Technical Report No. 93-354

EDGE-DISJOINT SPANNING TREES ON THE STAR NETWORK WITH APPLICATIONS TO FAULT TOLERANCE

by
Paraskevi Fragopoulou and Selim G. Akl

Department of Computing & Information Science
Queen's University
Kingston, Ontario, Canada

November 1993

This research was supported by the Telecommunications Research Institute of Ontario and by the Natural Sciences and Engineering Research Council of Canada.

Abstract

Data communication and fault tolerance are important issues in multiprocessor systems. One way to achieve fault tolerant communication is by exploiting and effectively utilizing the disjoint paths that exist between pairs of source, destination nodes. In this paper we construct a structure, called the *multiple edge-disjoint spanning trees*, on the star network, denoted by S_n. This is used for the derivation of an optimal single node broadcasting algorithm, which offers a speed up of $n-1$ compared to the straightforward single node broadcasting algorithm that uses a single breadth first spanning tree. It is also used for the derivation of fault tolerant communication algorithms. As a result, fault tolerant algorithms are presented for four basic communication problems: the problem of a single node sending the same message to all other nodes or *single node broadcasting*, the problem of simultaneous single node broadcasting from all nodes or *multinode broadcasting*, the problem of a single node sending distinct messages to each one of the other nodes or *single node scattering* and finally the problem of simultaneous single node scattering from all nodes or *total exchange*. Fault tolerance is achieved by sending multiple copies of the message through a number of disjoint paths. These algorithms operate successfully in the presence of up to $n-1$ faulty nodes or edges in the system. They also offer the flexibility of controlling the degree of fault tolerance, depending on how reliable the network is. As pointed out in [28], the importance of these algorithms lies in the fact that no knowledge of the faulty nodes or edges is required in advance. All of the algorithms presented make the assumption that each node can exchange messages of fixed length with all of its neighbors simultaneously at each time step, i.e. the *all-port* communication assumption, and that communication is bidirectional.

Key words and phrases: communication algorithm, edge-disjoint, fault tolerance, interconnection network, optimality, parallel algorithm, spanning tree, star network.

Figure 1: (a) The S_3 network. (b) The S_4 network: 4 interconnected S_3 networks.

1 Introduction

The star network was proposed in [1] as "an attractive alternative to the n-cube" topology for interconnecting processors in parallel computers. Since its introduction, the network received considerable attention. Let us denote by V_n the set of $n!$ permutations of symbols $\{1, 2, ..., n\}$. A star interconnection network on n symbols, denoted by $S_n = (V_n, E_{S_n})$, is an undirected graph with $n!$ nodes. Each node $i = i_1 i_2 ... i_n$ is connected to $n-1$ nodes that are obtained by transposing the first with the k^{th} symbols of i, i.e. $(\underline{i_1} i_2 ... i_{k-1} \underline{i_k} i_{k+1} ... i_n, \underline{i_k} i_2 ... i_{k-1} \underline{i_1} i_{k+1} ... i_n) \in E_{S_n}$, for $2 \leq k \leq n$, Fig.1. We call these $n-1$ connection *dimensions*. Thus each node is an endpoint of $n-1$ edges through dimensions $2, 3, ..., n$. S_n enjoys a number of properties desirable in interconnection networks. These include node and edge symmetry, maximal fault tolerance, and strong resilience. Because of its symmetry, the network is easily extensible, can be decomposed in various ways and allows for simple routing algorithms. In addition S_n is superior to $\mathcal{C}_n$ (the n-cube) with respect to two key properties: degree (number of edges at each node), and diameter (maximum distance between any two nodes) [1]. The degree of S_n is $n-1$, i.e. *sublogarithmic* to the number of its nodes while a hypercube with $\Theta(n!)$ nodes has degree $\Theta(\log n!) = \Theta(n \log n)$, i.e. *logarithmic* to the number of its nodes. The same can be said for the diameter of S_n which is $\lfloor \frac{3(n-1)}{2} \rfloor$. The network was shown to be Hamiltonian [24], and efficient algorithms for sorting [23] and Fourier transform computation [10, 11], were developed on it.

Data communication and fault tolerance are important issues in multiprocessor systems, in which processors are connected to each other according to a specific topology. In order for a network of processors to be candidate for parallel processing, it must lend itself to the derivation of optimal communication and fault tolerant algorithms. Working towards this direction in this paper, we construct the multiple edge-disjoint spanning trees structure on the star interconnection network. We say that a node h of S_n is the root of *multiple edge-disjoint spanning trees*, denoted by EDT_h, if each of the nodes adjacent to h is the root of a tree that spans all nodes of S_n except h and all of these trees are edge-disjoint. This structure is useful for

the construction of optimal communication and fault tolerant communication algorithms and has been used before for other popular interconnection networks such as the hypercube [16, 18] and the cube connected cycles [15] networks.

Using the multiple edge-disjoint spanning trees structure we derive an optimal algorithm for the single node broadcasting problem and optimal fault tolerant algorithms for the single node broadcasting, multinode broadcasting, single node scattering and total exchange problems under the all-port communication assumption on S_n. *Single node broadcasting* is the problem where a node wishes to transmit the same message to all other nodes. *Multinode broadcasting* is the problem of simultaneous single node broadcasting of the same message from every node to all other nodes. *Single node scattering* is the problem of a single node sending distinct messages to each one of the other nodes. Finally, *total exchange* is the problem of each node sending distinct messages to every other node. The optimal single node broadcasting algorithm derived offers a speed up of $n-1$ over the straightforward algorithm that uses a single breadth first spanning tree. The basic idea is to split the original message into $n-1$ packets of equal size, each of which is broadcast independently through a different edge-disjoint spanning tree. Each node receives part of the message through a different disjoint path from the source node and as a consequence the network resources are fully utilized. To achieve fault tolerant communication multiple copies of the same message are send through the edge-disjoint spanning trees. As a consequence each node receives a copy of the message through a number of disjoint paths from the source node and the reliability of the algorithm is increased. The algorithms presented can operate successfully in the presence of up to $n-1$ faulty nodes or edges in the system. They also offer the flexibility of controlling the degree of fault tolerance depending on the required reliability, by forcing the same message through a specific number of edge-disjoint subtrees. As pointed out in [28], the importance of these algorithms lies in the fact that no knowledge of the faulty nodes or edges is required in advance. In all of the algorithms the assumption that each node can exchange messages of fixed length with all of its neighbors at each time step, i.e. the *all-port* communication assumption, is adapted. Communication is assumed to be bidirectional. Other data communication algorithms and properties on S_n can be found in [1, 4, 5, 13, 14, 22, 25, 26, 27]. Fault tolerant algorithms and properties on S_n using different approaches can be found in [2, 8, 9, 17, 19, 29].

This paper is organized as follows: Following the introduction to the subject in section 1, notations and definitions that are used throughout the paper are introduced in section 2. Section 3 presents the multiple edge-disjoint spanning trees structure on the star network. In section 4 we demonstrate several applications of this structure in the areas of data communication and fault tolerance. More specificly, lower bounds for all the algorithms presented are derived in subsection 4.1. The optimal single node broadcasting algorithm of M messages under the all-port assumption is presented in subsection 4.2. Finally, the fault tolerant algorithms for the single node broadcasting, multinode broadcasting, single node scattering and total exchange problems, under the all-port assumption again, are presented in subsections 4.3 to 4.6 respectively. We conclude in section 5, along with a summary of the results and some suggestions for further research.

2 Notations and definitions

In what follows, node i is labeled by permutation $i_1 i_2 ... i_n$. By I_n we denote the sorted permutation on the n symbols $\{1, 2, ..., n\}$. Calligraphic letters are used for sets. We denote by $\mathcal{N}$ the set of symbols $\{1, 2, ..., n\}$. Symbols i, j and h are used for nodes of S_n. By $dim(i, j)$ we denote the dimension of edge (i, j). Two paths between a pair of nodes are parallel if they are node (and as an extension edge) disjoint. A misplaced symbol of a node is a symbol that does not occupy its correct position.

S^k_{n-1}, $2 \leq k \leq n$, is the subnetwork induced by all nodes of S_n with symbol 1, in the k^{th} position of their label. It is well known that S^k_{n-1}, $2 \leq k \leq n$, is an S_{n-1} defined on symbols $\{2, ..., n\}$, [1]. For notation purposes, in what follows, we use the symbol S^1_{n-1}, to denote the set of $(n-1)!$ nodes of S_n with symbol 1 in the first position of their label. It is known that S^1_{n-1} is a collection of $(n-1)!$ isolated nodes.

Definition 1: In the *cycle notation* of a node each symbols position is that occupied by the next symbol (cyclically) in the cycle (the position of a symbol is defined with respect to the sorted permutation I_n) [20]. Cycles with only one symbol are excluded from the cycle notation of a node. For example node 341526 has cycle notation (13)(245).

In what follows for node i, we denote by c_i, s_i, the number of cycles and the number of symbols that belongs to those cycles, respectively, in the cycle notation of i. The minimum distance of a node i from node I_n has been shown to be [1]:

$$d_{I_n}(i) = \begin{cases} c_i + s_i, & \text{if } i_1 = 1, \\ c_i + s_i - 2, & \text{otherwise} \end{cases}$$

We now define two operations on nodes of the star network, namely the translation and the rotation operations, that will be of primary importance for the construction of the multiple edge-disjoint spanning trees on S_n and the description of the fault tolerant communication algorithms.

Definition 2: Consider a node h of the star network. We define T_h, the *translation with respect to* h, of a node i as:

$$T_h(i) = h \cdot i$$

(this operation is often referenced as *permutation composition*). By *translation of a network with respect to* h we mean that each node of the network is translated with respect to h. The *inverse translation with respect to* h, denoted by T_h^{-1}, of a node i, is defined as:

$$T_h^{-1}(i) = h^{-1} \cdot i$$

Lemma 1: Let i, j and h represent nodes of S_n. Then (i, j) and $(T_h(i), T_h(j))$ are edges of the same dimension.

