Recursive Hierarchical Fully-Connected Networks: A Class of Low-Degree Small-Diameter Interconnection Networks

Chi-Hsiang Yeh and Behrooz Parhami
Department of Electrical and Computer Engineering
University of California
Santa Barbara, CA 93106-9560, USA

Abstract

In this paper, we propose a new class of interconnection networks called recursive hierarchical fully-connected (RHFC) networks for high-performance parallel processing. The degree of a partially-linked RHFC network is equal to that of its nucleus graph plus a fixed integer, leading to the scalability of such networks. The diameters of suitably constructed RHFC networks are asymptotically optimal within a small constant factor with respect to their node degrees. We then present the construction of recursive hierarchical swapped (RHS) networks and multi-dimensional hypernets, both of which are subclasses of RHFC networks and can emulate a hypercube efficiently. We conclude that RHFC networks are highly modularized, make the use of fixed-degree building blocks possible for any practically realizable system, and lead to the construction of high-performance interconnection networks with reasonable cost.

1 Introduction

High performance, low node-degree, and simplicity of algorithms are all important, but often conflicting, requirements for interconnection networks. To overcome the problem of unbounded node complexity in large hypercube networks, some constant-degree variants, such as cube-connected cycles (CCC) [7], de Bruijn graph, and butterfly, have been proposed and shown to have some desirable properties.

In practical applications, topologies that achieve better performance and have intermediate node degrees (e.g., higher than 3, but lower than $\log_2 N$) may be desirable. Many such interconnection schemes for parallel architectures have been proposed in the literature [2, 3, 4, 5, 8, 9]. However, some of them maintain logarithmic node degree, which still grows rapidly when the system size increases. Others may have larger diameter or incur congestion in some applications.

In this paper, we propose a wide class of interconnection networks called *recursive hierarchical fully-connected (RHFC) networks* for high-performance parallel processing with low node degree. RHFC networks have node degrees varying from $O(1)$ to $O(\log N)$, depending on the parameters chosen, and can achieve asymptotically optimal diameters with respect to their node degree. We present the constructions of RHS networks, a subclass of RHFC networks with specific connection rules, which are shown to emulate a hypercube with asymptotically optimal slowdown. We also propose a novel scalable interconnection network, *multi-dimensional hypernet*, which has fixed node degree and small diameter, and can emulate a hypercube with small dilation.

2 Multi-level Fully-Connected (MFC) Networks

We define a multi-level fully-connected (MFC) network as follows. An MFC network begins with N_2 identical copies of a nucleus with N_1 nodes, where $N_2 \leq N_1 + 1$ and the nucleus can be any connected graph or hypergraph. Each nucleus copy is viewed as a *level-2 cluster*, and connects to each of the other level-2 clusters via at least one *level-2 inter-cluster link*. The resultant interconnection network is called a 2-level MFC network. To construct an l-level MFC network, we use N_l identical copies of an $(l-1)$-level MFC network with $\prod_{i=1}^{l-1} N_i$ nodes as *level-l clusters*, and connect each level-l cluster to each of the other level-l clusters via at least one *level-l inter-cluster link*, where $N_l \leq \prod_{i=1}^{l-1} N_i + 1$. This recursive definition allow us to construct arbitrarily large MFC networks based on any type of nucleus. An l-level MFC network based on the nucleus G is denoted by MFC(l, G); in particular, MFC$(1, G)$ is the nucleus G.

Note that each node is allowed to have at most one link at a certain level, which is the reason we restrict N_l to be no more than $\prod_{i=1}^{l-1} N_i + 1$, one plus the number of nodes in a level-l cluster.

2.1 Topological Properties and Routing Algorithms

We start with a nucleus G having N_1 nodes of degree d_G and at each level i use N_i copies of the $(i-1)$-level MFC network. Thus the size N of an MFC(l, G) network is given by

$$N = \prod_{i=1}^{l} N_i.$$

We give a node in G a $\lceil \log_2 N_1 \rceil$-bit string $\in [0, N_1 - 1]$ as its node address. We also give each level-i cluster a $\lceil \log_2 N_i \rceil$-bit string $\in [0, N_i - 1]$ as its cluster address, where N_i is the number of level-i clusters in a level-$(i+1)$ cluster.

Given the address for a node in an MFC$(i-1, G)$ ($i = 2$ initially), we can assign the address for the node in MFC(i, G) by simply concatenating the cluster address for the level-i cluster (i.e., MFC($i-1, G$)) to which the node belongs and the address for the node within its cluster.

The degree of an MFC(l, G) may vary according to different connection rules. An upper bound on the degree d_l of MFC(l, G) is given by $d_l \leq d_G + l - 1$, according to the definition.

In what follows, we present a recursive routing algorithm to route a packet from node x to node y in an MFC(l, G) network.

Let Cx and Cy be level-l clusters to which nodes x and y belong, respectively, and assume that node x knows the address of at least one of the nodes (within the same level-l cluster) that have a link connecting Cx to the level-l cluster Cy within the same level-($l+1$) cluster (which is an MFC(l, G) containing nodes x and y). Given the routing algorithm for the nucleus G (denoted by MFC($1, G$)), here is how routing is done at level l.

- Case 1: Nodes x and y belong to the same level-l cluster: We use the routing algorithm for MFC($l-1, G$) to route the packet since any level-l cluster is an MFC($l-1, G$).
- Case 2: Nodes x and y belong to different level-l clusters: Let nodes x and y belong to level-l clusters Cx and Cy, respectively, and let (x', y') be a level-l inter-cluster link that connects Cx and Cy. To route a packet from node x to node y, we use the routing algorithm for MFC($l-1, G$) to route the packet from node x to node x', send the packet from node x' to node y' via the level-l inter-cluster link, and then use the routing algorithm for MFC($l-1, G$) again to route the packet from node y' to node y.

