Embedding Meshes on the Star Graph

Sanjay Ranka
Syracuse University

Jhy-Chun Wang
Syracuse University, School of Computer and Information Science, jcwang@cs.uiuc.edu

Nangkang Yeh

Journal of Parallel & Distributed Computing, to appear

CIS-89-9

EMBEDDING MESHES ON THE STAR GRAPH

Sanjay Ranka
Jhy-Chun Wang
Nangkang Yeh

School of Computer and Information Science
Syracuse University
Syracuse, New York 13244-4100, U.S.A.
August 1989

Embedding Meshes on the Star Graph

Sanjay Ranka, Jhy-Chun Wang and Nangkang Yeh
Syracuse University

August 25, 1989

Abstract

We develop algorithms for mapping n-dimensional meshes on a star graph of degree n with expansion 1 and dilation 3. We show that an n-degree star graph can efficiently simulate an n-dimensional mesh.

Keywords and Phrases

Star graph, mesh, embedding, sorting

1 Introduction

One of the primary design considerations for a large multiprocessor system is the topology of the interconnection network used for communication between processors. Some of the important interconnection networks which have been studied so far are hypercube, meshes, trees, mesh-of-trees and pyramids [HWAN84]. Hypercube is one of the more popular general purpose networks due to its small degree, short diameter, symmetry, optimal fault tolerant properties, embedding of other networks, and a fast sorting algorithm [RANK88] [AKER87]. Akers, *et al.* [AKER87] presented a new interconnection network "star graph" as an alternative to hypercube. They showed that with degree-n, $(n+1)!$ nodes could be connected using a star graph as compared to only 2^n nodes for a hypercube. The diameter of star graph is asymptotically superior to the hypercube. It has also been shown that the star graph is maximally fault tolerant [AKER87].

One of the important properties of any general interconnection networks is that it should be able to embed other interconnection networks with low overheads. Most of the problems, which utilize parallel processing, use n-dimensional matrices ($n \geq 2$) as one of the primary data structures. This is especially true in the areas of numerical analysis, image processing, computer vision and pattern recognition. These applications require operations which utilize data which is proximate from the point of view of the mesh topology. Thus efficient embedding of n-dimension meshes is an important feature of any general purpose interconnection network.

It has already been shown that meshes can be efficiently embedded in hypercubes [SAAD88] [CHAN88]. In this paper we develop algorithms for mapping an $(n-1)$-dimensional mesh on a star graph of size $N = 2 * 3 * 4 * \cdots * n$. This mapping has a dilation 3 and expansion 1.

One of the important data movement operations on an SIMD n-dimensional mesh is a unit route [NASS81]. In a unit route on an SIMD n-dimensional mesh, data can be moved along any one of the n dimensions by each processor. This is one of the primary operations used for development of almost all parallel algorithms on a mesh. We show that our mapping can perform this operation efficiently on the star graph.

Thus most algorithms for the $(n-1)$-dimensional mesh of size $2*3*4*\cdots*n$ can be efficiently simulated on the star graph of size $N = n!$. There have been a great deal of research done on parallel algorithms on meshes [ATAL84], [NASS79], [NASS80], [THOM77]. However most of these algorithms assume n-dimensional meshes which are equal in size along all the n dimensions. At this stage, it does not seem very obvious that uniform meshes can be efficiently mapped on the star graph. We use the results of Atallah [ATAL88] to give weak upper bounds on the mapping of uniform meshes on the star graph.

The rest of the paper is described as follows. In section 2, we describe the star graph and the model of computation. In section 3, we describe the embedding of an n-dimensional mesh on an n degree star graph. In section 4 we show that this embedding can be used efficiently to simulate a mesh on a star graph.

2 Model of Computation

A block diagram of an SIMD multicomputer is given in Figure 1. The features and programming notations used to describe data movement operations on the star graph are as follows:

1. There are $N = n!$ nodes connected together via an "n-star graph", S_n (to be described later). Each PE has a unique representation $(a_{n-1}a_{n-2}\cdots a_1a_0)$ which is a permutation of $(n-1\ n-2\cdots 3\ 2\ 1\ 0)$ (*i.e.*, $a_i \neq a_j, i \neq j$ and $0 \leq a_i \leq n-1$). We shall use $A(\pi)$ to refer to a register value in PE with a representation π. The local memory of each PE holds data only. Each PE is capable of performing only the basic arithmetic operations.

2. There is a separate program memory and control unit. The control unit performs instruction sequence, fetching, decoding. In addition, instructions and masks are broadcasted by the control unit to the PEs for execution. An instruction mask is a boolean function used to select certain PEs to execute an instruction. For example, in the instruction :

$$A(i) := A(i) + 1, \; (\; f(i) = y \;)$$

only PEs' i satisfying the condition $f(i) = y$ will execute the instruction.

