1

A NEW METRIC FOR PROCESSOR ALLOCATION SCHEMES IN MULTIPROCESSOR SYSTEMS

Sumit Roy and Vipin Chaudhary

Parallel and Distributed Computing Laboratory
Wayne State University
Detroit, MI – 48202
vipin@ece.eng.wayne.edu

ABSTRACT

Recently a number of scalable interconnection networks for connecting multiple processors have been proposed. Though these networks differ in their properties such as bisection bandwidth, node degree, network diameter, and average diameter, there is one common problem that must be addressed by their designers. It should be possible to subdivide the set of processors such that the subset forms a smaller version of the underlying topology. This task is termed as *Processor Allocation* in general and *Subcube* allocation for Hypercubes. Various techniques appear in literature, including tree collapsing, free-list based approaches and others. A common metric for evaluating a scheme is the extent to which it can recognize all the possible sub-graphs in a particular network. In this paper we show that recognizability alone is a poor yardstick for predicting the performance of a processor allocation scheme. We introduce the ideas of *Search Space* and *Impact Sets* and propose a new, more pragmatic metric to measure the efficiencies of an allocation scheme. We evaluate this metric for a simple buddy strategy and a hypothetical exhaustive scheme for two popular interconnection networks, the *Star Graph* and the *k-ary-n-cube.* We support our results by simulation experiments and conclude that a scheme with higher recognizability is unlikely to justify the associated increase in complexity and storage requirements.

I. INTRODUCTION

In recent years, multiprocessor systems have become very popular for solving large scale, computationally intensive problems. A fundamental aspect of these machines is the underlying interconnection network that links the individual processing elements. A group theoretic model has been proposed recently which can be used to model and analyze a certain class of symmetric interconnection networks, known as *Cayley Graphs* [1, 2]. Important members of this class include the Hypercube, the Rotator Graph, and the Star Graph. The Hypercube in particular has been used in many commercial machines like the *INTEL iPSC/1, iPSC/2,* and the *nCUBE.* Hence a great deal of study has been devoted to the properties and capabilities of this topology. Properties which are used to determine the quality of a particular network include the number of connections required at each node, the communication delay between pairs of nodes and the fault tolerance of the network. In graph theoretic terms, these are related to the degree of a node, the diameter of the graph, and the connectivity, respectively. Existence of an efficient message routing algorithm and the scalability of the network are additional features to be considered.

In practice one may want to run programs with varying degrees of parallelism on a multiprocessor machine. However, programs that do not require all the available computing power would waste the excess nodes. One would therefore like to allocate parts of the larger machine to various jobs or programs which can execute concurrently, leading to an improved utilization. These parts are generally constrained to form a smaller version of the original network. A suitable operating system could thus schedule a number of jobs concurrently, making optimum use of the available resources.

After a task is done, it must return the nodes comprising the sub-graph to the operating system, ie. the sub-graph has to be deallocated. The repeated allocation and deallocation of sub-graph could give rise to fragmentation in the network, which means that there are enough nodes in the network to form a sub-graph, but they cannot be linked up due to scattering. Thus the efficiency of a scheme would be greatly reduced if it introduced too much or avoidable fragmentation. Another parameter to be considered is the ability to recognize sub-graphs of various sizes in a network. Depending on the degree of hierarchy of the network, the number of distinct sub-graphs can be very large. However, a particular scheme may recognize only a subset of all existing sub-graphs. So it may be possible for a free sub-graph to exist, which

This research was supported in part by NSF grant MIP-9309489, US Army Contract DAEA 32-93D004 and Wayne State University Faculty research award

an inefficient allocation scheme does not detect. This condition has been termed as a *pseudo fault*. On the other hand an exhaustive scheme which searches all possibilities may require too much time or memory to be of any practical use. To date, the recognizability of a processor allocation scheme was considered a prime criteria for determining its quality. In this paper we introduce a *new* metric to compare the efficiencies of simple allocation schemes with complex and exhaustive schemes. We evaluate this metric for two kinds of networks which have received attention recently, the Star Graph and the k-ary n-cube. Extensive simulations were carried out to validate the metric.