Proof: This becomes obvious if we analytically express (i, j) and $(T_h(i), T_h(j))$ as:

$$(\underline{i_1} i_2 ... i_{k-1} \underline{i_k} i_{k+1} ... i_n, \underline{i_k} i_2 ... i_{k-1} \underline{i_1} i_{k+1} ... i_n)$$

$$(\underline{h_{i_1}} h_{i_2} ... h_{i_{k-1}} \underline{h_{i_k}} h_{i_{k+1}} ... h_{i_n}, \underline{h_{i_k}} h_{i_2} ... h_{i_{k-1}} \underline{h_{i_1}} h_{i_{k+1}} ... h_{i_n})$$

Clearly if (i, j) is an edge of dimension k then $(T_h(i), T_h(j))$ is also an edge of dimension k. □

Definition 3: Let us define the function r from $\mathcal{N}$ to $\mathcal{N}$ as:

$$r(k) = \begin{cases} k, & \text{if } k = 1 \\ (k-1) \bmod (n-1) + 2, & \text{otherwise} \end{cases}$$

(notice that r maps $\{1,2,3,...,n-1,n\}$ to $\{1,3,4,...,n,2\}$). The rotation of a node $i \in S_n$, denoted by R, is defined as:

$$R(i) = r(i_1)r(i_n)r(i_2)...r(i_{n-1})$$

or equivalently $i' = R(i)$ so that $i'_{r(k)} = r(i_k)$. By $R^k = R \circ R^{k-1}$ we denote k applications of rotation. By *rotation of a network* we mean that rotation is applied to each node of the network.

Lemma 2: Let i and j be nodes of S_n and $i' = R(i)$ and $j' = R(j)$ be the nodes obtained from i and j, respectively, by application of a rotation:

1. If (i,j) is an edge of dimension k, $2 \leq k \leq n$, then (i',j') is an edge of dimension $r(k)$. As an extension to this, the edges obtained after $1,2,...,n-2$ applications of rotation on (i,j) have dimensions $k+1, k+2, ..., n, 2, ...k-1$, respectively. With this observation we conclude that the $n-1$ edges, each obtained as a rotation of its previous one, are all of different dimensions.

2. If $i \in S^k_{n-1}$, $1 \leq k \leq n$, then $i' \in S^{r(k)}_{n-1}$.

3. The rotation operation preserves the distance between nodes of S_n, or equivalently, $d_{I_n}(i) = d_{I_n}(i')$.

Proof: We'll prove each part separately:

1. If we analytically express (i,j) and (i',j') as:

$$(\underline{i_1} i_2 ... i_{k-1} \underline{i_k} i_{k+1} ... i_n, \underline{i_k} i_2 ... i_{k-1} \underline{i_1} i_{k+1} ... i_n)$$

$$(\underline{r(i_1)} r(i_n) ... r(i_{k-2}) r(i_{k-1}) \underline{r(i_k)} ... r(i_{n-1}), \underline{r(i_k)} r(i_n) ... r(i_{k-2}) r(i_{k-1}) \underline{r(i_1)} ... r(i_{n-1}))$$

   we notice that if (i,j) is an edge of dimension k, then (i',j') is an edge of dimension $r(k)$. This is true because from the definition of rotation the position of symbol $r(i_k)$ in i' is $r(k)$.

2. If $i \in S^k_{n-1}$, $1 \leq k \leq n$, then $i_k = 1$. From the definition of r, if $i_k = 1$ then $i'_{r(k)} = r(i_k) = r(1) = 1$. As a result $i' \in S^{r(k)}_{n-1}$.

3. We must prove the following: (a) if $i_1 = 1$ then $i'_1 = 1$, else if $i_1 \neq 1$ then $i'_1 \neq 1$, (b) $s_i = s_{i'}$, and (c) $c_i = c_{i'}$. From part 2 of this lemma (a) is easily derived. We know from the definition of rotation that $i'_{r(k)} = r(i_k)$. This means that if symbol i_k occupies position k in i then symbol $r(i_k)$ occupies position $r(k)$ in i'. As a consequence if cycle $(i_{k_1}, i_{k_2}, ..., i_{k_l})$ belongs to the cycle notation of i then cycle $(r(i_{k_1}), r(i_{k_2}), ..., r(i_{k_l}))$ belongs to the cycle notation of i' and we conclude that $s_i = s_{i'}$ and $c_i = c_{i'}$. □

To summarize, the translation and the rotation operations preserve the distance between nodes of S_n. The rotation operation maps every edge in dimension d to an edge in dimension $r(d) = (d-1)\text{mod}(n-1)+2$. Application of rotation k times, or R^k, maps every edge in dimension d to an edge in dimension $r^k(d) = (d-2+k)\text{mod}(n-1)+2$. The translation operation preserves the dimension of every edge. Finally the topology of S_n, or a subgraph of S_n, remains unchanged under translation or rotation.

Definition 4: A group of nodes for which each one is derived from its previous one by application of a rotation is called a *necklace*

Lemma 3: Necklaces have the following properties:

1. Each node $i \in S^k_{n-1}$, $2 \leq k \leq n$, belongs to a necklace that includes $n-1$ distinct nodes.

2. Each node $i \in S^1_{n-1}$ belongs to a necklace that includes at most $n-1$ distinct nodes.

3. All nodes of a necklace have the same minimum distance from I_n.

Proof: We prove each part separately.

1. Node $i \in S^k_{n-1}$, $2 \leq k \leq n$, has $i_1 \neq 1$. From the definition of r, the $n-1$ nodes derived from i by consecutive rotations have first symbols $i_1, i_1+1, ..., n, 2, ..., i_1-1$. So the $n-1$ nodes that belong to a necklace of this type start with different symbols and as a consequence are different. Also a necklace of this type contains exactly $n-1$ nodes. From the definition of r, it is true that $r^{n-1}(k) = k$. If i' is produced by i after $n-1$ rotations then $i'_k = r^{n-1}(i_k) = i_k$, $1 \leq k \leq n$, and we conclude that $i' = R^{n-1}(i) = i$. For example node 4123 of S_4 belongs to necklace (4123, 2413, 3421).

2. Node $i \in S^1_{n-1}$ has $i_1 = 1$. From the definition of r, all nodes derived from i by consecutive rotations start with symbol 1. If i' is produced by i after $n-1$ rotations then $i'_k = r^{n-1}(i_k) = i_k$, $1 \leq k \leq n$, and we conclude that $i' = R^{n-1}(i) = i$. However it is possible that $i' = R^k(i) = i$, after $k < n-1$ rotations. For example node 13254 of S_5 is mapped to itself after only two and not $n-1=4$ applications of rotation, $R^2(13254) = 13254$, and belongs to a necklace that contains only two nodes (13254, 15432), while node 12435 belongs to a necklace that contains $n-1=4$ nodes (12435, 12354, 15342, 13245).

3. From part 3 of lemma 2 this is easily derived. □

From part 3 of lemma 3 we conclude that the nodes of S_n at each distance from I_n are grouped into necklaces. For example, the necklaces of S_4 at each distance from I_4 are given below enclosed in parentheses:

$d_{I_4} = 0$: (1234)
$d_{I_4} = 1$: (2134, 3214, 4231)
$d_{I_4} = 2$: (3124, 4213, 2431) (4132, 2314, 3241)
$d_{I_4} = 3$: (3142, 4312, 2341) (4123, 2413, 3421) (1243, 1432, 1324)
$d_{I_4} = 4$: (2143, 3412, 4321) (1342) (1423)

The size of a necklace of S_n is always a divisor of $n-1$.

Definition 5: An *unfolded necklace* is a group of exactly $n-1$ nodes, each obtained as a rotation of its previous one. Unfolded necklaces can contain the same node more than once. For example (13254, 15432, 13254, 15432) is an unfolded necklace of S_5.

The definitions of the rotation operation and the necklace will be of primary importance for the construction of the multiple edge-disjoint spanning trees and for the description of the fault tolerant algorithms on S_n. Both of these definitions have been developed in analogy to definitions with similar properties that exist for the hypercube interconnection network. The application of rotation on a node of S_n is analogous to the application of a right cyclic shift operation on a node of the hypercube. The definition of necklace for nodes of S_n is analogous to similar groups defined for nodes of the hypercube in [18]. The term necklace was initially used in [21] for similar groups of nodes in the shuffle-exchange graph. An interesting observation is that although the definitions in [18] were motivated by specific properties of the hypercube topology, similar definitions, with the same properties, can be derived for other networks, like the star network, which has a structure that is fundamentally different from that of the hypercube.

Figure 2: (a) A schematic representation of SPT_{I_n}. (b) The SPT_{I_4}.

3 Construction of the multiple edge-disjoint spanning trees

We say that node h of S_n is the root of *multiple edge-disjoint spanning trees*, denoted by EDT_h, if each of the nodes adjacent to h is the root of a tree that spans all nodes of S_n except h and all of these trees are edge-disjoint. In this section we construct EDT_{I_n} rooted at node I_n of S_n. The EDT_h, rooted at any other node h of S_n, will be obtained by applying the operation of translation with respect to h on EDT_{I_n}

Before we proceed to the construction of the EDT_{I_n}, we construct a balanced shortest path tree, rooted at node I_n, that includes all nodes of S^k_{n-1}, $2 \leq k \leq n$, denoted by SPT_{I_n}. For the definition of the SPT_{I_n} we need the following: Denote by $\mathcal{C}_k$, $1 \leq k \leq n$, the set of dimensions $\{2, 3, ..., n\} - \{k\}$ ($\mathcal{C}_1$ is the set of dimensions $\{2, 3, ..., n\}$). Assume node $i \in S^k_{n-1}$, $2 \leq k \leq n$. If we move from i along any of the dimensions in $\mathcal{C}_k$, the resulting node belongs to the same substar S^k_{n-1} that i belongs. We split $\mathcal{C}_k$ into two subsets

$$\mathcal{C}^1_{k,i} = \{c \in \mathcal{C}_k : i_c = c\} \cup \begin{cases} \{c : i_c = k\}, & \text{if } k \text{ is the first misplaced symbol cyclically} \\ & \text{to the right of symbol 1 in } i \text{ (excluding } i_1\text{)}, \\ \emptyset, & \text{otherwise} \end{cases}$$

and $\mathcal{C}^2_{k,i} = \mathcal{C}_k - \mathcal{C}^1_{k,i}$. Also let $p_i = i_1$ if $i_1 \neq k$, else let p_i be such that i_{p_i} is the first misplaced symbol cyclically to the right of symbol 1 in i (excluding i_1).

In what follows the k^{th} subtree of a spanning tree ST_h rooted at node h, is defined to be the subtree rooted at the neighbor of h over dimension k, and is denoted by $T^{ST_h}_k$.

