Ring Connected Star Networks

by

Duško S. Miloradović

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy

The City University of New York

1995

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

7/31/95 — Date

Azdak Ghosal — Chair of Examining Committee

8/1/95 — Date

Prof. Stanley Hobbs — Executive Officer

Dr. Theodor Brown

Dr. Frank Hsu

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

Ring Connected Star Networks

by

Duško S. Miloradović

Adviser: Professor S.A.Ghozati

A new topology for interconnection parallel systems is introduced and its static and dynamic properties are investigated. The topology, the Ring Connected Star Graphs, is defined as an enhancement of the Star Graph networks. The diameter of the network is shown to be smaller than that for the Hypercube or the Star graph of comparable size. Several multistage versions of this topology are also described and analyzed. Routing strategies for the network are proposed for both single stage and multistage implementations. Sufficient conditions for conflict free routing in the multistage version is determined. Necessary and sufficient conditions are given for different types of conflict free permutation routings and for single node broadcasting. Partitioning of any multistage Ring Connected Star Graph into a collection of independent subnetworks of the same type is also described. The containment properties of the network are investigated regarding to linear arrays, rings and spoke graphs. The number of parallel paths between any two nodes in the single stage implementation is determined.

ACKNOWLEDGMENTS

I would like to thank all these people who helped me to overcome the difficulties throughout this work, and to make it possible. First, I would like to acknowledge my adviser, Professor Seyed Ali Ghozati, for his invaluable technical guidance, assistance and encouragement during the development and completion of this document. His insights and experience were invaluable to me. He gave many helpful corrections and suggestions from the first draft to the final version.

I would also like to express my gratitude to Professor Stanley Habib for his guidance and suggestions. My thanks also to Professor Theodore Brown for his patience and time and Professor D. Frank Hsu for all his comments and useful advice.

I owe my many thanks to my parents who provided unwavering support and encouragement throughout my education.

Contents

List of Figures

1 Introduction

Since its introduction by Akers *et al.* in [1], the networks based on graphs of the symmetric groups have attracted a lot of attention. Being vertex and edge symmetric, strongly hierarchical and recursive, with a small diameter, these graphs are a very attractive alternative to some very popular topologies like the hypercube. In this thesis, we research properties of some new topologies which are based on the Star graph. The main effort will be focused on what we call the *Ring Connected Star Graph* (*RCS*). This has the features of the Star graph and in some aspects is similar to the Shuffle Exchange network. It should, thus, inherit some important and useful properties from both of its ancestors. The *RCS* is also a special case of a class of networks which we name *Starnets*. The Starnet is a network made of Star graphs, or in other words, a network which can be divided into a number of subnetworks, each of which is a Star graph.

In particular, our goal is research in the following topics:

- The principles of construction of star graph based networks and some other related topologies. The analysis of their properties using novel modeling technique based on finite automata theory.
- The development of distributed routing and broadcasting strategies for *RCS* and Star networks.
- Implementation of the Ring Connected Star graph as single stage and multi stage networks.
- The performances with regard to message routing complexity and fault toler-

ance.

- The development of algorithms for embedding of other topologies, such as rings,trees,spoke graphs and hypercubes.

In the first chapter we introduce the concept of multi-processor systems and parallel networks. We also describe basic terminology and define properties and characteristics of interconnection networks in general. Some of the more important types of interconnection networks are formally introduced.

In the next chapter we describe several formal models for networks and introduce a novel modeling approach based on finite automata theory. Some general results for different classes of bit and character controlled networks with emphasis on modeling of dynamical properties are presented.

In the chapter three we present and study a new topology, the Ring Connected Star Graph. Our research focuses on basic topological properties and routing problems for this interconnection network. The connection of RCS and the Star graph is established. Some results related to the genus of the graph are also proved. The main results in this chapter are: characterization of the Star graphs and the diameter of the Ring Connected Star Graphs. Several methods for routing in RCS are also developed.

The following chapter deals with the topic of multistage implementations for RCS networks. Several multistage versions of the Ring Connected Star Graph network are described. Routing permutations and broadcasting are examined. The conditions for conflict free paths are established for different types of point-to-point routing, permutations and broadcasts and some properties of parallel paths are

investigated. We also show how to partition the multistage Ring Connected Star graph into several subnetworks of the same type and smaller order.

Chapters four and five are about containment and fault tolerance properties of the introduced network. Using formalism introduced in chapter 2 embedding of linear arrays, rings, trees and spoke graphs is investigated. Here we also describe several applications for RCS based on containment properties of the network.

Finally some results are presented about fault tolerance of RCS in regard to some substructures like stars or rings and about parallel paths in single stage RCS networks.

In the final chapter we compare the Ring Connected Star Graph to the hypercube. We also list some of the more important questions that, in our opinion, should be further investigated.

