$              /* $u$ and $v$ are in the clusters of distinct classes */
                        $u' \leftarrow (u_0^k, u_1^k, v_1^k, u_3^k)$;
                        $v' \leftarrow (v_0^k, v_1^k, u_1^k, v_3^k)$;
                        if $k > 1$ then                                /* perform re-indexing */
                                $((u')_0^{k-1}, (u')_1^{k-1}, (u')_2^{k-1}, (u')_3^{k-1}) \leftarrow (v_1^k, u_3^k)$;
                                $((v')_0^{k-1}, (v')_1^{k-1}, (v')_2^{k-1}, (v')_3^{k-1}) \leftarrow (u_1^k, v_3^k)$;
                                HDN_routing(HDN($B, k-1, S$), $u$, $u'$);
                                HDN_routing(HDN($B, k-1, S$), $v$, $v'$);
                        else
                                Base_routing($B, u, u'$);
                                Base_routing($B, v, v'$);
                        endif
                        route $u'$ to $u''$ via a cross-edge of level $k$;        /* $u'' = (v_0^k, v_1^k, u_1^k, u_3^k)$ */
                        Base_route($B, u'', v'$);          /* route from $u_3^k$ to $v_3^k$ inside the super-node */
                Case 3: $u_0^k = v_0^k$ and $u_1^k \neq v_1^k$       /* $u$ and $v$ are in the clusters of the same class */
                        route $u$ to $w$ via the cross-edge of level $k$;
                        route node $w$ to node $v$ as in Case 2;
        endif
end
```

Theorem 2 *Assume that the routing algorithms in the base network B is available. In HDN(B, k, S) for $k > 0$, routing between any two nodes can be done in at most $2^k R(B) - \sum_{j=0}^{k-1} 2^j R(SN^{k-j}) + 2^{k+1} - 2$ steps, where $R(B)$ and $R(SN^i), 1 \leq i \leq k$, are the time complexities of the routing in B and SN^i, respectively.*

Proof: We show the correctness of Algorithm 1 by induction on k. Assume that the algorithm is correct for $k - 1 \geq 0$. From the algorithm, it is clear that we need to consider only Case 2. In Case 2, nodes u' and u are in the same cluster by the definition of u'. They can be connected by the induction hypothesis. Similarly, nodes v' and v can be connected. The node u'' that is connected to u' by a cross-edge of level k and node v' are in the same super-node as can be seen from their IDs. Therefore, they can be connected by Base_routing algorithm. Next, we derive the time complexity R_k of the algorithm. In Case 2, there are two recursive calls to connect u to u' and v to v', respectively. Since the nodeIDs of u and u' are the same (so are v and v'), a recursive call takes only $R_{k-1} - R(SN^k)$ time. Since the SupernodeIDs of u'' and v' are the same, the last call to Base_route to connect u'' to v' takes only $R(SN^k)$ time. In Case 3, there is an additional routing step via a cross-edge. Therefore, the time complexity R_k of HDN_Routing(HDN$(B, k, S), u, v)$ satisfies the recurrence $R_k = 2(R_{k-1} - R(SN^k)) + R(SN^k) + 2$ for $k > 0$. Solving this recurrence, we have

$$R_k = 2^k R(B) - \sum_{j=0}^{k-1} 2^j R(SN^{k-j}) + 2^{k+1} - 2$$

where $R(B)$ and $R(SN^i), 1 \leq i \leq k$, are the time complexities of the routing in B and SN^i, respectively. □

Example: Routing $u = (0, 0, 0, 0)$ to $v = (1, 19, 19, 14)$ on HDN$(B, 2, S)$, where $B = C_2 \times C_3 \times C_5$, a $2 \times 3 \times 5$ torus as shown as in Figure 7, $s_1 = 2 \times 3 = 6$, and $s_2 = 3 \times 5 = 15$. Figure 10 shows an HDN$(B, 1, S)$ with $|SN^1| = s_1 = 6$, where super-node SN^1 is a 2×3 torus. A cycle in the figure is a super-node. Because there are 5 super-nodes in the base network, the number of clusters in each class is 5 (numbering from 0 to 4 in the figure). A line connecting two super-nodes of distinct classes denotes 6 links, one link per node in the super-node. There are $2 \times 5 \times 5 \times 6$, or 300 nodes in the HDN$(B, 1, S)$.