Definition 6: The shortest path tree SPT_{I_n} rooted at node I_n of S_n is defined through the following parent and children functions:

$$\text{parent}^{SPT}(i, I_n) = \begin{cases} \emptyset, & \text{if } i = I_n, \\ i_{p_i} i_2 ... i_{p_i - 1} i_1 i_{p_i + 1} ... i_n, & \text{if } i \in S^k_{n-1},\ 2 \leq k \leq n \end{cases}$$

$$\text{children}^{SPT}(i, I_n) = \begin{cases} i_c i_2 ... i_{c-1} i_1 i_{c+1} ... i_n, & \forall c \in \mathcal{C}_1, \quad \text{if } i = I_n, \\ i_c i_2 ... i_{c-1} i_1 i_{c+1} ... i_n, & \forall c \in \mathcal{C}^1_{k,i}, \quad \text{if } i \in S^k_{n-1},\ 2 \leq k \leq n \end{cases}$$

It can be easily seen that the parent^{SPT} and children^{SPT} functions are consistent. A schematical representation of SPT_{I_n} along with SPT_{I_4} can be seen in Fig. 2.

Lemma 4: The SPT_{I_n} has the following characteristics:

1. All nodes of S_{n-1}^k, $2 \leq k \leq n$, belong to subtree $T_k^{SPT_{I_n}}$.

2. It is a shortest path tree.

Proof: We prove each part separately.

1. We'll prove that if $i \in S_{n-1}^k$, $2 \leq k \leq n$, then its parent $i_{p_i} i_2 ... i_n$ also belongs to S_{n-1}^k, except if i is a node adjacent to I_n in which case its parent is I_n. To show this we must prove that $p_i \neq k$ for all nodes that are not adjacent to I_n, which is true from the definition of p_i. If $i_1 \neq k$ then $p_i = i_1 (\neq k)$. If $i_1 = k$ then p_i is such that i_{p_i} is the first misplaced symbol cyclically to the right of symbol 1 (position k) in i, excluding symbol i_1. In this case $p_i = k$ if the only misplaced symbols in i are symbols $i_1 = k$ and $i_k = 1$, which concludes that node i is adjacent to I_n.

2. We'll prove that if $i \in S_{n-1}^k$, $2 \leq k \leq n$, then $d_{I_n}(i_{p_i} i_2 ... i_n) = d_{I_n}(i) - 1$, or that the parent of each node in SPT_{I_n} is closer to I_n than the node itself. This can be verified from a close look to the definition of p_i. If $i_1 \neq k$ then $p_i = i_1$, and the first symbol of i is moved to its correct position. If $i_1 = k$ then p_i is such that i_{p_i} is the first misplaced symbol (excluding i_1) cyclically to the right of symbol 1 in i, which has the effect of merging cycle $(1k)$ with the cycle that includes symbol p_i in the cycle notation of i. From the definition of d_{I_n} the above follows. □.

We now extend the definition of SPT_{I_n}, to include nodes of S_{n-1}^1. SPT_{I_n} is extended so that each one of its nodes has a child that belongs to S_{n-1}^1 (except nodes that are adjacent to I_n). The resulting structure is no more a spanning tree but a directed graph denoted by SPG_{I_n}.

Definition 7: The shortest path graph SPG_{I_n}, rooted at node I_n of S_n is defined through the following parent and children functions. By $\text{parent}^{SPG}(i, l, I_n)$ and $\text{children}^{SPG}(i, l, I_n)$, we denote the parent and children nodes, respectively, of node i in subtree $T_l^{SPG_{I_n}}$.

$$\text{parent}^{SPG}(i, l, I_n) = \begin{cases} \text{parent}^{SPT}(i, I_n), & \text{if } i = I_n \text{ or } i \in S_{n-1}^l, \\ i_l i_2 ... i_{l-1} i_1 i_{l+1} ... i_n, & \text{if } i \in S_{n-1}^1 - I_n \end{cases}$$

$$\text{children}^{SPG}(i, l, I_n) = \begin{cases} \text{children}^{SPT}(i, I_n), & \text{if } i = I_n \text{ or } i \text{ is adjacent to } I_n, \\ \text{children}^{SPT}(i, I_n) \cup \{1 i_2 ... i_n\}, & \text{if } i \in S_{n-1}^l \text{but } i \text{ is not adjacent to } I_n, \\ \emptyset, & \text{if } i \in S_{n-1}^1 - I_n \end{cases}$$

It can be easily seen that the parent^{SPG} and children^{SPG} functions are consistent. The SPG_{I_4} can be seen in Fig. 3.

Lemma 5: The SPG_{I_n} has the following characteristics:

1. Each node of S_{n-1}^1, except I_n, belongs $n-1$ times in SPG_{I_n}, once in each of the subtrees $T_l^{SPG_{I_n}}$, $2 \leq l \leq n$.

2. For each $i \in S_{n-1}^1 - I_n$, there are $n-1$ parallel paths that lead to node I_n through SPG_{I_n}, and these paths have minimum lengths [8].

Proof: We prove each part separately.

1. According to the definition of parent^{SPG}, each node $i \in S_{n-1}^1 - I_n$ is connected to $T_l^{SPG_{I_n}}$, $2 \leq l \leq n$, through dimension l.

Figure 3: The SPG_{I_4}.

2. Node $i \in S^1_{n-1}$ is connected to subtree $T_l^{SPG_{I_n}}$, $2 \leq l \leq n$, through dimension l, to node $i' = i_l i_2 ... i_{l-1} i_1 i_{l+1} ... i_n \in S^l_{n-1}$. From lemma 4, i' is connected with a shortest path to I_n through subtree $T_l^{SPT_{I_n}}$ that includes only nodes of S^l_{n-1}. As a consequence, these paths are parallel since the path through the l^{th} subtree includes only nodes of S^l_{n-1}. Using this type of reasoning it has been proven in [8] that these paths have minimum lengths. □

Up to this point only nodes of $S^1_{n-1} - I_n$ belong to all subtrees $T_l^{SPG_{I_n}}$, $2 \leq l \leq n$. However nodes of any other S^l_{n-1}, $2 \leq l \leq n$, belong only to subtree $T_l^{SPG_{I_n}}$. Now we further extend SPG_{I_n} so that each subtree includes all nodes $i \in S^l_{n-1}$, $2 \leq l \leq n$. The resulting structure will be the multiple edge-disjoint spanning trees, denoted by EDT_{I_n}. In order for the subtrees to be edge-disjoint, each node should be connected to each subtree through a different neighboring node and as an extension through a different one of its incident edges. Let us remind that node $i \in S^l_{n-1}$, $2 \leq l \leq n$, is connected to its parent in the l^{th} subtree through neighbor $i_{p_i} i_2 ... i_{p_i - 1} i_1 i_{p_i + 1} ... i_n$.

Definition 8: The EDT_{I_n} rooted at node I_n of S_n is now defined through the following parent and children functions. By parent$^{EDT}(i, l, I_n)$ and by children$^{EDT}(i, l, I_n)$, we denote the parent and children nodes, respectively, of node i in subtree $T_l^{EDT_{I_n}}$. For clarity of definition we distinguish among different kinds of nodes:

1. Node $i \in S^k_{n-1}$, $2 \leq k \leq n$, with $i_{p_i} = k$ (the parent of i in SPG_{I_n} starts with symbol k). Node i is connected to subtree $T_{i_1}^{EDT_{I_n}}$ through neighbor $1 i_2 ... i_n$, and to any other subtree $T_l^{EDT_{I_n}}$, $l \neq i_1$, $l \neq (k = i_{p_i})$, through neighbor $l i_2 ... i_n$. For example node 3124 of S_n is connected to subtree $T_3^{EDT_{I_4}}$ through neighbor 1324, and to subtree $T_4^{EDT_{I_4}}$ through neighbor 4123.

$$\text{parent}^{EDT}(i, l, I_n) = \begin{cases} i_{p_i} i_2 ... i_n, & \text{if } l = k = i_{p_i}, \quad (1) \\ 1 i_2 ... i_n, & \text{if } l = i_1, \quad (2) \\ l i_2 ... i_n, & \text{otherwise} \quad (3) \end{cases}$$

$$\text{children}^{EDT}(i, l, I_n) = \begin{cases} i_c i_2 ... i_{c-1} i_1 i_{c+1} ... i_n, & \forall c \in \mathcal{C}^1_{k,i} \cup \{k\}, \quad \text{if } l = k, \\ i_c i_2 ... i_{c-1} i_1 i_{c+1} ... i_n, & \forall c \in \mathcal{C}^2_{k,i}, \quad \text{if } l = i_1, \\ \emptyset, & \text{otherwise} \end{cases}$$

Figure 4: The EDT_{I_4}.

2. Node $i \in S^k_{n-1}$, $2 \leq k \leq n$, with $i_1 = k$ (node i starts with symbol k). Node i is connected to subtree $T^{EDT_{I_n}}_{i_{p_i}}$ through neighbor $1i_2...i_n$, and to any other subtree $T^{EDT_{I_n}}_{l}$, $l \neq i_{p_i}$, $l \neq (k = i_1)$, through neighbor $li_2...i_n$. For example node 2143 of S_4 is connected to subtree $T^{EDT_{I_4}}_4$ through neighbor 1243, and to subtree $T^{EDT_{I_4}}_3$ through neighbor 3142.

$$\text{parent}^{EDT}(i,l,I_n) = \begin{cases} i_{p_i}i_2...i_n, & \text{if } l = k = i_1, \quad (4) \\ 1i_2...i_n, & \text{if } l = i_{p_i}, \quad (5) \\ li_2...i_n, & \text{otherwise} \quad (6) \end{cases}$$

$$\text{children}^{EDT}(i,l,I_n) = \begin{cases} i_c i_2...i_{c-1}i_1 i_{c+1}...i_n, & \forall c \in \mathcal{C}^1_{k,i}, & \text{if } l = k \text{ and } i \text{ is adjacent to } I_n, \\ i_c i_2...i_{c-1}i_1 i_{c+1}...i_n, & \forall c \in \mathcal{C}^1_{k,i} \cup \{k\}, & \text{if } l = k \text{ and } i \text{ is not adjacent to } I_n, \\ i' = (i_c i_2...i_{c-1}i_1 i_{c+1}...i_n), & & \text{if } l = i'_{p_{i'}} \end{cases}$$

3. Node $i \in S^k_{n-1}$, $2 \leq k \leq n$, with $i_1 \neq k$ and $i_{p_i} \neq k$ (neither i nor its parent in SPG_{I_n} start with symbol k). Node i is connected to subtree $T^{EDT_{I_n}}_{i_1}$ through neighbor $1i_2...i_n$, and to subtree $T^{EDT_{I_n}}_{i_{p_i}}$ through neighbor $ki_2...i_n$. To any other subtree $T^{EDT_{I_n}}_{l}$, $l \neq i_1$, $l \neq i_{p_i}$, $l \neq k$, it is connected through neighbor $li_2...i_n$. For example node 4123 of S_4 is connected to subtree $T^{EDT_{I_4}}_4$ through neighbor 1423, and to subtree $T^{EDT_{I_4}}_3$ through neighbor 2143.

