Total Exchange Routing on Hierarchical Dual-Nets

Yamin Li, Wanming Chu

Total Exchange Routing on Hierarchical Dual-Nets

Yamin Li[1] and Wanming Chu[2]

[1] Hosei University, Tokyo 184-8584 Japan
[2] University of Aizu, Aizu-Wakamatsu 965-8580 Japan

Abstract. The hierarchical dual-net (HDN) is a newly proposed interconnection network for massive parallel computers. The HDN is constructed based on a symmetric product graph (base network). A k-level hierarchical dual-net, HDN(B, k, S), contains $n_k = (2n_0)^{2^k}/(2\prod_{i=1}^{k} s_i)$ 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 S is used mainly for adjusting the scale of the system. 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 is node and edge symmetric and can contain huge number of nodes with small node-degree and short diameter. The total exchange is one of the most dense communication patterns and is at the heart of numerous applications and programming models in parallel computing. In this paper, we show that the total exchange routing can be done on HDN efficiently.

Keywords: interconnection network; total exchange routing

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. 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, 6]. The supercomputer Roadrunner built by IBM adopts 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 first present a flexible interconnection network, called *Hierarchical Dual-Net* (HDN) [8]. 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. The total exchange, or all-to-all personalized communication, is one of the most dense communication patterns and is at the heart of numerous applications and programming models in parallel computing. In this paper, we present an efficient total exchange routing algorithm on a hierarchical dual-net. The time complexity $T_k(m)$ of the algorithm for an HDN(B, k, S) is $T_k(m) = (2^{k+1} - 2)(t_s + t_w m n_k/2) + 2^k T_0(m n_k/2)$, where n_k is the total number of nodes, t_s is startup latency, m is the message length in words, and t_w is the per-word transfer time.

The rest of this paper is organized as follows. Section 2 introduces the hierarchical dual-net in details. Section 3 describes the routing algorithm. Section 4 gives the total exchange routing algorithm on a hierarchical dual-net. Section 5 concludes the paper.

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.

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.

Meshes/tori or hypercubes are typical examples of product graphs. 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 $G = G_1 \times \ldots \times G_i \times \ldots \times G_r$ can be represented by $(a_1, \ldots, a_i, \ldots, a_r)$ with $0 \leq a_i \leq |G_i| - 1$. 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 $1 \leq j \leq q \leq r$ and $1 \leq i \leq r$, $G_j^{''} \neq G_k^{''}$ if $j \neq k$ for $1 \leq j, k \leq q$. Then a node in the sub-graph G' can be represented by $(b_1, \ldots, b_i, \ldots, b_q)$ with $0 \leq b_i \leq |G_i^{''}| - 1$. 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. 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).

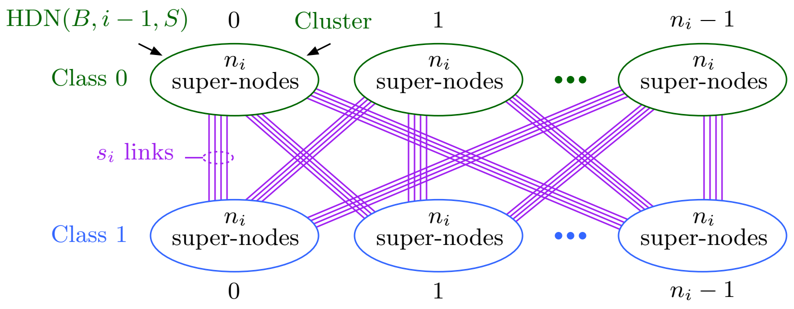


Fig. 1. Build an HDN(B, i, S) from HDN$(B, i-1, S)$

Referring to Figure 1, 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 $1 \le i \le k$.

The indexes of the nodes in HDN(B, k, S) can be defined as follows. Let SN_{id}^k be a *super-node_id* in a cluster of HDN(B, k, S) and N_{id}^k be a *node_id* in a super-node, then a node in the HDN(B, k, S) can be represented by $(C^k, U_{id}^k, SN_{id}^k, N_{id}^k)$

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 $(\overline{C^k}, SN^k_{id}, U^k_{id}, N^k_{id})$.

If we use a $2 \times 3 \times 5$ torus 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.

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

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 $N_i = 2(N_{i-1}/s_i)N_{i-1} = 2N^2_{i-1}/s_i$ nodes in the HDN(B, i, S) for $1 \leq i \leq k$, 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 $(2n_0)^{2^k}/(2\prod_{i=1}^{k} s_i)$, where n_0 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 $D(Q^{i-1}) = D_{i-1} - D(SN^i)$.