The main results of this thesis are:

- Basic topological properties of RCS, diameter and routing are determined.
- Topological characterization of the Star graph and RCS.
- Implementation and properties of the multistage RCS. Characteristics and conditions for conflict free point-to-point routing, broadcasting and permutation on the network.
- Partitioning of the Multistage Ring Connected Star Graph.
- Embedding of Star graphs,rings and spoke graphs in RCS is defined.

- Algorithms for finding the maximal element and for the single node broadcasting in RCS and Star graphs are described.
- Some properties related to fault tolerance of RCS and existence of parallel paths.

2 Preliminaries

2.1 Parallel Processors and Networks

Term parallel processing is applied rather loosely to a number of more or less similar concepts in computer science. The essence of the idea is that we have more than one computer or process active at any given instant. It is also applied, sometimes, on systems in which we have a number of potentially active processes but with only one running at any given moment (based on some type of time sharing). However, in this thesis, this will not be considered parallel processing. We will always assume existence of more than one process running concurrently. Speaking about computer systems it is also possible to talk about parallelism at different levels. Following Hayes [23] we can think about parallelism at the gate level (e.g. parallel adders), at the register level (vector and pipeline computers) or at the processor level (system with multiple processing elements). In fact almost all computers have some kind of parallelism at the processor level, usually related to IO operations. In this thesis we will be concerned mostly with processor level parallelism, or in other words, using Flynn's classification [18] with *Single Instruction Multiple Data* (SIMD) and *Multiple Instruction Multiple Data* (MIMD) computers.

The most important problem for any parallel system is the question of cooperation and coordination. Though we could imagine a system with multiple processing elements working simultaneously and independently in solving real and practical problems we need to assume some links between processing elements of a parallel system. The system which connects individual processing element and allows transfer of data and information between them is termed *interconnection network*

Thus we describe *multiprocessor system* or *Parallel Random Access Machine* (PRAM) (see Figure 1.) as a system made of a number of *Processing Elements* (PE) a *control unit* (CU) and an *interconnection network* (IN).

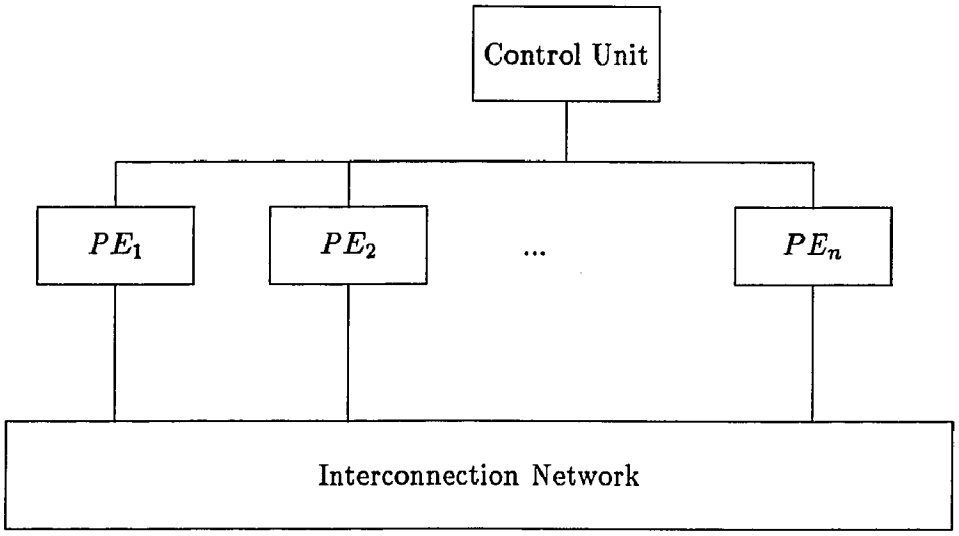

Figure 1: Multiprocessor system

The purpose of the control unit is to store the programs and controls the flow of instructions by sending signals one at a time to all PEs in the system.

Each PE is a processor (CPU) with its own memory. Terms node,point and vertex will be used as synonyms for processing elements. We assume that each PE contains a number of data and address registers $R_1, R_2, \ldots$ and also at least one *data transfer register* (DTR). DTRs are used to send and receive data from other PEs in the network. We can consider cases in which there is only one DTR used for both types of transfer or when we have separate input and output DTRs. In this way any data that has to be sent to another PE has to be moved from working registers into output DTR and any received data has to be fetched from input DTRs. Basically,

DTRs of a PE are assumed directly linked to DTRs of all PEs connected to the element. Also each PE can be in one of two states, *active* or *enabled* or *inactive* or *disabled.* A PE will execute any instruction it receives from the control unit only if it is in active state. Otherwise the instruction will not be performed, that is, inactive PEs retain the contents of their DTRs after a transfer step is done. In most cases we consider, all PEs will be active which means that all processing elements receive and send data during transfer stage. We will usually assume that local memory has constant size but also it can be a function of the size of the network, i.e. the number of processing element in it.