$$\text{parent}^{EDT}(i,l,I_n) = \begin{cases} i_{p_i}i_2...i_n, & \text{if } l = k, \quad (7) \\ 1i_2...i_n, & \text{if } l = i_1, \quad (8) \\ ki_2...i_n, & \text{if } l = i_{p_i}, \quad (9) \\ li_2...i_n, & \text{otherwise} \quad (10) \end{cases}$$

$$\text{children}^{EDT}(i,l,I_n) = \begin{cases} i_c i_2...i_{c-1}i_1 i_{c+1}...i_n, & \forall c \in \mathcal{C}^1_{k,i} \cup \{k\}, & \text{if } l = k, \\ i_c i_2...i_{c-1}i_1 i_{c+1}...i_n, & \forall c \in \mathcal{C}^2_{k,i}, & \text{if } l = i_1, \\ \emptyset, & \text{otherwise} & \end{cases}$$

4. Node $i \in S^1_{n-1}$, $i \neq I_n$ (nodes that start with symbol 1).

$$\text{parent}^{EDT}(i,l,I_n) = \{i_l i_2 ... i_{l-1} 1 i_{l+1} ... i_n\} \qquad (11)$$

$$\text{children}^{EDT}(i,l,I_n) = \{l i_2 ... i_n\} \cup \begin{cases} i' = (i_c i_2 ... i_{c-1} 1 i_{c+1} ... i_n), & \text{if } i_c = c \text{ and } l = i'_{p_{i'}}, \\ \emptyset, & \text{otherwise} \end{cases}$$

5. Finally, the parent and children nodes of I_n are:

$$\text{parent}^{EDT}(I_n) = \emptyset$$

$$\text{children}^{EDT}(I_n) = \{i_c i_2 ... i_{c-1} i_1 i_{c+1} ... i_n, \quad \forall c \in \mathcal{C}_1\}$$

The parent^{EDT} and children^{EDT} functions that define EDT_{I_n} are consistent. The EDT_{I_4} can be seen in Fig. 4. Notice that each edge belongs twice in EDT_{I_n}, once in each direction, since communication is bidirectional.

Lemma 6: The EDT_{I_n} has the following characteristics:

1. Subtrees $T_l^{EDT_{I_n}}$, $2 \leq l \leq n$, are all edge-disjoint.
2. Subtree $T_{r(l)}^{EDT_{I_n}}$ is a rotation of subtree $T_l^{EDT_{I_n}}$, $2 \leq l \leq n$ (its previous subtree cyclically).
3. For each node $i \in S^k_{n-1}$, $1 \leq k \leq n$, there are $n-1$ parallel paths of almost minimum lengths that lead to node I_n through EDT_{I_n}.
4. The depth of EDT_{I_n} is at most $\lfloor \frac{3(n-1)}{2} \rfloor + 4$.

Proof: See Appendix. □

The multiple edge-disjoint spanning trees, EDT_h, rooted at any other node h of S_n can be obtained from EDT_{I_n}, using the operation of translation with respect to h (see definition 1). Node i of S_n is connected to its parent, children nodes in subtree $T_l^{EDT_h}$ along the same dimensions that node $T_h^{-1}(i)$ is connected to its parent, children nodes in subtree $T_l^{EDT_{I_n}}$. This is easily derived because connectivity and the dimension of each edge are preserved under translation in S_n (lemma 1).

We need to pose an ordering to the children of each node in each of the subtrees $T_k^{EDT_{I_n}}$, $2 \leq k \leq n$. This will be useful in the construction of the algorithms described in the following section. We define the k^{th} ordering of numbers $\{2, 3, ..., n\}$, denoted by $\prec_k$ to be such that: $k+1 \prec_k k+2 \prec_k ... \prec_k n \prec_k 2 \prec_k ... \prec_k k-1 \prec_k k$. Each node arranges its children in each subtree $T_k^{EDT_{I_n}}$ according to the k^{th} ordering of the dimensions of the edges it is connected to them. This guarantees that if node i is connected to its children in subtree $T_k^{EDT_{I_n}}$ through dimensions $c_1, c_2, ...c_l$ in order, then node $R(i)$ is connected to its children in subtree $T_{r(k)}^{EDT_{I_n}}$, through dimensions $r(c_1), r(c_2), ..., r(c_l)$ again in order. This ordering in combination with the fact that subtrees $T_k^{EDT_{I_n}}$, $2 \leq k \leq n$, are rotations of each other guarantees that corresponding nodes of the subtrees form unfolded necklaces. For example the nodes enclosed in rectangulars of the same kind in Fig. 4 form unfolded necklaces. Also corresponding edges of the subtrees are rotations of each other and as consequence all of different dimensions (lemma 6). For example the dotted edges in Fig. 4 are rotations of each other and of different types. The ordering is carried by translation to EDT_h rooted at any other node h of S_n.

4 Applications

The multiple edge-disjoint spanning trees structure, defined in the previous section is used to derive optimal communication and fault tolerant communication algorithms on the star network. More specificly we derive an optimal single node broadcasting algorithm. We also derive optimal fault tolerant algorithms for four basic communication problems in interconnection networks, namely the single node broadcasting, multinode broadcasting, single node scattering and total exchange problems. All of the algorithms operate under the all-port communication assumption. Before we proceed to the description of the algorithms, we derive lower bound for the time and the number of message transmissions required for each of them.

4.1 Lower Bounds

Broadcasting on an interconnection network is the problem where a node wishes to send the same message to all other nodes in the network. To broadcast M messages from a node of S_n, by pipelining the communication from the root towards the leaves along any $\lfloor \frac{3(n-1)}{2} \rfloor$ depth, breadth first spanning tree, under the all-port communication assumption, the number of time steps required is $M + \lfloor \frac{3(n-1)}{2} \rfloor - 1$, which is not optimal. Since S_n is a regular network with degree $n-1$, the lower bound for the single node broadcasting algorithm of M messages assuming all ports of a node can be used simultaneously for message transmission is $\lceil \frac{M}{n-1} \rceil + \lfloor \frac{3(n-1)}{2} \rfloor$. To achieve this lower bound the M messages are grouped into $n-1$ packets of equal size, each of which is communicated over a different edge of the source node and is pipelined down a different edge-disjoint subtree rooted at a node adjacent to the source node. Since each node receives each of the M messages once, the minimum number of message transmissions required for an optimal single node broadcasting algorithm is $M(n!-1)$. In the fault tolerant single node broadcasting algorithm the M messages are pipelined down each one of the of the $n-1$ edge-disjoint spanning trees rooted at the nodes adjacent to the source node. The time required for this algorithm is $M + \lfloor \frac{3(n-1)}{2} \rfloor$. Since each node receives each of the M messages through $n-1$ parallel paths, the minimum number of message transmissions required is $M(n!-1)(n-1)$.

Multinode broadcasting on an interconnection network is the problem where each node of the network wishes to send a message to all other nodes. If each node wishes to broadcast M messages, then each node must receive a total of $M(n!-1)$ messages. As a consequence the minimum number of message transmissions required is $Mn!(n!-1)$. Under the all-port assumption all $n!(n-1)$ edges of the network can be used for message transmissions at each time step. Thus the minimum time required for the algorithm to complete is $\lceil \frac{M(n!-1)}{n-1} \rceil$. The lower bounds for the fault tolerant multinode broadcasting algorithm are easily derived from the lower bounds for the multinode broadcasting with a multiplication by factor $n-1$.

Single node scattering on an interconnection network is the problem where a node wishes to send a different message to each one of the other nodes. If the source node wishes to send M messages to each one of the other nodes, $M(n!-1)$ different messages must be transmitted by the source node. Under the all-port assumption all the $n-1$ edges incident to the source node can be used for message transmissions at each time step and as a consequence the minimum time required for the algorithm to complete is $\lceil \frac{M(n!-1)}{n-1} \rceil$. The number of message transmissions required can be found as follows: A message destined to a specific node must travel as many edges as the shortest distance from the source to this node. If we sum the shortest

distances from the source to each node, this will be the minimum number of message transmissions required for this problem:

$$\sum_{k=1}^{\lfloor \frac{3(n-1)}{2} \rfloor} k|N_k| = n! \frac{\sum_{k=1}^{\lfloor \frac{3(n-1)}{2} \rfloor} k|N_k|}{n!} = n!d$$

$|N_k|$ is the number of nodes at a distance k from the source and d has been shown to be [3]:

$$d = n + \frac{2}{n} + H_n - 4$$

H_n is the n^{th} harmonic number: $H_n = 1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}$. Thus the minimum number of message transmissions required for a single node scattering algorithm on S_n is:

$$Mn!(n + \frac{2}{n} + H_n - 4)$$

In the fault tolerant single node scattering algorithm the source node transmits the $M(n!-1)$ messages to all of its neighbors simultaneously. Each of the $n-1$ edge-disjoint spanning trees rooted at the nodes adjacent to the source node are used for a single node scattering algorithm. The number of message transmissions required is $M(n!-1)(n-1)+Mn!(n+\frac{2}{n}+H_n-4)(n-1)$. Since the source node must transmit $M(n!-1)(n-1)$ messages the time required for this algorithm is $M(n!-1)$.

Total exchange on an interconnection network is the problem where each node wishes to send a distinct message to every other node, in other words, every possible pair of nodes exchange distinct messages. The fault tolerant total exchange algorithm is equivalent to $n!$ different fault tolerant single node scattering algorithms, one from each node of S_n. Thus the minimum number of message transmissions required is $Mn!(n!-1)(n-1) + M(n!)^2(n + \frac{2}{n} + H_n - 4)(n-1)$. Under the all-port assumption $n!(n-1)$ edges can be used for message transmission at each time step simultaneously. Thus the minimum time required for the algorithm to complete is $M(n!-1) + Mn!(n + \frac{2}{n} + H_n - 4)$.

All the lower bounds were derived for degree of fault tolerance $n-2$. This means that each node receives each message through $n-1$ parallel paths. The lower bounds for the algorithms with controlled degree of fault tolerance will be derived in the following sections along with the description of the algorithms.