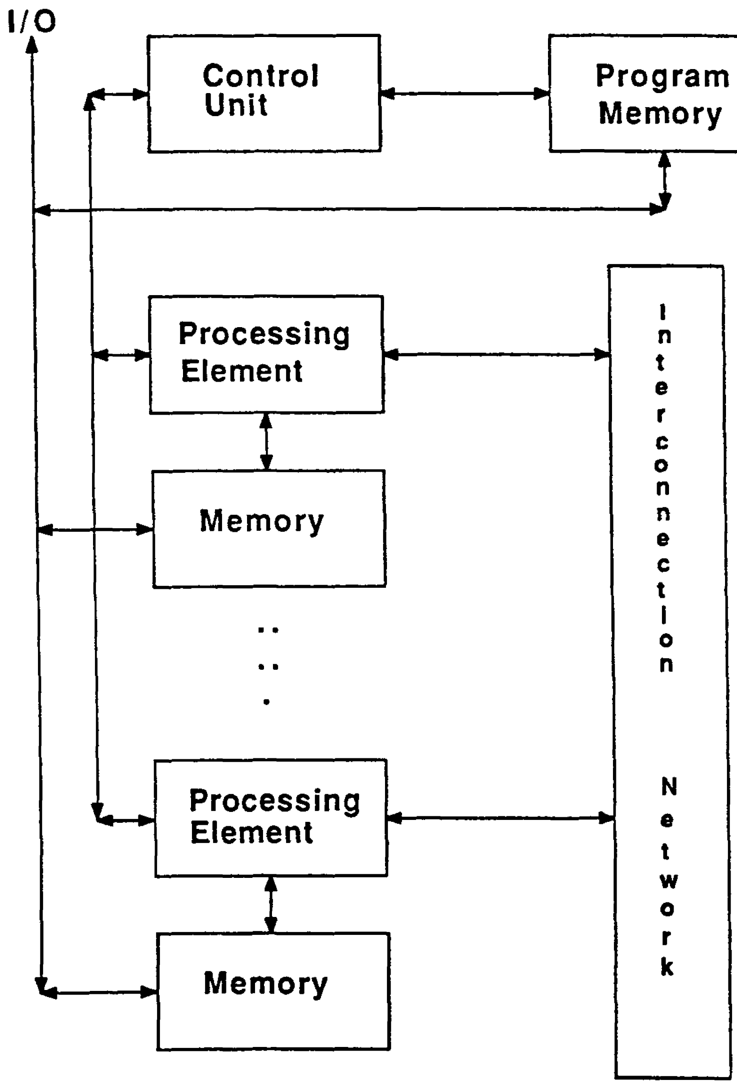

Figure 1: SIMD multicomputer

3. The topology of a 4-star graph is shown in Figure 2. An n-star graph, S_n, connects $N = n!$ PEs. Each PE $\pi = (a_{n-1} \cdots a_0)$, a permutation of $(n-1$ $n-2 \cdots 3\ 2\ 1\ 0)$, is connected to nodes $(a_i a_{n-2} a_{n-3} \cdots a_{i+1} a_{n-1} a_{i-1} \cdots a_0\ )$ ($0 \leq i \leq n-2$). We will use the notation $\pi^{(i)}$ to represent these nodes.

   The topology of a $2 * 3 * 4$ mesh is shown in Figure 3. An m-dimensional mesh M of size $N = l_1 * l_2 * \cdots * l_m$ can be represented by an m-dimensional array $D(l_m, l_{m-1}, \cdots, l_1)$ where l_i is the size of the i^{th} dimension. Each PE $d = (d_m, d_{m-1}, \cdots, d_1)$ is connected to PEs $(d_m, \cdots, d_j \pm 1, \cdots, d_1)$, $1 \leq j \leq m$, provided they exist.

