PLACEMENT OF RESOURCES IN THE STAR NETWORK

Ansaf I. Alrabady, Syed M. Mahmud, and Vipin Chaudhary*

Department of Electrical and Computer Engineering
Wayne State University Detroit, MI 48202

ABSTRACT In a large system with many processing elements (PE), it is very expensive to equip each PE with a copy of the resource. It is desirable to distribute a few copies of a given resource to ensure that every PE is able to reach a copy of that resource within a certain number of hops. Previous work has been done on the binary hypercube as well as on the *k*-ary *n*-cube. In this paper we consider the problem of resource sharing among PEs in the star interconnection network (SIN) and present three different placement strategies. First, we will consider the perfect 1-adjacency resource placement. In this placement, resources have to be distributed in such a way that every node without a copy of the resource will find exactly one node adjacent to it having a copy of the resource. Second, the perfect full adjacency placement will be considered. In this placement each node without a copy of the resource will find all nodes adjacent to it having a copy of the resource. Finally, the perfect 2-adjacency placement will be considered where each non resource node is adjacent to exactly two resource copies. We show that a perfect 2-adjacency resource placement does not exist for all star networks.

I INTRODUCTION

The star graph is a Cayley graph which has gained prominence due to its superior properties as compared to the binary hypercube [1-3]. The problem of distributing resources in a multiprocessor system has been studied for several interconnection networks such as the hypercube network [4-8] and the *k*-ary *n*-cube [9]. These resources can be hardware resources such as printers, disk drives, memory units, etc., or they can be software resources such as compilers, data files, library routines, etc. In large systems it is very expensive to provide each node with a copy of the resource. It may also lead to poor resource utilization, since not every node in the system requires a copy of the resource. On the other hand, having few copies of the resources in the system might lead to contention with other access requests to the same copy. It also reduces the reliability of the system because the failure of some of those resource nodes will result in the unavailability of that resource copy to some other non-failure non-resource nodes. This tradeoff among cost, performance, and availability has to be considered when determining the number of resources in the network and their placement.

In this work, we will distribute as few resource copies as possible in the star network which will satisfy the perfect *j*-adjacency placement, where *j* can be equal to one, two, or the degree of the network. Finding a general solution for any value of *j* is a very complicated problem and does not exist for any star network.
The rest of this paper is organized as follows. In section II, necessary background and notations used in the rest of the paper are described. Section III outlines the necessary conditions for the existence of a perfect placement allocation in star graph. In section IV we study the existence of perfect 1-adjacency placements and show that this placement always exists. In section V the perfect full-adjacency is studied. Section VI deals with the perfect 2-adjacency placement. Finally, section VII concludes this paper with directions of future research on this topic.

II PRELIMINARIES

We use the terms PE, node, vertex, and permutation; the terms graph and network; and the terms edge and generator interchangeably throughout this paper. An undirected graph $G=(V,E)$ is a set of vertices V connected by a set of edges $E \subset V \times V$. Two nodes $v,w \in V$ are connected *iff* there is an edge $(v,w) \in E$. In computer terminology the set V corresponds to the PEs of the multiprocessor, while E corresponds to the bi-directional communication links that connect the PEs. The degree of a node is defined as the number of edges incident upon that node. A graph is said to be regular *iff* all nodes have the same degree.

Definition 1: A set of nodes $I \subset V$ is called an independent set *iff* for every two nodes $v,w \in I$, $(v,w) \notin E$. A maximal independent set is one which can't absorb any extra node(s) and keep the set an independent set. The biggest maximal independent set is normally called the maximum independent set.

Definition 2: A set of nodes $C \subset V$ is called a dominating set *iff* for every node $v \in (V-C)$, there exist a node $w \in C$ such that $(v,w) \in E$.

Definition 3: A graph G is a bipartite graph *iff* the vertices of G can be partitioned into two sets, V_1 and V_2, such that for every $(v,w) \in E$, if $v \in V_1$ then $w \in V_2$, or vice versa. In other words every edge joins a vertex of V_1 to a vertex of V_2.

Definition 4: A cycle is a sequence of distinct nodes in V, such that the first node in the sequence is also the last one, and any two consecutive nodes in the sequence are directly connected by an edge. The length of the cycle is the number of nodes in that cycle.
Let $\zeta=\{x_1,x_2, .., x_n\}$ be the set of n different symbols. A star graph is an undirected regular graph with degree equal to $(n-1)$. It is normally called by the number of n different symbols used to label the $n!$ different nodes. So an n-star has $n!$ nodes labeled by the $n!$

* This author's research is supported in part by NSF grant MIP-9309489, U.S. Army contract DAEA32-93-D-0004, and Wayne State University Faculty Research Award..

0-7803-3529-5/96/$5.00 © 1996 IEEE

different permutations ~~using~~ the n different symbols in ζ. Two nodes $v, w \in V$ are adjacent in the star network *iff* the permutation of v and w differ in the first symbol as well as in one more symbol. The set of edges is partitioned into (n-1) classes denote~~d by~~ g_i, $2 \leq i \leq n$. An edge is labeled by a generator g_i which exchanges the first and the ith symbol in the permutations labeling the nodes it connects. Figures 1a, 1b, and 1c show the 2-star, 3-star, and 4-star, respectively.