If the routing algorithm in MFC(i, G) requires at most $T_{R,MFC}(i, G)$ time steps, and the routing algorithm in G requires no more than $T_{R,G}$ time steps, the recursive routing algorithm in MFC(l, G) requires time not exceeding

$$\begin{aligned} T_{R,MFC}(l, G) &= 2T_{R,MFC}(l-1, G) + 1 \\ &= 2^{l-1}T_{R,G} + 2^{l-1} - 1. \end{aligned} \qquad (1)$$

An upper bound on the diameter of an MFC network is obtainable from the preceding routing algorithm (assuming optimal routing algorithm in the nucleus graph), leading to the following theorem:

Theorem 2.1 *The diameter of an l-level MFC network is no more than $(D_G + 1)2^{l-1} - 1$ where D_G is the diameter of the nucleus graph.*

3 Recursive Hierarchical Fully-Connected Networks

When there are multiple links connecting a pair of clusters, the distribution of these links within a cluster may affect the efficiency of MFC networks. Roughly speaking, if they are distributed uniformly, it is easier for a node to access other clusters, leading to smaller internode distances, lower congestion, and better performance.

In this section, we present *recursive hierarchical fully-connected (RHFC) networks*, a subclass of MFC networks, which have approximately uniformly distributed links connecting a pair of clusters, leading to desirable topological and algorithmic properties. An RHFC network is characterized by its nucleus graph G, its recursive depth r (henceforth simply depth), numbers $l_r, l_{r-1}, ..., l_1$ of hierarchical levels at various depths, and numbers $N_{i,j}$ of depth-i level-j clusters within a depth-i level-($j+1$) cluster. We first define hierarchical fully-connected (HFC) networks, or RHFC networks of depth 1, and then present the construction of general r-deep RHFC networks. The top view of an r-deep RHFC network is shown in Fig. 1.

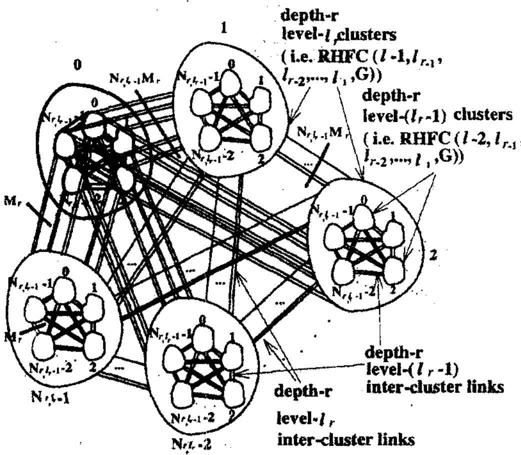

Fig. 1. Top view of an RHFC($l_r, l_{r-1}, ..., l_1, G$), where $M_r = \prod_{j=2}^{l_r-1} N_{r,j}$ is the number of depth-r level-2 clusters in a depth-r level-$r-1$ cluster.

Definition 3.1 (HFC(l, G)) *An l-level MFC(l, G) is called an HFC(l, G) if $N_i \leq N_1 + 1$, $N_i = \Theta(N_1)$, $i = 2, 3, ..., l$, and each nucleus of the MFC network has exactly one link connecting it to each of the other level-i clusters within the same level-($i+1$) cluster to which the nucleus belongs, $i = 2, 3, ..., l$, where N_i is the number of level-i clusters within a level-($i+1$) cluster, for $i = 2, 3, ..., l$, N_1 is the number of nodes in the nucleus G, and the level-($l+1$) cluster refers to the MFC(l, G) itself.*

The l-tuple

$$\mathcal{N}_{CV} \overset{\text{def}}{=} (N_1, N_2, ..., N_l)$$

is called the characteristic vector of the HFC(l, G). Given $\mathcal{N}_{CV}$ of an HFC network, there exist unique l-tuples $\mathcal{A}_{CV} = (a_1, a_2, ..., a_l)$ and $\mathcal{B}_{CV} = (b_1, b_2, ..., b_l)$ satisfying

$$\mathcal{N}_{CV} = \mathcal{A}_{CV} N_1 + \mathcal{B}_{CV},$$

where $a_i = O(1)$ and $b_i = o(N_1)$ [1], $i = 1, 2, ..., l$. We call $\mathcal{A}_{CV}$ and $\mathcal{B}_{CV}$ the coefficient vector and the remainder vector of the HFC network, respectively. Clearly, $a_1 = 1$ and $b_1 = 0$. When $N_1 = O(1)$, we have $b_i = 0$ for all i.

[1] If $f(n) = o(g(n))$, $\lim_{n\to\infty} f(n)/g(n) = 0$.

Definition 3.2 (RHFC(l, G))
An r-deep RHFC($l_r, l_{r-1}, ..., l_1, G$) network is recursively defined as an HFC(l_r, RHFC($l_{r-1}, l_{r-2}, ..., l_1, G$)).

Clearly, an RHFC($l_r, l_{r-1}, ..., l_1, G$) network is a p-level MFC(p, G) based on G, where $p = \sum_{i=1}^{r} l_i - r + 1$. The level-$j$ cluster of an HFC(l_i, RHFC($l_{i-1}, l_{i-2}, ..., l_1, G$)) is called a *depth-$i$ level-j cluster* of the RHFC($l_r, l_{r-1}, ..., l_1, G$) network; its level-$j$ inter-cluster link is called a *depth-i level-j inter-cluster link* of the RHFC($l_r, l_{r-1}, ..., l_1, G$) network. A depth-$i$ level-j cluster (inter-cluster link) of the RHFC($l_r, l_{r-1}, ..., l_1, G$) network is called a level-($\sum_{t=1}^{i-1} l_t - i + 1 + j$) cluster (inter-cluster link) of the MFC(p, G).

We define the characteristic matrix of RHFC($l_r, l_{r-1}, ..., l_1, G$) as

$$\mathcal{N}_{CM} \overset{\text{def}}{=} (\mathcal{N}1_{CV0}, \mathcal{N}2_{CV0}, ..., \mathcal{N}l_{CV0})^T,$$

where $\mathcal{N}i_{CV0} = (\mathcal{N}i_{CV}, \underbrace{0, 0, ..., 0}_{l_{max}-l_i})$, $l_{max} = \max(l_j,\ j = 1, 2, ..., r)$, and $\mathcal{N}i_{CV}$, $i = 1, 2, ..., r$, is the characteristic vector of a depth-i level-($l_i + 1$) cluster of the RHFC network, which is an HFC(l_i, RHFC($l_{i-1}, l_{i-2}, ..., l_1, G$)). We denote element i, j of $\mathcal{N}_{CM}$ as $N_{i,j}$ and abbreviate $N_{1,j}$ as N_j.