Table 1 below summarizes the lower bounds for all of the above problems, with degree of fault tolerance $n-2$, and M messages transmitted to each node. By t_n we denote the quantity $n!(n + \frac{2}{n} + H_n - 4)$.

problem	time	number of transmissions
single node broadcasting	$\lceil \frac{M}{n-1} \rceil + \lfloor \frac{3(n-1)}{2} \rfloor$	$M(n!-1)$
fault tolerant single node broadcasting	$M + \lfloor \frac{3(n-1)}{2} \rfloor$	$M(n-1)(n!-1)$
fault tolerant multinode broadcasting	$M(n!-1)$	$M(n-1)n!(n!-1)$
fault tolerant single node scattering	$M(n!-1)$	$M(n-1)(n!-1) + M(n-1)t_n$
fault tolerant total exchange	$M(n!-1) + Mt_n$	$M(n-1)n!(n!-1) + M(n-1)n!t_n$

Table 1: Lower bounds on the star network.

The algorithms derived here for all of the above problems are optimal in terms of time and number of message transmissions. Some of the methods used in this sections to derive lower bounds for the communications problems under consideration are similar to the methods used in [7] to derive lower bounds for similar problems on the hypercube network.

4.2 Optimal single node broadcasting

In a single node broadcasting algorithm one node wishes to transmit a single message or a group of messages to each other node. To broadcast M messages from a node of S_n, by pipelining the communication from the root towards the leaves along any $\lfloor \frac{3(n-1)}{2} \rfloor$ depth, breadth first spanning tree, under the all-port communication assumption, the number of time steps required is $M + \lfloor \frac{3(n-1)}{2} \rfloor - 1$, which is not optimal. Since S_n is a regular network with degree $n-1$, the lower bound for the single node broadcasting algorithm of M messages assuming all ports of a node can be used simultaneously for message transmission is $\lceil \frac{M}{n-1} \rceil + \lfloor \frac{3(n-1)}{2} \rfloor$. This lower bound can be achieved if the M messages are grouped into $n-1$ packets, each of size $\frac{M}{n-1}$. Each of the packets is communicated over a different edge of the source node h and is pipelined down a different edge-disjoint subtree of the EDT_h rooted at the source node. As soon as a node receives a message from its parent node in subtree $T_k^{EDT_h}$, $2 \leq k \leq n$, saves a copy, and forwards the message to its children nodes in the same subtree. The result is that each node receives each of the $n-1$ packets of the message through a different parallel path from the source node. The time required for this algorithm to complete is at most $\lceil \frac{M}{n-1} \rceil + \lfloor \frac{3(n-1)}{2} \rfloor + 3$, which is almost optimal, since the depth of the multiple edge-disjoint spanning trees structure is at most $\lfloor \frac{3(n-1)}{2} \rfloor + 4$. The number of message transmissions required for the algorithm is $M(n!-1)$, since each node receives each of the M messages once, which is the minimum possible. Using this algorithm the resources of the network are fully utilized since all communication edges contribute to the distribution of the information.

4.3 Fault tolerant single node broadcasting

The multiple edge-disjoint spanning trees structure can be used to derive a fault tolerant single node broadcasting algorithm under the all-port communication assumption. Assume that the source node h, wishes to broadcast M messages to all the other nodes. Node h sends the messages it wishes to broadcast through all its incident edges simultaneously and these are pipelined down each of the $n-1$ edge-disjoint subtrees rooted at the nodes adjacent to h. As soon as a node receives a message from its parent node in subtree $T_k^{EDT_h}$, $2 \leq k \leq n$, saves a copy and forwards the message to its children nodes in the same subtree. Using this algorithm each of the nodes of S_n receives the same message through $n-1$ parallel paths. If up to $n-2$ node or edge faults occur in the system that block the message from passing we are still guaranteed that each node receives a copy of the message and as a consequence the algorithm is $n-2$ fault tolerant. If we assume that the system has faults that alter the contents of the messages instead of just blocking or destroying it, the fault tolerance degree of the algorithm decreases since an election algorithm is required at each node in order to select the intact message. A brief discussion on the election algorithms can be found in [28]. The time required for this algorithm to complete using the multiple edge-disjoint spanning trees structure is at most $M + \lfloor \frac{3(n-1)}{2} \rfloor + 3$, which is almost optimal, since the depth of the multiple edge-disjoint spanning trees is at most $\lfloor \frac{3(n-1)}{2} \rfloor + 4$. The number of message transmissions required is $M(n!-1)(n-1)$ since each node receives each of the M messages $n-1$ times, which is the minimum possible.

Using a similar technique we can control the degree of fault tolerance of the single node broadcasting algorithm. Assume that the required degree of fault tolerance is $x-1 \leq n-2$. This means that each node must receive each message through x parallel paths, or in other words that each message must be pipelined

down at least x edge-disjoint subtrees rooted at the nodes adjacent to the source node. However the number of available edge-disjoint subtrees is $n-1$. In order to achieve maximum utilization of the network resources the M messages are grouped into $\frac{n-1}{x}$ packets, each of size $\frac{M}{(n-1)/x} = \frac{Mx}{n-1}$ (x must divide $n-1$ for this to work properly). Each of the $\frac{n-1}{x}$ packets is pipelined down x edge-disjoint subtrees. As a consequence, all of the $n-1$ ($x\frac{n-1}{x} = n-1$) edge-disjoint subtrees are used for message transmission. The result is that each node receives each of the $\frac{n-1}{x}$ packets through x of its incident edges, and as an extension through x parallel paths from the source node, and as a consequence the fault tolerance degree of the algorithm is $x-1$. The time required for the algorithm is at most $\frac{Mx}{n-1} + \lfloor\frac{3(n-1)}{2}\rfloor + 3$ which is almost optimal, since the depth of the multiple edge-disjoint spanning trees is at most $\lfloor\frac{3(n-1)}{2}\rfloor + 4$, and in addition we have the flexibility of controlling the degree of fault tolerance based on how reliable the system is. The number of message transmissions required is $M(n!-1)x$ which is again optimal, since each node receives each of the M messages x times. To illustrate the algorithm assume that node 12345 of S_5, which is the root of $n-1=4$ edge-disjoint subtrees $T_k^{EDT_{I_5}}$, $2 \leq k \leq 5$, whishes to broadcast $M=4$ messages with degree of fault tolerance $x-1=1$ (this means that up to one faulty node or edge should be tolerated by the algorithm). The message of size M is split into $\frac{n-1}{x} = \frac{4}{2} = 2$ packets m_1 and m_2, each of size $\frac{Mx}{n-1} = 2$. Each of the packets is pipelined down $x=2$ edge-disjoint subtrees i.e. m_1 down $T_2^{EDT_{I_5}}$, $T_3^{EDT_{I_5}}$ and m_2 down $T_4^{EDT_{I_5}}$, $T_5^{EDT_{I_5}}$. As a consequence, each node receives each packet through two parallel paths and the fault tolerance degree of the algorithm is one.

4.4 Fault tolerant multinode broadcasting

In a multinode broadcasting algorithm, each node wishes to transmit a single message, or a group of messages to each one of the other nodes. As a consequence each of the nodes should be the root of multiple edge-disjoint spanning trees. The EDT_{I_n} can be replicated at any other node h of S_n using the operation of translation with respect to h, as it was explained at the end of section 3 (see definition 1). Fault tolerance can be achieved if each node receives each message through $n-1$ parallel paths. However in this case we have to guarantee that no conflicts arise during the execution of the algorithm, since all nodes are sources of messages. Under the all-port assumption $n(n-1)$ edges are available on S_n at each time step for message transmission. This means that the messages originating from a specific node should be transmitted through at most $n-1$ edges, at each time step. Let us denote by $L_k(h)$ the set of edges on which messages originating at node h are transmitted at time step k of the algorithm. For each k, $L_k(h)$ is obtained from $L_k(I_n)$ using the operation of translation with respect to h (if $(i,j) \in L_k(I_n)$ then $(T_h(i), T_h(j)) \in L_k(h)$) (see definition 1). The following lemma is enough to guarantee that no conflicts arise during the execution of the algorithm.

Lemma 7: If for each k, the edges in $L_k(I_n)$ are all of different dimensions, then for each k, the sets $L_k(h)$, where h ranges over all nodes of S_n, are disjoint.

Proof: Assume two different edges $(i,j) \neq (i',j') \in L_k(I_n)$ for some k, and take the edges $(T_h(i), T_h(j)) \in L_k(h)$ and $(T_{h'}(i'), T_{h'}(j')) \in L_k(h')$ which are obtained by (i,j) and (i',j'), respectively, under translation with respect to two different nodes of S_n, h and h'. Also assume that $(T_h(i), T_h(j)) = (T_{h'}(i'), T_{h'}(j'))$. Since the dimension of each edge is preserved under translation (lemma 1), this means that $dim(i,j) = dim(T_h(i), T_h(j)) = dim(T_{h'}(i'), T_{h'}(j')) = dim(i',j')$ which contradicts our assumption that (i,j) and

(i', j') are two different edges of $L_k(I_n)$. □

The fault tolerant multinode broadcasting algorithm on S_n, assuming each node wishes to broadcast M messages, proceeds as follows:

1. Each source node sends the M messages it wishes to broadcast to all of its neighbors simultaneously.

2. As soon as a node receives a group of M messages from its parent in subtree $T_k^{EDT_h}$, it saves a copy, and forwards the messages to its leftmost child in the same subtree. However, if the node is a leaf of subtree $T_k^{EDT_h}$, it sends an acknowledgement to its parent node in the subtree.

3. When a node receives an acknowledgement from one of its children nodes in subtree $T_k^{EDT_h}$, it forwards the M messages it received from its parent in this subtree to its next child node in the subtree. However, if the node has no more children in this subtree, it sends an acknowledgement to its parent node in the subtree.

The algorithm terminates when each source node receives acknowledgements from all its neighbors. This algorithm corresponds to a depth first traversal of the edges in each of the edge-disjoint subtrees. This means that at each time step of the algorithm corresponding edges of the subtrees, $T_k^{EDT_h}$, $2 \leq k \leq n$, rooted at the nodes adjacent to h, are used simultaneously for message transmission. Since corresponding edges of the $n-1$ subtrees of EDT_{I_n} are all rotations of each other, they are all of different dimensions (lemma 6) and the requirement of lemma 7 for conflict avoidance is satisfied by the algorithm.

The time required for this algorithm to complete is $M(n!-1)$ which is optimal. The number of message transmissions required is $Mn!(n!-1)(n-1)$ which is the minimum possible, since each node receives each of the $M(n!-1)$ messages $n-1$ times. The way the algorithm was constructed, the degree of fault tolerance is $n-2$ which means that each message is transmitted through all of the edge-disjoint subtree rooted at the nodes adjacent to each source node. Controlling the degree of fault tolerance is possible by a technique similar to the one described in subsection 4.3.