Similar to the hypercube the star network possess a hierarchical structure. The n-star consists of smaller p-stars. It has been shown [2] that there are as many as $\binom{n-1}{n-p} \cdot \frac{n!}{p!}$ distinct p-stars, out of which $\frac{n!}{p!}$ are disjoint p-stars.

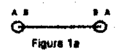
Figure 1a

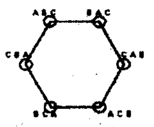
Figure 1b

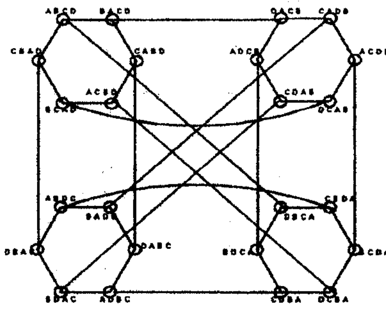
Figure 1c

Definition 5: The subset $Vr_k \subset V$ is defined as the set of permutations where the symbol $r \in \zeta$, is the same in the kth position, where $1 \leq k \leq n$. (e.g. in 4-star, let $\zeta = \{1,2,3,4\}$ then $V2_1 = \{2134, 2143, 2314, 2341, 2413, 2431\}$). It is clear that this set contains (n-1)! different permutations.

III PERFECT *j*-ADJACENCY PLACEMENT

In this section we will develop the necessary conditions for the existence of perfect j-adjacency placement. Throughout this paper we will refer to a node which has a copy of the resource as a resource node, and the set which contains all resource nodes as the resource set. A node without a resource copy is referred to as a non-resource node.

Definition 6: We define an allocation strategy to be perfect *iff* no two resource nodes are adjacent to each other, and all non-resource nodes are adjacent to the same number of resource nodes.

Definition 7: An allocation strategy is called a perfect j-adjacency *iff* it is perfect, and each non-resource node is adjacent to exactly j resource nodes.

Lemma 1: Any perfect resource set is a dominating set and a maximal independent set.

Proof: Since each node outside the resource set is a non-resource node which is adjacent to at least one resource node, this means that the resource set is a dominating set. From the definition of perfect placement, there are no two resource nodes adjacent to each other, hence the resource set is an independent set. Adding any non-resource node to the resource set will make the set a dependent set. Thus, the resource set is a maximal independent set. □

The importance of the above theorem lies in the fact that in looking for a perfect resource set in any interconnection network, we have to look for a dominating and a maximal independent set in that network. Such a problem is known so far to be an NP complete problem [5,16].

Let $R(n,j)$ be the minimum number of resource copies required to achieve the perfect j-adjacency placement in an n-star. Then $R(n,j)$ is given by the following lemma.

Lemma 2: The minimum number of resource copies required to achieve a j-adjacency placement in an n-star is given by $R(n,j) = \frac{j*n!}{n-1+j}$. For perfect j-adjacency (if it exists) $R(n,j)$ is an integer number. Where obvious, we use R instead of $R(n,j)$.

Proof: If R resources are required then there are R nodes out of the $n!$ nodes in the star network which have a resource, and ($n!$-R) nodes are non-resource nodes. All resource nodes can be adjacent to exactly (n-1)R non-resource nodes. This is equal to the number of resources required by the non-resource nodes, which is equal to $j(n!-R)$. By equating these two expressions and solving for R we have $(n-1)R = j(n!-R)$. Thus, $R(n,j) = \frac{j*n!}{n-1+j}$. Note that for a perfect j-adjacency the number of resources R must be integral. This is a necessary condition for perfect j-adjacency to exist in the star network. □

From the above formula we can see that it is not always possible to achieve a perfect j-adjacency for any arbitrary values of n and j. However, this condition is not sufficient to guarantee that a perfect j-adjacency exists in the star network.

Lemma 3: For any value of n and j there are no two non-resource nodes connected to more than one common resource copy.

Proof: Let us assume that there are two non-resource nodes connected to the same two or more resource copies. This means that there should exist a cycle of length four. However, the smallest cycle that can be embedded onto the star network is of length six [15]. This means that no two nodes can be connected to more than one common node. □

We will limit our approach for constructing the j-adjacency placement to three different cases. In the next three sections we study the existence of j-adjacency placements for various values of j, namely, 1, n-1, and 2.

IV PERFECT 1-ADJACENCY PLACEMENT

In this section we study the existence of perfect 1-adjacency placement and show that these placement always exists for all star networks.

Lemma 4: A perfect 1-adjacency placement exists in any star.

Proof: *(Necessity)* From lemma 2, substituting j by 1, we get $R(n,1)$=(n-1)! which is integral. This expression shows that there are (n-1)! nodes, equipped with a copy of the resource in order for each node in the star to have a resource or to be adjacent to a resource copy.

(Sufficiency) We can choose the subset Vr_1 to be the resource set and all other nodes in (V-Vr_1) to be the non-resource nodes. Note that Vr_1 contains exactly (n-1)! nodes.

In order to prove that Vr_1 is the resource set, we have to prove that

1. Vr_1 is an independent set, and

2. For every node $v \in (V\text{-}Vr_1)$ there exists an edge $e \in E$, such that e is the only edge which connects the node v to one and only one node in Vr_1. In other words each non-resource node is adjacent to exactly one resource node.

To prove the first assertion, let $w=rx_2x_3..x_{i-1}x_ix_{i+1}..x_n \in Vr_1$. Applying any generator g_i will generate a new permutation $v=x_ix_2x_3..x_{i-1}rx_{i+1}..x_n$. It is clear that $v \notin Vr_1$ since $x_i \neq r$.