We define the coefficient matrix and the remainder matrix of the RHFC($l_r, l_{r-1}, ..., l_1, G$) as

$$\mathcal{A}_{CM} \overset{\text{def}}{=} (\mathcal{A}1_{CV0}, \mathcal{A}2_{CV0}, ..., \mathcal{A}l_{CV0})^T,$$

$$\mathcal{B}_{CM} \overset{\text{def}}{=} (\mathcal{B}1_{CV0}, \mathcal{B}2_{CV0}, ..., \mathcal{B}l_{CV0})^T,$$

where

$$\mathcal{A}i_{CV0} = (\mathcal{A}i_{CV}\ \underbrace{0, 0, ..., 0}_{l_{max}-l_i}),\ \mathcal{B}i_{CV0} = (\mathcal{B}i_{CV}, \underbrace{0, 0, ..., 0}_{l_{max}-l_i}),$$

$l_{max} = \max(l_j,\ j = 1, 2, ..., r)$, and $\mathcal{A}i_{CV}$, ($\mathcal{B}_{i,CV}$), $i = 1, 2, ..., r$, are the coefficient (remainder) vectors of a depth-i level-($l_i + 1$) cluster of the RHFC network, which is an HFC(l_i, RHFC($l_{i-1}, l_{i-2}, ..., l_1, G$)). Clearly, $a_{i,1} = 1$ and $b_{i,1} = 0$, $i = 1, 2, ..., r$. When $N_1 = O(1)$, we have $b_{i,j} = 0$.

3.1 Network Diameter

The routing algorithm presented for MFC networks can be applied to an l-level HFC networks directly in considerably smaller time than $T_{R,MFC}(l, G)$ (Eq. (1)) due to the special structure of HFC networks..

Lemma 3.1 *A packet can be routed in $lT_{R,G} + l - 1$ time on an HFC(l, G), where $T_{R,G}$ is the time required for the packet routing algorithm on the nucleus G.*

Proof: Let $T_{R,HFC}(i, G)$ be the time required for the routing algorithm on an HFC(i, G). Since there always exists one node x' within the nucleus to which node x belongs (see Case 2 of the routing algorithm for an l-level MFC network), it requires only $T_{R,G}$ time to send a packet from node x to x'. Thus, the overall routing time is given by

$$\begin{aligned} T_{R,HFC}(l, G) &= T_{R,HFC}(l-1, G) + T_{R,G} + 1 \\ &= lT_{R,G} + l - 1. \end{aligned} \quad (2)$$

□

Note that the resultant routing time on an l-level HFC network is much better than the upper bound on a general l-level MFC network, which requires time $T_{R,MFC}(l, G) = 2^{l-1}(T_{R,G} + 1) - 1$.

Since an r-deep RHFC network can be decomposed into an HFC network based on an $(r-1)$-deep RHFC network, the preceding lemma can be applied to RHFC networks, leading to the following theorem.

Theorem 3.2 *A packet can be routed in $(T_{R,G} + 1)\prod_{i=1}^{r} l_i - 1$ time on an RHFC($l_r, l_{r-1}, ..., l_1, G$), where $T_{R,G}$ is the time required for the packet routing algorithm on a nucleus G.*

Proof: Let $T_{R,RHFC}(l_i, l_{i-1}, ..., l_1, G)$ be the time required for the routing algorithm on an RHFC($l_i, l_{i-1}, ..., l_1, G$). From Lemma 3.1, the routing algorithm on an RHFC($l_r, l_{r-1}, ..., l_1, G$) requires time at most equal to

$$\begin{aligned} &T_{R,RHFC}(l_r, l_{r-1}, ..., l_1, G) \\ &= l_r(T_{R,RHFC}(l_{r-1}, l_{r-2}, ..., l_1, G) + 1) - 1 \\ &= (T_{R,G} + 1)\textstyle\prod_{i=1}^{r} l_i - 1. \end{aligned} \quad (3)$$

□

3.2 Asymptotically Optimal Diameter

An N-node interconnection network with degree d will be said to have *asymptotically optimal diameter* if its diameter is asymptotically within a constant factor from the universal lower bound $D_L(d, N) = \log_2 N / \log_2 d$. Hypercubes, meshes, folded hypercubes, HCN, and WK-recursive networks, do not have asymptotically optimal diameters; while star, pancake, and complete graphs are examples that achieve asymptotically optimal diameter.

In this subsection, we show that a suitably constructed RHFC network can have optimal diameter within a constant factor.

Theorem 3.3 *The diameter of an N-node RHFC network based on the nucleus G is asymptotically optimal if the number of inter-cluster links per node is at most equal to a polynomial in d_G (i.e., $\log\left(\sum_{i=1}^{r} l_i\right) = O(\log d_G)$ for RHFC($l_r, l_{r-1}, ..., l_1, G$)) and the diameter of the nucleus G is asymptotically optimal, where d_G is the degree of the nucleus G.*

Proof: Let N_1 be the number of nodes in a nucleus G, and D_G be its diameter. Then we have

$$D_G = \Theta\left(\frac{\log N_1}{\log d_G}\right).$$

The diameter of the RHFC network is no more than

$$D_r = (D_G + 1)\prod_{i=1}^{r} l_i - 1 = \Theta\left(\frac{\log N_1}{\log d_G}\prod_{i=1}^{r} l_i\right)$$

from Eq. (3).