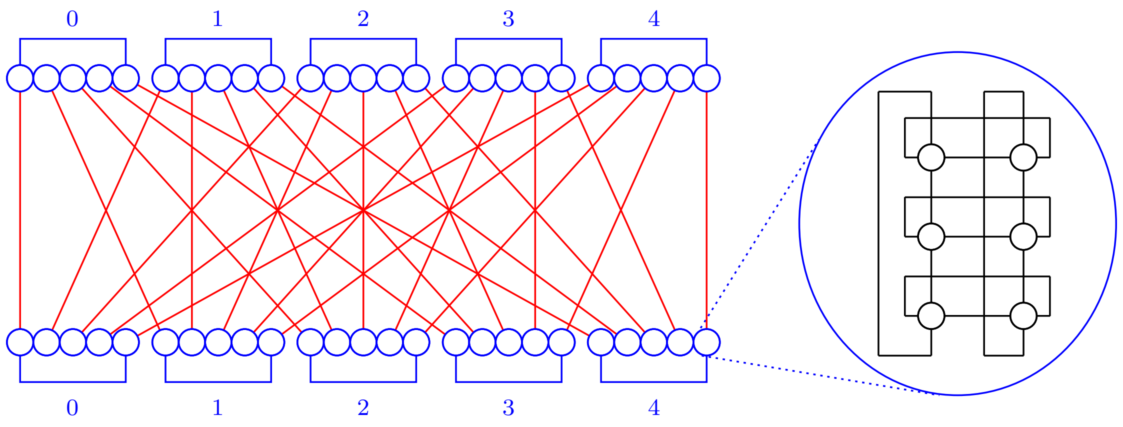

Figure 10: The HDN$(B, 1, S)$ with $|SN^1| = s_1 = 6$

Figure 11 shows an HDN$(B, 2, S)$ with $s_1 = 6$ and $|SN^2| = s_2 = 15$. A cluster in HDN$(B, 2, S)$ is an HDN$(B, 1, S)$ as shown in Figure 10. Because $s_2 = 15$, a 3×5 torus is mapped into a super-node SN^2. There are 2 super-nodes in the base network, and each has 15 nodes. In a cluster, there are $300/15 = 20$ super-nodes. Therefore, there are 20 clusters in each class, numbering from 0 to 19 in the figure. The number of nodes in total is $2 \times 20 \times 300 = 12{,}000$. A small cycle in the figure is a super-node which is a 3×5 torus and a big cycle is a cluster. The level 1 links inside the clusters are not shown in the figure. The two super-nodes shown with solid cycles, x and y, contain u and v, respectively. The thick line connects two super-nodes with plus mark (w_0 and w_1) in which u' and v' reside, respectively. Similarly, a line connecting two super-nodes of distinct classes denotes 15 links, one link per node in the super-node.

From the algorithm, we have $u' = (0, 0, 19, 0)$ and $v' = (1, 19, 0, 14)$. The two nodes u' and v' are nodes inside the super-nodes w_0 and w_1, respectively, as shown in Figure 11. For routings inside HDN$(B, 1, S)$, we should perform re-indexing from (SN^2_{id}, N^2_{id}) to $(C^1_{id}, U^1_{id}, SN^1_{is}, N^1_{id})$. As explained in the re-indexing process, we have $(0, 0) \rightarrow (0, 0, 0, 0)$; $(19, 0) \rightarrow (1, 4, 2, 3)$; $(0, 14) \rightarrow (0, 0, 2, 2)$; and $(19, 14) \rightarrow (1, 4, 4, 5)$.