To prove the second assertion let us assume that $v \in (V\text{-}Vr_1)$. Since $v \notin Vr_1$, the symbol r should not appear in the first position. Let $v= x_1x_2..x_{i-1}r..x_n$. Then there exists only one generator g_i which will take the index r (in the ith position) to the first position. This will result in a new node $w=rx_2..x_{i-1}x_1..x_n \in r_1$. □

Since r can be any symbol from ζ, this means that there are n different independent sets and we can chose any one of these sets to be the resource set.

Figure 2 shows the 1-adjacency placement for the 2-star, 3-star, and 4-star where V_{A_1} is chosen to be the resource set. In the figure, the nodes with the filled circle are the resource nodes.

Figure 2a

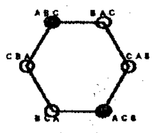
Figure 2b

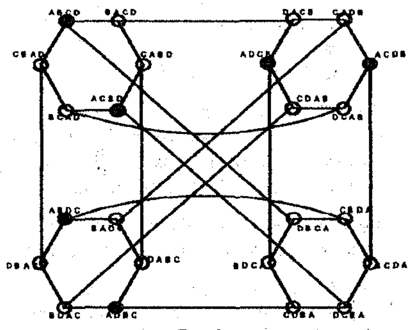
Figure 2c

V PERFECT FULL-ADJACENCY PLACEMENT

A perfect full adjacency resource placement is an allocation strategy where each node that does not have a copy of the resource will find all nodes adjacent to it having a copy of the resource. Since the star network is a regular network with degree (n-1), this means that j is equal to (n-1).

Lemma 5: A perfect full-adjacency resource placement exists in any star network.

Proof: *(Necessity)* From lemma 2, it is clear that $R(n,n\text{-}1)$ is integral.

(Sufficiency) We can prove that an allocation always exists for any star by noting that the star network is a bipartite graph [11]. If a graph is bipartite, then we can divide the set of nodes V into two independent sets, T and its neighbors $N(T)=V\text{-}T$, such that for every node $a \in T$ the neighbors of a, $N(a) \subset N(T)$, and for every node $b \in N(T)$, $N(b) \subset T$. Then we can choose one of these two sets to be the resource set. Algorithm "partition into two independent sets" (PTIS) outlines a scheme to partition V into two independent sets, namely T and $N(T)$. □

ALGORITHM PTIS

```
Begin
Input: The set of nodes V.
1.  Initially, T=N(T)=∅. Chose a node from V and put that node
    in T.
2.  Find all neighbors for each node in T from V-N(T). Put them
    in N(T).
3.  Find all neighbors for each node in N(T) from V-T. Put them
    in T.
4.  If V≠T∪N(T) then go to 2.
Output: Two Independent sets T, and N(T).
End.
```

Algorithm PTIS is a generic algorithm to divide any bipartite graph into two independent sets. It is always possible to find a perfect full-adjacency resource allocation for any bipartite interconnection network. Note that if the graph is connected and bipartite then all cycles in the graph have even length. In full-adjacency the neighbors of each resource node are non-resource nodes and vice versa. This means that along any path in the network we will have an alternate resource and non-resource nodes. An easy algorithm to find the nodes where a copy of the resource should be placed is to place the first copy of the resource in an arbitrary node v and then placing a copy of the resource in every node that is at even distance from v. Figure 3 shows the full-adjacency placement for the 2-star, 3-star, and 4-star.

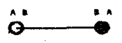
Figure 3a

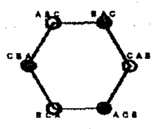
Figure 3b

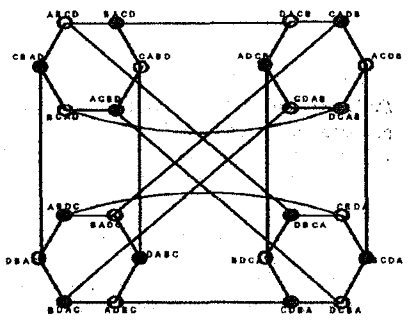
Figure 3c

Corollary 1: The number of resource copies required to achieve a perfect full-adjacency in any regular interconnection network which is bipartite is equal to half the total number of nodes.

Proof: If the graph is a bipartite graph, then we can divide the nodes into two independent sets. Since the network is regular then the number of nodes in one set is equal to the numbers of nodes in the other set, which is equal to half the total number of nodes in the network. □

VI PERFECT 2-ADJACENCY PLACEMENT

This placement is more complicated than the previous two cases. In this section we will see that a perfect 2-adjacency does not exist for every star (e.g. 4-star where R(4,2) is not an integral). However, we will see that this placement always exists for an n-star if n is an odd number.