We define *edge* or *link* as a line which physically connects two or more elements of an interconnection network. Interconnection network is made of a set of links connecting PEs. All such networks can be roughly divided into two groups. A single line can be used by more than two PEs (only two and in one direction at a time). Example of this type of INs are different single and multi bus and crossbar networks. However, if no more than two PEs share a single link then we have *graph network* which we define as connected graphs. We will be concerned in this thesis only with graph networks so a network N is ordered pair:

$$N = (P, N)$$

where P , the set of vertices, is a set of processing elements and N, the set of edges, is a collection of communication links between pairs of PEs. The term graph will be used interchangeably with network or IN.

We will be assuming that the system works in a synchronous manner, that is, its operation is a sequence of steps

$$s_1, s_2, \ldots, s_k, \ldots$$

and each step s_i is a computation or a transfer. During a computation step each active PE performs some kind of computation utilizing only its local memory. Transfer steps represent communication between PEs in which some or all PEs send or receive data thru the network. In each transfer step data can be passed only among directly connected PEs. Therefore if two PEs are not linked directly we need to perform several transfer steps to move data from one PE to another. The system works in a synchronous manner in a sense that all active PEs are either in computation or transfer stage at any instance.

We will suppose that in most cases connections are bidirectional, that is, that transfer can be performed in both directions but in only one way at a time (i.e. half duplex). If the network graph is defined as a diagraph we have a simplex so data can go only in one predefined direction between any two PEs. The length of any edge is supposed to be 1 and it takes 1 unit of time for a transfer to take place between two directly linked vertices.

The basic unit of data, sent or received, is called a *packet*. Packets are strings of binary words which cannot be broken into pieces. That means that processors can receive and output only whole packets. A part of each packet is usually defined as a *key* which is used for routing purposes.

The described model of SIMD machine can be generalized in the form of *multiple* SIMD machine. This type of multi-processor system has a control unit that can be modified dynamically to operate as a collection of one or more SIMD machines. Each of these machines can be of different size and operates independently. This concepts assumes also the partitioning of the network into set of independent networks one for each virtual SIMD system. This has numerous advantages, particularly for better

fault detection and tolerance and also allows running of multiple parallel program simultaneously.

MIMD systems on the other hand are made of independent PEs each having its own program. The main difference is that processors operate in an asynchronous manner. This makes it necessary to perform synchronization before communication between PEs is possible. Basically we can consider MIMD machines to be collections of independent SIMD systems. It also possible to consider multiple MIMD/SIMD machines as collection of dynamically reconfigurable sets of virtual MIMD and SIMD systems.

2.2 Attributes of Interconnection Networks

In this section we will describe some of the more important properties of interconnection networks.

First is the maximal *degree* of the network. Degree of a node is the number of PEs directly connected to the node if our IN can be represented as a graph. We will be considering mostly symmetric graphs so in most networks the degree is the same for all vertices. The second parameter is *throughput* or the maximal number of PEs that can send data simultaneously during a transfer step. The maximal degree of a network N will be denoted $D(N)$ and throughput as $T(N)$.

For example if a network can be represented as a complete graph $N = K_n$ of n vertices $D(N) = n - 1$ and $T(N) = n$. For a system with a single bus and n PEs it is $D(N) = n - 1$ but $T(N) = 1$. If we have n PEs and k line multibus $T(N) = min(k, \lfloor n/2 \rfloor)$. For a ring made of n nodes R_n the values are $D(R_n) = 2$

and $T(R_n) = n$

Symmetry. Symmetry basically means node independence. We say that a network is *symmetric* if it "looks the same" from every node. With symmetric networks we can use identical processors since every node plays an identical role in the system. This property simplifies the design of algorithms because any node can be the starting node. So if we devise a routing function for a single PE we have in fact created it for the whole network. It is also useful in discovery of subnetworks. If for example we can embed a tree in a network starting with a node then we can do that at any other node. Symmetric networks also minimize vertex/edge congestion during message routing. From the topological viewpoint symmetry is tantamount to *regularity* of the graph representing the topology. That is, the degree of each node is equal. Formally a graph is *vertex symmetric* if there is an automorphism of the graph which for each two vertices which maps one into another. We similarly define *edge symmetric* graphs. That is, there is an automorphism of the set of edges. The edge symmetry ensures that communication load is uniformly distributed so there is no congestion at any one of links.