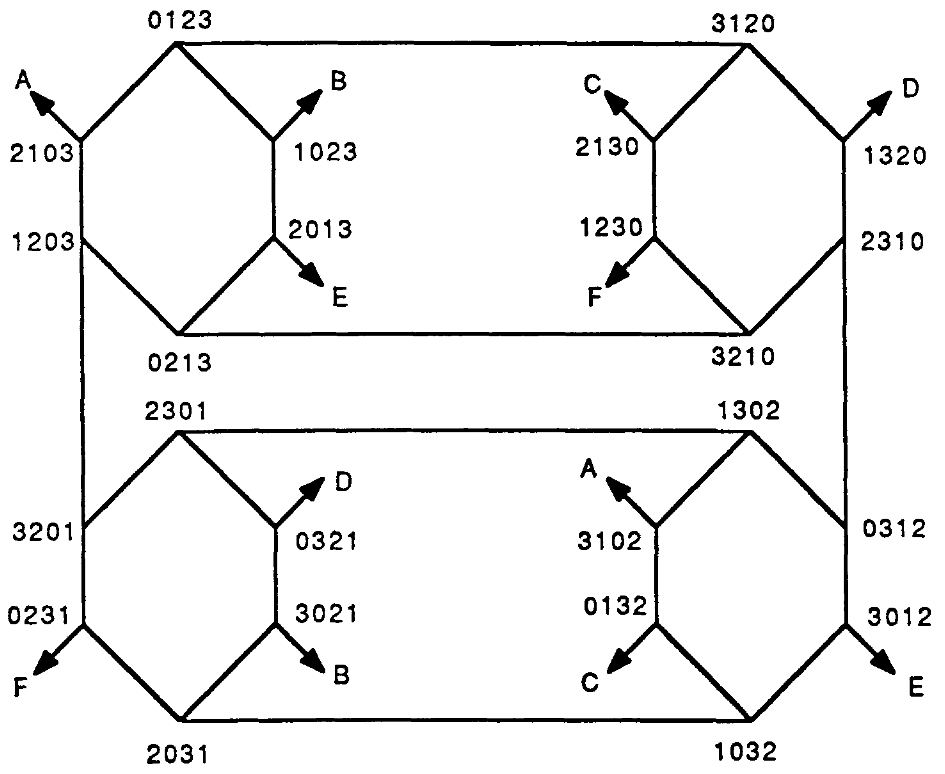

Figure 2: A star graph of degree 3

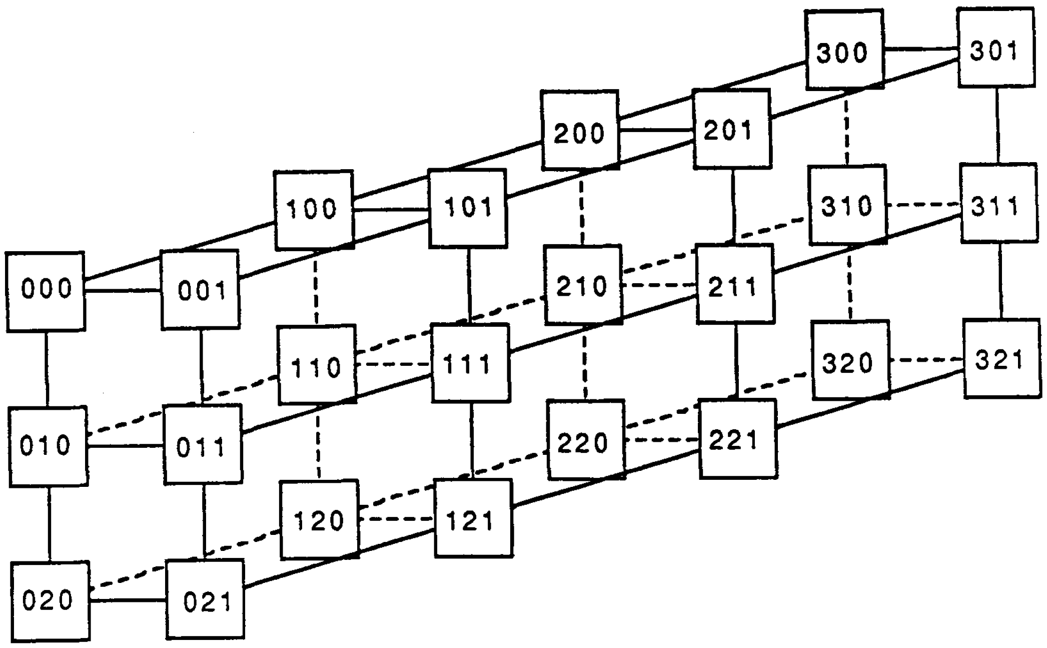

Figure 3: A $2 * 3 * 4$ mesh

4. Interprocessor assignments are denoted using the symbol $\leftarrow$ while intraprocessor assignments are denoted using the symbol :=. Thus the assignment statement:

$$B(i^{(2)}) \leftarrow B(i)$$

implies that all processors transmit their message to neighbor along dimension 2.

5. In a *unit route*, data may be transmitted from one processor to another if it is directly connected. We will assume two models of computation SIMD-A and SIMD-B. In model SIMD-A each processor can transmit data along a particular dimension i, $1 \leq i \leq n-1$, in a unit route. In model SIMD-B each processor can transmit data to any one neighbor in one unit route.

6. The asymptotic complexity of all our algorithms is determined by the number of unit routes. Thus our complexity analysis will only count these.

Some of the important properties of the star graph are as follows [AKER87], [AKER89]:

1. Each node is symmetrical to every other node.
2. The diameter k_n of the star graph S_n is $\lfloor\frac{3(n-1)}{2}\rfloor$.
3. Broadcasting can be performed on the star graph in at most $3(n\log n - \frac{n}{2})$ unit routes in a star graph S_n.
4. A star graph is maximally fault tolerant.

3 Mapping

3.1 Definitions

A set of interconnected processors may be modeled as an undirected graph in which each vertex denotes a unique processor and there is an edge between two vertices iff the corresponding processors are directly connected. Let G and S be two undirected graphs that model two sets of interconnected processors. For any undirected graph G we use $V(G)$ and $E(G)$, respectively, to denote the vertices and edges in G. An *embedding* of G into S is a mapping of $V(G)$ into $V(S)$ and of $E(G)$ into simple paths of S. We will denote this mapping from G to S by m. Thus any vertex x of G is mapped to $m(x)$ of S. The vertex mapping is such that each vertex of G is mapped to a distinct vertex of S (*i.e.*, $m(x) \neq m(y)$, $x \neq y$, where $x, y \in V(G)$). Note that for an embedding to exist $|S| \geq |G|$. Also, if S is connected and $|S| \geq |G|$ then an embedding always exists. As an example, consider the graphs G and S of Figure 4. The vertex mapping$(1 \rightarrow a, 2 \rightarrow b, 3 \rightarrow c, 4 \rightarrow d)$ and the edge to path mapping ($(1,2) \rightarrow ab, (2,4) \rightarrow bad, (4,3) \rightarrow dac, (3,1) \rightarrow ca)$ defines an embedding of G into S.

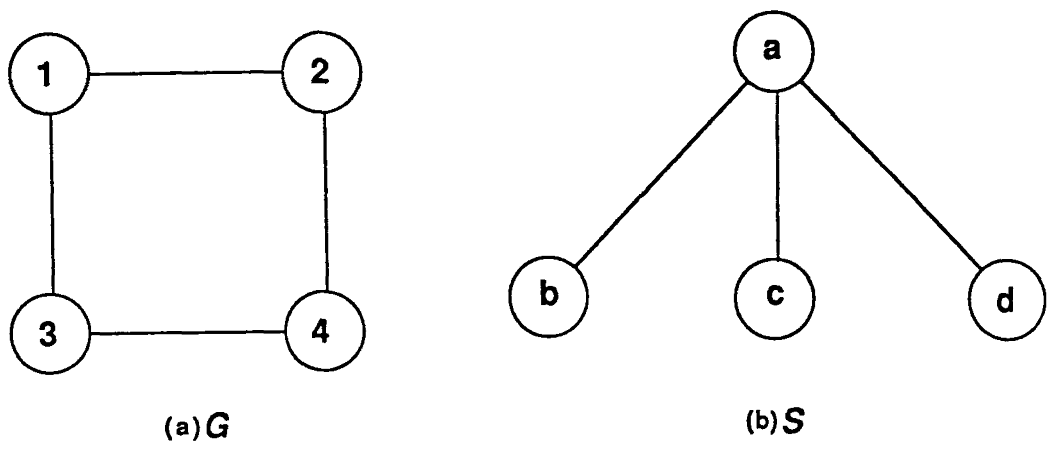

Figure 4: Example of graphs

The ratio $|S|/|G|$ is called the *expansion*. The *dilation* is defined as follows:

$$dilation = \max_{\forall(x,y)\in E(G)} \{length\ of\ the\ shortest\ path\ from\ m(x)\ to\ m(y)\}$$

The *congestion* of any edge e of S is the number of paths (each path representing an edge of G mapped to S) of G which contain e. The maximum of the congestions of all edges of S is the congestion of the embedding. For the above example, the expansion is 1 while the dilation and congestion are both 2. In the remainder of this section the graph S will always be the graph corresponding to a star graph interconnection. S_n will denote the n-star graph. This graph has $n!$ vertices and each vertex has degree $n-1$.