The degree d_r of the RHFC network is given by

$$d_r = d_G + \sum_{i=1}^{r}(l_i - 1)$$

from its definition. Since the number of inter-cluster links is at most equal to a polynomial in d_G, that is,

$$\log\left(\sum_{i=1}^{r}(l_i - 1)\right) = O(\log d_G),$$

we can easily show that N_1 cannot be a constant independent of N and that

$$\log d_r = \Theta(\log d_G).$$

Therefore, we have

$$D_r = \Theta\left(\frac{\log N_1}{\log d_r}\prod_{i=1}^{r} l_i\right).$$

The size of the RHFC network can be derived as

$$\begin{aligned} N \;=\; & \textstyle\prod_{j_r=1}^{l_r}\Big(a_{r,j_r}\cdots\Big(a_{3,j_3}\prod_{j_2=1}^{l_2}\big(a_{2,j_2}\cdot \\ & \textstyle\prod_{j_1=1}^{l_1}(a_{1,j_1}N_1 - b_{1,j_1}) - b_{2,j_2}\big) - b_{3,j_3}\Big)\cdots\Big). \end{aligned}$$

Therefore, we have

$$\log_2 N = \Theta(\log N'),$$

where

$$N' = N_1^{\prod_{i=1}^{r} l_i}\prod_{i=1}^{r}\left(\prod_{j=1}^{l_i} a_{i,j}\right)^{\prod_{i'=i+1}^{r} l_{i'}}$$

with $\prod_{i'=r+1}^{r} l_{i'} = 1$. Note that N' is obtained by removing the elements $b_{i,j}$ in the remainder matrix for simplicity. Therefore, we have

$$\log_2 N' \geq (\log_2 N_1 + \log_2 a_{min})\prod_{i=1}^{r} l_i,$$

where $a_{min} = \min(a_{i,j},\ i = 1, 2, ..., r,\ j = 2, 3, ..., l_i)$. Since the $a_{i,j}$ are positive constants for $j \leq l_j$ and N_1 is not a constant independent of N, it is clear that

$$\log_2 N = \Theta(\log N') = \Omega\left(\log N_1 \prod_{i=1}^{r} l_i\right).$$

Thus, the diameter of the RHFC network is

$$D_r = \Theta\left(\frac{\log N}{\log d_r}\right),$$

which is asymptotically optimal from the universal lower bound $D_L(d_r, N)$. □

Note that it is, in fact, impossible for the number of inter-cluster links per node to be a constant in an RHFC network that uses a nucleus of constant size, since some of the elements in the coefficient matrix will approach zero (i.e., $\lim_{N\to\infty} a_{i,j} = 0$ for some i, j), which conflicts with the definition of an RHFC network. It can be shown that the diameter of an RHFC network based on a generalized hypercube [1] can be asymptotically optimal within a factor of 1 with respect to its node degree by choosing appropriate parameters.

4 Parallel Algorithms on RHFC Networks

In this section, we present efficient algorithms for semigroup computation and on-line fault-tolerant routing on RHFC networks.

4.1 Semigroup Computation

Let $\otimes$ be an associative binary operator, i.e., $(u \otimes v) \otimes w = u \otimes (v \otimes w)$ for all $u, v, w \in S$. The operations $+, \times, \wedge, \vee$, *max*, and *min* are all examples of associative binary operators. A *semigroup* is a pair $(S, \otimes)$, where S is a set of elements on which the associative binary operator $\otimes$ is defined. Given a list of values $v_1, v_2, \ldots, v_n \in S$, *semigroup computation* is the problem of computing $v_1 \otimes v_2 \otimes \ldots \otimes v_n$.

We first present the algorithm on an HFC network, then generalize it to RHFC networks.

Assume that each node in HFC(l, G) holds at most one value v_j for computation. Nodes without values v_j are given padding values (e.g., ∞ for finding the minimum). Semigroup computation on the HFC network can be performed in three phases:

- Phase 1: Perform semigroup computation on each level-l cluster, separately.
- Phase 2: Each node exchanges its value via its level-l inter-cluster link with its level-l neighbor. A node that doesn't receive a value in this step assumes a padding value.
- Phase 3: Perform semigroup computation on each nucleus, separately.

This algorithm gives the following lemma.

Lemma 4.1 *Semigroup computation can be performed in $lT_{S,G} + l - 1$ time on an HFC(l, G), where $T_{S,G}$ is the time required on the nucleus G.*

Proof: Assume that the algorithm for an HFC(i, G) is available and requires $T_S(i, G)$ time, $i = 1, 2, ..., l-1$. Then semigroup computation on an HFC(l, G) requires time

$$T_S(l, G) = T_S(l-1, G) + T_{S,G} + 1 = lT_{S,G} + l - 1. \quad (4)$$

□

Since an r-deep RHFC($l_r, l_{r-1}, ..., l_1, G$) can be viewed as an l_r-level HFC(l_r, RHS($l_{r-1}, l_{r-2}, ..., l_1, G$)), the preceding algorithm can be applied to RHFC networks, leading to the following theorem.

Theorem 4.2 *Semigroup computation can be performed in $(T_{S,G}+1)\prod_{i=1}^{r} l_i - 1$ time on an RHFC($l_r, l_{r-1}, ..., l_1, G$) where $T_{S,G}$ is the time required to perform semigroup computation on the nucleus G.*

Proof: Let $T_S(l_i, l_{i-1}, ..., l_1, G)$ be the time required for semigroup computation on an RHFC$(l_i, l_{i-1}, ..., l_1, G)$. Then semigroup computation on an RHFC$(l_r, l_{r-1}, ..., l_1, G)$ requires time

$$\begin{aligned} & T_S(l_r, l_{r-1}, ..., l_1, G) \\ &= l_r(T_S(l_{r-1}, l_{r-2}, ..., l_1, G) + 1) - 1 \quad (5) \\ &= (T_{S,G} + 1)\prod_{i=1}^{r} l_i - 1. \end{aligned}$$

□

4.2 An On-Line Fault-Tolerant Routing Algorithm

On-line fault-tolerant routing can be easily implemented on an RHFC network in the presence of a small number of faulty nodes and links, given an on-line fault-tolerant routing algorithm on the nucleus G.

For an HFC network to route elegantly and efficiently, we assume that a node x knows the address of the node (within the same nucleus) that has the link connecting the nucleus containing x to each of the other level-i clusters within the same level-$(i+1)$ cluster, $i = 2, 3, ..., l$. Given the on-line fault-tolerant routing algorithm for the nucleus G (i.e., HFC$(1, G)$), here is how fault-tolerant routing, $\mathcal{FTR}(l, x, y)$, is done at level l.