Let us start by the set of n different symbols $\zeta=\{x_1,x_2, ..., x_n\}$, where n is an odd number. We want to divide ζ into two disjoint sets α, γ such that α contains $(n\text{-}1)/2$ symbols $(\alpha \subset \zeta)$, and γ contains the other $(n+1)/2$ symbols ($\gamma=\zeta-\alpha$). There is m different ways to partition ζ into α and γ, where $m=\binom{n}{\frac{n-1}{2}}$. Let us distinguish between these different partitions by using a different subscript that accompany each partition (i.e., $\alpha_1, \alpha_2, \ldots, \alpha_m; \gamma_1, \gamma_2, \ldots, \gamma_m$} where $\zeta=\alpha_i\cup\gamma_i$, for i =1,2,..,$m$, and $\alpha_i\neq\alpha_j$, $\gamma_i\neq\gamma_j$, for i≠j. For each γ_i we want to divide this set into two disjoint sets, δ_i which contains only one symbol, and β_i which contains the remaining $(n-1)/2$ symbols (i.e., $\gamma_i=\delta_i\cup\beta_i$, for i=1,2,..,$m$). The question now is whether it is possible to partition each γ_i into δ_i and β_i such that $\beta_i\neq\beta_j$, for i≠j? The following lemma states that such a partition is possible.

Lemma 6: Let $\zeta=\{x_1,x_2, \ldots, x_n\}$ be the set of n different symbols, where n is an odd number. Let each γ_i, i=1,2,..m, represent one of the m different combinations of $(n+1)/2$ symbols from ζ. Then we can construct a new set β_i of $(n-1)/2$ symbols from γ_i such that $\beta_i\subset\gamma_i$ and $\beta_i\neq\beta_j$, for i≠j, $1\leq i,j\leq m$.

Proof: See appendix A for a proof of this lemma.□

The next problem is to choose the symbols in β_i from γ_i under the condition that $\beta_i\neq\beta_j$, for i≠j? Algorithm "partition into disjoint sets" (PIDS) outlines a scheme that generates all different partitioning of ζ into disjoint sets δ_i, α_i, and β_i, where $\alpha_i\neq\alpha_j$, and $\beta_i\neq\beta_j$, for i=1,2..,m. Note that $\zeta=\delta_i\cup\alpha_i\cup\beta_i$, for i=1,2..,$m$.

ALGORITHM PIDS

```
Begin
1.   Let $\zeta=\{x_1,x_2, \ldots, x_n\}$. Let $\alpha_i\subset\zeta$ $(1\leq i\leq m)$ be the $m$ different
     sets, each containing $(n-1)/2$ different symbols, such that
     $\alpha_i\neq\alpha_j$ for i≠j.
2.   For i =1 to $m$ do
         $\gamma_i=\zeta-\alpha_i$, $\beta_i=\varnothing$, and $\delta_i=\varnothing$.
3.   Pick $(n-1)/2$ symbols from the set $\gamma_i$, and put them in $\beta_i$.
     Build $\delta_1$ as $\delta_1=\gamma_1-\beta_1$. Note that there is only one symbol in
     the set $\delta_1$
4.   i=1
5.   For j=1 to $m$ do
         if j≠i and $\beta_i\subset\gamma_j$ then $\beta_j=\beta_j\cup(\gamma_j-\beta_i)$.
6.   Select a new value for i in such a way that $\delta_i=\varnothing$ and for all
     j≠i, $1\leq j\leq m$, and the number of symbols in $\beta_i$ is greater than
     or equal to that in $\beta_j$.
7.   If $\beta_i$ does not have $(n-1)/2$ symbols then add more symbols
     from $\gamma_i$ to $\beta_i$ until $\beta_i$ has (n-1)/2 symbols. Then build $\delta_i$ as $\delta_i$
     $=\gamma_i-\beta_i$. If any $\delta_i$ set is still empty then go to 5.
End.
```

Example: To illustrate the above algorithm let's assume that $\zeta=\{A,B,C,D,E\}$. Then we can build the sets α_i $(1\leq i\leq m)$ as $\alpha_1=\{A,B\}$, $\alpha_2=\{A,C\}$, $\alpha_3=\{A,D\}$, $\alpha_4=\{A,E\}$, $\alpha_5=\{B,C\}$, $\alpha_6=\{B,D\}$, $\alpha_7=\{B,E\}$, $\alpha_8=\{C,D\}$, $\alpha_9=\{C,E\}$, $\alpha_{10}=\{D,E\}$. Now the above algorithm can be used to determine the sets β_i, and δ_i $(1\leq i\leq m)$.

From step-2 of the algorithm we get $\gamma_1=\{A,C,D\}$, $\gamma_2=\{C,D,E\}$, $\gamma_3=\{B,D,E\}$, $\gamma_4=\{B,C,E\}$, $\gamma_5=\{A,D,E\}$, $\gamma_6=\{A,C,E\}$, $\gamma_7=\{A,C,D\}$, $\gamma_8=\{A,B,E\}$, $\gamma_1=\{A,B,D\}$, $\gamma_1=\{A,B,C\}$, $\beta_1=\varnothing$, $\beta_2=\varnothing$, $\beta_3=\varnothing$, $\beta_4=\varnothing$, $\beta_5=\varnothing$, $\beta_6=\varnothing$, $\beta_7=\varnothing$, $\beta_8=\varnothing$, $\beta_9=\varnothing$, $\beta_{10}=\varnothing$, $\delta_1=\varnothing$, $\delta_2=\varnothing$, $\delta_3=\varnothing$, $\delta_4=\varnothing$, $\delta_5=\varnothing$, $\delta_6=\varnothing$, $\delta_7=\varnothing$, $\delta_8=\varnothing$, $\delta_9=\varnothing$, $\delta_{10}=\varnothing$.