3.2 Mapping Strategy

In this section, we describe a mapping of an $(n-1)$-dimensional mesh (no wrap arounds), D_n, of size $2 * 3 * 4 * \cdots * n$ on a star graph S_n. Since the number of nodes in the both graphs are equal, we are considering a mapping of expansion 1.

Lemma 1: There is no dilation 1 embedding existing for D_n on S_n for $n > 2$.

Proof: In any dilation 1 embedding the degree of a node in D_n should be less than or equal to the degree of a node in S_n. A node in D_n (namely $(1, 1, \cdots, 1)$) can have a degree $(2n - 3)$. Thus there is no dilation 1 embedding of D_n on S_n when $2n - 3 > n - 1$ (or $n > 2$). □

Let the mapping of a node $(d_{n-1}, d_{n-2}, \cdots, d_1)$ in D_n be mapped on to $(p(n-1)\ p(n-2) \cdots p(0))$ node on a star graph (Thus the array $p[0..n-1]$ represents the mapping). Assume that node $(0, 0, 0 \cdots, 0)$ gets mapped to $(n-1\ n-2 \cdots 2\ 1\ 0)$. The algorithms for mapping any node of D_n on S_n and for inverse mapping any of node of S_n on D_n are described in Figure 5 and Figure 6 respectively.

```
procedure CONVERT-D-S($d, n, p$)
    for $i := 0$ to $n - 1$ do
        $q(i) := i$
    end
    for $i := 1$ to $n - 1$ do
        for $j := 1$ to $d(i)$ do
            exchange $q(i - j)$ and $q(i - j + 1)$
        end
    end
    for $i := 1$ to $n - 1$ do
        $p(q(i)) := i$
    end
end
```

Figure 5: Algorithm for mapping any node in D_n on S_n

```
procedure CONVERT-S-D($p, n, d$)
    for $i := 0$ to $n-1$ do
        $q(i) := p(i)$
    end
    for $i := n-1$ to 1 do
        if $i > q(i)$ then
            [ $d(i) := i - q(i)$
            for $j := i-1$ to 1 do
                if $q(j) \geq i$ then $q(j) := q(j) - 1$
            end ]
        else $d(i) := 0$
    end
end
```

Figure 6: Algorithm for mapping any node in S_n on D_n

Let $(a\ b)$ represent the exchange of symbols a and b. The algorithm in Figure 5 can be explained by Table 1. The row i represents the sequence of exchanges along dimension i.

i	Sequence of exchanges
1	(0 1)
2	(1 2) (0 1)
.	.
.	.
$n-1$	$(n-2\ n-1)\ (n-3\ n-2) \cdots (2\ 3)\ (1\ 2)\ (0\ 1)$

Table 1: Sequence of exchanges

Consider the mapping of $(3,0,1)$ on S_4. Since $d_1 = 1$ a (0 1) exchange is performed on (3 2 1 0) giving (3 2 0 1). In the next iteration since $d_2 = 0$ there is no need to perform any exchange. In the next iteration $d_3 = 3$, thus we need to perform a (2 3) exchange followed by a (1 2) exchange and a (0 1) exchange. This gives the following sequence:

(2 3) (2 3 0 1)
(1 2) (1 3 0 2)
(0 1) (0 3 1 2)

Thus node (3,0,1) is mapped to node (0 3 1 2) on the star graph. The complexity of the algorithm is $O(n^2)$.

The algorithm in Figure 6 can be seen as to reverse the sequence of exchanges in Table 1 to rearrange $(p(n-1)p(n-2)\cdots p(0))$ into $(n-1\ n-2\cdots 1\ 0)$. The difference (operation $d(i) := i - q(i))$ means there are $d(i)$ symbols greater than $q(i)$ between $q(i-1)$ and $q(0)$, *i.e.*, it must takes exact $d(i)$ exchanges to rearrange i into $q(i)$.

Consider the inverse mapping of (0 2 1 3) on D_n. Since $p(3) = 0$, $d(3) = 3 - 0 = 3$ and we need to perform a (0 1) exchange followed by (1 2) and (2 3) exchanges to get $p(3) = 3$. This gives the following sequence:

(0 1) (1 2 0 3)
(1 2) (2 1 0 3)
(2 3) (3 1 0 2)

Then $p(2) = 1$, $d(2) = 2 - 1 = 1$ and a (1 2) exchange is performed on (3 1 0 2) giving (3 2 0 1). Now $p(1) = 0$, $d(1) = 1 - 0 = 1$ and a (0 1) exchange is performed on (3 2 0 1) resulting in (3 2 1 0). Thus node (0 2 1 3) is mapped to node (3,1,1) on the mesh. The complexity of this algorithm is also $O(n^2)$. Figure 7 describes the mapping between $V(D_4)$ and $V(S_4)$.

