

# A Star-Based I/O-Bounded Network for Massively Parallel Systems

*Shahram Latifi*

Department of Electrical and Computer Engineering  
University of Nevada, Las Vegas – Las Vegas, NV 89154-4026

email: latifi@jb.ee.unlv.edu

Phone: (702) 895-4016 FAX: (702) 895-4075

*Marcelo Moraes de Azevedo\* and Nader Bagherzadeh*

Department of Electrical and Computer Engineering  
University of California, Irvine – Irvine, CA 92717

email: mazevedo@ece.uci.edu, nader@ece.uci.edu

Phone: (714) 824-8720 FAX: (714) 824-2321

**Abstract** — *This paper describes a new interconnection network for massively parallel systems, referred to as star-connected cycles (SCC). The SCC graph presents an I/O-bounded structure that results in several advantages over variable-degree graphs like the star and the hypercube.*

*The description of the SCC graph given in this paper includes issues such as labeling of nodes, degree, diameter and symmetry. The paper also presents an optimal routing algorithm for the SCC and efficient broadcasting algorithms with  $O(n)$  running time, with  $n$  being the dimensionality of the graph. A comparison with the cube-connected cycles (CCC) and other interconnection networks is included, indicating that, for even  $n$ , an  $n$ -SCC and a CCC of similar sizes have about the same diameter. Also, we show that one-port broadcasting in an  $n$ -SCC graph can be accomplished with a running time better than or equal to that required by an  $n$ -star containing  $(n - 1)$  times fewer nodes.*

**Index terms** — *Broadcasting, Fixed-degree graphs, I/O-bounding, Interconnection networks, Parallel processing, Routing, Star graph.*

## 1 Introduction

Over the past years, many interesting graphs such as the star [1] and the hypercube [2], [3] have been proposed as interconnection networks for parallel processing applications. Some important properties shown by these graphs are node and edge symmetry, hierarchical structure, and high resilience [4], [5]. However, the star graph is asymptotically superior to the hypercube, presenting a sublogarithmic growth of the degree and diameter as a function of the number of processors in the graph [6].

Most graphs studied so far offer a high processor density and a reduced diameter. Nevertheless, graphs such as the star and the hypercube present a variable-degree structure and have low scalability from the viewpoint of network growth. Other disadvantages posed by variable-degree interconnection networks are

---

\*This research was supported in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq - Brazil), under the grant No. 200392/92-1.

the large number of I/O communication ports required at each processor and the complexity of the physical lay-out.

As an alternative to overcome these difficulties, we propose a new type of interconnection network: the *star-connected cycles (SCC)* graph [7]. The SCC offers an I/O-bounded, fixed-degree structure and can be viewed as a variant of its counterpart, the cube-connected cycles or CCC [8]. The SCC and CCC graphs are formed by connecting cycles or rings of nodes via a particular network communication topology. The underlying topology used to connect the cycles in an  $n$ -SCC graph is an  $n$ -star, while that of the  $n$ -CCC graph is the  $n$ -cube.

In this paper, we define the SCC graph and present issues such as labeling of nodes, diameter and symmetry. It is shown that although the diameter of the SCC graph is greater than that of a star and a hypercube of similar size, it is possible to select a proper combination of communication techniques in the links that yields low message transmission delays.

We present one-port and multiple-port broadcasting algorithms supporting message pipelining and show that one-port broadcasting in the  $n$ -SCC can be accomplished very efficiently, resulting in a running time better than or equal to that of an  $n$ -star graph containing  $(n - 1)$  times fewer nodes. Also included in this paper is an optimal routing algorithm for the SCC graph. Finally, we compare the SCC with the hypercube, the star and CCC graphs and show that the SCC is an interesting alternative for parallel systems.

## 2 Description of the SCC

An  $n$ -SCC graph is obtained by replacing each node of an  $n$ -star with a ring of  $(n - 1)$  nodes, namely a *supernode*. The  $i$ th dimension link originally connected to a node of the star is referred to as a *lateral link* and is connected to the  $i$ th node of the corresponding ring in the SCC graph. The connections between nodes inside the same supernode are referred to as *local links*. Figure 1 shows a 4-SCC graph. Although not shown in Figure 1, we assume that an  $i$ th dimension lateral link is labeled  $i$ .

The nodes in each ring are identified by a pair of labels  $\langle i, \pi \rangle$ , where  $2 \leq i \leq n$  and  $\pi$  is a permutation of  $n$  digits. Then two nodes  $\langle i, \pi \rangle$  and  $\langle i', \pi' \rangle$  are connected by a link  $(\langle i, \pi \rangle, \langle i', \pi' \rangle)$  in the SCC graph iff either

1.  $(\langle i, \pi \rangle, \langle i', \pi' \rangle)$  is a local link, i.e.  $\pi = \pi'$  and  $\min(|i - i'|, n - 1 - |i - i'|) = 1$ , or
2.  $(\langle i, \pi \rangle, \langle i', \pi' \rangle)$  is a lateral link, i.e.  $i = i'$  and  $\pi$  differs from  $\pi'$  only in the first and  $i$ th digits, such that  $\pi(1) = \pi'(i)$  and  $\pi(i) = \pi'(1)$ .

An  $n$ -SCC graph contains  $(n - 1)n!$  nodes,  $(n - 1)n!$  local links and  $(n - 1)n!/2$  lateral links. The SCC is a vertex-symmetric graph with degree  $\delta = 3$  for  $n \geq 4$ .

The existence of two different types of links in the SCC graph causes it to be not edge-symmetric. However, every lateral link in an SCC graph is edge-symmetric with any other lateral link in the graph. The local links within a ring or supernode are also transitive with any other local link in the graph.

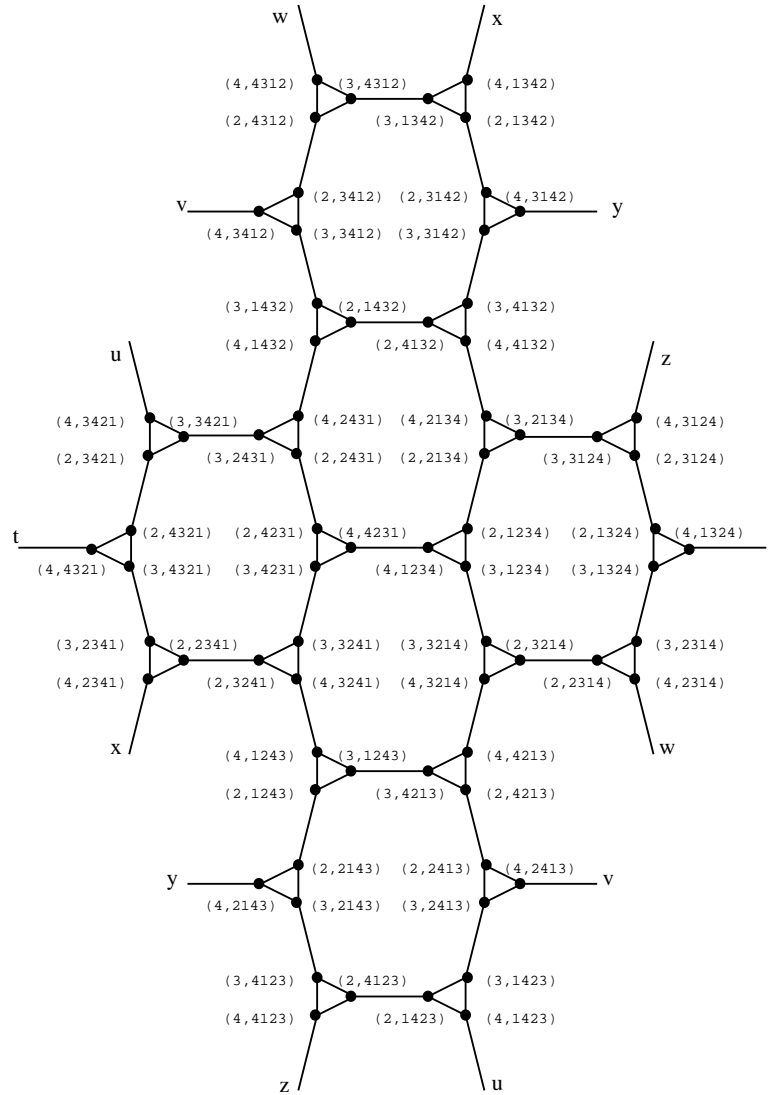


Figure 1: 4-SCC graph

### 3 Routing in the SCC graph

#### 3.1 Basic considerations

Routing between two nodes  $\langle i_s, \pi_s \rangle$  and  $\langle i_d, \pi_d \rangle$  in an SCC graph is equivalent to routing from  $\langle i_s, \pi_{ds} \rangle$  to  $\langle i_d, \pi_1 \rangle$ , where  $\pi_{ds} = \pi_d^{-1} \pi_s$  and  $\pi_1 = 123 \dots n$  [9], [7]. In particular, we note that routing from supernode  $\pi_s$  to supernode  $\pi_d$  (or, equivalently, from supernode  $\pi_{ds}$  to supernode  $\pi_1$ ) can be accomplished with the same routing techniques that have been proposed for the star graph [1], [6].