The Star Graph is an interconnection network with many properties superior to the Hypercube, as shown in section 2 of this paper. Though its growth in size is far more rapid than for the Hypercube, due to its strong hierarchical nature, it lends itself to a similar approach to increasing utilization, by allocating sub-stars to the various programs that need to execute on a machine. The k-ary n-cube is another important network that subsumes meshes with wrap-around connections a well as binary hypercubes, ie., k-ary 2-cubes are 2-D meshes with wrap-around and 2-ary n-cubes are binary hypercubes. This network has been used in several concurrent computers, like the Ametek 2020, the experimental J-machine, the Mosaic, the i-Warp and the Cray T3D. Meshes, rings and hypercubes can be embedded in this network [3].

We devised a Simple Buddy scheme for allocation of sub-stars and studied its performance through extensive simulation. We compared our scheme with an Extended Buddy system which is more exhaustive than the Simple Buddy scheme. The simulations show that there is hardly any gain from using a more exhaustive scheme, as predicted by our efficiency metric. Hence we would expect that future research should be addressed towards devising more efficient scheduling strategies as shown by Krueger [4].

The rest of the paper is organized as follows: Section 2 presents some background information about the Star Graph and some useful theorems are derived. Relevant properties of the k-ary n-cube are discussed in section 3. The efficiency metric is introduced in section 4. The simulation results are shown in section 5 and we conclude in section 6.

II. STAR GRAPH

We first define the Star Graph and look at various graph parameters which will be used to establish the efficiency metric.

An interconnection network can be represented by an undirected graph $G = (V, E)$ where the vertex set V corresponds to the processing elements and the edge set E corresponds to bidirectional communication channels.

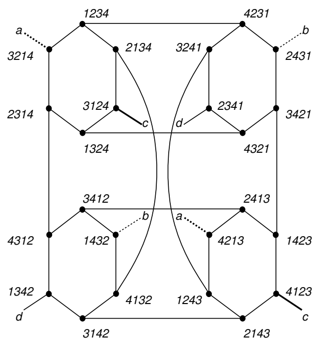

Figure 1: 4-Star

Definition 1 *A Star Graph with $n!$ nodes is called an n-Star and denoted by $S_n = (V_n, E_n)$ where*

$$\begin{aligned} V_n &= \{\star = \star_1 \cdots \star_n \mid \star_i \in \{a_1, \ldots, a_n\}, 1 \leq i \leq n \\ &\quad \wedge \star_i = \star_j \Leftrightarrow i = j\ \} \\ E_n &= \{(\star, \star') \mid \exists!\, i, 2 \leq i, j \leq n and \star_j = \star'_j \forall j \neq i\} \end{aligned}$$

We assume $\{a_1, a_2, \ldots, a_n\} = \{1, 2, \ldots, n\}$ so that V_n becomes the set of $n!$ permutations of the n numbers $1, 2, \ldots, n$ and an edge exists in the Star Graph if two permutations differ only in two positions, the first one and any other. For example in a 4-Star, 4123 can be obtained from 2143 by interchanging the first symbol with the third, so there exists a link between the corresponding nodes. Figure 1 a) shows a 4-Star. The lines indicate links between nodes, and like letters are connected by links which have been omitted for clarity.

The quality of an interconnection network can be gauged by looking at various parameters such as the degree of the network, the diameter, and the average diameter. An n-Hypercube, or n-cube, has 2^n processing nodes, a degree of n, a diameter of n, and an average diameter of $n/2$ [5]. The diameter thus grows as the logarithm of the size of the Hypercube. On the other hand an n-Star Graph connects $n!$ nodes with a degree $n-1$ and a diameter which is at most $3(n-1)/2$. This means that the degree as well as the diameter grow slower than a logarithmic function of the size. Hence the Star Graph offers a network with less interconnection edges and smaller communication delays when compared to a Hypercube of comparable size.