D_4	S_4	D_4	S_4
(0,0,0)	(3 2 1 0)	(2,0,0)	(1 3 2 0)
(0,0,1)	(3 2 0 1)	(2,0,1)	(1 3 0 2)
(0,1,0)	(3 1 2 0)	(2,1,0)	(1 2 3 0)
(0,1,1)	(3 1 0 2)	(2,1,1)	(1 2 0 3)
(0,2,0)	(3 0 2 1)	(2,2,0)	(1 0 3 2)
(0,2,1)	(3 0 1 2)	(2,2,1)	(1 0 2 3)
(1,0,0)	(2 3 1 0)	(3,0,0)	(0 3 2 1)
(1,0,1)	(2 3 0 1)	(3,0,1)	(0 3 1 2)
(1,1,0)	(2 1 3 0)	(3,1,0)	(0 2 3 1)
(1,1,1)	(2 1 0 3)	(3,1,1)	(0 2 1 3)
(1,2,0)	(2 0 3 1)	(3,2,0)	(0 1 3 2)
(1,2,1)	(2 0 1 3)	(3,2 1)	(0 1 2 3)

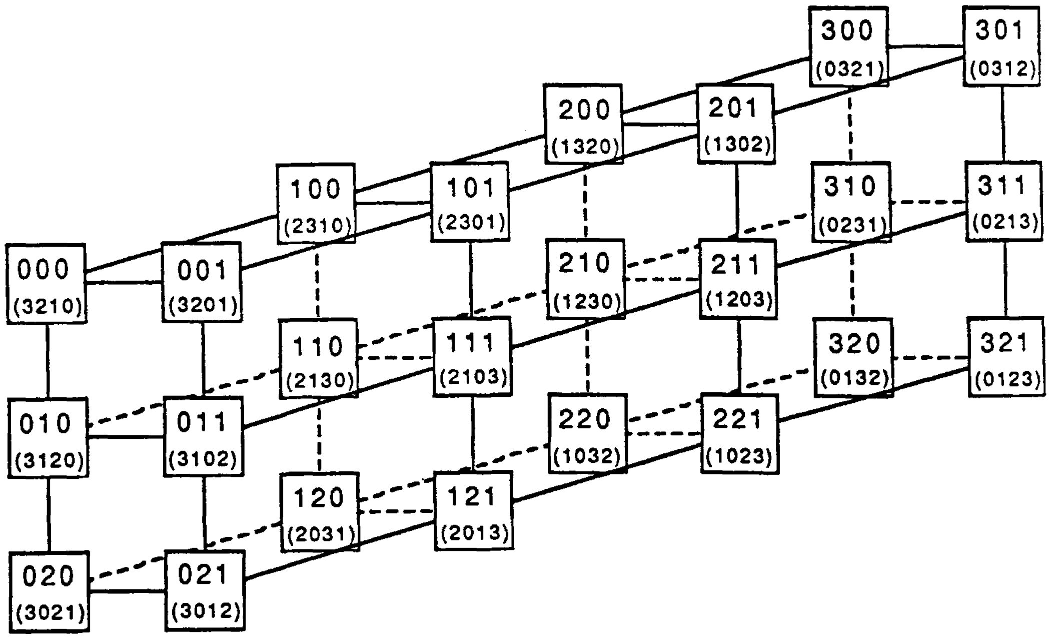

Figure 7: Mapping of $V(D_4)$ into $V(S_4)$

Definition 1: Let $\pi_{(i,j)}$ denotes the node by exchanging symbols i and j, $i \neq j$, $0 \leq i,j \leq n-1$ of a node π.
Example: Let $\pi = (3\ 1\ 4\ 2\ 0)$, then $\pi_{(2,3)} = (2\ 1\ 4\ 3\ 0)$.

Lemma 2: The shortest distance between π and $\pi_{(i,j)}$ is either 1 or 3.

Proof: If $\pi = (i \cdots)$ or $(j \cdots)$ the distance is easily seen to be 1. Without loss of generality $\pi = (k \cdots i \cdots j \cdots)$. Let $\pi_1 = (i \cdots k \cdots j \cdots)$ and $\pi_2 = (j \cdots k \cdots i \cdots)$. Then one of the shortest path of π to $\pi_{(i,j)}$ is via π_1 and π_2 as the only intermediate nodes. Thus the shortest distance between π and $\pi_{(i,j)}$ in this case is 3. $\Box$

Definition 2: Let $\pi = (a_{n-1} \cdots a_0)$ be the node corresponding to the mesh node $(d_{n-1}, d_{n-2}, \cdots, d_1)$. Let π_{k+} and π_{k-} represent the nodes corresponding to mesh nodes $(d_{n-1}, \cdots d_k + 1, d_{k-1}, \cdots, d_1)$ and $(d_{n-1}, \cdots d_k - 1, d_{k-1}, \cdots d_1)$, respectively (if they exist).

Lemma 3: π_{k+} and π_{k-} satisfy the following properties:
$\pi_{k+} = (a_{n-1} \cdots a_{k+1} a_l a_{k-1} \cdots a_{l+1} a_k a_{l-1} \cdots a_0)$,
where $a_l = \max \{\ a_t | a_t < a_k, 0 \leq t < k\ \}$,and
$\pi_{k-} = (a_{n-1} \cdots a_{k+1} a_m a_{k-1} \cdots a_{m+1} a_k a_{m-1} \cdots a_0)$,
where $a_m = \min \{\ a_t | a_t > a_k, 0 \leq t < k\ \}$
Thus $\pi_{k+} = \pi_{(a_k, a_l)}$
and $\pi_{k-} = \pi_{(a_k, a_m)}$

Proof: Let π^k denote the node corresponding to the node $(0, 0, \cdots, 0, d_k, \cdots, d_1)$. From Table 1, d_k implies taking the sequence of exchanges $(k-1\ k)\ (k-2\ k-1) \cdots (r\ r+1)$ where $r = k - d_k$. Let $\pi^k = (n-1\ n-2 \cdots k+1\ r\ j_{k-1} \cdots r-1 \cdots j_0)$. π^k_{k+} requires one more exchange $(r-1\ r)$. Thus $\pi^k_{k+} = \pi^k_{(r\ r-1)} = (n-1\ n-2 \cdots k+1\ r-1\ j_{k-1} \cdots r \cdots j_0)$.