- Case 1: Nodes x and y belong to the same level-l cluster: We use $\mathcal{FTR}(l-1, x, y)$, the fault-tolerant routing algorithm for HFC$(l-1, G)$, to route the packet since any level-l cluster is an HFC$(l-1, G)$.
- Case 2: Nodes x and y belong to different level-l clusters: Let nodes x and y belong to level-l clusters Cx and Cy, respectively, and let (x', y') be the level-l inter-cluster link that connects Cx and Cy, where node x and x' are in the same nucleus.
    - Phase 1: To route a packet from node x to node y, we use $\mathcal{FTR}(l-1, x, x')$, the fault-tolerant routing algorithm for HFC$(l-1, G)$, to route the packet from node x to node x'.
    - Phase 2:
        * Case (i): Node x' and link $(x'y')$ are normal: We send the packet from node x' to node y' via the level-l inter-cluster link (x', y'), and then use $\mathcal{FTR}(l-1, y', y)$, the routing algorithm for HFC$(l-1, G)$, to route the packet from node y' to node y.
        * Case (ii): Either node x' or link $(x'y')$ fails: We divert the packet to a node x'' that has a normal level-l inter-cluster link, send the packet via its level-l inter-cluster link (denoted by (x'', y'')), then perform the fault-tolerant routing $\mathcal{FTR}(l, y'', y)$ again.

Note that node x'' in Case (ii) can be picked either randomly, according to some rule, or perhaps by probing the vicinity of x' for a node that has a healthy level-l inter-cluster link. As a result, no a-priori knowledge of the locations of normal level-l links is required. For this algorithm to work on-line without the knowledge of the locations of faulty nodes/links, the number of faults must be reasonably small.

Since an r-deep RHFC network can be decomposed into an HFC network based on an $(r-1)$-deep RHFC network, the preceding routing algorithm can be applied to RHFC networks without modification. It can be easily established that the penalty paid when a packet encounters a faulty depth-i level-j inter-cluster link (or the corresponding node failure) is at most equal to $T_R(l_{i-1}, l_{i-2}, ..., l_1, G)+1$, where $T_R(l_{i-1}, l_{i-2}, ..., l_1, G)$, the time required for routing in a depth-$(i-1)$ level-$(l_{i-1}+1)$ cluster, is usually negligible, especially when i is smaller than r.

5 Fully-Linked RHFC Networks

A subclass of RHFC networks, fully-linked RHFC networks, can perform many algorithms efficiently, and achieves asymptotically optimal diameter within a small constant. We define this class formally as follows.

Definition 5.1 (Fully-Linked RHFC(l, G))
An RHFC$(l_r, l_{r-1}, ..., l_1, G)$ network is said to be fully-linked if and only if

$$a_{i,j} = 1 \textit{ for } j = 2, 3, ..., l_i,$$

where $a_{i,j}$ is element i, j of the coefficient matrix A_{CM} of the RHFC network.

Fully-linked RHFC networks form a high-performance subclass of RHFC networks. It is also easier to develop elegant algorithms on such RHFC networks.

5.1 Recursive Hierarchical Swapped Networks

Although packet routing and semigroup computation can be performed on general RHFC networks efficiently, some algorithms such as ascend/descend algorithms cannot be performed unless appropriate connectivity is specified. In this section, we introduce recursive hierarchical swapped (RHS) networks, a subclass of swapped networks [10] and fully-linked RHFC networks with $a_{i,j} = 1$ and proper connection rule, and show that they can emulate a hypercube of the same size efficiently. We first present the construction of a hierarchical swapped network [11], or an RHS network of depth 1, and then generalize the definition to general RHS networks.

An l-level hierarchical swapped network, HSN(l, G), begins with a nucleus G, which forms an HSN$(1, G)$. To build an l-level hierarchical swapped network, HSN(l, G), we use M identical copies of HSN$(l-1, G)$. Each copy of HSN$(l-1, G)$ is viewed as a level-l cluster, and is given a k-bit string X_l as its address; each node is already given a $k(l-1)$-bit string $X'_{l-1} = X_{l-1:1}$ as its address within the level-l cluster to which it belongs, where $X_{i:j} = X_iX_{i-1}\cdots X_{j+1}X_j$. Node X'_{l-1} within the level-l cluster X_l has a kl-bit string $X'_l = X_lX'_{l-1} = X_{l:1}$ as its address within the HSN(l, G). Each of the M level-l clusters has M^{l-1} nodes and M^{l-2} links connecting it to each of the other $M-1$ level-l clusters, via which node $X_lX_{l-1:2}X_1$ connects to node $X_1X_{l-1:2}X_l$. This connectivity and the hierarchical construction are the reasons we call such networks "hyerarchical swapped networks." The recursive definition allows us to construct arbitrary-level HSNs based on any type of nucleus, as a subclass of HFC networks. The construction of an HSN$(3, Q_2)$ is shown is Fig. 2.

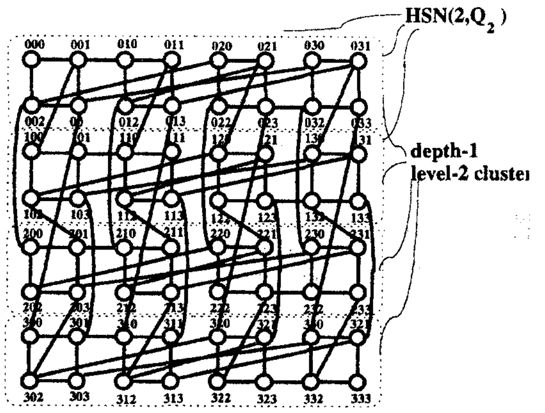

Fig. 2. The complete structure of the 64-node HSN$(3, Q_2)$, with node addresses expressed as radix-4 numbers.

An r-deep RHS$(l_r, l_{r-1}, ..., l_1, G)$ is recursively defined as HSN$(l_r,$ RHS$(l_{r-1}, l_{r-2}, ..., l_1, G))$.

5.2 Mixed Radix Number Systems

In this subsection, we introduce a mixed radix number system, which is useful in presenting the time complexity of emulation algorithms and their operation steps.