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 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. Therefore, we have the following recurrence:

$$D_i = 2(1 + D(Q^{i-1})) + D(SN^i) = 2D_{i-1} - D(SN^i) + 2$$

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 $D(SN^i), 1 \le i \le k$, 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.

Theorem 1 *Assume that the base network B is a symmetric, product graph and $SN^i, 1 \le i \le k$, 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 $SN^i, 1 \le i \le k$, 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 $1 \le i \le k$. 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 [9], 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.

Table 2. Comparison of topological properties

Network	# of nodes	Degree	Diameter
3D Torus	$x * y * z$	6	$(x + y + z)/2$
n-cube	2^n	n	n
CCC(n)	$n * 2^n$	3	$2n + \lfloor n/2 \rfloor - 2$
Dual-Cube(n)	2^{2n-1}	n	$2n$
RDN(m, k)	$(2m)^{2^k}/2$	$d_0 + k$	$2^k * D_0 + 2^{k+1} - 2$
HDN(B, k, S)	$(2\|B\|)^{2^k}/(2\prod_{i=1}^{k} s_i)$	$d_0 + k$	$2^k(D(B) - \sum_{j=0}^{k-1} 2^j (D(SN^{k-j})) + 2^{k+1} - 2$

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 $CR_w(G)$ 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

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

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.

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

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.

3 Routing on HDN

Given two nodes u and v in HDN(B, k, S), we first present a simple routing algorithm that finds a shortest path from u to v. 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 $(C^{i-1}, U_{id}^{i-1}, SN_{id}^{i-1}, N_{id}^{i-1})$ 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_{id}^i and N_{id}^i, of the node index in HDN(B, i, S). The other two parts, C^i and U_{id}^i, are generated from the construction at the ith level. The re-indexing process that generates a 1-to-1 mapping between $(C^{i-1}, U_{id}^{i-1}, SN_{id}^{i-1}, N_{id}^{i-1})$ and (SN_{id}^i, N_{id}^i) 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_{id}^i is $2|U^{i-1}/(SN^i - SN^{i-1})| \times |(SN^{i-1} - SN^i)|$. If $s_{i-1} = s_i$ then the re-indexing is simple: 1-1 mapping between SN_{id}^i and the 3-tuple $(C_{id}^{i-1}, U_{id}^{i-1}, SN_{id}^{i-1})$. However, when $s_{i-1} \neq s_i$, the re-indexing is a little complicated 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 $(C^{i-1}, U_{id}^{i-1}, SN_{id}^{i-1}, N_{id}^{i-1})$ to (SN_{id}^i, N_{id}^i) moves the indexes of those $B_j \subset SN^i - SN^{i-1}$ into N_{id}^i 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.

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*

```
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 $u_1^k = v_1^k$                   /* $u$, $v$ 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: $u_0^k \neq v_0^k$                    /* $u$, $v$ 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 $u_1^k \neq v_1^k$          /* $u$, $v$ 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
```

$R(SN^i), 1 \leq i \leq k$, 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. $\square$

4 Total Exchange Routing on HDN

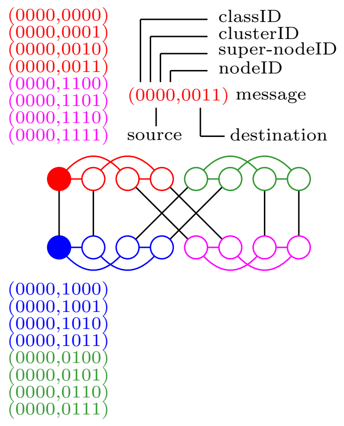
Design of efficient routing algorithms for collective communications is the key issue in parallel computers or networks. Collective communications are required in load balancing, event synchronization, and data exchange. Based on the number of sending and receiving processors, these communications can be classified into one-to-many, one-to-all, many-to-many and all-to-all. The nature of the messages to be sent can be classified as personalized or non-personalized (multicast or broadcast). The all-to-all personalized communication (total exchange) is at the heart of numerical applications.

An important metric used to evaluate efficiency of communication is *transmission latency*, or *communication time*. The communication time depends on many factors such as contentions, switching techniques, network topologies etc. Therefore, we first define the communication model used in this paper.