Due to the hierarchical nature of the Star Graph we can define an r sub-star as follows:

Definition 2 *An r-sub-star is a sub-graph of an n-Star and denoted by $R_r(S_r, p_r) = (V_r(S_r, p_r), E_r)$ where*

$$\begin{aligned} S_r &\subset \{a_1, a_2, \ldots, a_n\}, |S_r| = (n - r) \\ p_r &: S_r \mapsto \{2, 3, \ldots, n\} \wedge p_r \textit{ is injective} \\ V_r &= \{\star_1 \cdots \#_i \cdots \star_l \cdots \in V_n \mid \# \in S_r, i = p_r(\#)\} \end{aligned}$$

$$E_r = \{ (\star, \star') \in E_n \mid \star, \star' \in V_r \}$$

Here S_r is a set of $n-r$ symbols and the relation p_r fixes their positions. The first position cannot be fixed due to the definition of a Star Graph. A particular 4-sub-star in an 7-Star would be represented as: $\star_1\ 5\ \star_3\ 3\ \star_5\ \star_6\ 1$ where $S_4 = \{1,3,5\}$ and $p_4(1) = 7, p_4(3) = 4$ and $p_4(5) = 2$.

Remark 1 *Two nodes $\star$ and $\star'$ are said to identical if $\forall\, i\ \ \star_i = \star'_i, 1 \leq i \leq n$.*

Remark 2 *Two r sub-stars R_r and R'_r are said to be identical if $V_r(S_r, p_r) = V'_r(S'_r, p'_r)$.*

Theorem 1 *Two r sub-stars R_r and R'_r are identical if and only if: $S_r = S'_r and p_r = p'_r$*

Proof: Suppose R_r and R'_r are identical, by *Remark 3*
$\Rightarrow V_r(S_r, p_r) = V'_r(S'_r, p'_r)$
assume $S_r \neq S'_r$ (which implies $p_r \neq p'_r$)
Hence $\exists\, \#^{\,k}$ such that $\#^{\,k} \in S_r \wedge \#^{\,k} \notin S'_r$
$\Rightarrow \forall \star_1 \cdots \#_{\,i} \cdots \star_l \cdots \in V_r \quad p_r(\#^{\,k}) =$ constant, say K
$\Rightarrow \forall \star'_1 \cdots \#'_{\,i} \cdots \star'_l \cdots \in V'_r$ the symbol at position K is $\#^{\,k}$, since $V_r(S_r, p_r) = V'_r(S'_r, p'_r)$
Hence there are $|S'_r| + 1 = n - r + 1$ fixed symbols in V'_r, and it cannot be an r sub-star.
Next assume $S_r = S'_r$ but $p_r \neq p'_r$
Hence $\exists\, \#^{\,k}$ such that $p_r(\#^{\,k}) = K$ and $p'_r(\#^{\,k}) = K'$ with $K \neq K'$
$\Rightarrow \forall \star_1 \cdots \#_{\,i} \cdots \star_l \cdots \in V_r \quad p_r(\#^{\,k}) = K$
$\Rightarrow \forall \star'_1 \cdots \#'_{\,i} \cdots \star'_l \cdots \in V'_r$ symbol at position K $= \#^{\,k}$, since $V_r(S_r, p_r) = V'_r(S'_r, p'_r)$
a contradiction since $p'_r(\#^{\,k}) = K'$ with $K \neq K'$. Since the sets S_r, S'_r and the relations p_r, p'_r specify V_r and V'_r completely, the reverse proof is trivial.

From the proof of the theorem we get the following corollary:

Corollary 1 *Two r sub-stars R_r and R'_r are distinct if and only if $p_r \neq p'_r$.*

Proof: From the above theorem for the two r sub-stars to be distinct, either $p_r \neq p'_r$ or $S_r \neq S'_r$. However if $S_r \neq S'_r$, then the mappings cannot be identical, since the pre-images are different. Hence $p_r \neq p'_r$ in both cases.