Single Stage and Multi Stage. There are two main type of interconnection networks based on the number of immediate links between processing units. If there is a direct connection between at least two PEs we term such a network *single stage network.* If between any two PEs we have one or more stages made of crossbar or similar switching elements we have a *multistage networks.* This naming is a bit misleading. Namely, not all nodes in a single stage network are directly connected and so any connection of two PEs can take in fact several steps (stages) using

intermediate PEs. On the other hand with multistage networks there are no PEs in any path between two nodes so disregarding switching elements we have a kind of single stage link between any two nodes. The main advantage of multistage networks is lower hardware cost. Most are built on $log_2 N$ stages where N is the number of processors. The path length is fixed, they can provide simultaneous connection and support distributed routing algorithms so there is no need for a control unit. On the down side if we use only $log_2 N$ stages we have only single unique path between any two nodes. Certain input output combination can cause blocking that is the case in which simultaneous communication between two pairs of PEs is impossible. This is called *congestion.*

Diameter. Diameter of a network is defined as the maximum of the set of shortest possible distances between any two nodes. It determines the longest possible path (without loops) in the network and thus should be as small as possible. It measure the compactness of the network and also communication delay.

Scalability. Scalability reflects the ability of a network to be extended in the same topology. In other words it is the question of adding new PEs to an existing network so that the new networks retains the old topology. The main problems is how to connect new nodes and what is the minimal numbers of nodes that has to be added. It is desirable if this can be done in a uniform way thus simplifying the integration of components.

Fault Tolerance and Reliability. It is always possible for some components of the interconnection network to become inoperable either due to imperfections during

manufacturing or some failures later. Thus we need to provide some mechanism to cope with this possibility. Reliability is the measure of the network to tolerate failures. One approach is to use additional hardware as a backup for possible faulty components. The other approach is to provide software solution. In the case of failures the system has to be reconfigured so that with some penalty in performance it can emulate the original network. Closely related is the problem of the maximal number of failures such that the system still can be functional.

We say that a graph is *k fault tolerant* [28, 3] if when k or fewer nodes are removed the resulting graph remains connected. The fault tolerance of a graph is the maximal k such that the graph is k fault tolerant. It is obvious that fault tolerance one less than the connectivity of the graph. With regular graphs the upper bound for fault tolerance is $d-1$ where d is the degree of the graph. We term graphs with tolerance equal $d-1$ *maximally fault tolerant.* Closely related are fault tolerant routing algorithms. These are algorithms which can find a path between two nodes in the presence of at most k faults (where k is not greater than the fault tolerance of the graph). The *fault diameter* of a graph with fault tolerance k is the maximum diameter of any graph obtained from the original graph by removing at most k vertices. Graphs are termed *strongly resilient* if the fault diameter is at most $d+c$ where d is the diameter of the graph and c is a fixed number independent of the size of the graph.

2.3 Problems related to Interconnection Networks

This is a list of most important problem that we should be able to answer in any network.

Labeling. By labeling we understand the naming the nodes or PEs of the network such that each node has its unique name. Successful labeling can implicitly define connectedness and convey other properties of the network. It is also very important in sorting algorithms where the ordering of labels determines the final configuration.

Connectivity. Connectivity of an interconnection network means that there is a possible path between each two nodes of the network. In the case when real directed graphs or diagraphs are used to describe a network we can distinguish between weekly and strongly connected networks. A network is strongly connected if whenever A and B are two nodes there are paths from A to B and also from B to A. We say that a network is weakly connected if there is a path between any two nodes but not necessarily in both directions.

Point to Point routing. This describes path that we have to take in order to make a transfer between given source and destination PEs. It should be also the minimal length path between given nodes. Node independent routing function are of special interest. That is to say, routing which doesn't depend on particular nodes.

Alternative and Parallel Paths. This problem is closely related to the question of reliability and failure recovery. It consists of finding alternative (possible minimal) paths between any two nodes. The paths should be disjoint, that is, all nodes in alternative paths should be different except for the source and the destination. Such paths are also called *parallel.*

Permutations. Very often we have a case in which each processor is the source and destination and this gives rise to the problem of *permutation routing.* Permutations are one to one and onto mappings from the set of PEs into itself. We can talk about *off-line* routing when we have prior or global knowledge about permutation that we need to route and *on-line* or local if the next node is determined at each particular node.

Broadcasting. Broadcasting is a routing policy in which data from a single node is sent to more than one node during the transfer phase. More generally we consider the case in which a number of PEs broadcast each to a different set of destination PEs. Broadcasting is closely related to embedding of trees into a graph. The following are the most common forms of broadcasting

- Single-node broadcast a case when a node is sending a packet to all other nodes.
- Single-node scatter when a node is sending a different packet to every node in the network.
- Multi-node broadcast when all nodes perform their own broadcasts simultaneously.
- Total exchange when all nodes perform their scatters simultaneously.

For all mentioned cases we can also define duals by reversing the direction of broadcasting.

Sorting The problem is to sort values that belongs to different PEs in the network. Close to this is finding the maximal element among numbers distributed among processors located at the vertices of the network.