Let $\pi(k)$ denote the symbol in the position k of π and $\pi[k]$ denote the position of symbol k in π. *e.g.*, $\pi^k(k) = r$ and $\pi^k_{k+}[r-1] = k$. We make the following observation:

if $t - d_t > r,\ k+1 \leq t < s$ and $k+1 \leq s \leq n-1$ $(r = k - d_k)$

then $\pi^t_{k+} = \pi^t_{(r\ r-1)},\ k+1 \leq t < s$

This is true since there is no exchange $(r\ r+1)$ (or $(r-1\ r)$) among these sequences of exchanges along t dimensions. Hence we only concentrate on the following cases:

$s - d_s \leq \pi^{s-1}(k),\ k+1 \leq s \leq n-1$.

We will use mathematical induction to prove our result. Let d_s be the first dimension such that $s - d_s \leq \pi^{s-1}(k)$. Let $(r \equiv \pi^{s-1}(k))$.

Thus $\pi_{k+}^{s-1} = \pi_{(r\ r-1)}^{s-1}$ when d_s is the first dimension such that $s - d_s \leq \pi^{s-1}(k)$.

When $s - d_s = r$:

$\pi^s(k) = r+1,\ \pi_{k+}^s(k) = r-1,$

$\pi^s[r-1] = \pi^{s-1}[r-1],\ \pi_{k+}^s[r+1] = \pi_{k+}^{s-1}[r] = \pi^{s-1}[r-1]$

Thus $\pi_{k+}^s = \pi_{(r+1\ r-1)}^s$ and $\pi^s[r] = s > k$.

When $s - d_s < r$:

$\pi^s(k) = r+1,\ \pi_{k+}^s(k) = r,$

$\pi^s[r] = \pi^{s-1}[r-1] = \pi_{k+}^{s-1}[r], \pi_{k+}^s[r+1] = \pi_{k+}^{s-1}[r]$

Thus $\pi_{k+}^s = \pi_{(r+1\ r)}^s$.

Assume d_u be the m^{th} dimension such that $u - d_u \leq \pi^{u-1}(k)$ and $\pi_{k+}^u = \pi_{(x\ y)}^u$ where $x > y$, $\pi^u(k) = x, \pi^u[y] < k$ and $\pi^u[t] > k,\ y+1 \leq t \leq x-1$.

Let d_v be the $(m+1)^{th}$ dimension such that $v - d_v \leq \pi^{v-1}(k)$.

Thus $\pi_{k+}^{v-1} = \pi_{(x\ y)}^{v-1}$, $\pi^{v-1}(k) = x$, $\pi^{v-1}[y] < k$ and $\pi^{v-1}[t] > k,\ y+1 \leq t \leq x-1$.

Consider the following cases:

When $y < v - d_v \leq x$:

$\pi^v(k) = x+1,\ \pi_{k+}^v(k) = y,$

$\pi^v[y] = \pi^{v-1}[y],\ \pi_{k+}^v[x+1] = \pi_{k+}^{v-1}[x] = \pi^{v-1}[y]$

Thus $\pi_{k+}^v = \pi_{(x+1\ y)}^v$ and $\pi^v[t] = \pi^{v-1}[t-1] > k,\ v - d_v < t \leq x$; $\pi^v[t] = \pi^{v-1}[t] > k$, $y < t < v - d_v$; $\pi^v[v - d_v] = v$.

When $v - d_v \leq y$:

$\pi^v(k) = x+1,\ \pi_{k+}^v(k) = y+1,$

$\pi^v[y+1] = \pi^{v-1}[y] = \pi_{k+}^{v-1}[x],\ \pi_{k+}^v[x+1] = \pi_{k+}^{v-1}[x]$.

Thus $\pi_{k+}^v = \pi_{(x+1\ y+1)}^v$, and $\pi^v[t] = \pi^{v-1}[t-1] > k,\ y+2 \leq t \leq x$.

From the above $\pi_{k+} = \pi_{(a_k, a_l)}$ and satisfies the condition of the lemma. The proof of π_{k-} is similar. □

As an example, consider π=(2 3 4 0 1) (corresponding to node (2,1,0,1)), then

π_{3+} =(2 1 4 0 3) and π_{3-} =(2 4 3 0 1). And the edge to path mapping is (((2,1,0,1), (2,2,0,1)) $\rightarrow$ (2 3 4 0 1) (3 2 4 0 1) (1 2 4 0 3) (2 1 4 0 3), ((2,1,0,1),(2,0,0,1)) $\rightarrow$ (2 3 4 0 1) (3 2 4 0 1) (4 2 3 0 1) (2 4 3 0 1)).

Theorem 4: The above embedding, m, of an $(n-1)$-dimensional mesh D_n on S_n has a dilation 3.

Proof: From lemma 2 and lemma 3. □

From theorem 4 we know that the maximum distance between any two neighboring nodes in D_n are mapped onto nodes such that distance between the mapped nodes in S_n is at most 3. A unit route for data movement in an SIMD-A mesh is defined to be the movement of data along either the positive or negative direction along dimension i, $1 \leq i \leq n-1$, by all the PE's in the mesh. This definition is similar to the definition of a unit route on the star graph defined in the previous section. The following lemma shows that a unit route of an SIMD-A mesh can be simulated on an SIMD-B star graph in a constant number of unit routes. Thus any algorithm which can be computed in $T(n)$ communication steps on an SIMD-A mesh can be completed in at most $3T(n)$ communication steps on the SIMD-B star graph.

Lemma 5: Let d_1 and d_2 represent two distinct nodes in D_n. Let $\pi^1 = m(d_1)$ and $\pi^2 = m(d_2)$ corresponding to the nodes on S_n. Then the path from π^1 to π^1_{k+} (if π^1_{k+} exists) is not blocked at any node by a path from π^2 to π^2_{k+} (if π^2_{k+} exists).