Theorem 2 *The number of* distinct *r-sub-stars in an n-Star is given by $\frac{n!\,(n-1)!}{r!\,(n-r)!\,(r-1)!}$ [1]*

*Proof:*Given a set of symbols in S_r we can choose the mapping p_r in P^n_{n-r} ways. By *Corollary 1*, each of these is a distinct r sub-star. For the set S_r we have to select $n-r$ symbols out of a possible n. This can be done in C^{n-1}_{n-r} ways. The number of distinct r sub-stars is the product $C^{n-1}_{n-r} \times P^n_{n-r}$

Definition 3 *Two r sub-stars R_r and R'_r are said to be disjoint if $V_r(S_r, p_r) \cap V'_r(S'_r, p'_r) = \phi$.*

Theorem 3 *The number of* disjoint *r-sub-stars in an n-Star is given by $\frac{n!}{r!}$*

Proof: If a symbol is fixed in one position in an n-Star, the resulting set of nodes forms an $(n-1)$ sub-star. The particular symbol can be chosen in n ways, and all the corresponding sets $V_r(S_r, p_r)$ can be seen to be disjoint, since $S_r \cap S'_r = \phi$. Hence there are n disjoint $(n-1)$ sub-stars in a n-Star. Similarly there are $(n-1)$ disjoint $(n-2)$ sub-stars in each $(n-1)$ (sub-)star so formed, $n \times (n-1)$ in total. Continuing thus, we get $n \times (n-1) \ldots \times (r+1)$ disjoint r sub-stars in a n-Star. This number can also be written as $\frac{n!}{r!}$.

Theorem 4 *A necessary and sufficient condition for two r-sub-stars R_r and R'_r in an n-Star to have common nodes is:*
$\forall\, \# \in S_r$ *and* $\#' \in S'_r, p_r(\#) = p'_r(\#') \Leftrightarrow \# = \#'$

Proof: To prove that it is necessary, assume that:
$\exists\, \# \ \in S_r$ and $\#' \in S'_r$ such that $p_r(\#) = p'_r(\#')$ but $\# \neq \#'$
$\Rightarrow \forall \star \in V_r(S_r, p_r)$ the symbol at $p_r(\#)$ is $\#$.
At the same time $\forall\ \star' \in V'_r(S'_r, p'_r)$ the symbol at $p_r(\#)$ is $\#'$.
Since $\# \neq \#'$ it implies that $V_r(S_r, p_r)$ and $V'_r(S'_r, p'_r)$ cannot have any common nodes. Similarly if we have $\# = \#'$ but $p_r(\#) \neq p'_r(\#')$), then this particular symbol appears in two different places for all nodes in $V_r(S_r, p_r)$ and $V'_r(S'_r, p'_r)$. So again there are no common nodes.

To prove that it is sufficient we construct the common nodes. Let $\star \in V_r(S_r, p_r)$ and $\star' \in V'_r(S'_r, p'_r)$ such that
$\forall\, \# = \#', \star_{p_r(\#)} = \# = \#' = \star'_{p'_r(\#')}$
$\forall\, \# \neq \#', \star'_{p_r(\#)} = \#$ and $\star_{p'_r(\#)'} = \#'$
This will use exactly $|S_r \cup S'_r|$ symbols out of a total n. The remaining symbols can then be assigned so that $\star_i = \star'_i$. Nodes constructed thus are identical.

Remark 3 *The above theorem does not assume that $S_r \cap S'_r \neq \phi$, ie., the two sets could be disjoint.*

Corollary 2 *The number of common nodes between two r sub-stars satisfying* **Theorem 4** *is given by:* $(n - |S_r \cup S'_r|)!$

Proof: From the construction of the nodes $|S_r \cup S'_r|$ symbols are fixed, the remaining can be arranged in any permutation.

Theorem 5 *The number of r sub-stars that share at least one common node with a given r sub-star is:* $\sum_i^{(n-r)} C_i^{(n-r)} \times C^{(r-1)}_{(n-r-i)} \times P^r_{(n-r-i)}$ *where* $\max(0, (n-2r+1)) \leq i \leq (n-r)$