We may organize the digits of permutation  $\pi_{ds}$  as a set of cycles – i.e., cyclically ordered sets of digits with the property that each digit's desired position is that occupied by the next digit in the set. For example, a permutation  $\pi_{ds} = 23154$  labeling a supernode of a 5-SCC graph can be represented in cyclic format by  $(1\ 2\ 3)(4\ 5)$ . This cyclic representation has the particularity that the cycles can be listed in any order. In

addition, any cyclic shift of the digits within a cycle can be used without affecting the unique permutation label of  $\pi_{ds}$ . For example, (1 2 3)(4 5), (1 2 3)(5 4), (2 3 1)(4 5), (2 3 1)(5 4), (3 1 2)(4 5), and (3 1 2)(5 4) are all valid cyclic representations of permutation  $\pi_{ds} = 23154 \in 5\text{-SCC}$ . For simplicity, we assume in this paper that all cycles are written in canonical form, i.e. the smallest digit appears first in each cycle. However, the cycles can be listed in any order.

A cycle containing  $r$  digits is referred to as an  $r$ -cycle. Note that any digit already in its correct position in  $\pi_{ds}$  appears as a 1-cycle. Let  $C_i \in \pi_{ds}$  be an  $r$ -cycle of the form  $C_i = (a_1 a_2 \dots a_r)$ ,  $2 \leq r \leq n$ . The execution of  $C_i$  corresponds to a path  $R_i$  in the quotient  $n$ -star embedded in the  $n$ -SCC graph and can be expressed as a sequence of lateral links as follows [7], [10]:

- If  $a_1 = 1$ :

$$R_i = (a_2, a_3, \dots, a_r) \quad (1)$$

- If  $a_1 \neq 1$ :

$$R_i = (a_{1+k \bmod r}, a_{1+(k+1) \bmod r}, \dots, a_{1+(k+r-1) \bmod r}, a_{1+k \bmod r}), \quad \text{for } 0 \leq k \leq r-1 \quad (2)$$

The execution of each cycle  $C_i \in \pi_{ds}$  is part of a route containing a minimum number of lateral links between the source and the destination supernodes. For example, one possible path  $R_{ds}$  from supernode  $\pi_{ds} = 23154 = (1\ 2\ 3)(4\ 5)$  to supernode  $\pi_1 = 12345 = (1)(2)(3)(4)(5)$  in a 5-SCC graph is the sequence of lateral links (2, 3, 4, 5, 4):

$$23154 \xrightarrow{2} 32154 \xrightarrow{3} 12354 \xrightarrow{4} 52314 \xrightarrow{5} 42315 \xrightarrow{4} 12345$$

The minimum number of lateral links required in the routing between supernodes  $\pi_{ds}$  and  $\pi_1$  is [1]:

$$|R_{ds}| = \begin{cases} c + m, & \text{if the first digit in } \pi_{ds} \text{ is 1} \\ c + m - 2, & \text{if the first digit in } \pi_{ds} \text{ is not 1,} \end{cases} \quad (3)$$

where  $c$  is the number of cycles of length at least 2 in  $\pi_{ds}$  and  $m$  is the total number of digits in these cycles.

Note that according to the order chosen to execute the cycles in  $\pi_{ds}$  different paths result. Furthermore, the cycles do not necessarily have to be executed independently (i.e., it is possible to interleave the execution of the cycles in  $\pi_{ds}$ ). Some possible paths from  $\pi_{ds} = 23154 = (1\ 2\ 3)(4\ 5)$  to  $\pi_1 = 12345 = (1)(2)(3)(4)(5)$  are (2, 3, 4, 5, 4), (2, 3, 5, 4, 5), (4, 5, 4, 2, 3), (5, 4, 5, 2, 3), (2, 4, 5, 4, 3), and (2, 5, 4, 5, 3).

The order chosen to execute the cycles in permutation  $\pi_{ds}$  affects the number of local links that have to be traversed in the corresponding path in the SCC graph. The number of local links that is required to execute a particular  $r$ -cycle  $C_i = (a_1 a_2 \dots a_r) \in \pi_{ds}$ ,  $r \geq 2$ , is:

$$loc(C_i) = \begin{cases} \sum_{j=2}^r d(a_{j-1}, a_j) + d(a_r, a_1), & \text{if } a_1 \neq 1 \\ \sum_{j=3}^r d(a_{j-1}, a_j), & \text{if } a_1 = 1, \end{cases} \quad (4)$$

where the function  $d(i, j) = \min(|i - j|, n - 1 - |i - j|)$  represents the minimum cost required to traverse a supernode between its  $i$ th and  $j$ th nodes.

Note that if  $a_1 \neq 1$  Equation 4 holds for any of the  $r$  different sequences  $R_i$  that may be used to execute  $C_i$  (Equation 2). This observation immediately follows from the cyclic nature of these sequences.

The total cost of a path  $P(\ell_1 \mapsto \ell_f)$  from  $\langle i_s, \pi_s \rangle$  to  $\langle i_d, \pi_d \rangle$  in an  $n$ -SCC graph along a sequence of  $f$  lateral links  $R(\ell_1 \mapsto \ell_f) = (\ell_1, \ell_2, \dots, \ell_f)$  is

$$|P(\ell_1 \mapsto \ell_f)| = f + d(i_s, \ell_1) + \sum_{j=1}^{f-1} d(\ell_j, \ell_{j+1}) + d(\ell_f, i_d) \quad (5)$$

To illustrate the cost of local links in the routing, assume routing between nodes  $\langle 3, 34125 \rangle$  and  $\langle 2, 12345 \rangle$  in a 5-SCC. A path along the sequence  $R(2 \mapsto 3) = (2, 4, 2, 3)$  contains 4 lateral links and 7 local links ( $|P(2 \mapsto 3)| = 11$ ). However, if the sequence of lateral links  $R(3 \mapsto 2) = (3, 2, 4, 2)$  is used, a path with 4 lateral links and 5 local links results ( $|P(3 \mapsto 2)| = 9$ ).

### 3.2 Optimal routing algorithm

We now present an optimal routing algorithm that provides the shortest path between a pair of nodes  $\langle i_s, \pi_s \rangle$  and  $\langle i_d, \pi_d \rangle$  in an SCC graph. As stated earlier, this is equivalent to routing from  $\langle i_s, \pi_{ds} \rangle$  to  $\langle i_d, \pi_1 \rangle$ , where  $\pi_{ds} = \pi_d^{-1} \pi_s$  and  $\pi_1 = 123 \dots n$ .

The goal of the algorithm is to find a sequence of lateral links  $R(\ell_1 \mapsto \ell_f)$ , such that  $|P(\ell_1 \mapsto \ell_f)|$  (Equation 5) is minimal. The algorithm performs a depth-first search on a weighted tree structure. The tree is built by expanding at each step only those cycle orderings that seem to result in a minimal number of local links. Although the search tree can virtually examine all possible cycle orderings, including interleaved cycles, its size is significantly constrained in our algorithm. To guarantee that an optimal route is always found, backtracking is used to enable expansion of previous cycle orderings that seem to be better than the most recently expanded orderings.

In the following discussion, we use the term *vertex* to refer to an element of the search tree. In addition, we use the term *edge* to refer to the logical connection between vertices in the search tree, which is usually implemented with pointers or some form of indexing. The following data structures are stored within each vertex  $v_i$  of the search tree and are used by our routing algorithm:

- $\langle \ell_i, \pi_i \rangle$  – the label of the node reached so far by the routing algorithm. If the corresponding label used in the child of a vertex is  $\langle \ell'_i, \pi'_i \rangle$ , and  $\pi_i \neq \pi'_i$ , then the routing algorithm guarantees that supernode  $\pi'_i$  is one lateral link closer to the destination supernode ( $\pi_1$ ) and one lateral link farther from the source supernode ( $\pi_{ds}$ ) than is supernode  $\pi_i$ .
- $B_i$  – a subset of the digits of  $\pi_i$ , such that:

- If  $(1 a_2 a_3 \dots a_r)$  is an  $r$ -cycle of  $\pi_i$ ,  $2 \leq r \leq n$ , then  $a_2 \in B_i$  and  $1, a_3, \dots, a_r \notin B_i$ .