4.5 Fault tolerant single node scattering

In a single node scattering algorithm one node wishes to transmit distinct messages to each one of the other nodes. The single node scattering algorithm on S_n, under the all-port assumption, can become fault tolerant using the multiple edge-disjoint spanning trees. A message destined to a specific node is transmitted through each of the edge-disjoint subtrees rooted at the nodes adjacent to the source node. In each subtree, messages destined to nodes that are the furthest from the source are transmitted first.

If each node is the destination of M messages, the time required for this algorithm to complete is $M(n!-1)$, which is optimal, since each edge incident to the source node constitutes a bottleneck for $M(n!-1)$ messages. The number of message transmissions required is $M(n!-1)(n-1)+O(Mt_n(n-1))$ which is asymptotically optimal, because the lengths of the $n-1$ parallel paths between two nodes of S_n are not all equal the length of a shortest path between the two nodes [8]. Controlling the degree of fault tolerance is possible using a technique similar to the one described in subsection 4.3.

4.6 Fault tolerant total exchange

In a total exchange algorithm each node wishes to transmit distinct messages to each other node. As a consequence, each of the nodes should be the root of multiple edge-disjoint spanning trees. The EDT_{I_n} can be replicated at any other node h of S_n using the operation of translation with respect to h (see definition 1). Fault tolerance can be achieved if each node receives each message through $n-1$ parallel paths. As in the fault tolerant multinode broadcasting algorithm, we have to guarantee that no conflicts arise during the execution of the algorithm, since all nodes are sources of messages, or in other words we have to guarantee that the requirement of lemma 7 is satisfied.

The way node I_n transmits the messages through the edge-disjoint subtrees rooted at its neighbors is the following: For each node i of S_n, I_n sends the messages destined to nodes $R^{k-2}(i)$, $2 \leq k \leq n$, respectively through subtrees $T_k^{EDT_{I_n}}$, $2 \leq k \leq n$, simultaneously. As soon as a group of messages reaches its destination another group is send from I_n. Nodes $R^{k-2}(i)$, $2 \leq k \leq n$, form an unfolded necklace of nodes (see definition 5) at a specific level of EDT_{I_n}, since subtrees $T_k^{EDT_{I_n}}$, $2 \leq k \leq n$, are all rotations of each other (lemma 6). As a consequence the $n-1$ paths that lead from I_n to nodes $R^{k-2}(i)$, $2 \leq k \leq n$, respectively through subtrees $T_k^{EDT_{I_n}}$, $2 \leq k \leq n$, are all rotations of each other. This means that the $n-1$ edges at each level of the paths are of different dimensions and the requirement of lemma 7 for conflict avoidance is satisfied. If at a specific instance of the algorithm node I_n transmits messages to nodes $R^{k-2}(i)$, $2 \leq k \leq n$, respectively through subtrees $T_k^{EDT_{I_n}}$, $2 \leq k \leq n$, simultaneously, then any other node h of S_n transmits messages to nodes $T_h(R^{k-2}(i))$, $2 \leq k \leq n$, respectively through subtrees $T_k^{EDT_h}$, $2 \leq k \leq n$, simultaneously. This is a simple application of the operation of translation with respect to h.

If M messages must be transmitted to each node from each other node the time required for the algorithm to compete is $M(n!-1)+O(Mt_n)$ which is asymptotically optimal. The number of message transmissions required is $Mn!(n!-1)(n-1)+O(Mn!t_n(n-1))$ which is again asymptotically optimal. This algorithm is only asymptotically optimal because the lengths of the $n-1$ parallel paths between two nodes of S_n are not all equal to the length of a shortest path between the two nodes [8]. The way the algorithm was described the degree of fault tolerance is $n-2$ which means that each message is transmitted through each different edge-disjoint subtree rooted at the nodes adjacent to each source node. Controlling the degree of fault tolerance is possible by a technique similar to the one described in subsection 4.3.

5 Conclusions

We presented several algorithms on the star interconnection network, in the areas of data communication and fault tolerance. New definitions like that of the rotation operation and the necklace for nodes of S_n were introduced to facilitate the construction of multiple edge-disjoint spanning trees on S_n. As a result a multiple edge-disjoint spanning trees structure of optimal depth was constructed on the star interconnection network. Using this structure an optimal single node broadcasting algorithm and optimal fault tolerant algorithms for the single node broadcasting, multinode broadcasting, single node scattering and total exchange problems on the star network were presented. All of the algorithms operate under the all-port assumption and are optimal in terms of time and number of message transmissions. Constructing multiple edge-disjoint spanning trees on the star network that would offer optimal solutions to the above problems under the assumption

that each node can exchange a message of fixed length with only one of its neighbors at each time step, i.e. the *one-port* communication assumption, is a problem that remains open.

We now provide a comparison of the algorithms presented in this paper for the four communication problems under consideration on the star network, with algorithms for the same problems, under exactly the same assumptions, on the popular hypercube network. Tables 2 and 3 below give the number of message transmissions and the communication time required for each of the problems on the S_n and the hypercube network of dimension k, denoted by C_k, respectively. For the fault tolerant communication algorithms the degree of fault tolerance is assumed to be x.

problem	time	number of transmissions
single node broadcasting	$\lceil \frac{M}{n-1} \rceil + \lfloor \frac{3(n-1)}{2} \rfloor$	$M(n!-1)$
fault tolerant single node broadcasting	$\lceil \frac{Mx}{n-1} \rceil + \lfloor \frac{3(n-1)}{2} \rfloor$	$Mx(n!-1)$
fault tolerant multinode broadcasting	$\lceil \frac{Mx(n!-1)}{n-1} \rceil$	$Mxn!(n!-1)$
fault tolerant single node scattering	$\lceil \frac{Mx(n!-1)}{n-1} \rceil$	$Mx(n!-1)+Mxt_n$
fault tolerant total exchange	$\lceil \frac{Mx(n!-1)}{n-1} \rceil + \lceil \frac{Mxt_n}{n-1} \rceil$	$Mxn!(n!-1)+Mxn!t_n$

Table 2: Lower bounds on the star network of dimension n.

problem	time	number of transmissions
single node broadcasting	$\lceil \frac{M}{k} \rceil + k$	$M(2^k-1)$
fault tolerant single node broadcasting	$\lceil \frac{Mx}{k} \rceil + k$	$Mx(2^k-1)$
fault tolerant multinode broadcasting	$\lceil \frac{Mx(2^k-1)}{k} \rceil$	$Mx2^k(2^k-1)$
fault tolerant single node scattering	$\lceil \frac{Mx(2^k-1)}{k} \rceil$	$Mx(2^k-1)+Mx2^{k-1}k$
fault tolerant total exchange	$\lceil \frac{Mx(2^k-1)}{k} \rceil + Mx2^k$	$Mx2^k(2^k-1)+Mx2^{2k-1}k$

Table 3: Lower bounds on the hypercube network of dimension k.

In table 4 below the performances of the two networks are compared. Since the star network is defined for numbers of nodes which are factorials, while the hypercube is defined for powers of two, the comparison cannot be exact. In the comparison below a hypercube network with $O(2^k) = O(n!)$ nodes and degree $O(k) = O(\log n!) = O(n \log n)$ is assumed.

From table 4 we notice that whenever the performance of an algorithm depends on the degree of the network, as for example the communication times of the fault tolerant multinode broadcasting, single node scattering and total exchange algorithms, the hypercube network performs better than the star network by a factor of $\log n$. On the other hand, whenever the performance of an algorithm depends on the diameter of the network, or the lengths of the shortest paths between nodes, as for example the number of message transmissions of the fault tolerant single node scattering and total exchange algorithms, the star network performs better by a factor of $\log n$. The communication times of the single node broadcasting and the fault tolerant single node broadcasting algorithms depend on both the degree and the diameter of the networks and this is reflected at the comparison of their performances. In any other case the performance of the two

networks is the same. However we should not forget that the star network has smaller degree resulting in processors with a smaller number of ports and as a consequence smaller cost.

problem	net	time	number of transmissions
single node broadcasting	S_n	$O(\frac{M}{n}+n)$	$O(Mn!)$
	C_n	$O(\frac{M}{n\log n}+n\log n)$	$O(Mn!)$
fault tolerant single node broadcasting	S_n	$O(\frac{Mx}{n}+n)$	$O(Mxn!)$
	C_n	$O(\frac{Mx}{n\log n}+n\log n)$	$O(Mxn!)$
fault tolerant multinode broadcasting	S_n	$O(\frac{Mxn!}{n})$	$O(Mx(n!)^2)$
	C_n	$O(\frac{Mxn!}{n\log n})$	$O(Mx(n!)^2)$
fault tolerant single node scattering	S_n	$O(\frac{Mxn!}{n})$	$O(Mxn!+Mxn!n)$
	C_n	$O(\frac{Mxn!}{n\log n})$	$O(Mxn!+Mxn!n\log n)$
fault tolerant total exchange	S_n	$O(\frac{Mxn!}{n}+Mxn!)$	$O(Mx(n!)^2+Mx(n!)^2n)$
	C_n	$O(\frac{Mxn!}{n\log n}+Mxn!)$	$O(Mx(n!)^2+Mx(n!)^2n\log n)$

Table 4: Comparison of star and hypercube performances.

References

[1] S.B. Akers, D. Harel, and B. Krishnamurthy, "The Star Graph: An Attractive Alternative to the Hypercube", in *Proceedings of the International Conference on Parallel Processing*, St. Charles, IL, pp. 393-400, 1987.

[2] S.B. Akers, and B. Krishnamurthy, "The Fault Tolerance of Star Graphs", in *Proceedings of the International Conference on Supercomputing*, San Francisco, CA, pp. 270-276, 1987.

[3] S.B. Akers, and B. Krishnamurthy, "A Group Theoretic Model for Symmetric Interconnection Networks", *IEEE Transactions on Computers*, vol. c-38, no. 4, pp. 555-566, 1989.

[4] S.G. Akl, and K. Qiu, "A Novel Routing Scheme on the Star and Pancake Networks and its Applications", *Parallel Computing*, vol. 19, no. 1, pp. 95-101, 1993.

[5] P. Berthomé, A. Ferreira, and S. Perennes, "Decomposing Hierarchical Cayley Graphs, with Applications to Information Dissemination and Algorithm Design", Technical Report no. 92-32, Laboratoire de l' Informatique du Parallélism, Ecole Normale Superiéure de Lyon, Lyon, France, 1992.

[6] D.P. Bertsekas, and J.N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Englewood Cliffs, NJ, Prentice Hall, 1989.

[7] D.P. Bertsekas, C. Ozveren, G.D. Stamoulis, P. Tseng, and J.N. Tsitsiklis, "Optimal Communication Algorithms for Hypercubes", *Journal of Parallel and Distributed Computing*, vol. 11, pp. 263-275, 1991.