Let i_a be an integer, $i_a \in [0, \prod_{j=0}^{r} l_j - 1]$. It can be seen that i_a has a unique $(r+1)$-tuple $(a_r, a_{r-1}, ..., a_1, a_0)$, $a_j \in [0, l_i - 1]$, such that

$$i_a = \sum_{j_1=0}^{r} a_{j_1} \prod_{j_2=0}^{i} l_{j_2}.$$

The $(r+1)$-tuple $(a_r, a_{r-1}, ..., a_1, a_0)_{(l_r, l_{r-1}, ..., l_1, l_0)}$ is the *mixed radix representation* of i_a, and $(l_r, l_{r-1}, ..., l_1, l_0)$ is the *base* of the mixed radix number system.

5.3 Emulating a Hypercube with Single-Dimension Communication

In this subsection, we assume single-port communication, with all the nodes only capable of using links of the same dimension at the same time. This assumption is used in some SIMD architectures and their algorithms, in order to reduce the cost of implementation, and is called *single-dimension communication* in this paper. Such a model is also suitable for a parallel system using wormhole routing, and is assumed by some researchers (e.g., [6]), especially when one input can only drive one output.

Theorem 5.1 *The action of all the dimension-i_a links in a pk-cube can be emulated on an N-node RHS$(l_r, l_{r-1}, ..., l_1, Q_k)$, in $2|S_a| + 1$ time steps, where $p = \prod_{i=1}^{r} l_i$, $i_a = (a_r, a_{r-1}, ..., a_1, a_0)_{(l_r, l_{r-1}, ..., l_1, k)}$, $|S_a|$ is the number of elements in $S_a = \{i | a_i \neq 0, i = 1, 2, ..., r\}$ (i.e., the number of non-zero digits in the mixed-radix representation of i_a), and Q_k denotes a k-cube.*

Proof: Emulation is done by exchanging data via depth-i level-$(a_i + 1)$ links for all nonzero a_i with the ordering of i, $i = r, r-1, ..., 2, 1, 0, 1, 2, ..., r-1, r$, where the depth-0 level-j link denotes a nucleus dimension-j link. No congestion will occur. □

Clearly, any pk-cube algorithm with single-dimension communication can be emulated on an N-node RHS based on a k-cube with $2r + 1$ slowdown from Theorem 5.1, since $|S_a| \leq r$. We refer to this algorithm as $\mathcal{EQS}$ algorithm. It can be shown that no interconnection network can emulate a pk-cube with dilation asymptotically smaller than an RHS$(l_r, l_{r-1}, ..., l_1, Q_k)$ with $\log r = O(\log l_{max})$ and $|S| = \Theta(r)$, by a nonconstant factor, if they use nodes of degree bounded by a polynomial in k (i.e. $O(k^c)$, where c is a constant). In the above statement, $|S|$ is the number of elements in the set $S = \{i | \log l_i = \Theta(\log l_{max}), i = 1, 2, ..., r\}$ (i.e., $\sum_{i=1}^{r} \log l_i = \Theta(r \log l_{max})$), $p = \prod_{i=1}^{r} l_i$, and $l_{max} = max(k, l_i, i = 1, 2, ..., r)$.

5.4 Emulating a Hypercube with All-Port Communication

In this subsection, we assume all-port communication, with all the nodes capable of using links of all dimensions at the same time.

Theorem 5.2 *Any pk-cube algorithm with all-port communication can be emulated on an RHS$(l_r, l_{r-1}, ..., l_1, Q_k)$ with $2r - 1 + max_{i=1,2,...,r} \left(\frac{2pk}{l_i}, p\right)$ slowdown, where $p = \prod_{i=1}^{r} l_i$.*

Proof: In Subsection 5.3, we presented an algorithm emulating a hypercube on RHS networks based on a small hypercube with single-dimension communication. The emulation algorithm with all-port communication can be done simply by performing single-dimension emulation for all dimensions concurrently and with proper scheduling. Since $\frac{2pk}{l_i}$ packets will pass a depth-i level-j inter-cluster link; p packets will pass a nucleus link, and the dilation for emulating the pk-cube is $2r - 1$ (Theorem 5.1), the time required is given by $2r - 1 + max\left(\frac{2pk}{l_i}, p\right)$, $i = 1, 2, ..., r$. □

By properly choosing the nucleus size and, the number of recursive and hierarchical levels, we can emulate a hypercube with all-port communication on RHS networks with optimal slowdown.

6 Partially-Linked RHFC Networks

In this section, we define *partially-linked RHFC networks*, a subclass of RHFC networks, which can be constructed with fixed-degree modules, leading to the scalability and low cost.

An e-partially-linked RHFC$(l_r, l_{r-1}, ..., l_1, G)$ network has coefficient matrix

$$\sum_{i=1}^{r} a_{i,j} \leq e < r,$$

where $a_{i,j}$ is element i, j of the coefficient matrix $\mathcal{A}_{CM}$ of the RHFC network. In view of the above condition, we must assign the inter-cluster links such that at most e such *external links* are required per node. As a consequence, the degree of a partially-linked RHFC network is equal to that of its nucleus graph plus the fixed integer e.

For example, we can let $a_{1,j} = 2/3, a_{2,j} = 2/3, a_{3,j} = 1/2$, leading to only 2 external links per node. As another example, let $l_i = l$ and

$$a_{i,j} \approx (1-g)g^{i-1},\ 0 < g < 1, \tag{6}$$

for $i = 1, 2, ..., r$, $j = 1, 2, ..., l$. Then only one external link is required for each node in the resultant partially-linked RHFC network. If the recursive depth is a constant, $a_{i,j}$ will not approach zero and the diameter of such networks is guaranteed to be asymptotically optimal if a nucleus that has non-constant nodes and asymptotically optimal diameter is used (Theorem 3.3).

An r-deep hypernet (called (k, r) cubelet-based hypernet in [5]) is a special case of RHFC$(\underbrace{2, 2, ..., 2}_{r}, Q_k)$, whose coefficient matrix has elements $a_{i,j} = 2^{-i}$. If we let $l \geq 2$ and $a_{i,j} \in (0, 1]$, we obtain a generalization of hypernets, called *l-dimensional hypernet.* In this paper, we refer to the original hypernet [5] as a 2-dimensional hypernet.

6.1 Multi-Dimensional Hypernet

In this subsection, we present the construction of a multi-dimensional hypercube-based hypernet, which can emulate a hypercube with considerably smaller dilation and congestion than a 2-dimensional hypernet having similar node degree.