- If  $(a_1 a_2 \dots a_r)$  is an  $r$ -cycle of  $\pi_i$ ,  $2 \leq r \leq n$ , such that  $a_1, a_2, \dots, a_r \neq 1$ , then  $a_1, a_2, \dots, a_r \in B_i$ .

The digits in  $B_i$  represent all possible lateral links that can be selected by the routing algorithm while expanding the search tree from a given vertex  $v_i$ . For convenience, we define a function to generate  $B_i$  from  $\pi_i$ . Let this function be referred to as *bdigits*, such that  $B_i = \text{bdigits}(\pi_i)$ .

- $F_i$  – a subset of the digits of  $\pi_i$ , such that:

- If  $(1 a_2 a_3 \dots a_r)$  is an  $r$ -cycle of  $\pi_i$ ,  $2 \leq r \leq n$ , then  $a_r \in F_i$  and  $1, a_2, \dots, a_{r-1} \notin F_i$ .
- If  $(a_1 a_2 \dots a_r)$  is an  $r$ -cycle of  $\pi_i$ ,  $2 \leq r \leq n$ , such that  $a_1, a_2, \dots, a_r \neq 1$ , then  $a_1, a_2, \dots, a_r \in F_i$ .

The digits in  $F_i$  represent all lateral links that can be possibly selected by the routing algorithm to enter supernode  $\pi_1$  (i.e., all possible cycle orderings that can be selected by the routing algorithm from a given vertex  $v_i$  necessarily end with a lateral link  $\ell_f \in F_i$ ). For convenience, we define a function to generate  $F_i$  from  $\pi_i$ . Let this function be referred to as *fdigits*, such that  $F_i = \text{fdigits}(\pi_i)$ .

- $L_i$  – the number of local links used so far by the routing algorithm in the path from  $\langle i_s, \pi_{ds} \rangle$  to  $\langle \ell_i, \pi_i \rangle$ .
- $M_i$  – an estimate of the minimum number of local links that may be needed by the routing algorithm to reach node  $\langle i_d, \pi_1 \rangle$  from node  $\langle i_s, \pi_{ds} \rangle$ , using the path already constructed by the algorithm up to the intermediate node  $\langle \ell_i, \pi_i \rangle$ . For convenience, we use a function dubbed as *minloc* to compute  $M_i$  from  $L_i, \ell_i, \pi_i, i_d$ . Such a function is defined as:

$$M_i = \text{minloc}(L_i, \ell_i, \pi_i, i_d) = L_i + \min(d(\ell_i, b_i)) + \sum_{C_i \in \pi_i} \text{loc}(C_i) + \min(d(f_i, i_d)), \quad (6)$$

where  $b_i \in B_i$ ,  $B_i = \text{bdigits}(\pi_i)$ , and  $f_i \in F_i$ ,  $F_i = \text{fdigits}(\pi_i)$ .

Note that *minloc* is computed under the optimistic assumption that the route from  $\langle \ell_i, \pi_i \rangle$  to  $\langle i_d, \pi_1 \rangle$  selects the best possible lateral links in the sets  $B_i$  and  $F_i$ . In addition, the summation term used to compute the number of local links that are required to execute all cycles  $C_i \in \pi_i$  (see Equation 4) assumes that an optimal cycle ordering requiring no local links to move from one cycle to the next can be found by the routing algorithm.

- $e_i$  – an enable/disable bit that indicates whether the tree should continue to be expanded from vertex  $v_i$  or not.

In addition, the tree structure generated by the optimal routing algorithm has the following characteristics:

- The number of levels in the search tree is at most  $|R_{ds}| + 2$ , with  $|R_{ds}|$  being given by Equation 3. We number these levels from 0 to  $|R_{ds}| + 1$ , starting from the root level.

- Let  $v_i$  be the parent of a vertex  $v'_i$  in the search tree. We refer to the data stored in  $v_i$  and  $v'_i$  as  $\{\langle \ell_i, \pi_i \rangle, B_i, F_i, L_i, M_i, e_i\}$  and  $\{\langle \ell'_i, \pi'_i \rangle, B'_i, F'_i, L'_i, M'_i, e'_i\}$ , respectively. The weight of the edge connecting  $v_i$  to  $v'_i$  corresponds to the number of local links that are required to route from  $\langle \ell_i, \pi_i \rangle$  to  $\langle \ell'_i, \pi'_i \rangle$  in the  $n$ -SCC graph and is given by  $d(\ell_i, \ell'_i) = \min(|\ell_i - \ell'_i|, n - 1 - |\ell_i - \ell'_i|)$ . Hence,  $L'_i = L_i + d(\ell_i, \ell'_i)$ . Note that routing from  $\langle \ell_i, \pi_i \rangle$  to  $\langle \ell'_i, \pi'_i \rangle$  also requires one lateral link if  $\pi_i \neq \pi'_i$ , and zero lateral links otherwise. Since the total number of lateral links that must be traversed in the route from  $\langle i_s, \pi_{ds} \rangle$  to  $\langle i_d, \pi_1 \rangle$  can be computed in advance (Equation 3), the routing algorithm focuses on accounting for the local links only.
- The root vertex is initialized with  $\langle \ell_i, \pi_i \rangle = \langle i_s, \pi_{ds} \rangle$ ,  $B_i = \text{bdigits}(\pi_{ds})$ ,  $F_i = \text{fdigits}(\pi_{ds})$ ,  $L_i = 0$ ,  $M_i = \text{minloc}(0, i_s, \pi_{ds}, i_d)$  and  $e_i = \text{ON}$ . All vertices located at level  $|R_{ds}| + 1$  in the tree have  $\langle \ell_i, \pi_i \rangle = \langle i_d, \pi_1 \rangle$ ,  $B_i = F_i = \emptyset$  and  $M_i = \text{minloc}(L_i, i_d, \pi_1, i_d) = L_i$ . All vertices located at level  $|R_{ds}|$  in the tree have  $\langle \ell_i, \pi_i \rangle = \langle \ell_f, \pi_1 \rangle$  (with  $\ell_f$  being the lateral link used to enter supernode  $\pi_1$ ),  $B_i = F_i = \emptyset$ , and  $M_i = \text{minloc}(L_i, \ell_f, \pi_1, i_d) = L_i + d(\ell_f, i_d)$ .
- The backtracking mechanism is triggered by comparing the estimated minimum number of local links ( $M_i$ ) stored in the most recently generated child vertices with a global variable referred to as  $T$ . This variable is updated whenever a backtracking procedure occurs, meaning that the minimum number of local links that is required in the route from  $\langle i_s, \pi_{ds} \rangle$  to  $\langle i_d, \pi_1 \rangle$  is actually greater than the previous value of  $T$ . As expected, the search for an optimal route becomes more selective as  $T$  increases, which not only limits the width of the search tree but also makes the backtracking mechanism less likely to be triggered again.

Given the definitions above, the optimal routing algorithm for the SCC graph follows :

Algorithm 1 (Optimal routing in the SCC graph):

1. Compute permutation  $\pi_{ds}$  from the labels of the source ( $\langle i_s, \pi_s \rangle$ ) and destination ( $\langle i_d, \pi_d \rangle$ ) nodes, such that  $\pi_{ds} = \pi_d^{-1} \pi_s$ .
2. Create a root vertex with  $\langle \ell_i, \pi_i \rangle = \langle i_s, \pi_{ds} \rangle$ ,  $B_i = \text{bdigits}(\pi_{ds})$ ,  $F_i = \text{fdigits}(\pi_{ds})$ ,  $L_i = 0$ ,  $M_i = \text{minloc}(0, i_s, \pi_{ds}, i_d)$ , and  $e_i = \text{ON}$ .
3. Initialize  $T$  with the value  $T = \text{minloc}(0, i_s, \pi_{ds}, i_d)$ .
4. Generate child vertices for all enabled vertices, such that the label  $\ell'_i$  for each child corresponds to exactly one of the digits stored in the set  $B_i$  of each parent vertex. Set  $e_i = \text{OFF}$  at each recently expanded parent vertex. Also, obtain permutation  $\pi'_i$  for each child vertex by swapping the 1st and the  $\ell'_i$ th digits of the corresponding permutation stored in the parent vertex ( $\pi_i$ ), and make  $B'_i = \text{bdigits}(\pi'_i)$ ,  $F'_i = \text{fdigits}(\pi'_i)$ ,  $L'_i = L_i + d(\ell_i, \ell'_i)$ ,  $M'_i = \text{minloc}(L'_i, \ell'_i, \pi'_i, i_d)$ .