Embedding and Partitioning. Since many types of interconnection networks are created there is a problem of portability of parallel programs. That is how to translate parallel programs written for a particular topology if we want to process it in a network with different topology. The usual way to solve this is to *embed* one topology into another. Embedding is a bijection between the set of nodes in one topology into the set of nodes of another topology such that the image creates a subnetwork having the properties of the source topology. Informally, it measures the ability of the host network to emulate the source network. Special case is embedding of the same topology. It is usually called *partitioning* of the network. This is the case where we embed one or more subnetworks of the same topology into given network. Formally we define embedding of a graph $S = (P, L)$ in a graph $D = (Q, K)$ as a pair of maps (φ, ν) where $\varphi : P \rightarrow Q$ assigns the nodes of S to the nodes of D and ν is a routing of the edges of S (denoted by $E(S)$) to paths of D such that if $(u, v) \in L$ then the path $\nu(u, v)$ starts at $\varphi(u)$ and ends in $\varphi(v)$ in D. The *load* of embedding is the maximal number of vertices that are mapped to a single vertex.

Another way to describe embedding is using finite automata approach where it reduces to *transition preserving function* [6]. So given two networks $A = (L_A, Q_A, X_A, \delta_A)$ and $B = (L_B, Q_B, X_B, \delta_B)$ embedding is a pair

$$\alpha = (\alpha_Q, \alpha_X)$$

where

$$\alpha_Q : Q_A \rightarrow Q_B$$

and

$$\alpha_X : X \to Y$$

where α is transition preserving, that is, where

$$\alpha_Q(\delta_A(q,a)) = \delta_B(\beta_Q, \beta_X(a))$$

for $\forall q \in Q_A, \forall x \in X_A^*$

2.4 Important Interconnection Networks

Finally, we describe and formally define, using different models, some of the most important types of interconnection networks. Historically the first implemented networks were rather simple like one (Line) or two dimensional arrays or rings. However with the advent of VLSI technology we witness design of more complex and sophisticated topologies. In this survey we consider only *binary* versions of some topologies. That is to say, it is possible to replace single bits in all n-bit binary words with k-ary $k > 1$ strings of bits in which case we have generalized or enhanced versions of original topologies.

Ring and Line. We denote n-vertex line by L_n and n-cycle as an n-ring R_n.

Mesh. A mesh $M_{n,m}$ is a network graph made of nm nodes. It is a graph product

$$M_{n,m} = L_m \times L_n$$

Illiac. We define generalized Illiac graphs using interconnection functions. The set of vertices is $\{0, 1, 2, \ldots, n^k\}$ and we have $2(k-1)$ functions:

$$Illiac_{+j}(x) = x + n^{j-1} \text{ modulo } n^k \text{ for } x \in V, j \in \{1, k\}$$

$$Illiac_{-j}(x) = x - n^{j-1} \text{ modulo } n^k \text{ for } x \in V, j \in \{1, k\}$$

The most often used version of Illiac is a case when $k = 2$.

PM2I. The PM2I is very similar to generalized Illiac. It has the same set of nodes as Illiac with $n = 2$ and the same set of interconnection functions.

Hypercube. n-order hypercube (called an n-dimensional hypercube) H_n is defined as graph product of n copies of R_n. Generalized n,m-hypercube $H_{n,m}$ is a graph product of m copies of R_n.

Shuffle-Exchange. The order n Shuffle-Exchange graph D_n has the vertex set of 2^n numbers labeled as binary numbers of length n. Each node is therefore represented as $(p_{n-1} \ldots p_1 p_0)$ where $p_i \in \{0, 1\}$. We have two types of edges

- Shuffle edges where $(p_{n-1} \ldots p_1 p_0)$ is connected to $(p_{n-2} \ldots p_0 p_{n-1})$.
- Exchange edges where $(p_{n-1} \ldots p_1 p_0)$ is connected to $(p_{n-1} \ldots p_1 \overline{p_0})$

de-Bruijn. The order n de-Bruijn graph D_n is defined as a set of 2^n numbers labeled as binary numbers of length n just as with Shuffle-Exchange as $(p_{n-1} \ldots p_1 p_0)$ where $p_i \in \{0, 1\}$. We have two types of edges

- Shuffle edges where $(p_{n-1} \ldots p_1 p_0)$ is connected to $(p_{n-2} \ldots p_0 p_{n-1})$.
- Shuffle-exchange edges where $(p_{n-1} \ldots p_1 p_0)$ is connected to $(p_{n-2} \ldots p_0 \overline{p_{n-1}})$ and $(\overline{p_0} p_{n-1} \ldots p_1)$

Butterfly. The number of nodes in an order n Butterfly network is $n2^n$ and consists of the set of pairs $\langle l, B \rangle$ where l is an integer between 0 and $n-1$ and B is an n-bit integer.There is an edge between $\langle l_1, B_1 \rangle$ and $\langle l_2, B_2 \rangle$ if and only if

- $B_1 = B_2$ and $|l_1 - l_2| \bmod n = 1$ or
- $|l_1 - l_2| \bmod n = 1$ and B_1 and B_2 differ in the l_1-th bit.