For simplicity, we construct a special case of r-deep l-dimensional hypernets based on a k-cube nucleus, with $a_{i,j} = 2^{-i}$, $i = 1, 2, ..., r$, $j = 2, 3, ..., l$, as its coefficient matrix, and call it (r, l, Q_k) net.

An r-deep l-dimensional hypernet based on a k-cube, or (r, l, Q_k) net, begins with a nucleus k-cube, which forms a $(1, 1, Q_k)$ net. A node in the (r, l, Q_k) net has k nucleus links, and $l - 1$ external links. We give each of the $l - 1$ external links of a node an integer ID ranging from 2 to l. A link with ID i, $i = 2, 3, ..., l$, is called a *level-i link* or dimension-i link. To build a $(1, 2, Q_k)$ net, we use $N_{1,2} = a_{1,2}N_{1,1} = 2^{k-1}$ identical copies of the nucleus Q_k, each having $N_{1,1} = 2^k$ nodes. Each nucleus is viewed as a depth-1 level-2 cluster, and is given a $(k-1)$-bit string X_2 as its address; we also give each node a k-bit string X_1 as its address within the nucleus to which it belongs. Node X_1 within nucleus X_2 has a $(2k-1)$-bit string $X_2' = X_2X_1$ as its address within the $(1, 2, Q_k)$ net. Each of the $N_{1,2} = 2^{k-1}$ nucleus copies has a link with ID equal to 2 connecting it to each of the other $N_{1,2} - 1 = 2^{k-1} - 1$ nuclei, via which node $X = X_2X_1$ connects to node $x_{k:2}X_20$ if $x_1 = 0$, where $x_{j_1:j_2}$ denotes the bits of the address of node X at positions $j_1, j_1 - 1, ..., j_2$, $j_2 \geq 1$. These links are called ***depth-1 level-2 external links*** or ***depth-1 dimension-2 external links.*** Clearly, $N_{1,1} - N_{1,2} + 1 = 2^{k-1} + 1$ dimension-2 external links remain unassigned in each nucleus (2^{k-1} of them belong to nodes whose least significant bit in the address is equal to 1).

To build a $(1, l, Q_k)$ net, we also use $N_{1,l} = a_{1,l}N_{1,1} = 2^{k-1}$ identical copies of a $(1, l-1, Q_k)$ net, which is viewed as a depth-1 level-l cluster. Each copy of the $(1, l-1, Q_k)$ net is given a $(k-1)$-bit string X_l as its address; each node is already given an $(lk - l - k + 2)$-bit string $X_{l-1}' = X_{l-1:1}$ as its address within the level-l cluster to which it belongs, where $X_{i:j} = X_iX_{i-1}\cdots X_{j+1}X_j$. Node X_{l-1}' within the level-l cluster X_l has a bit string $X_l' = X_lX_{l-1}' = X_{l:1}$ of length $k_1 = lk - l + 1$ as its address within the $(1, l, Q_k)$ net. Each of the $\prod_{j=2}^{l} N_{1,j} = 2^{(k-1)(l-1)}$ nuclei has a link connecting it to each of the other $N_{1,l} - 1 = a_{1,l}N_{1,1} - 1 = 2^{k-1} - 1$ level-l clusters via which node $X_lX_{l-1:2}X_1$ connects to node $x_{k:2}X_{l-1:2}X_l0$ if $x_1 = 0$. These links are called *depth-1 level-l external links* or *depth-1 dimension-l external links.* Clearly, there are $N_{1,1} - N_{1,l} + 1 = 2^{k-1} + 1$ unassigned dimension-l external links in a nucleus (2^{k-1} of them belong to nodes whose least significant bit in the address is equal to 1).

To build an $(r, 2, Q_k)$ net, we use $N_{r,2} = a_{r,2}N_{r,1} = 2^{k_{r-1}-r}$ identical copies of a $(r, 1, Q_k)$ net, which is the same as an $(r-1, l, Q_k)$ net and is viewed as a depth-r level-2 cluster. Each copy of the $(r, 1, Q_k)$ net is given a $(k_{r-1} - r)$-bit string $X_{r,2}$ as its address; each node is already given a k_{r-1}-bit string $X_{r,1}' = X_{r-1,l}' = X_{r-1,l:1,1}$ as its address within the $(r-1, l, Q_k)$ net to which it belongs, where $X_{i_1,j_1:1,1} = X_{i_1,j_1}X_{i_1,j_1-1}\cdots X_{i_1,1}$, $X_{i_1,1} = X_{i_1-1,l}$, and $X_{1,i} = X_i$. Each of the nuclei has a link connecting it to each of the other depth-r level-2 clusters via which node $X_lX_{l-1:2}X_1$ connects to node $x_{k_{r-1}:r+1}X_{r:1}X_{r-1,l-1}\cdots 0\underbrace{11\cdots 1}_{r-1}$, if the least significant r bits of the address of the node $x_{r:1} = 0\underbrace{11\cdots 1}_{r-1}$. These links are called *depth-r level-2 external links* or *depth-r dimension-2 external links.* Clearly, there is a depth-r dimension-2 external link for every 2^r nodes. The construction of an (r, l, Q_k) net follows similar rules, except that dimension-l external links are used. This recursive definition allow us to build an l-dimensional hypernet of arbitrary depth.

6.2 Emulating a Hypercube on Multi-Dimensional Hypernets

Multi-dimensional hypernets based on a k-cube can emulate a hypercube of the same size efficiently using an algorithm similar to that for a 2-dimensional hypernet [5].

For example, to route a packet to the dimension-n neighbor of the hypercube (which is the worst case) on an r-deep 2-dimensional hypernet, we have to first send the packet to the node that has a depth-r dimension-2 external link (i.e. whose address ends with $0\underbrace{11\cdots 1}_{r-1}$), requiring r steps in the worst case. Then we have to send it via depth-$(r-1)$, depth-$(r-2)$,...,depth-1 dimension-2 external links, requiring $\Theta(r)$ steps, and we finally send it to the destination.

To route a packet to the hypercube dimension-n neighbor on an r-deep l-dimensional hypernet, we use an algorithm that is essentially the same, except that the packet is sent via dimension-l external links.