[8] K. Day, and A. Tripathi, "A Comparative Study of Topological Properties of Hypercubes and Star Graphs," to appear in the *IEEE Transaction on Parallel and Distributed Systems.*

[9] M. Dietzfelbinger, S. Madhavapeddy, and I.H. Sudborough, "Three Disjoint Path Paradigms in Star Networks", in *Proceedings of the IEEE Symposium on Parallel and Distributed Processing*, Dallas, TX, pp. 400-406, 1991.

[10] P. Fragopoulou, "Parallel Algorithms for the Fourier and Other Mathematical Transforms," Master's Thesis, Department of Computing and Information Science, Queen's University, Kingston, ON, 1990.

[11] P. Fragopoulou, and S.G. Akl, "A Parallel Algorithm for Computing Fourier Transforms on the Star Graph," in *Proceedings of the International Conference on Parallel Processing*, St. Charles, IL, vol. III, pp. 100-106, 1991, to appear in the *IEEE Transactions on Parallel and Distributed Systems.*

[12] P. Fragopoulou, and S.G. Akl, "Optimal Communication Algorithms on the Star Interconnection Network", to appear in the *Proceedings of the fifth IEEE Symposium on Parallel and Distributed Processing*, Dallas, TX, 1993.

[13] P. Fragopoulou, and S.G. Akl, "Optimal Communication Algorithms on Star Graphs Using Spanning Tree Constructions", Technical Report no. 93-346, Department of Computing and Information Science, Queen's University, Kingston, ON, 1993, to appear in the *Journal of Parallel and Distributed Computing.*

[14] P. Fraigniaud, E. Lazard, "Methods and Models of Communication in Usual Networks" Technical Report, Laboratoire de l' Informatique du Parallélism, Ecole Normale Superiéure de Lyon, Lyon, France, 1991.

[15] P. Fraigniaud, and C.T. Ho, "Arc Disjoint Spanning Trees on Cube Connected Cycles Network", in *Proceedings of the International Conference on Parallel Processing*, St. Charles, IL, vol. I, pp. 225-229, 1991.

[16] P. Fraigniaud, "Fault-tolerant Gossiping on Hypercube Multicomputers", in *Proceedings of the EDMCC2*, Munchen, France, pp. 463-472, 1991.

[17] L. Gargano, U. Vaccaro, and A. Vozella, "Fault Tolerant Routing in the Star and Pancake Interconnection Network", Dipartimento di Informatica ed Applicazioni, Universita di Salerno, Baronissi, SA, Italy.

[18] S.L. Johnson, and C.T. Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes", *IEEE Transactions on Computers*, vol. c-38, no. 9 , pp. 1249-1268, 1989.

[19] J. Jwo, S. Lakshmivarahan, and S.K. Dhall, "Characterization of Node Disjoint (parallel) Paths in Star Graphs", in *Proceedings of the Fifth International Parallel Processing Symposium*, pp. 404-409, 1991.

[20] D.E. Knuth, *The Art of Computer Programming*, Addison-Wesley, Reading, MA, 1973.

[21] F.T. Leighton, *Complexity Issues in VLSI: Optimal Layouts for the Shuffle-Exchange and Other Networks*, MIT Press, 1983.

[22] V.E. Media, and D. Sarkar, "Optimal Broadcasting on the Star Graph", *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 4, pp. 389-396, 1992.

[23] 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*, St. Charles, IL, vol. III, pp. 1-8, 1990.

[24] M. Nigam, S. Sahni, and B. Krishnamurthy, "Embedding Hamiltonians and Hypercubes in Star Interconnection Graphs", in *Proceedings of the International Conference on Parallel Processing*, St. Charles, IL, pp. 340-343, 1990.

[25] K. Qiu, S.G. Akl, and I. Stojmenović, "Data Communication and Computational Geometry on the Star and Pancake Networks", in *Proceedings of the IEEE Symposium on Parallel and Distributed Processing*, Dallas, TX, 1991.

[26] K. Qiu, P. Fragopoulou, and S.G. Akl, "On the Tree Structure of the Star Graph", Technical Report no. 93-349, Department of Computing and Information Science, Queen's University, Kingston, ON, 1993.

[27] K. Qiu, and S.G. Akl, "On the Properties of Breadth First Spanning Tree of the Star Graph", Technical Report no. 93-350, Department of Computing and Information Science, Queen's University, Kingston, ON, 1993.

[28] P. Ramanathan, and K.G. Shin, "Reliable Broadcast in Hypercube Multicomputers", *IEEE Transaction on Computers*, vol. c-37, no. 12, pp. 1654-1657, 1988.

[29] S. Sur, and P.K. Srimani, "A Fault Tolerance Routing Algorithm in Star Graphs", Technical Report no. 90-108, Department of Computer Science, Colorado State University, Ft. Collins, CO, 1990.

Appendix

Lemma 6: The EDT_{I_n} has the following characteristics:

1. Subtrees $T_l^{EDT_{I_n}}$, $2 \leq l \leq n$, are all edge-disjoint.
2. Subtree $T_{r(l)}^{EDT_{I_n}}$ is a rotation of subtree $T_l^{EDT_{I_n}}$, $2 \leq l \leq n$ (its previous subtree cyclically).
3. For each node $i \in S_{n-1}^k$, $1 \leq k \leq n$, there are $n-1$ parallel paths of almost minimum lengths that lead to node I_n through EDT_{I_n}.
4. The depth of EDT_{I_n} is at most $\lfloor \frac{3(n-1)}{2} \rfloor + 4$.

Proof: We prove each part separately:

1. From the definition of the parent$^{EDT_{I_n}}$ function, we see that each node of S_n is connected to each different subtree $T_l^{EDT_{I_n}}$, $2 \leq l \leq n$, through a different neighboring node, and as an extension through a different one of its incident edges. This concludes that subtrees $T_l^{EDT_{I_n}}$, $2 \leq l \leq n$, are all edge-disjoint.
2. We have to prove that if node i of S_n has parent node j in subtree $T_l^{EDT_{I_n}}$, $2 \leq l \leq n$, then node $R(i)$, obtained from i by a rotation, has parent node $R(j)$ in subtree $T_{r(l)}^{EDT_{I_n}}$.

   From the definition of rotation we make the following observations:

(a) If node $i \in S_{n-1}^k$, then node $i' = R(i) \in S_{n-1}^{r(k)}$ (lemma 2, part 2).

(b) $p_{i'} = r(p_i)$.

If $i \in S_{n-1}^k$ and $i_1 \neq k$ then $p_i = i_1$. In this case $i' \in S_{n-1}^{r(k)}$ and $i'_1 = r(i_1) \neq r(k)$, and we conclude that $p_{i'} = i'_1 = r(i_1) = r(p_i)$.

If $i \in S_{n-1}^k$ and $i_1 = k$ then p_i is such that i_{p_i} is the first misplaced symbol cyclically to the right of symbol 1 (position k) in i (excluding i_1). In this case $i' \in S_{n-1}^{r(k)}$ and $i'_1 = r(i_1) = r(k)$, then $p_{i'}$ is such that $i'_{p_{i'}}$ is the first misplaced symbol cyclically to the right of symbol 1 (position $r(k)$) in i' (excluding i'_1). From the definition of p_i, $i_m = m$ for all m cyclically between positions k and p_i in i. From the definition of rotation $i'_{r(m)} = r(i_m) = r(m)$ for all $r(m)$ cyclically between positions $r(k)$ and $r(p_i)$ in i'. So the first misplaced symbol cyclically to the right of symbol 1 in i' is in position $r(p_i)$, and we conclude that $p_{i'} = r(p_i)$.

(c) $i'_{p_{i'}} = i'_{r(p_i)} = r(i_{p_i})$.

In what follows we use the notation $k' = r(k)$ and $l' = r(l)$. We now show how the parent nodes of i and i' (j and j' respectively) in subtrees l and l', respectively, can be obtained from the definition of the parent^{EDT} function. For clarity we distinguish again among different kinds of nodes.

(a) If $i \in S_{n-1}^k$ with $i_{p_i} = k$, then $i' \in S_{n-1}^{k'}$ with $i'_{p_{i'}} = k'$ ($i'_{p_{i'}} = r(i_{p_i}) = r(k) = k$).

$$\text{parent}^{EDT}(i,l,I_n) = \begin{cases} j = (i_{p_i} i_2 \ldots i_n), & \text{if } l = k, \\ j = (1 i_2 \ldots i_n), & \text{if } l = i_1, \\ j = (l i_2 \ldots i_n), & \text{otherwise} \end{cases} \Rightarrow \text{parent}^{EDT}(i',l',I_n) = \begin{cases} j' = (i'_{p_{i'}} i'_2 \ldots i'_n), & \text{if } l' = k', \quad (l' = r(l) = r(k) = k') \\ j' = (1 i'_2 \ldots i'_n), & \text{if } l' = i'_1, \quad (i'_1 = r(i_1) = r(l) = l') \\ j' = (l' i'_2 \ldots i'_n), & \text{otherwise} \end{cases}$$

(b) If $i \in S_{n-1}^k$ with $i_1 \neq k$ and $i_{p_i} \neq k$, then $i' \in S_{n-1}^{k'}$ with $i'_1 \neq k'$ ($i'_1 = r(i_1) \neq r(k) = k$) and $i'_{p_{i'}} \neq k'$ ($i'_{p_{i'}} = r(i_{p_i}) \neq r(k) = k$).

$$\text{parent}^{EDT}(i,l,I_n) = \begin{cases} j = (i_{p_i} i_2 \ldots i_n), & \text{if } l = k, \\ j = (1 i_2 \ldots i_n), & \text{if } l = i_1, \\ j = (k i_2 \ldots i_n), & \text{if } l = i_{p_i}, \\ j = (l i_2 \ldots i_n), & \text{otherwise} \end{cases} \Rightarrow \text{parent}^{EDT}(i',l',I_n) = \begin{cases} j' = (i'_{p_{i'}} i'_2 \ldots i'_n), & \text{if } l' = k', \quad (l' = r(l) = r(k) = k') \\ j' = (1 i'_2 \ldots i'_n), & \text{if } l' = i'_1, \quad (i'_1 = r(i_1) = r(l) = l') \\ j' = (k' i'_2 \ldots i'_n), & \text{if } l' = i'_{p_{i'}}, \quad (i'_{p_{i'}} = r(i_{p_i}) = r(l) = l') \\ j' = (l' i'_2 \ldots i'_n), & \text{otherwise} \end{cases}$$