From step-3 we get $\beta_1=\{C,D\}$, and $\delta_1=\{E\}$; Using step-4 and 5 we have $\beta_4=\{B\}$, and $\beta_7=\{A\}$. Using step-6 we find that value of i is 4. Now using step-7 we can build $\beta_4=\{B,C\}$, and $\delta_4=\{D\}$. Since there are many empty δ's we must go back to step-5 with i=4.

Now using step-5 again we get $\beta_3=\{E\}$ and $\beta_{10}=\{A\}$. Using step-6 we find that the value of i is equal to 3. From step-7 we can build $\beta_3=\{B,E\}$ and $\delta_3=\{C\}$. At this point the values of different β and δ sets are $\beta_1=\{C,D\}$, $\beta_2=\varnothing$, $\beta_3=\{B,E\}$, $\beta_4=\{B,C\}$, $\beta_5=\varnothing$, $\beta_6=\varnothing$, $\beta_7=\{A\}$, $\beta_8=\varnothing$, $\beta_9=\varnothing$, $\beta_{10}=\{A\}$, $\delta_1=\{E\}$, $\delta_2=\varnothing$, $\delta_3=\{C\}$, $\delta_4=\{D\}$, $\delta_5=\varnothing$, $\delta_6=\varnothing$, $\delta_7=\varnothing$, $\delta_8=\varnothing$, $\delta_9=\varnothing$, $\delta_{10}=\varnothing$. Since all the deltas except δ_1,δ_3, and δ_4 are empty we must go to step-5 with i=3.

Now using step-5 again we get $\beta_2=\{D\}$ and $\beta_8=\{A\}$. Using step-6 we find that the value of i is equal to 2. From step-7 we can make $\beta_2=\{B,D\}$ and $\delta_2=\{E\}$. At this point the values of different β and δ sets are $\beta_1=\{C,D\}$, $\beta_2=\{B,D\}$, $\beta_3=\{B,E\}$, $\beta_4=\{B,C\}$, $\beta_5=\varnothing$, $\beta_6=\varnothing$, $\beta_7=\{A\}$, $\beta_8=\{A\}$, $\beta_9=\varnothing$, $\beta_{10}=\{A\}$, $\delta_1=\{E\}$, $\delta_2=\{E\}$, $\delta_3=\{C\}$, $\delta_4=\{D\}$, $\delta_5=\varnothing$, $\delta_6=\varnothing$, $\delta_7=\varnothing$, $\delta_8=\varnothing$, $\delta_9=\varnothing$, $\delta_{10}=\varnothing$. Since there are more empty deltas, then we have to go to step-5 with i=2.

If we continue the above process of building the different sets, then after few iterations we will get the following β and δ sets: $\beta_1=\{C,D\}$, $\beta_2=\{B,D\}$, $\beta_3=\{B,E\}$, $\beta_4=\{B,C\}$, $\beta_5=\{D,E\}$, $\beta_6=\{C,E\}$, $\beta_7=\{A,C\}$, $\beta_8=\{A,E\}$, $\beta_9=\{A,D\}$, $\beta_{10}=\{A,B\}$, $\delta_1=\{E\}$, $\delta_2=\{E\}$, $\delta_3=\{C\}$, $\delta_4=\{D\}$, $\delta_5=\{A\}$, $\delta_6=\{A\}$, $\delta_7=\{D\}$, $\delta_8=\{B\}$, $\delta_9=\{B\}$, $\delta_{10}=\{C\}$.

Let us represent each partition of ζ into its disjoint sets by a 3-tuple $\pi_i=(\delta_i, \alpha_i, \beta_i)$. Let $\Pi=\{\pi_1, \pi_2,.., \pi_m\}$ be the set which contains all different 3-tuples. From the previous example, Π={(\{E\}, \{A,B\}, \{C,D\}), (\{E\}, \{A,C\}, \{B,D\}), (\{C\}, \{A,D\},\{B,E\}), (\{D\}, \{A,E\}, \{B,C\}), (\{A\}, \{B,C\}, \{D,E\}), (\{A\}, \{B,D\}, \{C,E\}), (\{D\}, \{B,E\}, \{A,C\}), (\{B\}, \{C,D\}, \{A,E\}), (\{B\}, \{C,E\}, \{A,D\}), (\{C\}, \{D,E\}, \{A,B\})}.

For each 3-tuple we want to generate the permutations $P=x_1x_2\ldots x_{(n+1)/2}x_{(n+3)/2}\ldots x_n$, which satisfy the following three conditions :

1. $x_i\in\delta$ if $i=1$;

2. $x_i\in\alpha$ if $2\leq i\leq(n+1)/2$;

3. $x_i\in\beta$ if $(n+3)/2\leq i\leq n$;

For example, from the 3-tuple ({E},{A,B},{C,D}) we can generate the following four permutations {EABCD, EABDC, EBACD, EBADC}.