Enabled vertices located at level  $|R_{ds}|$  of the search tree must be expanded similarly. However, they generate a single child with  $\ell'_i = i_d$ ,  $\pi'_i = \pi_i$ ,  $B'_i = \emptyset$ ,  $F'_i = \emptyset$ ,  $L'_i = L_i + d(\ell_i, i_d)$ ,  $M'_i = L'_i$ .

In any case, a child vertex is enabled with  $e'_i = \text{ON}$  if  $M'_i \leq T$ . Otherwise, we set  $e'_i = \text{OFF}$ .

5. If one of the recently generated child vertices has  $\langle \ell'_i, \pi'_i \rangle = \langle i_d, \pi_1 \rangle$  and  $e'_i = \text{ON}$ , then an optimal route has already been found. The optimal sequence of lateral links  $R(\ell_1 \mapsto \ell_f)$  can be obtained in reverse order by backing up towards the root of the tree and listing the value of the digit  $\ell_i$  stored in each vertex located between the  $|R_{ds}|$ th and the 1st levels. Once  $R(\ell_1 \mapsto \ell_f)$  has been obtained, exit the algorithm.
6. If all the recently generated child vertices characterized by  $e'_i = \text{ON}$  have  $\langle \ell'_i, \pi'_i \rangle \neq \langle i_d, \pi_1 \rangle$ , go to Step 4.
7. If none of the recently generated child vertices has been enabled, do a backtracking search in the tree. Among all existing leaf vertices, select those with the smallest value of  $M_i$  and set  $T$  to this value. Also, enable the selected nodes and go to Step 5.

An example of a search tree built by the optimal routing algorithm is given in Figure 2.

### 3.3 Complexity analysis of the algorithm

#### Space complexity

The amount of memory required to hold the variables contained in each vertex of the search tree is  $O(n)$ . The height of the search tree is also  $O(n)$ , since its maximum value is  $d_{star} + 2 = \lceil 3(n-1)/2 \rceil + 2$ , where  $d_{star}$  is the diameter of the quotient  $n$ -star graph embedded in the  $n$ -SCC [1].

A worst-case analysis of the width of the search tree can be done under the following pessimistic assumption: considering that all possible orderings of cycles in permutation  $\pi_{ds}$  are examined by the routing algorithm, the lowest level in the search tree would have at most  $m!$  vertices. This is justified by the fact that there are at most  $m!$  possible ways to move the  $m$  misplaced digits in  $\pi_{ds}$  to their correct positions, using the minimum number of lateral links given by Equation 3. Hence, under this worst-case scenario the space complexity of the optimal routing algorithm is  $O(m!n^2)$ .

However, the actual space requirements of the algorithm are far more modest, due to the constraints placed on the number of expanded vertices by the heuristics of the algorithm (i.e., the estimated minimum number of local links  $M_i$ ), which is quite selective. Simulations carried out for  $4 \leq n \leq 9$  reveal that a very small number of vertices is enabled at each step, which makes the maximum width of the tree virtually proportional to  $m$ . A typical example that illustrates this behavior is given in Figure 2. Considering this reasoning, we estimate that the space complexity of our optimal routing algorithm is  $O(mn^2)$ , in the average case.

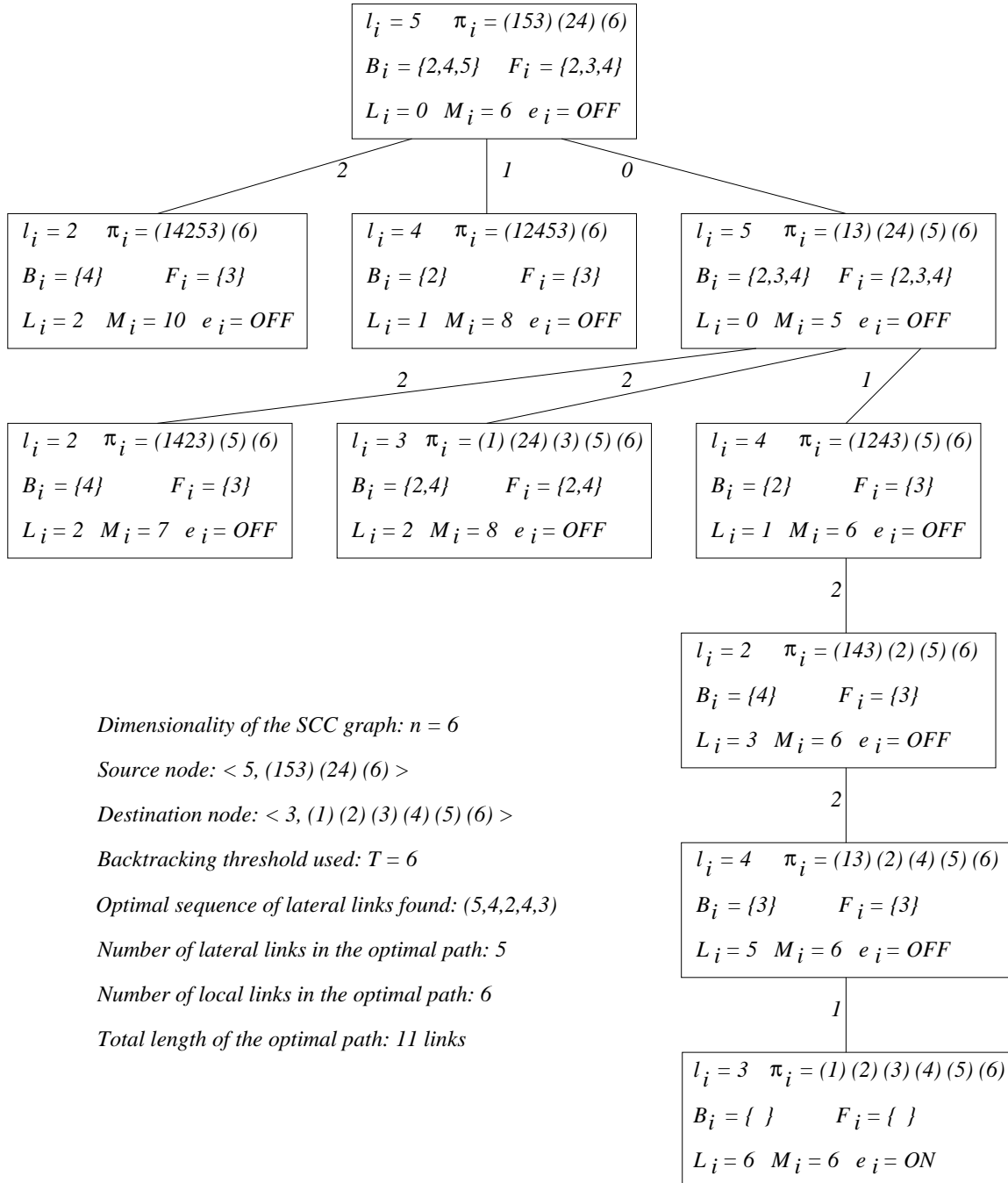


Figure 2: Example of search tree used for optimal routing in the SCC

### Time complexity

The main computations that must be performed upon creation of a vertex of the search tree refer to  $B'_i$ ,  $F'_i$  and  $M'_i$ . Fortunately, each of these computations can be accomplished in  $O(1)$  time by using the corresponding values  $B_i$ ,  $F_i$  and  $M_i$  that are stored in the parent vertex, and taking into account the differences in the cycle structures of permutations  $\pi_i$  and  $\pi'_i$ .

A straightforward extension of the reasoning used in the worst-case analysis of the space complexity of the algorithm results in a worst-case time complexity of  $O(m!n)$ . As expected, such time requirements were not observed during simulations of the algorithm. The potential need for backtracking searches in the tree, added to fact that the maximum width of the tree is in practice proportional to  $m$ , results in a time complexity of  $O(mn^2)$ , on the average.

### 3.4 Implementation aspects of the optimal routing algorithm

One possible approach that can be used in an implementation of the proposed routing algorithm consists of having the source node to compute the optimal sequence of lateral links to be used in the path to the destination node. Such a sequence can be used in the header of the message, making the algorithm usable with different communication mechanisms (e.g., packet-switching, circuit-switching and virtual cut-through routing) [11].