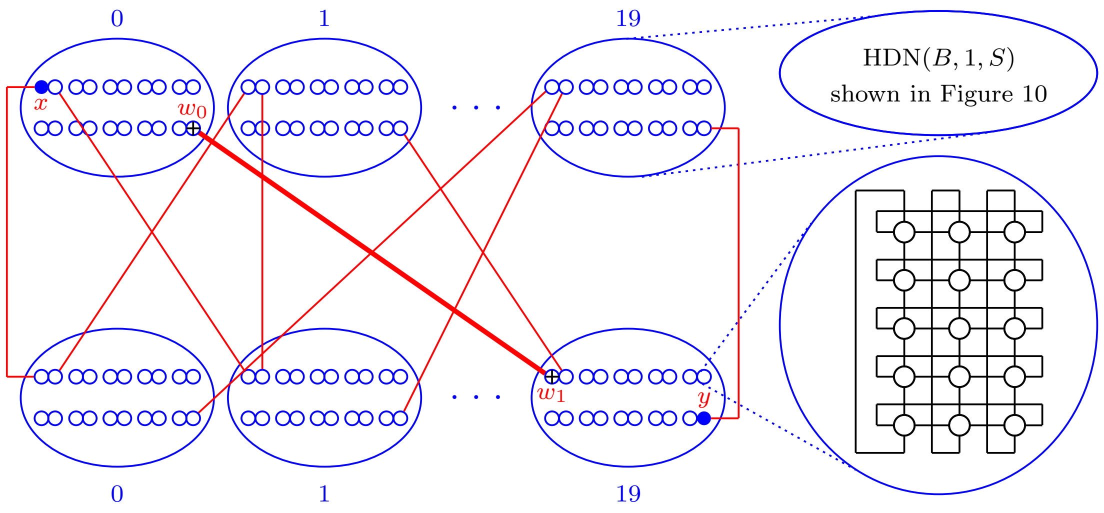

Figure 11: The HDN$(B, 2, S)$ with $|SN^2| = s_2 = 15$

The full path from $u = (0, 0, 0, 0)$ to $v = (1, 19, 19, 14)$ is shown below, where "$\rightarrow$" denotes a direct link and "$\Rightarrow$" is a sub-path:

- Routing from u to u' in the cluster with ID $(C^2_{id}, U^2_{id}) = (0, 0)$: $(0, 0) = (0, 0, 0, 0)$ (by re-indexing) $\rightarrow (0, 0, 4, 0) \rightarrow (1, 4, 0, 0) \rightarrow (1, 4, 1, 0) \rightarrow (1, 4, 2, 0) \rightarrow (1, 4, 2, 1) \rightarrow (1, 4, 2, 2) \rightarrow (1, 4, 2, 3) = (19, 0)$ (by re-indexing);
- Routing from v to v' in the cluster with ID $(C^2_{id}, U^2_{id}) = (1, 19)$: $(19, 14) = (1, 4, 4, 5)$ (by re-indexing) $\rightarrow (1, 4, 0, 5) \rightarrow (0, 0, 4, 5) \rightarrow (0, 0, 3, 5) \rightarrow (0, 0, 2, 5) \rightarrow (0, 0, 2, 0) \rightarrow (0, 0, 2, 1) \rightarrow (0, 0, 2, 2) = (0, 14))$ (by re-indexing);
- Routing from u' to v' in HDN$(B, 2, S)$: $u' = (0, 0, 19, 0) \rightarrow (1, 19, 0, 0) = u'' \Rightarrow (1, 19, 0, 14) = v'$.

The sub-path $u'' \Rightarrow v'$ is the routing path of length 2 inside the SN^2, a 3×5 torus: $(0, 0) \rightarrow (2, 0) \rightarrow (2, 4)$. Therefore, the length of the path $u \rightarrow v$ is $7 + 7 + 1 + 2 = 17$.

5 Broadcasting on HDN

We assume the 1-port communication model in which a node can not send (receive) the message simultaneously to (from) more than one of its neighbors. Assume also that the broadcasting algorithm in each component $B_i, 1 \leq i \leq r$, of the base network B is available.