Cube-connected cycles. The number of nodes in an order n Cube-connected cycles network is $n2^n$ and consists of the set of pairs $\langle l, B \rangle$ where l is an integer between 0 and $n-1$ and B is an n-bit integer. There is an edge between $\langle l_1, B_1 \rangle$ and $\langle l_2, B_2 \rangle$ if and only if

- $B_1 = B_2$ and $|l_1 - l_2| \bmod n = 1$ or
- $l_1 = l_2$ and B_1 and B_2 differ in the l_1-th bit.

3 Formal Models

Though a large body of knowledge has been been developed in investigation of different types of interconnection networks [17, 42, 33] we still lack a standard modeling technique which could be applied to all classes of networks. Each network has two types of properties. First we have those which are the consequence of its structure and can not be changed once the system is established and therefore are static in its nature. So we use term *static properties* to describe links that connect nodes in the network, in other words the topology.

In the second group are attributes and functions which can be dynamically modified while the network is operational. That includes characteristics like different types of routing or emulation of other topologies in the given network. It is obvious that any useful network should be able to adjust itself dynamically, that is, without reconfiguration. We name this type of attributes *dynamic properties* of a network.

We begin this part by describing several mathematical models that were proposed to define static and dynamic properties of interconnection networks and then we introduce a new modeling technique based on finite automata theory. This approach introduced in [20] has advantage over some other models that it combines static and dynamic characteristics in a very natural way.

Interconnection Functions. In this model interconnection network is described as a set of functions [39]. Let assume that $N = \{1, 2, \ldots, n\}$ is a set representing nodes. An interconnection function is a mapping from N into N, that is,

$$f \subset N \times N$$

The network can be defined as a pair (N, F) where F is a set of interconnection functions having the following property:

- Identity map i defined as $i(x) = x, \forall x \in N$ is in F.
- For any two $x, y \in N$ there is a finite sequence $f_1, f_2, \ldots, f_k$ of function (not necessarily distinct) from F such that

$$f_1 \circ f_2 \circ \ldots \circ f_k(x) = y \text{ where } \circ \text{ stands for composition}$$

We say that nodes $x, y \in N$ are directly connected if there is $f \in F$ such that $f(x) = y$ or $f(y) = x$.

The main shortcoming of this approach is that it gives rather local view of the system. Dynamic properties in particular are difficult to define or discover.

Networks as Algebraic Structures. This model [1, 5, 19] is applied to cases in which the graph of the network can be represented as a graph of an algebraic structure. We define here algebraic structure $\mathcal{A}$ as a groupoid i.e. a finite set with a defined binary operation $\mathcal{A} = (A, *)$ where * denotes the operation. A generating set S for $\mathcal{A}$ is a subset of A (without identity element, if there is one) if any element of A can be expressed (not necessarily uniquely) as a product of elements in S. We denote this as $A = gp(S)$. The graph of the structure $\mathcal{A}$ with respect to S is

$$\Gamma(\mathcal{A}, S) = (A, E)$$

where the set of edges E is defined as follows: There is an edge between $a, b \in A$ iff $as = b$ or $bs = a$ for some $s \in S$ The degree of the network is equal to the cardinality of the generating set.

The most interesting structures are semigroups,monoids, groups and with additional operation even vector spaces. In the case of groups the underlying networks are called *Cayley networks* or *graphs.* So in a case of a Cayley graphs the vertices correspond to the elements of the group and edges depict actions of the generators [7, 14]. It is usually assumed that the set of generators is closed under inverses so that the resulting graph is undirected. All Cayley graphs are vertex and edge symmetric. A Cayley graph is termed *hierarchical* if it generators can be ordered as $\{s_1, s_2, \ldots, s_d\}$ such that for each $i, 1 < i \leq d, s_i$ is outside the subgroup generated by the first $i-1$ generator. Group graphs that are hierarchical under any ordering of the set of generators are termed *strongly hierarchical.* Hierarchical graphs have important property of being recursively partitionable. That is, a graph of size n can be presented as a collection of graphs of the same type of the order $n-1$ or less.

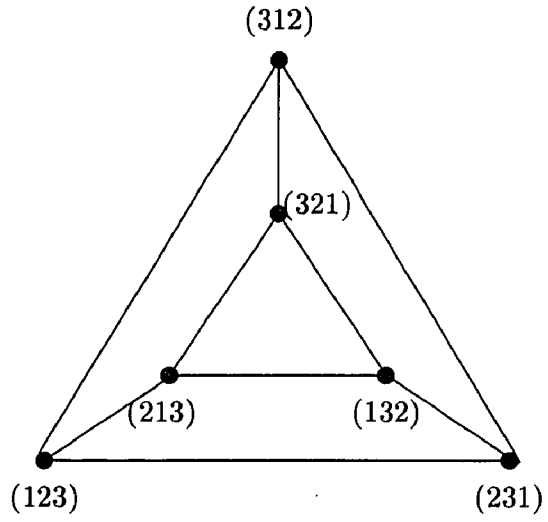

Figure 2: Cayley graph of Σ_3

Example 3.1. Cayley graphs of the symmetric group Σ_3. The elements of the group are 3! permutations of the set $\{1,2,3\}$ This group has several possible sets of generators (See Figure 2.). For example for

$$\Sigma_3 = gp(\{\sigma = (1,2), \tau = (1,2,3)\})$$

The generating relations are

$$\sigma^2 = \tau^2 = 1$$

$$(\sigma\tau)^3 = 1$$

$$(\sigma(\tau\sigma))^2 = 1$$

Important subclass of Cayley networks built around symmetric groups are *star networks.*