## 4 Diameter

The diameter of the  $n$ -SCC graph can be calculated from its *antipodes*. An antipode is the farthest node from a given node along the shortest path [10]. In particular, we consider the antipodes of an *identity* node  $\langle i_1, \pi_1 \rangle$  when deriving the diameter of the SCC graph. We recall that in the  $n$ -SCC graph there are  $(n - 1)$  nodes  $\langle i, \pi_1 \rangle$ , where  $2 \leq i \leq n$  and  $\pi_1 = 123 \dots n$ . We must therefore choose one of these  $(n - 1)$  nodes as the identity node for the  $n$ -SCC graph. Let such a node be  $\langle i_1, \pi_1 \rangle = \langle 2, 123 \dots n \rangle$ . Now, we define an antipode  $\langle i_a, \pi_a \rangle$  in the  $n$ -SCC graph as follows:

- $\pi_a$  is a permutation chosen such that node  $\langle i_a, \pi_a \rangle$  is located  $\lfloor 3(n - 1)/2 \rfloor$  lateral links away from  $\langle i_1, \pi_1 \rangle$ , while keeping the number of local links in the path to the identity node to a maximum.
- $i_a$  is a digit chosen so as to include a maximum number of additional local links in the path to  $\langle i_1, \pi_1 \rangle$ .

**Lemma 1:** The following nodes are antipodes in the  $n$ -SCC graph:

- For odd  $n$ :

$$\langle 2, (1)(2 \ a+1)(3 \ a+2) \dots (a-1 \ n-1)(a \ n) \rangle, \quad \text{where } a = (n + 1)/2 \quad (7)$$

- For even  $n$ :

$$\langle 2, (1 \ b+1)(2 \ b+2) \dots (b-1 \ n-1)(b \ n) \rangle, \quad \text{where } b = n/2 \quad (8)$$

Reference [13] gives a proof of correctness for these antipodes.

**Theorem 1** *The diameter of the  $n$ -SCC,  $n \geq 3$ , is :*

$$d_{SCC} = \begin{cases} 6, & \text{for } n = 3 \\ \frac{1}{2}(n^2 + n - 4), & \text{for even } n \\ \frac{1}{2}(n^2 + 3n - 8), & \text{for odd } n \geq 5 \end{cases} \quad (9)$$

*Proof:* The diameter of the 3-SCC graph can be obtained by inspection. We recall that since the 3-star is a hexagon [1], replacing each node of a 3-star with 2 connected nodes results in a dodecagon (i.e., a 3-SCC graph), whose diameter is 6. The case  $n \geq 4$  is considered in the remaining of the proof.

Routing from any of the above antipodes requires  $d_{star} = \lfloor 3(n-1)/2 \rfloor$  lateral links. The number of local links can be calculated from the terms  $d_{loc}(1)$ ,  $d_{loc}(2)$  and  $d_{loc}(3)$  as follows:

- $d_{loc}(1)$  – permutation  $\pi_a$  has  $\lfloor (n-1)/2 \rfloor$  cycles  $(p\ q)$  that do not include digit 1. Execution of each of these cycles requires  $2d(p, q) = 2\lfloor (n-1)/2 \rfloor$  local links (Equation 4). Thus, the total number of local links required for execution of all cycles in  $\pi_a$  that do not include digit 1 is  $d_{loc}(1) = 2(\lfloor (n-1)/2 \rfloor)^2$ .
- $d_{loc}(2)$  – permutation  $\pi_a$  has  $\lfloor n/2 \rfloor$  cycles of length 2 that must be executed in the route to the identity node. The cycles in  $\pi_a$  may be ordered such that only one local link is required to move between the execution of adjacent cycles. This adds  $d_{loc}(2) = \lfloor n/2 \rfloor - 1$  local links to the routing from the antipode to the identity.
- $d_{loc}(3)$  – digit  $i_a$  in the antipode is such that at most  $d_{loc}(3) = \lfloor n/2 \rfloor - 1$  local links may be added to the routing.

Hence, the diameter of the  $n$ -SCC graph,  $n \geq 4$ , is

$$d_{SCC} = d_{star} + d_{loc}(1) + d_{loc}(2) + d_{loc}(3) = 2 \left( \left\lfloor \frac{n-1}{2} \right\rfloor \right)^2 + \left\lfloor \frac{3(n-1)}{2} \right\rfloor + 2 \left\lfloor \frac{n}{2} \right\rfloor - 2 \quad (10)$$

This result can be written in terms of even and odd values of  $n$  as shown in Equation 9.  $\square$

## 5 Broadcasting in the SCC graph

A broadcasting algorithm for the SCC graph consists of a sequence of transmissions over lateral and local links, such that a particular piece of information originated by a node is passed on to all other processors in the interconnection network.

At each step of the algorithm, every node communicates with at least one of its neighbors and compares notes on whether either of them has already received the information that is being broadcast. If only one node has received it, then additional communication takes place to relay the broadcast information to the

uninformed node. If both or none of the nodes have already received the information, then no additional messages are exchanged between the nodes. We assume that node comparison between any pair of nodes is accomplished with full-duplex (i.e., bidirectional) communication links, both in the case of lateral and local links.

Broadcasting algorithms can be based on two distinct communication models, namely *one-port* communication or *multiple-port* communication. In the one-port communication model, each node sends messages using only one port at each step of the algorithm. With this scheme, the number of informed nodes can at most double at each step of the algorithm. Therefore, broadcasting in a graph with  $N$  nodes using a one-port communication algorithm requires at least  $\log N$  steps<sup>1</sup>.

In a multiple-port communication model, each node sends messages using two or more ports at each step. Assuming a regular graph with  $N$  nodes and degree  $\delta$ , a broadcasting algorithm based on an  $m$ -port communication model ( $1 \leq m \leq \delta$ ) requires at least  $\log_{(m+1)} N$  steps.

## 5.1 One-port broadcasting

Broadcasting in an SCC graph is an extension of the broadcasting algorithms already introduced for the star [1], [12]. The approach used in [1], for example, consists of finding a sequence of lateral links  $\sigma$ , such that for every pair of nodes in the graph there exists a subsequence of lateral links that forms a path from one node to the other. As long as  $\sigma$  satisfies this condition, it constitutes a one-port broadcasting algorithm in the star graph. This approach can be extended to the SCC graph by interleaving  $\sigma$  with a proper choice of local links, such that the information is correctly broadcast among the nodes inside each supernode.

Ideally, we should find an  $O(\log N) = O(n \log n)$  broadcasting sequence for the  $n$ -SCC. However, if we recall that the diameter of the  $n$ -SCC contains a quadratic term (Equation 9), the following theorem holds:

**Theorem 2** *One-port broadcasting in an  $n$ -SCC graph requires a sequence with  $\Omega(n^2)$  steps.*

*Proof:* The proof follows from the observation that any broadcasting sequence must include subsequences allowing the node originating the broadcast message to communicate with all other nodes in the graph. Clearly, the broadcasting sequence must be at least as long as the longest communication path existing in an  $n$ -SCC graph (i.e., the diameter). If we recall that the diameter of the  $n$ -SCC is  $\Theta(n^2)$  (Equation 9), the theorem follows.  $\square$

With that limitation in mind, different possible broadcasting sequences have been investigated [13]. The general approach for analyzing a broadcasting sequence for the SCC graph is done in two steps. First, we must consider how many lateral link steps each sequence requires, and then we evaluate the number of local link steps. In any case, the sequence of lateral links required by the broadcasting algorithm is defined by the quotient  $n$ -star graph embedded in the  $n$ -SCC.

---

<sup>1</sup>All logarithms in this paper are base 2, unless otherwise indicated.

An analysis of broadcasting sequences presented in [13] shows that an efficient algorithm can be obtained from a cyclic sequence of digits  $\sigma_c$ . Such a sequence is formed by repeating a cyclic pattern of the digits  $(2\ 3\ \dots\ n)$  as follows, with each digit representing a lateral link. Note that  $\sigma_c$  repeats the pattern  $(2\ 3\ \dots\ n)$   $d_{star}$  times, where  $d_{star}$  is the diameter of the quotient  $n$ -star embedded in the  $n$ -SCC graph. Therefore,  $\sigma_c$  includes all possible paths between any two supernodes in the  $n$ -SCC, and as long as a proper sequence of local links is also chosen,  $\sigma_c$  can be used as a broadcasting algorithm for the  $n$ -SCC graph.

$$\sigma_c = 2\ 3\ 4\ \dots\ (n-1)\ n\ 2\ 3\ 4\ \dots\ (n-1)\ n\ \dots\ (d_{star}\ \text{times}) \quad (11)$$

Although the length of the above sequence is  $O(n^2)$ ,  $\sigma_c$  can actually run in  $O(n)$  lateral link steps in an  $n$ -star graph by using parallel transmissions in those links:

**Theorem 3** *The use of a  $\delta$ -port communication model in an  $n$ -star graph with degree  $\delta = n-1$  and diameter  $d_{star} = \lceil 3(n-1)/2 \rceil$  yields an optimal broadcasting algorithm requiring only  $d_{star}$  steps.*

*Proof:* At each step, a node running a  $\delta$ -port broadcasting algorithm uses all its ports to propagate the information to be broadcast. Suppose that at the beginning of the algorithm, only node  $\pi_i$  holds the information. After the first step of the algorithm, all nodes within a distance of one lateral link from  $\pi_i$  will also have received the information. After the second step, the information has been propagated to all nodes within two lateral links of  $\pi_i$ , and so on. After  $d_{star}$  steps, all nodes in the  $n$ -star graph have received the broadcast information.