(c) If $i \in S_{n-1}^k$ with $i_1 = k$, then $i' \in S_{n-1}^{k'}$ with $i'_1 = k'$ ($i'_1 = r(i_1) = r(k) = k'$).

$$\text{parent}^{EDT}(i,l,I_n) = \begin{cases} j = (i_{p_i} i_2 \ldots i_n), & \text{if } l = k, \\ j = (1 i_2 \ldots i_n), & \text{if } l = i_{p_i}, \\ j = (l i_2 \ldots i_n), & \text{otherwise} \end{cases} \Rightarrow \text{parent}^{EDT}(i',l',I_n) = \begin{cases} j' = (i'_{p_{i'}} i'_2 \ldots i'_n), & \text{if } l' = k', \quad (l' = r(l) = r(k) = k') \\ j' = (1 i'_2 \ldots i'_n), & \text{if } l' = i'_{p_{i'}}, \quad (i'_{p_{i'}} = r(i_{p_i}) = r(l) = l') \\ j' = (l' i'_2 \ldots i'_n), & \text{otherwise} \end{cases}$$

(d) If $i \in S_{n-1}^1$, $i \neq I_n$, then $i' \in S_{n-1}^1$ and $i' \neq I_n$.

$$\text{parent}^{EDT}(i,l,I_n) = (j = i_l i_2 \ldots i_n) \quad \Rightarrow \quad \text{parent}^{EDT}(i',l',I_n) = (j' = i'_{l'} i'_2 \ldots i'_n)$$

We should now prove that $j' = R(j)$. Another way to prove this is that if edge (i,j) is of dimension d then edge (i',j') is of dimension $r(d)$. We have the following cases:

(a) $j = i_{p_i} i_2 .. i_n$, and $j' = i'_{p_{i'}} i'_2 ... i'_n$.
Edge (i, j) is of dimension p_i and edge (i', j') is of dimension $p_{i'}$. However $p_{i'} = r(p_i)$ and we conclude that $j' = R(j)$.

(b) $j = 1 i_2 ... i_n$, and $j' = 1 i'_2 ... i'_n$.
The dimension of edge (i, j) is k because $i \in S^k_{n-1}$. The dimension of edge (i', j') is k' because $i' \in S^{k'}_{n-1}$. But $k' = r(k)$ and we conclude that $j' = R(j)$.

(c) $j = k i_2 ... i_n$, and $j' = k' i'_2 ... i'_n$.
The dimension of edge (i, j) is d so that $i_d = k$. The dimension of (i', j') is d' such that $i'_{d'} = k'$. From the definition of rotation we know that $i'_{d'} = k' = r(k) = r(i_d) = i'_{r(d)} \Rightarrow d' = r(d)$ and we conclude that $j' = R(j)$.

(d) $j = l i_2 ... i_n$, and $j' = l' i'_2 ... i'_n$.
The proof is the same as in (c).

3. In what follows we characterize the paths that lead from node i of S_n to I_n through each of the subtrees $T_l^{EDT_{I_n}}$, $2 \le l \le n$. For clarity we distinguish again among different kinds of nodes. The labels above the arrows show which parts of the parent^{EDT} function could be applied each time (definition 8).

(a) For node $i \in S^k_{n-1}$, $2 \le k \le n$, with $i_{p_i} = k$, the paths have the following forms:

$i \overset{(1)}{\rightarrow} i_{p_i} i_2 ... i_n \overset{(4)}{\rightarrow}$ shortest path through nodes S^k_{n-1} of $T_k^{EDT_{I_n}} \overset{(4)}{\rightarrow} I_n$
$i \overset{(2)}{\rightarrow} 1 i_2 ... i_n \overset{(11)}{\rightarrow}$ shortest path through nodes $S^{i_1}_{n-1}$ of $T_{i_1}^{EDT_{I_n}} \overset{(4)}{\rightarrow} I_n$
$i \overset{(3)}{\rightarrow} l i_2 ... i_n \overset{(2,8)}{\longrightarrow} 1 i_2 ... i_n \overset{(11)}{\rightarrow}$ shortest path through nodes S^l_{n-1} of $T_l^{EDT_{I_n}} \overset{(4)}{\rightarrow} I_n \quad (l \ne i_1, l \ne (i_{p_i} = k))$

All of these paths are parallel since they go through different substars S_{n-1} of S_n and of minimum lengths.

(b) For nodes $i \in S^k_{n-1}$, $2 \le k \le n$, with $i_1 = k$, the paths have the following forms:

$i \overset{(4)}{\rightarrow} i_{p_i} i_2 ... i_n \overset{(7)}{\rightarrow}$ shortest path through nodes S^k_{n-1} of $T_k^{EDT_{I_n}} \overset{(4)}{\rightarrow} I_n$
$i \overset{(5)}{\rightarrow} 1 i_2 ... i_n \overset{(11)}{\rightarrow}$ shortest path through nodes $S^{i_{p_i}}_{n-1}$ of $T_{i_{p_i}}^{EDT_{I_n}} \overset{(4)}{\rightarrow} I_n \quad (i$ not adjacent to $I_n)$
$i \overset{(6)}{\rightarrow} l i_2 ... i_n \overset{(2,8)}{\longrightarrow} 1 i_2 ... i_n \overset{(11)}{\rightarrow}$ shortest path through nodes S^l_{n-1} of $T_l^{EDT_{I_n}} \overset{(4)}{\rightarrow} I_n \quad (l \ne i_{p_i}, l \ne (i_1 = k))$

All of these paths are parallel since they go through different substars S_{n-1} of S_n and of minimum lengths.

(c) For node $i \in S^k_{n-1}$, $2 \le k \le n$, with $i_1 \ne k$ and $i_{p_i} \ne k$, the paths have the following forms:

$i \overset{(7)}{\rightarrow} i_{p_i} i_2 ... i_n \overset{(1,7)}{\longrightarrow}$ shortest path through nodes S^k_{n-1} of $T_k^{EDT_{I_n}} \overset{(4)}{\rightarrow} I_n$
$i \overset{(8)}{\rightarrow} 1 i_2 ... i_n \overset{(11)}{\rightarrow}$ shortest path through nodes $S^{i_1}_{n-1}$ of $T_{i_1}^{EDT_{I_n}} \overset{(4)}{\rightarrow} I_n$

The path through neighbor $i' = k i_2 ... i_n$ of i has one of the following forms:
If $i_{p_i} = i'_{p_{i'}}$ then (parent nodes of i and i' start with the same symbol)

$i \overset{(9)}{\rightarrow} k i_2 ... i_n \overset{(5)}{\rightarrow} 1 i_2 ... i_n \overset{(11)}{\rightarrow}$
$\hookrightarrow$ shortest path through nodes $S^{i_{p_i}}_{n-1}$ of $T_{i_{p_i}}^{EDT_{I_n}} \overset{(4)}{\longrightarrow} I_n \quad$ (path 1)

If $i_{p_i} \neq i'_{p_{i'}}$ then (parent nodes of i and i' start with different symbols)

$$i \overset{(9)}{\rightarrow} ki_2...i_n \overset{(6)}{\rightarrow} i_{p_i}i_2...i_n \overset{(2,8)}{\longrightarrow} 1i_2...i_n \overset{(11)}{\rightarrow}$$
$$\hookrightarrow \text{shortest path through nodes } S_{n-1}^{i_{p_i}} \text{ of } T_{i_{p_i}}^{EDT_{I_n}} \overset{(4)}{\longrightarrow} I_n \quad \text{(path 2)}$$

Finally,

$$i \overset{(10)}{\rightarrow} li_2...i_n \overset{(2,8)}{\rightarrow} 1i_2...i_n \overset{(11)}{\rightarrow} \text{shortest path through nodes } S_{n-1}^{l} \text{ of } T_{l}^{EDT_{I_n}} \overset{(4)}{\rightarrow} I_n \quad (l \neq i_1, i_{p_i}, k)$$

All these paths are parallel since they go through different substars S_{n-1} of S_n and are of minimum possible lengths. Path 2 has length two more than path 1 which is the minimum length path that goes through nodes $S_{n-1}^{i_{p_i}}$ of $T_{i_{p_i}}^{EDT_{I_n}}$. However in some cases path 2 must be used to guarantee that all subtrees are edge-disjoint.

(d) Finally, for nodes $i \in S_{n-1}^1$, $i \neq I_n$, the paths have the following forms:

$$i \overset{(11)}{\rightarrow} li_2...i_n \overset{(1,4,7)}{\longrightarrow} \text{shortest path through nodes } S_{n-1}^{l} \text{ of } T_{l}^{EDT_{I_n}} \overset{(4)}{\rightarrow} I_n$$

All of these paths are parallel since they go through different substars S_{n-1} of S_n and of minimum lengths.

4. The depth of SPT_{I_n} is $\lfloor \frac{3(n-2)}{2} \rfloor + 1$. This is because each of the subtrees $T_k^{SPT_{I_n}}$, $2 \leq k \leq n$, is a shortest path tree in S_{n-1}^k and $\lfloor \frac{3(n-2)}{2} \rfloor$ is the diameter of S_{n-1} [1]. The depth of SPG_{I_n}, which is defined as an extension of SPT_{I_n}, is $\lfloor \frac{3(n-2)}{2} \rfloor + 2$ for obvious reasons. The EDT_{I_n} is constructed as an extension of SPG_{I_n}. From part 3 of this lemma, each node of S_{n-1}^k, $2 \leq k \leq n$, is connected to I_n through subtree $T_l^{EDT_{I_n}}$ $(k \neq l)$ with a path of the form $i \rightarrow \ldots \rightarrow 1i_2...i_n \rightarrow \ldots \rightarrow I_n$. From part 3 of this lemma we know that the path from i to $1i_2...i_n$ has length at most three (path 2). So the depth of EDT_{I_n} is less or equal to the depth of $SPG_{I_n}+3 = \lfloor \frac{3(n-2)}{2} \rfloor + 2 + 3 \leq \lfloor \frac{3(n-1)}{2} \rfloor - 1 + 2 + 3 \leq \lfloor \frac{3(n-2)}{2} \rfloor + 4$. This depth is almost optimal because it is greater only by one from the best up to now estimate of the fault diameter of S_n [2] which is the lower bound for the depth of EDT_{I_n}.

Form this lemma we notice that not only the EDT_{I_n} is of almost minimum depth but that the $n-1$ parallel paths that lead from each one of the nodes of S_n to I_n are also of minimum length except in one case. In this case $n-2$ of the paths are of minimum length and one of the paths (path 2) has length two more than the corresponding minimum path. This structure leads to algorithms that are optimal not only in terms of communication time, but also in term of message transmissions as we will see in section 4. □.