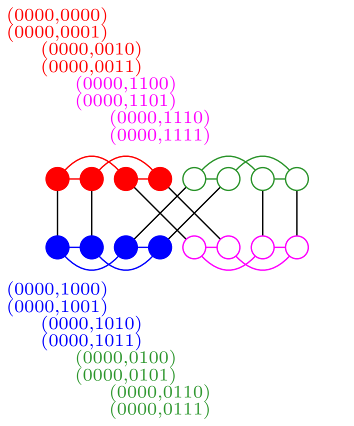
We assume that the communication links are bidirectional, that is, two directly-connected processors can send messages to each other simultaneously. We also assume the processor-bounded model (one-port model) in which each node can access the network through a single input port and a single output port at a time. The port model of a network system refers to the number of internal channels at each node. In order to reduce the complexity of communication hardware, many systems support one-port communication architecture. We also assume the linear cost model in which the transfer time for a message is linearly proportional to the length of the message.

There are many switching methods. In this paper, we assume the packet switching model [5]. In this model, each packet is maintained as an entity that is passed from node to node as it moves through the network. The long message can be partitioned and transmitted as fixed-length word w. The first few bytes of a packet contains routing and control information and are referred as packet header. A packet is completely buffered at each intermediate node before it is forwarded to the next node (for this reason, the model is also called store-and-forward switching). In this paper, we allow packages that are headed for the same destination to be combined into a single message. The time to pack and unpack messages is included in the startup latency. The packet switching model is suitable for collective communication in MPP since it is safer than other switching models such as virtual cut-through switching. With packet switching model, the communication time for a message of length m (number of fixed-length words) to be sent to a node of distance d is $d(t_s + mt_w)$, where t_s is startup latency, the time required for the system to handle the message at the sending node, t_w is the per-word transfer time ($1/t_w$ is the bandwidth of the communication links). Through this paper, we will use the formula above for estimating the communication times of the proposed algorithms.

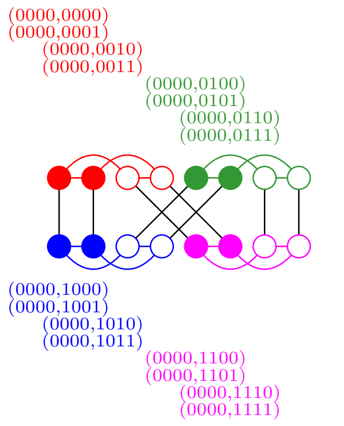
In total exchange, each node sends a distinct message of size m to every other node. The total number of messages is p^2 (a node also has a message for itself). Referring to Figures 2 and 3, the algorithm for total exchange in HDN can be described in four stages. Note that all nodes in the figures do the same operations but only the cases of node 0 are shown for clarity. Also note that the commas of node address in Figure 2 are omitted for saving spaces.