Optimality results from the observation that no broadcasting algorithm can use a shorter sequence of steps than the longest distance between any pair of nodes in the  $n$ -star (i.e., the diameter). Since exactly  $d_{star}$  steps are used, the algorithm is optimal.  $\square$

Note that the technique described in Theorem 3 for the  $n$ -star graph results in a multiple-port algorithm with  $O(n)$  steps, while a one-port broadcasting algorithm in the same graph requires  $O(n \log n)$  steps [1], [12].

A main disadvantage of using such a  $\delta$ -port algorithm in a star graph is that we may impose severe communication overhead on the nodes. The algorithm may even be difficult to implement or require special hardware support for  $\delta$ -port communication, specially in large degree star graphs.

These restrictions do not apply to the  $n$ -SCC graph, since the task of simultaneous communication over the lateral links is equally distributed over  $(n-1)$  nodes belonging to the same supernode. Therefore, we may take advantage of parallel transmissions in the lateral links to implement a faster broadcasting algorithm in the  $n$ -SCC graph.

An analysis of sequence  $\sigma_c$  shows that such a sequence can be efficiently executed by using all lateral links of each supernode simultaneously. To illustrate this reasoning, Figure 3 shows the initial steps required to run  $\sigma_c$  in a  $\theta$ -SCC graph. This approach allows a fast execution of  $\sigma_c$ , since it initially broadcasts the

information inside the supernode and then uses all lateral links simultaneously to pass the information on to adjacent supernodes. The full broadcasting algorithm requires this operation to be repeated  $d_{star}$  times. Note that we still have a one-port broadcasting algorithm, since at each step every node tries to compare notes using only one communication port. Of course, the nodes may also receive a communication request in a second port while transmitting in another port. Another interesting observation is that this technique actually runs  $\delta = (n - 1)$  lateral link steps of  $\sigma_c$  in parallel, therefore reducing the number of steps that would be required if we used only one lateral link per supernode at a time by a factor of  $(n - 1)$ .

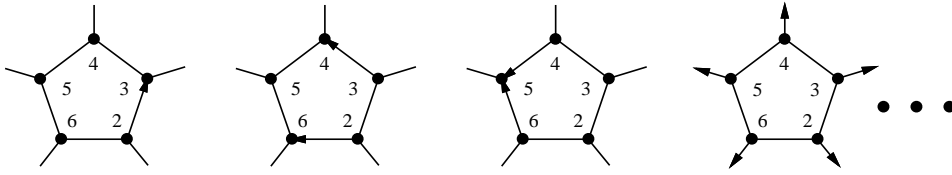


Figure 3: One-port broadcasting in a  $\theta$ -SCC (cyclic sequence)

Note that the concept of parallelism is also used inside each supernode by forcing one of the nodes to use different local links in the first two steps of a subsequence of local link transmissions (Figure 3). Each remaining node in the same supernode simply propagates the information using the same direction chosen by their informed neighbors. A total of  $\lfloor n/2 \rfloor$  local link steps is required to accomplish one-port broadcasting inside a supernode. The particular node that initiates the broadcasting in a supernode is either a node that originated a piece of information to be broadcast or a node that has just been informed of it via a lateral link. In this case, the node may be assumed as the first informed node in the ring, and therefore proceeds with transmissions over different local links in the next two steps.

**Theorem 4** *A one-port broadcasting algorithm for an  $n$ -SCC graph based on the cyclic sequence  $\sigma_c$  and using parallel transmissions over both lateral and local links requires  $d(\sigma_c)$  steps, where:*

$$d(\sigma_c) = \left\lfloor \frac{n+2}{2} \right\rfloor \left\lfloor \frac{3(n-1)}{2} \right\rfloor \quad (12)$$

*Proof:* The number of lateral link steps in  $\sigma_c$  is  $\sigma_c(lat) = d_{star} = \lfloor 3(n-1)/2 \rfloor$ . Also, the number of local link steps in  $\sigma_c$  is  $\sigma_c(loc) = \lfloor n/2 \rfloor d_{star} = \lfloor n/2 \rfloor \lfloor 3(n-1)/2 \rfloor$ . The total number of steps required to run sequence  $\sigma_c$  is then:  $d(\sigma_c) = \sigma_c(lat) + \sigma_c(loc) = \lfloor (n+2)/2 \rfloor \lfloor 3(n-1)/2 \rfloor$ .  $\square$

Due to space constraints, we do not present in this paper pseudo-code listings of our broadcasting algorithms. We refer the interested reader to [13] for a more formal description of our algorithms.

#### Message pipelining with the one-port broadcasting algorithm

Our one-port broadcasting algorithm can be easily extended to support pipelined broadcasting of  $B$  consecutive messages in the SCC graph. As depicted in Figure 3, after  $\lfloor (n+2)/2 \rfloor$  steps all nodes within a supernode have received the broadcast message and have sent it to each node's lateral link neighbor.

Following these steps, these nodes do not participate actively in the process of disseminating the broadcast message anymore, being able to start broadcasting a new message. With an initiation interval of  $\lfloor (n+2)/2 \rfloor$  steps between messages, broadcasting of  $B$  pipelined messages using a one-port communication model can be accomplished in  $\lfloor (n+2)/2 \rfloor \lfloor B-1+3(n-1)/2 \rfloor$  steps. This initiation interval does not increase the buffering capacity required at each node for proper execution of the broadcasting algorithm (i.e., both the pipelined and the non-pipelined versions of the algorithm require buffering of one message).

## 5.2 Multiple-port broadcasting

A multiple-port broadcasting algorithm for the SCC can be built as an extension of the previous one-port algorithm. Notably, an efficient multiple-port execution of  $\sigma_c$  can be achieved by using both local links of each node simultaneously while broadcasting the information inside each supernode. Such an approach reduces the number of local link steps that are required to broadcast the message inside each supernode to  $\lfloor (n-1)/2 \rfloor$ .

**Theorem 5** *A multiple-port broadcasting algorithm for an  $n$ -SCC graph based on the cyclic sequence  $\sigma_c$  and using parallel transmission over both lateral and local links requires a total of  $d(\sigma_c)_m$  steps, where:*

$$d(\sigma_c)_m = \left\lfloor \frac{n+1}{2} \right\rfloor \left\lfloor \frac{3(n-1)}{2} \right\rfloor \quad (13)$$

*Proof:* The proof is analogous to that of Theorem 4.

Accordingly, message pipelining can also be supported by a multiple-port broadcasting algorithm. If an initiation interval of  $\lfloor (n+1)/2 \rfloor$  steps between messages is used, broadcasting of  $B$  pipelined messages using a multiple-port communication model can be accomplished in  $\lfloor (n+1)/2 \rfloor \lfloor B-1+3(n-1)/2 \rfloor$  steps. The buffering capacity required in this case is the same for a non-pipelined execution of the algorithm, i.e. each node must store one message. The multiple-port communication model also allows the utilization of an initiation interval of 1 step between messages, which reduces the total number of steps required to broadcast  $B$  pipelined messages to  $\lfloor (n+1)/2 \rfloor \lfloor 3(n-1)/2 \rfloor + B-1$  steps. In this case, however, the buffering requirements at each node increase to  $\lfloor (n+1)/2 \rfloor$  messages.