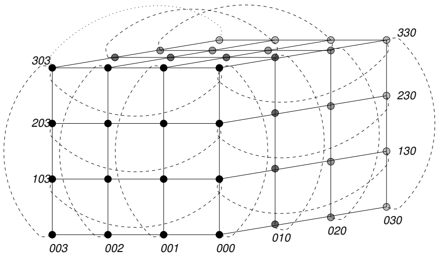

Figure 2: 4-ary 3-cube

Proof: Let R_r be the given r sub-star and R'_r be any other r-sub-star of a n-Star Graph. Assume that R_r and R'_r have i symbols identical which must be at the same position for the two r sub-stars to have at least one node in common, by Theorem 4. Since $|S_r| = (n-r)$ these i symbols can be chosen in $C_i^{(n-r)}$ ways. Also, to create an r sub-star R'_r must have $(n-r-i)$ fixed positions different from R_r and with different symbols, again by Theorem 4. The $(n-r-i)$ positions can be chosen out of $n-1-(n-r) = r-1$ available ones, and the symbols can be any permutation of $(n-r-i)$ symbols out of the $n-(n-r)$ remaining ones. This can hence be done in $C_{(n-r-i)}^{(r-1)} \times P_{(n-r-i)}^{r}$ ways. We require $(n-r-i) \leq (r-1)$ as a constraint on i. Summing over all possible values of i we get the above formula.

III. K-ARY N-CUBE

We can derive similar theorems for the k-ary n-cube. Detailed proofs have been omitted for brevity and follow those presented in the previous section.

Definition 4 *A k-ary-n-cube has k^n nodes and is denoted by $Q_n^k = (V_n, E_n)$ where*

$$\begin{aligned} V_n &= \{\star = \star_1\star_2\star_3\cdots\star_{n-1}\star_n \mid \star_i \in \{a_1, a_2, \ldots, a_n\}, \\ &\quad 1 \leq i \leq n\} \\ E_n &= \{(\star, \star') \mid \exists!\, i, 1 \leq i, j \leq n \text{ such that} \\ &\quad (\forall j \neq i \star_j = \star'_j) \wedge \star_i \equiv_k (\star'_i + 1)\} \end{aligned}$$

If $\{a_1, a_2, \ldots, a_n\} = \{1, 2, \ldots, n\}$ the nodes of the k-ary n-cube can be thought of as n-digit radix k numbers. A link exists between two node if they differ only in one position and that difference is one modulus k. This provides a wrap-around connection at the edges. Figure 2 b) shows a 4-ary 3-cube with internal nodes and some links omitted for clarity.

A k-ary n-cube has a diameter n and each node has $2n$ neighbors, for $k > 2$.

We define a k-ary r-sub-cube formally as follows:

Definition 5 *A k-ary r-sub-cube is a sub-graph of a k-ary n-cube and is denoted by $R_r^k(P_r, s_r) = (V_r(P_r, s_r), E_r)$ where*

$$\begin{aligned} P_r &\subset \{1, 2, \ldots, n\}, |P_r| = (n-r) \\ s_r &: P_r \mapsto \{a_1, a_2, a_3, \ldots, a_n\} \\ V_r &= \{\star_1 \cdots \#_i \cdots \star_l \cdots \in V_n \mid i \in P_r, \# = s_r(i)\} \\ E_r &= \{ (\star, \star') \in E_n \mid \star, \star' \in V_r \} \end{aligned}$$

Thus P_r defines a set of fixed positions and s_r maps any one of k symbols to each of the positions.

Definition 6 *Two k-ary r-sub-cubes are said to be identical if $V_r(P_r, s_r) = V'_r(P'_r, s'_r)$.*

Theorem 6 *Two k-ary r-sub-cubes R_r^k and R'^k_r are identical if and only if:*

$$\begin{aligned} P_r &= P'_r \\ s_r &= s'_r \end{aligned}$$

Proof: The proof follows the lines of proof for theorem 1.

We also have a corollary analogous to corollary 1 for Star graphs.

Corollary 3 *Two k-ary r-sub-cubes R_r^k and R'^k_r are distinct if and only if $s_r \neq s'_r$.*

Theorem 7 *The number of distinct k-ary r-sub-cube in a k-ary n-cube is given by $\frac{n!}{r!\,(n-r)!} k^{(n-r)}$*

Proof: The $n-r$ fixed positions can be chosen out of n possible ones in $C_{(n-r)}^{n}$ ways. Once the positions have been chosen, there are k choices for the symbols to occupy them. This gives a total of:

$$C_{n-r}^{n} \times k^{(n-r)} = \frac{n!}{(n-r)!\, r!} \times k^{(n-r)}$$

Definition 7 *Two k-ary r-sub-cubes R_r^k and R'^k_r are said to be disjoint if $V_r(P_r, s_r) \cap V'_r(P'_r, s'_r) = \phi$.*