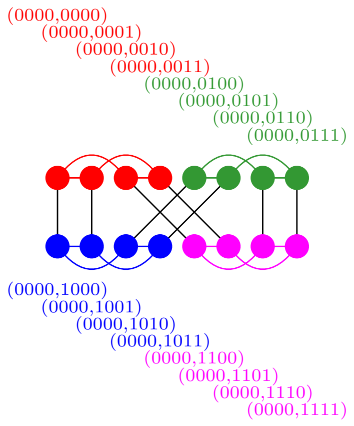
(a) Send message via cross-edge



(b) Total exchange inside cluster

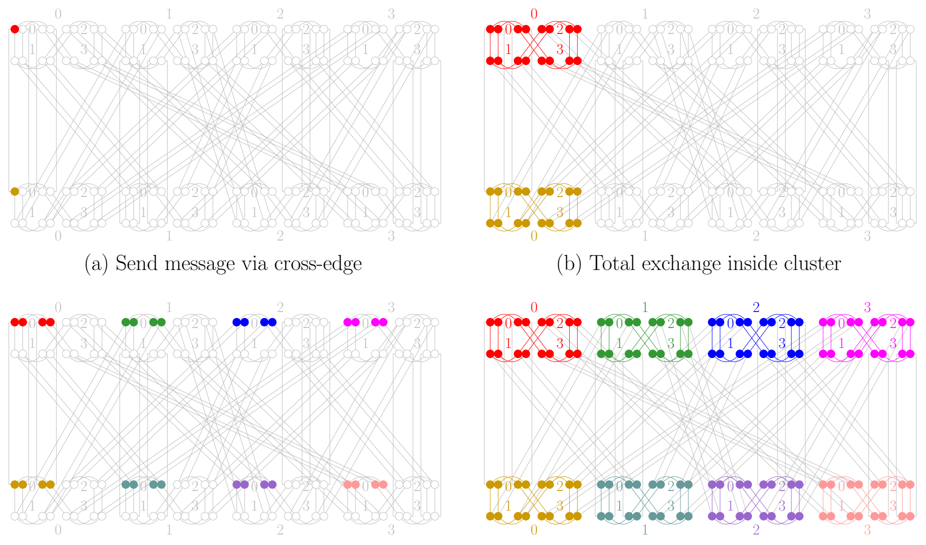


(c) Send message via cross-edge



(d) Total exchange inside cluster

Fig. 2. Four stages of total exchange on HDN$(B, 1, S)$ (only shows the case of node 0)



(a) Send message via cross-edge (b) Total exchange inside cluster

(c) Send message via cross-edge (d) Total exchange inside cluster

Fig. 3. Four stages of total exchange on HDN$(B, 2, S)$ (only shows the case of node 0)

1. In the first stage, we first divide M_{my_id} into two parts, $M1_{my\_id}$ and $M2_{my\_id}$, where $M1_{my\_id}$ contains all messages to be sent to the nodes in the clusters of *my_type*, and $M2_{my\_id}$ contains the rest message. Then, the first part of personalized messages $M1$ is exchanged between *my_id* and *partner*, the neighbor via cross-edge of level k. The time this stage takes is $t_s + t_w m n_k/2$.
2. In the second stage, we first pack all messages that are to be sent to the nodes in the cluster of level k with clusterID $= q$ into a single message msg_q. Then, we perform total exchange inside each cluster, where msg_q is to be sent to node with nodeID $= q$. The time this stage takes is denoted as $T_{k-1}(n_k/n_{k-1})$, the time for total exchange inside the cluster.
3. In the third stage, each node packs the received messages into a single message of length $n_k m$ and sends the packed message to its neighbor along the cross-edge of level k. After receive the message, each node unpacks the message received from its neighbor into n_{k-1} messages, $msg_{q'}$, where $msg_{q'}$ is the collection of all messages destinated to node with nodeID $= q'$ in the cluster. The time this stage takes is also $t_s + t_w m n_k/2$.
4. In the last stage, we perform total exchange again within each cluster of level k. This can be done since the packed messages sent through the level-k cross-edge are all destinated to the nodes inside the cluster. The time this stage takes is also $T_{k-1}(n_k/n_{k-1})$, which is the time for total exchange inside the cluster.

The algorithm is showed in Algorithm 2. All nodes execute the algorithm concurrently. In Algorithm 4, *my_id* is the *id* of the node. The initial message to be sent is M_{my_id} which contains p messages of length m. At the end of the algorithm, each node stores the collection of all p messages in *result*.

```
Algorithm 2: TOTAL_EXCHANGE (HDN($B, k, S$), $M_{my\_id}$)
begin
  if $k = 0$ result ← TOTAL_EXCHANGE ($B, M_{my\_id}$);
  else
    Divide $M_{my\_id}$ into two parts, $M1_{my\_id}$ and $M2_{my\_id}$, where $M1_{my\_id}$ contains
      all messages to be sent to the nodes in the clusters of type my_type, and
      $M2_{my\_id}$ contains the rest messages;
    partner ← the neighbor via cross-edge of level $k$;
    send message $M1_{my\_id}$ to partner;
    Receive message $M1_{partner}$ from partner;
    $M'_{my\_id} \leftarrow M2_{my\_id} \cup M1_{partner}$;
    Pack all messages in $M'_{my\_id}$ that are to be sent to the nodes in the cluster of
      level $k$ with cluster_ID = $q$ into a single message $msg_q$ to be sent to the
      node with node_ID = $q$;
    $T_{my\_id}$ ← TOTAL_EXCHANGE (HDN($B, k-1, S$), $M'_{my\_id}$);
    send message $T_{my\_id}$ to partner;
    receive message $T_{partner}$ from partner;
    $T'_{my\_id} \leftarrow T_{partner}$;
    Unpack $T'_{my\_id}$ into $n_{k-1}$ messages, $msg_{q'}$, such that $msg_{q'}$ is the collection of
      messages destinated to the node with node_ID = $q'$;
    result ← TOTAL_EXCHANGE (HDN($B, k-1, S$), $T'_{my\_id}$);
  endif
end
```