## 5.3 $O(n)$ broadcasting algorithms for the SCC

The broadcasting algorithms presented so far require  $O(n)$  lateral link steps and  $O(n^2)$  local link steps. We can actually accomplish broadcasting in  $O(n)$  running time by making proper assumptions on the time spent by the algorithm on lateral and local link steps. As an example, assume that we want to have the total running time of a broadcasting algorithm equally divided between lateral and local link steps. This results in the following theorem:

**Theorem 6** *Both one-port and multiple-port broadcasting using sequence  $\sigma_c$  in an  $n$ -SCC have linear running time if the transmission rate in the local links is  $O(n)$  times faster than the transmission rate in the lateral links.*

*Proof:* Suppose that the transmission rates on the lateral and the local links are, respectively,  $TR(lat)$  and  $TR(loc)$ . If we want an even distribution of time over lateral and local link steps, then we may choose:

$$\frac{\sigma_c(lat)}{TR(lat)} = \frac{\sigma_c(loc)}{TR(loc)} \quad (14)$$

The ratio between  $TR(loc)$  and  $TR(lat)$  is then:

$$\frac{TR(loc)}{TR(lat)} = \begin{cases} \left\lfloor \frac{n}{2} \right\rfloor, & \text{for one-port broadcasting} \\ \left\lfloor \frac{n-1}{2} \right\rfloor, & \text{for multiple-port broadcasting} \end{cases} \quad (15)$$

As described in Section 6, use of higher bandwidth communication channels in the local links adds little complexity to the lay-out of the SCC graph, which makes the requirements on the transmission rates given by Equation 15 quite simple to obtain in practice. Note that for both the one-port and multiple-port cases, the result is that the time spent on a quadratic number of local link steps can be made equal to that spent on a linear number of lateral link steps, as long as the transmission rate in the local links is  $O(n)$  times faster than the transmission rate in the lateral links. If we suppose that the broadcasting algorithm spends most of the time transmitting data (i.e., the overhead or start-up time associated with the messages is small when compared to the time required to transmit them), then the resulting running time is linear with  $n$  and approximately equal to  $2d_{star}/TR(lat)$ .  $\square$

#### 5.4 Analysis of broadcasting algorithms for the SCC graph

The efficiency of broadcasting algorithms for interconnection networks can be measured by comparing the required number of steps with the diameter of the graph. We present such a comparison in Table 1 in terms of percentages. More specifically, we define the *relative distance to the diameter* as:

$$\begin{aligned} \frac{d(\sigma_c) - d_{SCC}}{d_{SCC}} \times 100\%, & \quad \text{for one-port broadcasting} \\ \frac{d(\sigma_c)_m - d_{SCC}}{d_{SCC}} \times 100\%, & \quad \text{for multiple-port broadcasting} \end{aligned} \quad (16)$$

Note that the total number of steps required by the proposed broadcasting algorithms is very close to the diameter of the SCC graph. We have shown that the proposed algorithms are optimal from the viewpoint of lateral link steps, and by inspecting Table 1 we note that optimality from the viewpoint of local link steps is very likely to have already been achieved.

Particularly in the case of multiple-port broadcast, note that for  $4 \leq n \leq 7$  the relative distance to the diameter varies from 0 to 16.1%. Note also that, for odd  $n$ , the one-port broadcasting algorithm performs

<i>Type of broadcasting algorithm</i>	<i>n</i>	<i>Size</i>	<i>Diameter</i>	<i>Lateral link steps</i>	<i>Local link steps</i>	<i>Total number of steps</i>	<i>Running time in lateral link steps</i>	<i>Relative distance to the diameter</i>
<i>One-port</i>	4	72	8	4	8	12	8	50%
	5	480	16	6	12	18	12	12.5%
	6	3600	19	7	21	28	14	47.4%
	7	30240	31	9	27	36	18	16.1%
<i>Multiple-port</i>	4	72	8	4	4	8	8	0%
	5	480	16	6	12	18	12	12.5%
	6	3600	19	7	14	21	14	10.5%
	7	30240	31	9	27	36	18	16.1%

Table 1: Performance figures of broadcasting algorithms in the SCC graph

as well as its multiple-port counterpart. However, for even  $n$  the number of steps required by the one-port broadcasting algorithm is about 50% greater than the diameter of the  $n$ -SCC graph. Therefore, it is more efficient to use multiple-port broadcasting in this case.

## 6 Comparison with other graphs

A comparison between different interconnection network graphs is shown in Table 2. The SCC graph offers a fixed-degree structure with bounded I/O requirements at the node level, which allows the construction of interconnection networks of different sizes with higher scalability in network growth than the hypercube and the star. In other words, processors with just 3 communication ports can be used to build any SCC graph. Variable-degree graphs such as the hypercube and the star require a growing number of communication ports at each processor as we increase the number of nodes in the graph. The result is increased complexity and higher pin count at each processor than that required by fixed-degree graphs.

One of the trade-offs of fixed-degree graphs is an increased diameter. However, due to the cluster-like nature of the supernodes, the SCC can be built with high bandwidth parallel buses in the local links, at the expense of a minimum increase in the lay-out complexity. The lay-out of the lateral links, however, is characterized by similar wiring constraints found in high-dimensional networks such as the star graph and the hypercube [14], which often have to be implemented with narrower bandwidth, serial communication links. Therefore, an  $n$ -SCC can present communication delays comparable to an  $n$ -star graph, if we consider that the delay in the local links is small in comparison to the delay in the lateral links. In addition, many practical algorithms present locality of operation and use short-distance communication patterns. This reduces even more the impact of longer communication paths on the performance of parallel computers.

Table 2 also shows another type of I/O-bounded interconnection network, namely the cube-connected cycles or CCC. An  $n$ -CCC graph can be built by replacing each node of an  $n$ -cube with a ring of  $n$  or more nodes. We consider in this paper only the case of  $n$ -CCC graphs containing  $n$  nodes per ring. The number

Graph	$n$	Topological properties			One-port broadcasting			
		Size	Degree	Diameter	Lateral link steps	Local link steps	Total number of steps	Running time in lateral link steps
$n$ -cube	7	128	7	7	7	-	7	7
	8	256	8	8	8	-	8	8
	9	512	9	9	9	-	9	9
$n$ -star	5	120	4	6	12	-	12	12
	6	720	5	7	16	-	16	16
	7	5040	6	9	20	-	20	20
$n$ -CCC	4	64	3	8	4	5	9	6.5
	5	160	3	10	5	7	12	8.5
	6	384	3	13	6	8	14	8.7
	7	896	3	15	7	10	17	10.3
	8	2048	3	18	8	11	19	10.8
$n$ -SCC	9	4608	3	20	9	13	22	12.3
	4	72	3	8	4	8	12	8
	5	480	3	16	6	12	18	12
	6	3600	3	19	7	21	28	14
	7	30240	3	31	9	27	36	18

Table 2: Comparison of interconnection network graphs

of nodes and diameter of an  $n$ -CCC graph formed under such a structure are respectively  $N = n2^n$  and  $d_{CCC} = 2n + \lfloor n/2 \rfloor - 2$  [15].

Compared to a CCC graph of similar size, the  $n$ -SCC graph presents about the same diameter for the cases where  $n$  is even. The diameter of the  $n$ -SCC graph shows a sharp discontinuity when  $n$  changes from an even to an odd value. Such a behavior is due to the presence of the term  $2(\lfloor (n-1)/2 \rfloor)^2$  in the expression of the diameter of the  $n$ -SCC graph (Theorem 1).

The diameter of the CCC compares favorably with that of the  $n$ -SCC graph for odd  $n$ . However, the underlying topology or quotient graph used to connect the cycles in the  $n$ -SCC (i.e., the  $n$ -star) has several advantages over that used in the CCC (i.e., the  $n$ -cube) [1], [6]. Among these advantages, we may cite a smaller degree from the viewpoint of the supernodes and a shorter average distance. Also, an  $n$ -SCC graph requires fewer lateral links and fewer nodes at each ring than does a CCC graph with similar number of nodes. Implementation of supernodes as single multiprocessor VLSI devices is therefore less complex in the case of the SCC graph.

Table 2 also compares the SCC with other graphs from the viewpoint of one-port broadcasting algorithms. Theorem 6 shows that the selection of a proper value for the ratio of the transmission rates in the local and lateral links of an  $n$ -SCC graph can reduce the running time of a broadcasting algorithm to about twice the time spent in the lateral link steps. This results in an  $O(n)$  running time, while a one-port broadcasting in an  $n$ -star has an  $O(n \log n)$  running time.

The values shown for one-port broadcasting in the  $n$ -star in Table 2 are optimal and have been extracted

from [12]. By inspecting Table 2, we note that one-port broadcasting in an  $n$ -SCC graph can be accomplished with running time better than or equal to that of an  $n$ -star containing  $(n - 1)$  times fewer nodes.