For each 3-tuple $\pi\in\Pi$ there exist k different permutations in V that satisfy the above three conditions, where $k=1!*\left(\frac{n-1}{2}\right)!*\left(\frac{n-1}{2}\right)!=\left(\left(\frac{n-1}{2}\right)!\right)^2$. Let $\Omega\subset V$ be the set which contains all the different permutation that are generated from every $\pi\in\Pi$. Since we have m different 3-tuples in Π, the total number of different permutations in Ω are equal to $r=k*m=\frac{2*n!}{n+1}$. From the previous example, Π contains ten 3-tuples and from each 3-tuple we can generate 4 different permutations, a total of 40 permutations. By using the 3-tuples in Π as in the previous example, Ω contains the following permutations, Ω={EABCD, EABDC, EBACD, EBADC, EACBD, EACDB, ECABD, ECADB, CADBE, CADEB, CDABE, CDAEB, DAEBC, DAECB, DEABC, DEACB, ABCDE, ABCED, ACBDE, ACBED, ABDCE, ABDEC, ADBCE, ADBEC, DBEAC, DBECA, DEBAC, DEBCA, BCDAE, BCDEA, BDCAE, BDCEA, BCEAD, BCEDA, BECAD, BECDA, CDEAB, CDEBA, CEDAB, CEDBA}

Lemma 7: A perfect 2-adjacency resource allocation exists in an n-star network if n is an odd number

Proof: *(Necessity)* From lemma 2, $R(n,2) = \frac{2*n!}{n+2-1} = \frac{n!}{\frac{n+1}{2}}$. Since n is an odd number then $\frac{n+1}{2}$ is integral and it is less than n for $n>1$. This means that there is a factor in $n!$ which is equal to $\frac{n+1}{2}$ which will cancel with the denominator. Hence, $R(n,2)$ is integral.

(Sufficiency) By choosing Ω to be the resource set, we need to prove that

1. Ω is an independent set.

2. Each non-resource node is exactly adjacent to two resource nodes.

The first statement can be proved by noting that any two permutations P and Q in Ω are generated either from the same 3-tuple, or from two different 3-tuples in Π. In the first case, the two permutations have the same symbol in the first position since they have the same δ, hence they are not adjacent. In the second case, P and Q are generated from two different 3-tuples, namely π_P and π_Q. In this case $\alpha_P \neq \alpha_Q$ and $\beta_P \neq \beta_Q$ which means that there are at least two symbols in P different from those in Q in certain positions between 2 to n. For the two permutations to be adjacent they should differ in one symbol only from position 2 to n. Hence these two permutations are not adjacent. Thus, Ω is an independent set.

To prove the second statement, let us assume that $P \notin \Omega$ is a non-resource node. This means that P is generated from a 3-tuple $\pi_P \notin \Pi$. However, there are two 3-tuples π_Q, $\pi_R \in \Pi$, such that $\alpha_P = \alpha_Q$ but $\beta_P \neq \beta_Q$; and $\beta_P = \beta_R$ but $\alpha_P \neq \alpha_R$. Note that these conditions imply that $\delta_P \neq \delta_Q$ and $\delta_P \neq \delta_R$. In the first case where $\alpha_P = \alpha_Q$ but $\beta_P \neq \beta_Q$, there exists one and only one permutation Q that is generated from π_Q which is different from P in only one symbol other than the one in the first position. This symbol is in a position between $(n+3)/2$ to n. Hence P is adjacent to Q. In the second case where $\beta_P = \beta_R$ but $\alpha_P \neq \alpha_R$ there exists one and one permutation R that is generated from π_R which is different from P in one symbol other than the one in the first position. This symbol is in a single position between 2 to $(n+1)/2$. Hence P is adjacent to R. This implies that P is adjacent to two and only two nodes labeled by the permutations Q and R. □

The 5-star is very complicated to draw, so we represent the 5-star by the table in figure 4, where the columns correspond to one independent set and the rows correspond to the other independent set. Such a representation is possible since the star graph is a bipartite graph. The number in the entry (i,j) in the table represents the generator that links node v_i to node v_j. If there is no number then there is no link between the two nodes. Note that each row, or column has four different numbers in four different entries. The resource nodes are represented by bold letters which occupy the last twenty columns and the last twenty rows.

It should be clear at this point that for any non-resource node, there are two generators g_i and g_j, where $2 \le i \le (n+1)/2$, $(n+3)/2 \le j \le n$, each corresponding to a different communication link which connects that non-resource node to a different resource node.

The previous statement have several useful information which can be summarized in the following points.

1. We can group the generators into two groups: Group A which have the generators that switch the first symbol in the permutation with another symbol in a position between 2 to $(n+1)/2$; and group B which have the generators that switch the first symbol in the permutation with another symbol in a position between $(n+3)/2$ to n.

2. There is no non-resource node which is connected to two different resource nodes by links labeled by generators that belong to the same group defined in 1.

3. There exist $n!/\left(\frac{n+1}{2}\right)!$ disjoint $\frac{n+1}{2}$-stars where each star satisfies the perfect 1-adjacency placement considered earlier. One possible way to see this is by fixing the symbols in positions from $(n+3)/2$ to n and permuting the symbols in positions 1 to $(n+1)/2$ to obtain the $\frac{n+1}{2}$-star. Remember that there are $\frac{n!}{p!}$ disjoint p-stars. Note that for any non-resource node in this $\frac{n+1}{2}$-star there is one and only one generator which belongs to group A that connects a non-resource node to a resource node in this substar. Another possible way is by fixing the symbols in positions from 2 to $(n+1)/2$ and permuting the symbols $(n+3)/2$ to n in addition to the first symbol. In this case the generators belong to group B.

4. It is not necessary that group A has the first 2 to $(n+1)/2$ generators and group B has the other $(n+3)/2$ to n generators. We can generalize this by choosing any of the $(n-1)/2$ generators to belong to group A and the other $(n-1)/2$ generators to belong to group B. In this case, when we build the Ω set, for each 3-tuple $(\delta,\alpha,\beta) \in \Pi$, α should be permuted in positions corresponding to the generators in group A, and β should be permuted in positions corresponding to the generators in group B. It is clear that we have $\binom{n-1}{\frac{n-1}{2}}$ different possibilities to chose group A (group B is then implied).