Theorem 8 *The number of disjoint k-ary r-sub-cubes in a k-ary n-cube is given by $k^{(n-r)}$*

Proof: By fixing any one position in a k-ary n-cube, one can get k disjoint k-ary (n-1)-sub-cubes. Similarly, each resultant k-ary (n-1)-cube can again be decomposed into k disjoint k-ary (n-2)-cubes, giving a total of $k \times k$ such sub-cubes. Continuing thus we get k^r disjoint k-ary r-sub-cubes.

Theorem 9 *The number of k-ary r-sub-cubes that have at least one node in common with a given k-ary r-sub-cube is given by: $\sum_i^{(n-r)} C_i^{(n-r)} \times C_{(n-r-i)}^{r} \times k^{(n-r-i)}$ where $\max(0, (n-2r)) \leq i \leq (n-r)$*

Proof: The proof goes along the lines of that for theorem 5. If R_r^k is the given r-sub-cube and ${R'}_r^k$ is any other r-sub-cube in the same k-ary n-cube, then they can have i positions identical in $C_i^{(n-r)}$ ways. The remaining $(n-r-i)$ fixed positions in ${R'}_r^k$ must be different from those in R_r^k and hence can be chosen out of r possible ones in $C_{(n-r-i)}^r$ ways. Once the positions are chosen, they can be filled by any of k symbols, in $k^{(n-r-i)}$ ways. The total number of overlapping k-ary r-sub-cubes is obtained by summing over all possible values of i, hence the result.

IV. EFFECTIVENESS OF ALLOCATION SCHEMES

Processor allocation schemes fall into two major types, bottom-up or bit-mapped techniques and top-down or list-based methods.[6, 7]. While the former are usually less complex, they tend to suffer from poorer recognition ability.

The Buddy Strategy in particular has been used for bottom-up allocation for hypercubes [8] and can be readily adapted for other interconnection networks, such as the substar. It has been proven that the buddy strategy is statically optimal under a LIFO release scheme [9]. A general algorithm for a Buddy system is as shown below:

```
function Buddy (r,n)
searches for an r sub-graph in an n graph

1. for 0 ≤ i < size of n-graph
     if node(i) is FREE
       if node(i + 1) through
       node(i + size of r-graph − 1) are all FREE.
         Mark them BUSY and return this list
       else
         continue
       endif
     else
       set i = i + size of r-graph.
     endif
   endfor

2. if no r sub-graph was found, return
   UNSUCCESSFUL.
```

The only requirement for extending this scheme to other graphs is a mapping of the nodes to a linear list, such that the appropriate contiguous set of nodes indeed forms a sub-graph. We were able to adapt the Buddy strategy for Star Graphs by adopting the indexing scheme proposed by Menn and Somani [10].

The above algorithm detects only a particular disjoint set of sub-graphs. We can evaluate the recognition percentage for Star graphs and k-ary n-cubes from Theorems 1 and 2 and Theorems 7 and 8 respectively, refer Tables 1 and 2.

Based on this data, one may expect that the Buddy system should perform very poorly, especially when compared to an exhaustive scheme. We introduce the concepts *Search Space* and *Impact Set* to develop a new metric for predicting efficiencies of allocation schemes.

Sub-star	Recognition (%)			
	4	5	6	7
1	100.00	100.00	100.00	100.00
2	33.33	25.00	20.00	16.67
3	33.33	16.67	10.00	6.67
4	100.00	25.00	10.00	5.00
5		100.00	20.00	6.67
6			100.00	16.67
7				100.00

Table 1: Recognition for the Buddy Scheme in Star Graphs

Sub cube	Host k-ary n-cube	
	k-ary-3	k-ary-4
0	100.00	100.00
1	33.33	25.00
2	33.33	16.67
3		25.00

Table 2: Recognition for the Buddy Scheme in k-ary n-cubes

Definition 8 *The* Search Space *for an r-sub-graph is the number of possible r-sub-graphs in an n-graph, detectable by the processor allocation scheme at that time.*