Example 3.2. n-cube is a Cayley graph. For example 3-cube can be described as a group graph built from the generating set made of following transpositions $C_3 = \{s_1 = (12), s_2 = (34), s_3 = (56)\}$. The 8 elements are obtained by actions of C_3 on the identity permutation of 6 symbols (123456). Another way is to see the Hypercube as the Cayley graph of the nth direct product of Σ_2, that is

$$C_n = \Sigma_2 \times \Sigma_2 \times \ldots \times \Sigma_2$$

Using some other group theoretic methods it is possible to find underlying groups for many other topologies. For example the n dimensional cube-connected cycles is the Cayley graph of the *wreath product* $\Sigma_2 \wr \mathbf{Z}_n$. Further results and some generalizations can be found in [7].

3.1 Networks as Finite Automata.

In this paradigm [20] the network is represented as a finite automation (FA). Nodes of the network are states and strings accepted by this particular FA describe possible paths. Any node is starting and finite state at the same time. Though we can use any finite alphabet X for the set of inputs it is very convenient if we use the one used for labeling of nodes. By suitable choice the strings accepted by the automation relate to labels of source and destination nodes.

Formally interconnection network as FA is defined as follows.

Definition 3.1. Interconnection network of rank k is 4-tuple

$$N_k = (L, Q, X, \phi)$$

Here Q is the set of states represented as node labels, that is, as nonempty words of length k in some finite alphabet L. X is the input alphabet so any input is a word over X^*. All three sets are finite. The transition function ϕ is defined as

$$\phi \subset X^* \times Q \times Q$$

and described recursively

$$\phi(\lambda, q) = q \text{ where } \lambda \text{ is empty string } q \in Q$$

$$\phi(\alpha\beta, q) = \phi(\beta, \phi(\alpha, q))$$

where $\alpha, \beta \in X^*$ and $q \in Q$.

If there is a set $C \subseteq X^*$ of X words such that any word accepted by N_k can be represented as a sequence of elements of C then we name C a *control set* of the network. So an equivalent definition of a network would be:

Definition 3.2. Interconnection network of rank k is the 4-tuple, (L, N, C, ϕ), where L is a finite alphabet, N is the set of nodes represented as L words of length k, C is control set of words in L^* and ϕ is a mapping $N \times C \rightarrow N$, called the transfer function.

The function ϕ may be *partial,* that is, not defined for all pairs in $N \times C$. If there is $\phi(x, c) \in N$ for some $x \in N, c \in C$ then we say that ϕ is defined or that c is *applicable* to x. If ϕ is defined for all pairs in $N \times C$ we say that the transfer function is *total.* If ϕ is not total then we say that the network *partial* or *incomplete.*

We term a communication network as *bit controlled* if $L = \{0, 1\}$ and the size of N is 2^k where k is the rank. If L is a set made of n symbols such that labels are permutations and the size of N is $n!$ we have *character controlled* networks. We are interested mostly in regular and symmetric networks (because of their node independence) so will consider primarily transition functions which can be described using a fixed set of permutations.

Example 3.3. n-cube can be modeled as the following automation: $L = X = \{0, 1\}$ and Q is made of all tuples of length n, so we have $|Q| = 2^n$ of them. In other words we form a rank n network over L. C contains n elements represented as all n-tuples made of one symbol 1 and $n - 1$ symbols 0, that is,

$$C = \{10\ldots0, 010\ldots0, \ldots, 00\ldots01\}$$

The ϕ is defined naturally as

$$\phi(\alpha, p) = \alpha^p \text{ for } \forall p \in Q, \forall \alpha \in X$$

The operation α^p is standard p-complement of n-dimensional Boolean algebra i.e. $\alpha +_2 p$.

Example 3.4. With $L = \{1, 2, \ldots, n\}$ we can define a ring of length n as a rank n network with the following transition function:

$$\phi(\alpha, p) = \alpha p$$

where $C = \{(1, 2, \ldots, n)\}$ or the input set is made of a single permutation which rotates n symbols. Shuffle and Exchange networks can be defined similarly. The control set for Shuffle is $C = \{(2, 3, \ldots, n, 1), (n, 1, 2, \ldots, n-1)\}$ and for Exchange is $C = \{(1, 2)\}$.