VII CONCLUSION

In this paper we have investigated the distribution of resources in the star network. First, we found necessary conditions for the perfect j-adjacency to exist. A perfect 1-adjacency and full-adjacency placement solution was found and we show that these placement always exist for any star. We find that a perfect 2-adjacency placement exists in an n-star if n is an odd number, and we give an algorithm to find the resource nodes. The problem of finding a general solution for perfect j-adjacency for any value of j is still an open problem. Also, finding a general solution for resource placement which distribute as few resource copies as possible in the star network such that each node will reach a given number of resources within a certain number of hops is still an open problem.

	ABECD	ACDBE	ACEDB	ADCEB	ADEBC	AEBDC	AECBD	AEDCB	BACED	BADCE	BAEDC	BCADE	BDAEC	BDECA	BEACD	BEDAC	CABDE	CAEBD	CBAED	CBDAE	CBEDA	CDBEA	CEADB	CEBAD	DABEC	DACBE	DBACE	DBCEA	DCAEB	DCBAE	DCEBA	DECAB	EADBC	EBCAD	EBDCA	ECBDA	ECDAB	EDACB	EDBAC	EDCBA	ABCDE	ABDEC	ACBED	ADBCE	BCDEA	BCEAD	BDCAE	BECDA	CADEB	CDABE	CDEAB	CEDBA	DAECB	DBEAC	DEABC	DEBCA	EABCD	EACDB	EBADC	ECABD
ABEDC											2										5																																4					3	
ACDEB																													3								4								5				2												
ACEBD																			2														5														4													3
ADCBE																										2														5							4			3										
ADECB															5																													3														4		2							
AEBCD															3											4																																				5	2			
AECDB																								3									4																5										2		
AEDBC																	4																								2																				5			3					
BACDE																	3										4																							2																5		
BADEC																									3							4										2							5												
BAECD	2																		4																																						5				3			
BCAED																				2										5																3																	4
BDACE																												2											5						3						4										
BDEAC					4																																						3																5			2						
BEADC						3																			5																																		4				2	
BEDCA								5																																	2																		4				3				
CABED									3																						5																								2													4			
CAEDB			2								5																																																						4					3		
CBADE													2																	4																3																		5	
CBDEA																												3							4									5			2														
CBEAD	4																																					3													2									5						
CDBAE																														2											5						4			3														
CEABD							3														4																																												5					2
CEBDA						5																																					2																	3											4				
DABCE										3											4																															2															5			
DACEB				2					5																																											3											4		
DBAEC													2						5																								3																			4	
DBCAE																				3														5							4						2													
DCABE		3												4																																												2										5
DCBEA																							2										4									5		3															
DCEAB			4																																								3												5						2									
DECBA							5																																						2												4						3									
EADCB								2		5																																						4						3							
EBCDA																					3							4												5							2													
EBDAC																	2				5																					4													3						
ECBAD																									2						5												4			3																
ECDBA		5																														3														4									2								
EDABC					3									4																																																5					2					
EDBCA															3									4																												5											2				
EDCAB				4																															2															5				3									
ABCED									2											3											5										4																											
ABDCE										2											4										3										5																									
ACBDE													3					2														4														5																							
ADBEC														3															5			2																	4																					
BCDAE		4																				2												3											5																							
BCEDA			5																					2										4					3																											
BDCEA				5																						3						2											4																				
BECAD							4																							3								5		2																										
CADBE		2								4																			3								5																											
CDAEB				3											5																		2															4																						
CDEBA					5																4																							2									3																					
CEDAB								4															5																							3					2																						
DAEBC					2						4							5																								3																											
DBECA	5														2							4															3																					
DEACB								3													5												4																		2																					
DEBAC						4													3																5																					2																					
EABDC						2						3						5											4																																						
EACBD							2		4													3												5																																							
EBACD	3																2				4										5																																										
ECADB			3													5														2						4																																

Figure 4

APPENDIX A

Theorem of Matching [12-14]

If $G=(V,E)$ is a bipartite graph with parts $V_1,V_2 \subset V$, then there exists a complete matching from V_1 to V_2, *iff* $|A| \leq |N(A)|$ for all subsets A of V_1, where $N(A)$ denotes the set of vertices in V_2 that are adjacent to vertices in A

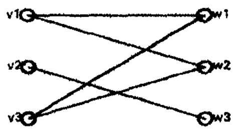

Figure 5a

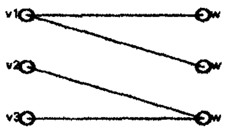

Figure 5b

Think of this problem as a set of men represented by V_1 and a set of women represented by V_2, with a set of acceptable pairs E, such that an edge from v in V_1 to w in V_2 means that the man v knows the woman w. The problem now is to determine if it is possible to match each man with a woman he knows such that every man is happy and no man is married twice. A necessary and sufficient condition for the matching problem to have a solution is that for every set of k men must, together, know at least k women, for every k, $1 \leq k \leq n$. For example, there is a complete matching from V_1 to V_2 in figure 5a, but not in figure 5b, since $|\{v_2,v_3\}|=2 \geq |N(\{v_2,v_3\})|=|\{w_3\}|=1$.