The dilation for both networks is $\Theta(r)$. However, for networks of similar size and node degree, an l-dimensional hypernet can have considerably smaller depth. For example, for a network with N nodes of degree $\Theta(\sqrt{\log N})$, a 2-dimensional hypernet requires $\Theta(\log\log N)$ depth, since the degree of a 2-dimensional hypernet based on a hypercube will be reduced to approximately half when increasing the depth by one. As a result, the dilation is $\Theta(\log\log N)$ for a 2-dimensional hypernet to emulate a hypercube of the same size. On the other hand, an l-dimensional hypernet can have depth as small as 1 and node degree as small as $\Theta(\sqrt{\log N})$ when $l = \Theta(\sqrt{\log N})$. As a result, the dilation can be a constant for an l-dimensional hypernet of degree $\Theta(\sqrt{\log N})$ to emulate a hypercube of the same size.

Moreover, congestion for a multi-dimensional hypernet to emulate a hypercube can also be much lower than that for a 2-dimensional hypernet. For example, the slowdown is at least $\Theta(\log N)$ for an N-node 2-dimensional hypernet to emulate a hypercube with all-port communication, even its depth is 1 and the node degree is $\Theta(\log N)$; while only a slowndown of $\Theta(\sqrt{\log N})$ is required on a 1-deep l-dimensional hypernet with N-nodes of degree $\Theta(\sqrt{\log N})$ when $l = \Theta(\sqrt{\log N})$ (using an algorithm similar to that for an HSN [11]).

An advantage of 2-dimensional hypernet over the previous l-diemnsional hypernet example is that the traffic on each of its links is approximately balanced (when the sources and destinations are uniformly distributed over the network nodes). To overcome the unbalanced traffic problem, we can choose $a_{i,j} \approx (1-g)g^{i-1}$, $g \approx 1/l$, and the resultant traffic on the l-dimensional hypernet with the new coefficient matrix $\mathcal{A}_{C\mathcal{M}}$ will also be approximately balanced.

7 Scaling Up RHFC Networks

To scale up an r-deep RHFC network with smaller step size, we can use $N_{top} \leq M_{i,j}$ identical copies of the RHFC network, where $M_{i,j}$ is the number of nodes in a depth-i level-j cluster of the original RHFC network. Each copy is given a $k_{i,j}$-bit string as its address, which forms the most significant $k_{i,j}$ bits of the addresses of the nodes within that copy, where $k_{i,j} = \lceil \log_2 N_{top} \rceil$. The links connecting these copies can be obtained in a way similar to the construction of an HFC network. It can be seen that most algorithms presented in this paper can be applied to such networks with minor modifications. For example, to perform the semigroup computation on such networks, we simply perform the semigroup computation on each copy in parallel, exchange data via the new links, and then perform the semigroup computation on each depth-i level-j cluster in parallel. Analysis of the properties and performance of such networks is straightforward and is omitted.

8 Conclusion

In this paper, we have proposed a new class of interconnection networks, RHFC networks, for modular construction of massively parallel computers. We have shown that suitably constructed RHFC networks have asymptotically optimal diameter. We introduced RHS networks which can emulate a hypercube of the same size with asymptotically optimal slowdown with respect to their node degrees, assuming either single-port or all-port communication. We have also proposed multi-dimensional hypernets, a special case of partially-linked RHFC networks, that inherit many advantages of the 2-dimensional hypernet [5], such as scalability, small diameter, and low node degree, and can emulate a hypercube with considerably smaller dilation and congestion.

One can also build RHFC networks based on other nucleus graphs or hypergraphs such as a 2-D mesh, folded-hypercube, and high-bandwidth buslet. With various nucleus graphs and flexible connectivity, RHFC networks are adaptable to the needs of a wide range of applications.

References

[1] Bhuyan L.N. and D.P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Trans. Comput.*, vol. 33, no. 4, Apr. 1984, pp. 323-333.

[2] Breznay, P.T. and M.A. Lopez, "Tightly connected hierarchical interconnection networks for parallel processors," *Proc Int'l Conf. Parallel Processing,* vol. I, 1993, pp. 307-310.

[3] Duh D., G. Chen, and J. Fang, "Algorithms and properties of a new two-level network with folded hypercubes as basic modules," *IEEE Trans. Parallel Distrib. Sys.*, vol. 6, no. 7, Jul. 1995, pp. 714-723.

[4] Ghose, K. and R. Desai, "Hierarchical cubic networks," *IEEE Trans. Parallel Distrib. Sys.*, vol. 6, no. 4, Apr., 1995, pp. 427-435.

[5] Hwang, K. and J. Ghosh, "Hypernet: a communication efficient architecture for constructing massively parallel computers," *IEEE Trans. Comput.*, vol. 36, no. 12, Dec. 1987, pp. 1450-1466.

[6] Mišić J. and Z. Jovanović, "Communication aspects of the star graph interconnection network," *IEEE Trans. Parallel Distrib. Sys.*, Vol. 5, no. 7, Jul. 1994, pp. 678-687.

[7] Preparata, F.P. and J.E. Vuillemin, "The cube-connected cycles: a versatile network for parallel computation," *Commun. of the ACM,* vol. 24, no. 5, May 1981, pp. 300-309.

[8] Vecchia, G.D. and C. Sanges, "Recursively scalable networks for message passing architectures," *Proc. Conf. Parallel Processing and Applications,* 1987, pp. 33-40.

[9] Yeh, C.-H. and B. Parhami, "Unified formulation of a wide class of scalable interconnection networks based on recursive graphs," *Proc. Int'l Conf. Systems Engineering,* July 1996, to appear.

[10] Yeh, C.-H. and B. Parhami, "Swapped networks: unifying the architectures and algorithms of a wide class of hierarchical parallel processors," *Proc. Int'l Conf. Parallel and Distributed Systems,* June 1996, to appear.

[11] Yeh, C.-H. and B. Parhami, "Hierarchical swapped networks: efficient low-degree alternatives to hypercubes and generalized hypercubes," *Proc. Int'l Symp. Parallel Architectures, Algorithms, and Networks,* June 1996, to appear.