For  $n = 4$  and  $n = 6$ , the total number of steps required by the one-port broadcasting algorithm is about 50% greater than the diameter of the corresponding SCC graph (Table 1). For  $n = 5$  and  $n = 7$ , the total number of steps is at most 16.1% greater than the diameter. Therefore, the higher discontinuity observed in the diameter of the  $n$ -SCC graph for odd  $n$  is somehow compensated by the increased efficiency of one-port broadcasting algorithms, which may be beneficial to different parallel processing applications.

A comparison of one-port broadcasting algorithms for the SCC and the CCC graph is also presented in Table 2. A one-port broadcasting algorithm requiring  $\lceil 5n/2 \rceil - 1$  steps [16] is used for the CCC graph. Broadcasting in the CCC is initially accomplished with alternated transmissions on local and lateral links<sup>2</sup>. After  $2n - 1$  such steps, reception of the broadcast message by at least one node in every ring of the CCC is guaranteed. The remaining steps of the algorithm use only local link transmissions to ensure proper distribution of the broadcast message to all nodes in each ring of the CCC. A total of  $n$  lateral link steps and  $\lceil 3n/2 \rceil - 1$  local link steps is used by the algorithm.

For a fair comparison with the SCC graph, we also consider the possibility of having the transmission rate in the local links of the  $n$ -CCC to be  $O(n)$  times faster than the transmission rate in the lateral links. Specifically, we assume that the ratio between the transmission rates in local and lateral links of the  $n$ -CCC graph is  $\lfloor n/2 \rfloor$ .

One-port broadcasting is accomplished more efficiently in the CCC graph than in the SCC. Note in Table 2 that, for graphs of similar size, the CCC graph requires fewer steps than does the SCC. In addition, the CCC is superior to the SCC in respect to the running time required for one-port broadcasting, even when an  $O(n)$  times faster transmission rate is used in the local links of both graphs when compared to the transmission rate in the lateral links. The superiority of the CCC graph in respect to one-port broadcasting can be explained by the fact that the quotient graph of the CCC (i.e., the hypercube) allows for an optimal broadcasting algorithm [1], [16], [17] which can be nicely extended to the case of the CCC [16]. The same is not true in the case of the SCC and its quotient graph (i.e., the star).

Although multiple-port broadcasting figures are not depicted in Table 2, we recall that it is possible to obtain a performance improvement for the  $n$ -SCC graph if  $n$  is even (see Table 1). It is also likely that multiple-port communication may reduce the number of steps required for broadcasting in the CCC graph, but the amount of improvement should be small since one-port broadcasting in the CCC already requires only 1 or 2 steps more than the diameter of the graph. Therefore, use of multiple-port broadcasting can reduce the difference in performance between the CCC and the  $n$ -SCC for even  $n$ .

---

<sup>2</sup>By analogy with the SCC, *local links* in a CCC graph connect nodes inside a ring or supernode, while *lateral links* correspond to a dimension link of the quotient hypercube.

## 7 Conclusion

We have described in this paper a new interconnection network: the star-connected cycles or SCC. The SCC is an I/O-bounded graph and has been proposed as a variant of the previous cube-connected cycles or CCC.

Aspects such as labeling of nodes, degree, diameter and symmetry have been presented. We have also proposed an optimal routing algorithm for the SCC graph.

We introduced one-port and multiple-port broadcasting algorithms for the  $n$ -SCC and showed that an  $O(n)$  running time can be achieved by making the transmission rate in the local links  $O(n)$  times faster than the transmission rate in the lateral links. The proposed broadcasting algorithms can be easily extended to support message pipelining.

We have compared the SCC with variable-degree graphs such as the star graph and the hypercube. We claim that the SCC is an attractive alternative for parallel systems, since it overcomes some disadvantages of variable-degree graphs such as the high number of communication ports per node and the issue of scalability.

An apparent disadvantage of I/O-bounded interconnection networks is the larger diameter when compared to variable-degree graphs. Nevertheless, the SCC shows some symmetry properties that allow the selection of different communication techniques between its nodes. A proper selection of transmission rates can be done to reduce the communication delay to levels comparable to those obtained in an  $n$ -star. Particularly for one-port broadcasting, we have shown that the total running time of the algorithm in the  $n$ -SCC graph is better than or equal to that required by an  $n$ -star containing  $(n - 1)$  times fewer nodes.

We have also compared the SCC with another fixed-degree graph, namely the CCC graph. We have shown that the diameter of the  $n$ -SCC is approximately the same as that of a CCC graph containing a similar number of nodes whenever  $n$  is even. For odd  $n$ , the diameter of an  $n$ -SCC graph is slightly greater than that of a CCC of similar size. The CCC graph also exhibits some superiority from the viewpoint of broadcasting. However, implementation of the supernodes seems to be simpler in the case of the SCC graph since less nodes and lateral links per supernode are required in this case.

### Acknowledgements

We are particularly grateful to referee 1, who provided valuable comments to improve the presentation of this paper.

## References

- [1] S. B. Akers, D. Harel and B. Krishnamurthy, "The Star graph: An Attractive Alternative to the  $n$ -Cube," *Proceedings of the International Conference on Parallel Processing*, 1987, pp. 393-400.
- [2] Y. Saad and M. H. Schultz, "Topological Properties of Hypercubes," *IEEE Transactions on Computers*, Vol. 37, No. 7, July 1988, pp. 867-872.

Appears in *IEE Proceedings - Computers and Digital Techniques*, Vol. 142, No. 1, January 1995, pp. 5-14.

- [3] J. P. Hayes and T. Mudge, "Hypercube Supercomputers," *Proceedings of the IEEE*, Vol. 77, No. 12, December 1989, pp. 1829-1841.
- [4] M. S. Krishnamoorthy and B. Krishnamurthy, "Fault Diameter of Interconnection Networks," *Computers & Mathematics with Applications*, Vol. 13, No. 5/6, 1987, pp. 577-582.
- [5] S. Latifi, "On the Fault-Diameter of the Star Graph," *Information Processing Letters*, 46 (1993), pp. 143-150.
- [6] K. Day and A. Tripathi, "A Comparative Study of Topological Properties of Hypercubes and Star Graphs," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 1, January 1994, pp. 31-38.
- [7] S. Latifi, M. M. Azevedo and N. Bagherzadeh, "The Star-Connected Cycles: a Fixed-Degree Interconnection Network for Parallel Processing," *Proceedings of the International Conference on Parallel Processing*, 1993, Vol. 1, pp. 91-95.
- [8] F. P. Preparata and J. Vuillemin, "The Cube-Connected Cycles: A Versatile Network for Parallel Computation," *Communications of the ACM*, Vol. 24, No. 5, May 1981, pp. 300-309.
- [9] S. B. Akers and B. Krishnamurthy, "A Group-Theoretic Model for Symmetric Interconnection Networks," *IEEE Transactions on Computers*, Vol. 38, No. 4, April 1989, pp. 555-566.
- [10] S. Latifi, "Parallel Dimension Permutations on Star Graph," *IFIP Transactions A: Computer Science and Technology*, 1993, A23, pp. 191-201.
- [11] P. T. Gaughan and S. Yalamanchili, "Adaptive Routing Protocols for Hypercube Interconnection Networks," *Computer*, Vol. 26, No. 5, May 1993, pp. 12-23.
- [12] V. E. Mendia and D. Sarkar, "Optimal Broadcasting on the Star Graph," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 3, No. 4, July 1992, pp. 389-396.
- [13] M. M. Azevedo, N. Bagherzadeh and S. Latifi, *The Star-Connected Cycles: a Fixed-Degree Interconnection Network for Massively Parallel Systems*, Department of Electrical and Computer Engineering, University of California, Irvine, Technical Report ECE 93-02, March 1993.
- [14] W. J. Dally, "Performance Analysis of  $k$ -ary  $n$ -cube Interconnection Networks," *IEEE Transactions on Computers*, Vol. 39, No. 6, June 1990, pp. 775-785.
- [15] D. S. Meliksetian and C. Y. R. Chen, "Communication Aspects of the Cube-Connected Cycles," *Proceedings of the International Conference on Parallel Processing*, Vol. 1, 1990, pp. 579-580.
- [16] A. L. Liestman and J. G. Peters, "Broadcast Networks of Bounded Degree," *SIAM Journal on Discrete Mathematics*, Vol. 1, No. 4, November 1988, pp. 531-540.

Appears in *IEE Proceedings - Computers and Digital Techniques*, Vol. 142, No. 1, January 1995, pp. 5-14.

- [17] S. L. Johnsson and C. T. Ho, "Optimum Broadcasting and Personalized Communications in Hypercubes," *IEEE Transactions on Computers*, Vol. 38, No. 9, September 1989, pp. 1249-1268.