The time to complete the total exchange on an HDN$(B, 1, S)$ is

$$\begin{aligned} T_1(m) &= (t_s + mt_w n_1/2) + T_0(mn_1/2) + (t_s + mt_w n_1/2) + T_0(mn_1/2) \\ &= 2(t_s + mt_w n_1/2) + 2T_0(mn_1/2). \end{aligned}$$

Generally, on an HDN(B, k, S), the time to complete the total exchange is

$$\begin{aligned} T_k(m) &= (t_s + t_w mn_k/2) + T_{k-1}(n_k/n_{k-1}) + (t_s + t_w mn_k/2) + T_{k-1}(n_k/n_{k-1}) \\ &= 2(t_s + t_w mn_k/2) + 2T_{k-1}(n_k/n_{k-1}). \text{ That is,} \end{aligned}$$

$$T_k(m) = (2^{k+1} - 2)(t_s + t_w mn_k/2) + 2^k T_0(mn_k/2)$$

where n_k is the total number of nodes and $T_0(m)$ is the time complexity for total exchange in B. In the examples of Figures 2 and 3 where B is a 2-cube, $T_0(m) = 2(t_s + t_w m)$. If B is an n-cube, then $T_0(m) = n(t_s + t_w m 2^n/2)$.

For the HDN$(B, 1, S)$ shown in Figures 2, $T_1 = 2(t_s + 8t_w m) + 4(t_s + 8t_w m) = 6(t_s + 8t_w m)$. In contrast, $T = 4(t_s + 8t_w m)$ for a 4-cube of same size. For the HDN$(B, 2, S)$ shown in Figures 3, $T_2 = 6(t_s + 64t_w m) + 4 \times 2(t_s + 64t_w m) = 14(t_s + 64t_w m)$. In contrast, $T = 7(t_s + 64t_w m)$ for a 7-cube of same size. The times of total exchange for HDNs are longer than that for hypercubes but an HDN has much less links than a hypercube of the same size. We summarize this result in the following theorem.

Theorem 3 *Assume that the time complexity $T_0(m)$ for total exchange in the base network B is known, where m is the length of each message. The time complexity $T_k(m)$ for total exchange on an HDN(B, k, S), $k > 0$, is $T_k(m) = (2^{k+1} - 2)(t_s + t_w m n_k/2) + 2^k T_0(mn_k/2)$, where $n_k = (2n_0)^{2^k}/(2\prod_{i=1}^{k} s_i)$.*

5 Concluding Remarks

The hierarchical dual-net can connect a large number of nodes with a small node-degree and a short diameter. It is a potential candidate for the interconnection network of the supercomputers of the next generation that may have more than one million of nodes. We can select a popular network of small size that is a product graph as the base network and then connect multiple base modules with cross links (cables) to construct a very large-scale hierarchical dual-net. We can also select a suitable set of integers based on the base network to control the number of nodes in the supercomputer. The base networks can be implemented in a NoC VLSI and high-speed line cables may be used as the cross links to connect PCB modules in cabinets. We presented an efficient algorithm for total exchange on recursive dual-net. There are many problems, such as disjoint path and fault-tolerant routing, on recursive dual-net that are worth further research.

References

1. Adiga, N.R., Blumrich, M.A., Chen, D., Coteus, P., Gara, A., Giampapa, M.E., Heidelberger, P., Singh, S., Steinmacher-Burow, B.D., Takken, T., Tsao, M., Vranas, P.: Blue gene/l torus interconnection network. IBM Journal of Research and Development 49(2/3), 265–276 (2005)
2. Beckman, P.: Looking toward exascale computing, keynote speaker. In: International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'08). Dunedin, New Zealand (December 2008)
3. Ghose, K., Desai, K.R.: 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 (January 2009)
5. Kumar, V., Grama, A., Gupta, A., Karypis, G.: Introduction to parallel computing: design and analysis of algorithms. Benjamin/Cummings Press (1994)
6. Li, Y., Peng, S., Chu, W.: Efficient collective communications in dual-cube. The Journal of Supercomputing 28(1), 71–90 (April 2004)
7. Li, Y., Peng, S., Chu, W.: 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, Taipei, Taiwan (June 2009)
8. Li, Y., Peng, S., Chu, W.: Hierarchical dual-net: A flexible interconnection network and its routing algorithm. In: Proceedings of the Second International Conference on Networking and Computing. pp. 58–67. Osaka, Japan (November 2011)
9. Preparata, F.P., Vuillemin, J.: The cube-connected cycles: a versatile network for parallel computation. Commun. ACM 24, 300–309 (May 1981)
10. TOP500: Supercomputer Sites. http://top500.org/ (June 2013)