First, we describe a subroutine for broadcasting in a super-node SN. Without loss of generality, assume that $SN = B_1 \times \ldots \times B_q$, $q \leq r$. Let the source node $u = (u_1, \ldots, u_r) \in SN$. The broadcast from u to all nodes $(*, \ldots, *, u_{q+1}, \ldots, u_r)$ inside SN can be done easily as follows. We first broadcast message to nodes $(*, u_2, \ldots, u_r)$ by broadcasting algorithm for B_1. Then, we broadcast message to nodes $(*, *, u_3, \ldots, u_r)$ by broadcasting algorithm for B_2 using nodes $(*, u_2, \ldots, u_r)$ as source nodes. We continue until all nodes $(* \ldots, *, u_{q+1}, \ldots, u_r)$ in the super-node where u belongs to have the message. It is easy to see from the above description that the time for broadcasting in SN is $T(B_1) \times \ldots \times T(B_q)$, where $T(B_i), 1 \leq i \leq q$, is the time to broadcast in B_i. We call this subroutine broadcast(SN).

The proposed broadcasting algorithm in HDN(B, k, S) for $k \geq 1$ is a recursive algorithm, where the source SN of broadcasting is a set of nodes that hold message. The recursive algorithm, formally specified in Algorithm 2, has two inputs: an HDN of level $k > 0$ and the source SN^k. The algorithm works as follows. First, we perform broadcasting in a SN^{k-1} with the source $SN^{k-1} \cap SN^k \neq \emptyset$ if $k > 1$. Second, we have the first recursive call for $k-1$ to broadcast message in the cluster C of level k with the source SN^{k-1}. Third, we spread the message from C to all clusters of distinct class with the class of C via cross-edges of level k. Forth, we have the second recursive calls on those clusters in which there exists a super-node SN^k just receiving message. Fifth, the nodes in the broadcasted clusters except the cluster C pass the message over the cross-edges of level k again. After the five steps above, the broadcasting is done.

The algorithm for broadcasting from a source node s in HDN(B,k,S) with $k>0$ containing two phases. First, call broadcast(SN^k), and then call HDN_broadcast(HDN(B,k,S), SN^k). The results for broadcasting in HDN are shown in Theorem 3.

```
**Algorithm 2:** HDN_broadcast(HDN($B,k,S$), $SN^k$)
          input: HDN($B,k,S$);
          input: a super-node $SN^k$ that holds a message;
          output: broadcast the message;
begin
          if $k = 1$ then
                    $C \leftarrow$ broadcast($B/SN^1$);
          else
                    broadcast($SN^{k-1}/(SN^k \cap SN^{k-1})$);
                    $C \leftarrow$ HDN_broadcast(HDN($B, k-1, S-\{s_k\}$),$SN^{k-1}$);
          endif
          for each $u$ in $C$ do
                    send message to $u'$ via the cross-edge of level $k$;
          endfor
          if $k = 1$ then
                    for each $SN^1$ containing $u'$ do
                              broadcast($B/SN^1$);
                    endfor
          else
                    for each $SN^{k-1}$ containing $u'$ do
                              HDN_broadcast(HDN($B, k-1$), $S-\{s_k\}$),$SN^{k-1}$);
                    endfor
          endif
          for each $v$ in the clusters of level $k$ that are fully broadcasted except cluster $C$ do
                    send message to $v'$ via the cross-edge of level $k$;
          endfor
end
```

Theorem 3 *Assume the one-port communication model. Assume also that the broadcasting algorithms in each component $B_i, 1 \leq i \leq r$, of the base network B is available. In HDN(B,k,S) for $k>0$, broadcasting from any source node can be done in $2^k T(B) - \sum_{i=0}^{k-1} 2^i T(SN^{k-i}) + 2^{k+1} - 2$ steps, where $T(B)$ and $T(SN^i), 1 \leq i \leq k$, are the time complexities of broadcasting in B and SN^i, respectively.*

Proof: The correctness of the recursive algorithm can be derived easily from the induction and the five steps description of Algorithm 2.