Proof: We assume that π^1_{k+} and π^2_{k+} exist. Since d_1 and d_2 are distinct $\pi^1 \neq \pi^2$. In the case $k = n-1$ the two paths are, $(\pi^1 \rightarrow \pi^1_{(n-1)+})$ and $(\pi^2 \rightarrow \pi^2_{(n-1)+})$, respectively. Since $\pi^1 \neq \pi^2$ the paths do not block each other.

Consider the case $k < (n-1)$. Let the path between π^1 and π^1_{k+} be $(\pi^1 \rightarrow X_1 \rightarrow Y_1 \rightarrow \pi^1_{k+})$ and the path between π^2 and π^2_{k+} be $(\pi^2 \rightarrow X_2 \rightarrow Y_2 \rightarrow \pi^2_{k+})$. Since $\pi^1 \neq \pi^2$, $\pi^1_{k+} \neq \pi^2_{k+}$. Thus $(\pi^1 X_1) \neq (\pi^2 Y_2)$ and $(Y_1 \pi^1_{k+}) \neq (Y_2 \pi^2_{k+})$. We have to now show $(X_1 Y_1) \neq (X_2 Y_2)$. Clearly $X_1 \neq X_2$ as the first edge represents exchange of the k^{th} symbol with the 1^{th} symbol and $\pi^1 \neq \pi^2$ (Similarly $Y_1 \neq Y_2$ because $\pi^1_{k+} \neq \pi^2_{k+}$). So $(X_1 Y_1) \neq (X_2 Y_2)$. Thus the path from π^2 to π^2_{k+} does not block the path from π^1 to π^1_{k+}. □

Theorem 6: A unit route of an SIMD-A mesh, D_n, can be performed in 3 unit routes on an SIMD-B, S_n.

Proof: From theorem 4 and lemma 5. □

4 Embedding Uniform Meshes

In the previous section, we described an embedding of a $2 * 3 * 4 \cdots * n$ mesh on S_n. Thus any algorithm which can be executed in $T(n)$ communication steps on an SIMD-A mesh can be executed on the SIMD-B star graph in $O(T(n))$ units of time. We will assume the SIMD-A model for the mesh and SIMD-B model for the star graph for the rest of this section. The results can be easily derived for the SIMD-A model of the star graph by multiplying with a factor of $O(n)$.

Most previous algorithms for the mesh have been designed assuming all the dimensions are of equal length. We will refer to them as uniform meshes. Let U denote a d-dimensional uniform mesh of N processors, *i.e.*, U is an $N^{1/d} * N^{1/d} * \cdots * N^{1/d}$ mesh. Let R denote an $l_1 * l_2 * \cdots * l_d$ mesh when $\Pi_{i=1}^{d} l_i = N$. The following theorem by Atallah [ATAL88] shows that R can efficiently simulate U when $d = O(1)$.

Theorem 7: [ATAL88] If $d = O(1)$, then mesh R can simulate every step of mesh U in $O((\max_i\ l_i)/N^{1/d})$ steps.

Since $N = n!$, using Stirling's approximation we have

$$\begin{aligned} N^{1/(n-1)} &= (n!)^{1/(n-1)} \\ &\approx (\sqrt{2\pi n}(n/e)^n)^{1/(n-1)} \\ &= (n/e)(2\pi)^{1/2(n-1)} n^{1/2(n-1)} (n/e)^{1/(n-1)} \\ &= \Theta(n) \end{aligned}$$

The above theorem may suggest that R (and hence the star graph) can simulate a uniform meshes efficiently ($n/N^{1/n} = O(1)$). However the above theorem assumes $d = O(1)$. The following theorem modifies the above theorem to take d in account.

Theorem 8: Mesh R can simulate every step of mesh U in $O((\max_i\ l_i)2^d/N^{1/d})$ steps.

Proof: The proof follows from the proof of theorem 7 giving in [ATAL88]. □

Using theorem 8 the number of steps required to simulate an $N = (N)^{1/(n-1)} * (N)^{1/(n-1)} * \cdots * (N)^{1/(n-1)}$ mesh on an $N = 2 * 3 * 4 \cdots * n$ mesh requires in the worst case $O(2^{n-1}n/N^{1/(n-1)})$ $=O(2^n)$ $=O(N^{n/\log_2 N})$ steps. This can be approximated by $O(N^{1/\log n + \log_2 e/\log^2 n})$ using Stirling's approximation and neglecting the lower order terms in the exponent.

Theorem 9: Any step of an $(n-1)$-dimensional uniform mesh of size $N^{1/(n-1)} * N^{1/(n-1)} * \cdots * N^{1/(n-1)}$ can be completed on a star graph in $O(N^{n/\log_2 N})$ steps.

Proof: From theorem 7 and theorem 8. □

For the same number of processors it is typical to expect a higher dimensional mesh to perform better than a lower dimensional mesh. This is due to smaller diameter and higher degree of each node. Let $F(N, d)$ be the number of steps required to solve a problem on an SIMD-A d-dimensional mesh with N processors. Thus by theorem 8 it can be solved on an SIMD-B star graph in $O((\max_i\ l_i)2^d/N^{1/d}) * F(N, d))$ time $(N = l_1 * l_2 * \cdots * l_d)$. This suggest that a dimension lower than n may be performed to simulate a uniform mesh algorithm on the star graph (see Appendix).