We want to extend this theorem so that every woman is happy too. A necessary and sufficient condition to match each man with a woman so that every one is happy, and no man or woman is married twice is that there should exist a complete matching from V_1 to V_2 and from V_2 to V_1.

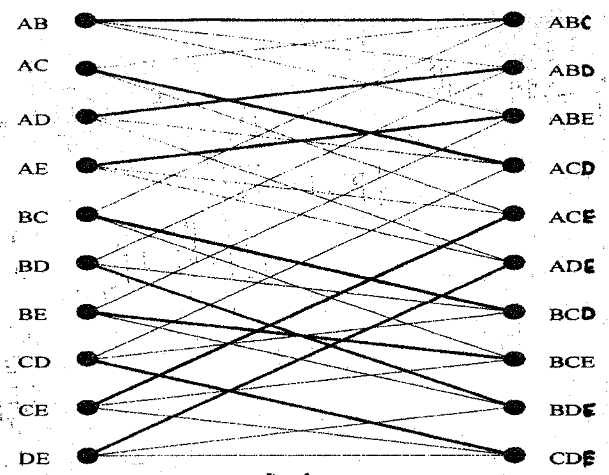

Figure 6

Theorem 2: If G is a regular bipartite graph with parts V_1 and V_2, then there exists a complete matching from V_1 to V_2 and from V_2 to V_1.

Proof: Since G is regular with degree m, then for each subset A of V_1, there are $|A|m$ edges incident with the nodes in A. Since the nodes in V_1 have the same degree as those in V_2 which is equal to m, then those edges from A must be incident with $|A|$ or more nodes in V_1. Thus, there is a complete matching from V_1 to V_2. With similar argument we can prove that there is a complete matching from V_2 to V_1, hence the theorem. □

Proof of lemma 6: Let each β_i represents a node in V_1 and each γ_i represents a node in V_2, $1 \le i \le m$. Draw an edge from β_i to all γ_i's if $\beta_i \subset \gamma_i$. An example is given in figure 6 for $n=5$. It should be clear that the resulting graph is a regular bipartite graph with degree $\lceil \frac{n}{2} \rceil$ ($\lceil k \rceil$: is the smallest integer greater than or equal to k). Thus there exists a complete matching from V_1 to V_2 and from V_2 to V_1. These matching are represented by heavy the lines in figure 6. □

Acknowledgments

The authors wish to thank Professor Gregory Bachelis from the math department for his comments during this work.

References

[1] S.B. Akers and B.Krishnamurthy, "A Group-Theoretical Model For Symmetric Interconnection networks," , *IEEE Trans. Computers,* Vol. 38, pp. 555-566, April 1989.

[2] S.B. Akers, D. Horel, and B. Krishnamurthy, "The Star Graph: an Attractive Alternative to the n-Cube" *Proc. Intl. Conf. Parallel Processing,* pp. 393-400, 1987

[3] K. Day and A. Tripathi, "A Comparative Study of Topological Properties of Hypercubes and Star Graphs", *IEEE Trans. on Parallel and Distributed Systems,* Vol. 5 January 1994.

[4] M. Livingston and Q. F. Stout, "Distributing Resources in Hypercube Computers," *Proc. 3rd Conf. Hypercube Concurrent Computers and Applications,* Jan. 1988, pp. 222-231.

[5] A. L. Narasimha Readdy, P. Banerjee, and Santosh G. Abraham. "I/O embedding in hypercubes," *Int. Conf. on parallel Processing,* August 1988, pp. 331-338.

[6] G-M. Chiu and C.S. Raghavendra, "Resource Allocation in Hypercube System," in *Proc. 5th Distributed Memory Computing Conf.,* April 1990, pp. 894-902.

[7] H.-L. Chen and N.-F. Tzeng, "Fault-Tolerant Resource Placement in Hypercube Computers", *Proc. Intl. Conf. Parallel Processing,* pp.I-517-524, 1991.

[8] H.-L. Chen and N.-F. Tzeng, "Efficient Resource Placement in Hypercubes Using Multiple-Adjacency Codes", *IEEE Trans. Computers,* Vol.43, pp.23-32, January 1994

[9] P. Ramanthan and S. Chalasani, "Resource Placement in k-Ary n-Cubes," *Int. Conf. on parallel Processing,* 1992, pp. 133-140.

[10] A. Menn and A. Somani, "An Efficient Sorting Algorithm for Star Interconnection Network," *Int. Conf. on parallel Processing,* 1990, pp. III-1 to III-8.

[11] F. Harary, *Graph Theory,* Addison Wesley, 1969, pp. 47-48.

[12] R.C Bose and B. Manvel, *Introduction to combinatorial theory,* Wiley & Sons, 1984, pp. 205-211.

[13] C.L. Liu, *Introduction to Combinatorial Mathematics.* New York, McGraw-Hill, 1968.

[14] M.-S. Chen and K. Shin. "Processor allocation in an n-cube multiprocessor using gray codes" *IEEE Trans. Computers,* December 1987, pp.1396-1407.

[15] J. Jow, S. Lakshmivarahan, and S. Dhall. "Embedding of Cycles and Grids in Star Graphs," *Journal of Circuits, Systems, and Computers.* Vol. 1, No.1, 1991, pp. 43-74.

[16] M.R.Garey and D.S.Johnson, *Computers and Intractability: A Guide to the Theory of NP- Completeness.* New York: W.H. Freeman & Co., 1997.