Definition 9 *The* Impact Set *is the set of recognizable r-sub-graphs lost after allocating r-sub-graphs using a particular processor allocation scheme.*

We evaluate the efficiency (η) of a scheme by comparing the *Search Space* left after allocating one r-sub-graph to the original *Search Space.* Thus our efficiency metric becomes:

$$\eta = \frac{\text{Original Search Space} - \text{Size of Impact Set}}{\text{Original Search Space}}$$

We can now compare the efficiency after allocating one r-sub-graph using the Buddy Scheme and a hypothetical Exhaustive Scheme, as shown in Tables 3, 4 and 5

As can be seen from the tables, the efficiency of the Buddy strategy is consistently higher than that for an exhaustive search scheme. Since the *Search Space* for the exhaustive scheme shrinks very rapidly it is difficult to justify the extra time and memory overhead of more complex allocation strategies, even though the recognition of the Buddy strategy is comparatively poor.

Sub-star	Efficiency (%)					
	5		6		7	
	Bud.	Ex.	Bud.	Ex.	Bud.	Ex.
1	99.17	99.17	99.86	99.86	99.98	99.98
2	98.33	97.08	99.72	99.50	99.96	99.93
3	95.00	84.17	99.17	96.92	99.88	99.52
4	80.00	35.00	96.67	79.67	99.52	95.98
5			83.33	30.00	97.62	74.44
6					85.71	26.19

Table 3: Efficiency of Buddy (Bud.) and Exhaustive (Ex.) Allocation Schemes for Star Graphs

Sub-cube	Efficiency (%)					
	5-ary-3		9-ary-3		17-ary-3	
	Bud.	Ex.	Bud.	Ex.	Bud.	Ex.
0	99.20	99.20	99.86	99.86	99.98	99.98
1	96.00	85.33	98.77	92.18	99.65	95.96
2	80.00	26.67	88.89	29.63	94.12	31.37

Table 4: Efficiency of Buddy (Bud.) and Exhaustive (Ex.) Allocation Schemes for various k-ary 3-cubes

We tested this metric by running extensive simulations using two allocation schemes on various Star Graphs. Further evaluation on k-ary n-cubes are planned.

V. SIMULATION MODEL AND RESULTS

We carried out extensive simulations on various sized Star Graphs to determine whether a more exhaustive scheme would lead to better results than a Buddy strategy. We used an event driven simulator, running on an IBM RS 6000 workstation. A central server accepts requests for sub-stars arriving at some allocation rate. If the request can be satisfied the corresponding nodes are marked as busy. The nodes are released after some residence time. If the allocation strategy is not able to grant the request, it is put in a First Come First Served Queue. All the following requests are also put into this queue. When nodes are deallocated, the server examines the queue, and if it is found non-empty, it tries to allocate the request at the head of the queue. If this request is successful, the next request in the queue is attempted, else the head is replaced. This is continued until either no more requests can be satisfied or the queue becomes empty. While a First Come First Served scheduling strategy avoids starvation of requests as long as queues are bounded, it tends to under utilize the system, since larger requests at the head of the queue would in general block smaller requests.

The residence times and allocation rates were assumed to follow an exponential distribution. The means of these distributions were varied over various simulation runs. Two types of distributions were used for

Sub-cube	Efficiency (%)					
	3-ary-4		5-ary-4		8-ary-4	
	Bud.	Ex.	Bud.	Ex.	Bud.	Ex.
0	98.77	98.77	99.84	99.84	99.98	99.98
1	96.30	90.74	99.20	96.80	98.80	98.78
2	88.89	59.26	96.00	69.33	98.44	74.74
3	66.67	16.67	80.00	20.00	87.50	21.88

Table 5: Efficiency of Buddy (Bud.) and Exhaustive (Ex.) Allocation Schemes for various k-ary 4-cubes

the size of the sub-star requested, uniform and uniformly decreasing. The uniform distribution uses a constant probability $p(r) = 1/(n-1)$ for the request size, while the uniformly decreasing distribution uses probability $p(r) = c/r$ such that $\sum_1^{n-1} c/r = 1$.