Let the time complexity of algorithm HDN_broadcast(HDN(B,k,S),SN^k) be T_k. Then, T_k satisfies the following recurrence equation for $k>1$:

$$T_k \leq T(SN^{k-1}) + 2T_{k-1} + 2$$

and for $k = 1$:

$$T_1 = 2(T(B) - T(SN^1)) + 2$$

Solving the recurrence, we get the time complexity for broadcasting in HDN(B, k, S) from a source node as below:

$$T(SN^k) + T_k \leq 2^k T(B) - \sum_{i=0}^{k-1} 2^i T(SN^{k-i}) + 2^{k+1} - 2$$

where $T(B)$ is the time complexity of the broadcasting in B, and $T(SN^i), 1 \leq i \leq k$, is the time complexity of the broadcasting in SN^i. $\square$

6 Concluding Remarks

In this paper, we proposed a family of flexible high-performance interconnection network, called HDN, that is practical and suitable for a wide range of large-scale parallel computers. The HDN is practical in many aspects. First, it uses a product graph such as a torus as a base network. The torus is a very practical network for parallel computers. However, for the supercomputers of next generations that might contain millions of nodes, the torus networks can not fulfill the task as high performance networks. Second, the hierarchical, cluster-based architecture is becoming popular. We think that the reason the IBM new supercomputer, Roadrunner, changed its architecture from traditional 3D torus to the cluster-based, fat-tree network is a symptom of such trend. The new architecture of Roadrunner bears some similarity with the HDN. From the current speed of advancing technology, we can expect that the hierarchical cluster-based network that uses torus network as its base network such as the proposed HDN will become practical and promising in the near future.

In order to show that the proposed HDN can be a good candidate of interconnection networks for the large scale supercomputers, there are some important research issues that should be addressed in the future. We list some of these issues below.

1. Efficient embedded schemes: For example, is the HDN Hamiltonian? If yes, what is the scheme for constructing Hamiltonian cycle?

2. Fault-tolerance capacity: For example, are there efficient fault-tolerant routing algorithms handling node failures?

3. Efficient collective communication: For example, flexible and efficient algorithms for multicast, all-to-all broadcast, total exchange, etc.

4. Efficient algorithms for basic computations in computer science.

Other important issues include the questions of determining the proper base network. That is, how do we decide the base network of HDN in order to achieve high performance? Furthermore, how do we select the sizes of super-nodes at each dual-construction? These problem should be investigated and evaluated from the practical view point.

References

[1] N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, P. Heidelberger, S. Singh, B. D. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas. Blue gene/l torus interconnection network. *IBM Journal of Research and Development*, 49(2/3):265–276, 2005.

[2] P. Beckman. Looking toward exascale computing, keynote speaker. In *International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'08)*, University of Otago, Dunedin, New Zealand, December 2008.

[3] K. Ghose and K. R. Desai. Hierarchical cubic networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(4):427–435, April 1995.

[4] IBM. *Roadrunner: Hardware and Software Overview.* IBM Corporation, http://www.redbooks.ibm.com/redpapers/pdfs/redp4477.pdf, 2009.

[5] Y. Li and S. Peng. Dual-cubes: a new interconnection network for high-performance computer clusters. In *Proceedings of the 2000 International Computer Symposium, Workshop on Computer Architecture*, pages 51–57, ChiaYi, Taiwan, December 2000.

[6] Y. Li, S. Peng, and W. Chu. Efficient collective communications in dual-cube. *The Journal of Supercomputing*, 28(1):71–90, April 2004.

[7] Y. Li, S. Peng, and W. Chu. Recursive dual-net: A new universal network for supercomputers of the next generation. In *Proceedings of the 9th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'09)*, pages 809–820, Taipei, Taiwan, June 2009. Springer, Lecture Notes in Computer Science (LNCS).

[8] F. P. Preparata and J. Vuillemin. The cube-connected cycles: a versatile network for parallel computation. *Commun. ACM*, 24:300–309, May 1981.

[9] Y. Saad and M. H. Schultz. Topological properties of hypercubes. *IEEE Transactions on Computers*, 37(7):867–872, July 1988.

[10] TOP500. *Supercomputer Sites.* http://top500.org/, Nov. 2011.