5 Conclusion

In this paper we have developed an efficient mapping of the mesh D_n on the star graph S_n. We have also shown that S_n can efficiently simulate D_n. The analysis in this paper indicates that the algorithms developed for uniform meshes may not be efficiently simulated on the star graph. Thus a sorting algorithm which can be executed in $O(d^2N^{1/d})$ time on a d-dimensional mesh [NASS79] may require $O(nN^{1/n}N^{1/\log n})$ time on the star graph. Further most of the sorting algorithms which sort an $N^{1/d} * N^{1/d} \cdots * N^{1/d}$ mesh requires $N^{1/d}$ to be power of 2 ([ATAL84], [NASS79], [NASS80], [THOM77]) as they used divide and conquer. When $N^{1/d}$ is not a power of 2, then a factor of (2^d) needs to be multiplied to the time. When $d = O(1)$, this can be considered as a constant factor. However while considering the sorting on an $N = 2*3*4 \cdots *n$ mesh, $d \neq O(1)$ and hence those algorithms cannot be efficiently executed.

Shear sort ([SCHE89]) is one method which does not use divide and conquer, but it does not seem that it can be easily extended to dimensions greater than 2. We are considering the problem of sorting on a rectangular mesh under the above conditions and have discovered an algorithm which is useful for sorting on the star graph.

Appendix

It is easy to see that an $N = 2 * 3 * 4 \cdots * n$ mesh can simulate a d-dimensional mesh of size $N = l_1 * l_2 * l_3 * \cdots * l_d$ in $O(1)$ time such that:

$$\begin{aligned}
l_1 &= n * (n-d) * (n-2d) * \cdots * (n - \lceil n/d - 1 \rceil d) \\
l_2 &= (n-1)*((n-1)-d)*((n-1)-2d)*\cdots*((n-1)-\lceil n/d-1\rceil d) \\
l_3 &= (n-2)*((n-2)-d)*((n-2)-2d)*\cdots*((n-2)-\lceil n/d-1\rceil d) \\
&\vdots \\
&\vdots \\
&\vdots \\
l_d &= ((n-(d-1))*((n-(d-1))-d)*\cdots*((n-(d-1))-\lceil n/d-1\rceil d)
\end{aligned}$$

Further $l_1/l_d \leq n(1 + n \bmod d) \leq nd$. Thus $l_1 \leq d(N^{1/d}n^{1-1/d})$. Consider a problem of size N that can be completed in $F(N,d)$ time on a d-dimensional mesh. Thus a d-dimensional mesh ($N = N^{1/d} * N^{1/d} * \cdots * N^{1/d}$) can be simulated on a star graph in $O(dN^{1/d}n^{1-1/d}2^d/N^{1/d} * F(N,d)) = O(d2^dN^{1/d} * F(N,d))$ ($n \approx N^{1/d}$) time.

Let us look at an algorithm that requires $O(N^{1/d})$ time to compute on a d-dimensional mesh. Thus it can be completed in $O(N^{1/d}d2^dN^{1/d})$ time on a star graph. In this case the optimal dimension for direct simulation can be shown to be ($\approx \sqrt{\log N}$) which gives a total time of $O(\sqrt{\log N}N^{c/\sqrt{\log N}})$ which can be shown to be asymptotically superior to the time required by simulating the sorting algorithm on an n-dimensional uniform mesh on the star graph. The above conversion from $(n-1)$ to d-dimensional mesh is not the best possible. We would like achieve this so that the mesh is as uniform as possible. We leave this as an open problem.

6 Bibliography

[AKER87] S. B. Akers, D. Harel and B. Krishnamurthy, "The star graph: an attractive alternative to the n-cube," *Proceedings of 1987 International Conference on Parallel Processing*, pp. 393-400.

[AKER89] S. B. Akers and B. Krishnamurthy, "A group-network theoretic model for symmetrical interconnection network," *IEEE transaction on Computers*, vol. 38, no. 4, pp. 555-566, Apr. 1989.

[ATAL84] M. J. Atallah and S. R. Kosaraju, "Graph problems on a mesh-connected processor array," *Journal of ACM*, vol. 31, pp. 649-667, July 1984.

[ATAL88] M. J. Atallah, "On multidimensional arrays of processors," *IEEE Transaction on Computers*, vol. 37, no. 10, pp. 1306-1309, Oct. 1988.

[CHAN88] M. Y. Chan and F. Y. L. Chin, "On embedding rectangular grids in hypercubes," *IEEE Transaction on Computers,* vol. 37, no. 10, pp. 1285-1288, Oct. 1988.

[HWAN84] K. Hwang and F. A. Briggs, "Computer architecture and parallel processing," *McGraw Hill, New York,* 1984.

[NASS79] D. Nassimi and S. Sahni, "Biotonic sort on a mesh-connected parallel computer," *IEEE Transaction on Computers*, vol. c-27, pp. 2-7, Jan. 1979.

[NASS80] D. Nassimi and S. Sahni, "Finding connected components and connected ones on a mesh-connected parallel computer," *SIAM Journal of Computing*, vol. 9, no. 4, pp.744-757, Nov. 1980.

[NASS81] D. Nassimi and S. Sahni, "Data broadcasting in SIMD computers," *IEEE Transaction on Computers,* vol. c-30, no. 2, pp. 101-106, Feb. 1981.

[RANK88] S. Ranka, S. Sahni and Y. J. Won, "Programming a hypercube multicomputer," *IEEE Software*, pp. 67-77, Sep. 1988.

[SAAD88] Y. Saad and M. H. Schults, "Topological properties of hypercubes," *IEEE Transaction on Computers,* vol. 37, no. 7, pp. 867-872, July 1988.

[SCHE89] I. D. Scherson, S. Sen and Y. Ma, "Two nearly optimal sorting algorithms for mesh-connected processor array using shear-sort," *Journal of Parallel and Distributed Computer*, vol. 6, pp. 151-165, Apr. 1989.

[THOM77] C. D. Thompsom and H. T. Kung, "Sorting on a mesh-connected parallel computers," *Communication of the ACM*, vol. 20-4, no. 4, pp. 263-271, Apr. 1977.