The *utilization* of the system can be determined by [4]:

$$\frac{\text{mean request size} \times \text{mean allocation rate} \times \text{mean residence time}}{\text{Total number of nodes in the Star Graph}}$$

We compared a Simple Buddy and an Extended Buddy scheme. The Extended Buddy required n times the storage as compared to the Simple Buddy scheme. Their respective recognization percentages are as given in Table 6.

Sub-star	Recognition (%)					
	5		6		7	
	S.B.	E.B.	S.B.	E.B.	S.B.	E.B.
1	100.00	100.00	100.00	100.00	100.00	100.00
2	25.00	50.00	20.00	40.00	16.67	33.33
3	16.67	50.00	10.00	30.00	6.67	20.00
4	25.00	100.00	10.00	40.00	5.00	20.00
5	100.00	100.00	20.00	100.00	6.67	33.33
6			100.00	100.00	16.67	100.00
7					100.00	100.00

Table 6: Recognition for the Simple Buddy (S.B.) and the Extended Buddy (E.B.) Schemes

Our simulation results confirmed that the usefulness of the Extended Buddy scheme was severely limited, since most of the times, whenever the simple scheme would fail, so would the extended one. Table 7 shows a typical run. A sub-star size of 0 indicates a failure in locating a free sub-star. As can be seen, the total number failures is far larger than the times that the Extended Buddy System was successful and the Simple Buddy failed. The utilization in this case was 80 %. These results agree very well with the efficiency metric shown in Tables 3, 4 and 5.

VI. CONCLUSION

In this paper we introduced a new efficiency metric for processor allocation in multiprocessor computer

Size	S.B. Found	E.B. Found
0	97568	97556
1	19643	19643
2	19708	19711
3	19271	19276
4	19892	19896
5	19529	19529

Table 7: Sub-stars found by Simple (S.B.) and Extended (E.B.) Buddy Schemes

systems. The current metrics like recognizability and average of allocation and deallocation times are not adequate to predict the performance of an allocation scheme. We evaluated our metric for various Star Graphs and k-ary n-cubes. Our simulation results for Star Graphs confirm that a complex scheme with high recognizability, requiring more memory and time to determine, hardly outperforms a simple scheme . The overhead is difficult to justify in a dynamic system. Changes in the scheduling policy of the queue are likely to provide a far larger benefit [4].

VII. REFERENCES

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

[2] S. B. Akers and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection netowrks," *IEEE Transactions on Computers*, vol. 38, pp. 555–566, April 1989.

[3] Y. Bela Bose, B.Broeg and Y.Ashir, "Lee distance and topological properties of k-ary n-cubes," *IEEE Transactions on Computers*, vol. 44, pp. 1021–1030, August 1995.

[4] P. Krueger, T. Lai, and V. Radiya, "Job scheduling is more important than processor allocation for hypercube computers," *IEEE Trans. Parallel and Distrib. Systems.*, vol. 5, pp. 488 – 497, May 1994.

[5] Y. Saad and M. H. Schultz, "Topological properties of hypercubes," *IEEE Transactions on Computers*, vol. C-37, pp. 867–872, July 1988.

[6] S. Rai, J. Trahan, and T. Smailus, "Processor allocation in hypercube computers," *IEEE Transaction on Parallel and Distributed System*, vol. 2, pp. 1220–1224, November 1995.

[7] J. Kim, C. R. Das, and W. Lin, "A top-down processor allocation scheme for hypercube computers," *IEEE Transaction on Parallel and Distributed System*, vol. 2, pp. 20–30, January 1991.

[8] M. S. Chen and K. G. Shin, "Processor allocation in an n-cube multiprocessor using gray codes," *IEEE Transactions on Computers*, vol. C-36, pp. 1396–1407, December 1987.

[9] S. Dutt and J. Hayes, "Subcube allocation in hypercube computers," *IEEE Trans. Comput.*, pp. 341–351, March 1991.

[10] A. Menn and A. K. Somani, "An efficient sorting algorithm for the star graph interconnection network," in *Proceedings of the International Conference on Parallel Processing*, vol. III, pp. 1–8, 1990.