Example 3.5. Here we create a partial network of nine nodes. $L = \{0, 1\}$, and the rank is 4 ($\lceil \log_2 9 \rceil$). N is made of 9 words in L representing first nine numbers in binary representation. The control set $C = \{0001, 0010, 0100, 1001\}$. The function ϕ is defined as in Example 3.1. In this case $\phi(0101, 1001)$ is not defined because $(0, 1, 0, 1) +_2 (1, 0, 0, 1) = (1, 1, 1, 0)$ is not in N.

This automata view is not an accident. It is possible to see group graphs as state diagrams of sorting of so called *transposition trees* [1].

3.2 Properties of control sequences

To establish paths between two nodes in the network we need to form a sequence made of elements of the control set. Such a sequence identifies a possible routing in the system and defines an extension of the routing function.

Definition 3.3. Let C be the finite set of control symbols and k a non-negative integer. We term the concatenation of k elements of C as a *control sequence* of length k and denote as p^k.

The number of elements in a control sequence is called the *length* of the sequence. Empty control sequence can be also considered and is defined as p^0. We will use C^* to denote the set of all control sequences built over a set C.

Definition 3.4. Let ϕ be a routing function. We define *extension* of ϕ, ϕ^m recursively as:

1. $\phi^0(x,p) = x$
2. $\phi^m(x,p^1p^{m-1}) = \phi^{m-1}(\phi(x,p^1),p^{m-1})$ for $\forall x \in X, p^m \in C^*$.

In order to facilitate comparison of control sequences we define a mapping that will play important role in the further investigation of control sequences.

Definition 3.5. Let $p^k = c_1c_2\ldots c_k \in C^*$. The mapping σ defined on the set C^* as $\sigma(p^k) = \phi^k(\mathbf{I}c_1c_2\ldots c_k)$ where **I** is the identity and concatenation represents the product of permutations in p^k is named as the *signature* of p^k.

Since our main goal is investigation of a character controlled type network which are complete, that is to say, have total routing function, we can define the following relation:

Definition 3.6. Let p^α and p^β represent two control sequences of length α and β respectively. We say that p^α and p^β are *x-equivalent*, and denote it $p^\alpha \equiv^x p^\beta$ if and only if $\phi^\alpha(x,p^\alpha) = \phi^\beta(x,p^\beta)$ for $x \in X$ and for $\forall p^\alpha, p^\beta \in C^*$.

From this definition it follows

Lemma 3.1. x-equivalence is an equivalence relation.

Furthermore this relation on the set C^* is node independent.

Proposition 3.1. If $p^\alpha \equiv^x p^\beta$ for some $x \in X$ then $p^\alpha \equiv^y p^\beta$ for $\forall y \in C^*$.

Proof: Let $p^\alpha = a_1 a_2 \ldots a_\alpha$ and $p^\beta = b_1 b_2 \ldots b_\beta$ where a_i, b_i are control symbols. If

$$\phi^\alpha(x, p^\alpha) = \phi^\alpha(x, p^\beta)$$

then

$$x a_1 a_2 \ldots a_\alpha = x b_1 b_2 \ldots b_\beta$$

Since x is a permutation it has inverse x^{-1} and thus from the associativity of functional compositions it follows

$$x^{-1} x a_1 a_2 \ldots a_\alpha = x^{-1} x b_1 b_2 \ldots b_\beta$$

or

$$a_1 a_2 \ldots a_\alpha = b_1 b_2 \ldots b_\beta$$

so

$$y a_1 a_2 \ldots a_\alpha = y b_1 b_2 \ldots b_\beta \text{ for } \forall y \in X$$

and therefore p^α and p^β are y-equivalent. □

We will denote the members of the equivalence class of p^α as $[p^\alpha]$.

Proposition 3.2. Let $C^*_{\equiv}$ be the set of equivalence classes defined on C^* by the routing function ϕ. If we define the concatenation operation on $C^*_{\equiv}$ by

$$[p^\alpha][p^\beta] = [p^\alpha p^\beta], \ \forall p^\alpha, p^\beta \in C^*_{\equiv}$$

then $C^*_{\equiv}$, with concatenation is a monoid.

Proof: First we need to prove that the operation is well defined. Let $p^k \in [p^\alpha]$ and $p^l \in [p^\beta]$. Then $p^k \equiv p^\alpha$ and $p^l \equiv p^\beta$ and so follows

$$\phi^k(x, p^k) = \phi^\alpha(x, p^\alpha) \text{ and } \phi^l(x, p^l) = \phi^\beta(x, p^\beta) \text{ for } \forall x$$

If we expand function ϕ^{k+l} as

$$\phi^{k+l}(x, p^{k+l}) = \phi^k(\phi^l(x, p^l), p^k) = \phi^k(\phi^\beta(x, p^\beta), p^k) = \phi^\alpha(\phi^\beta(x, p^\beta), p^\alpha)$$

whence $p^k p^l \equiv p^{k+l}$ and therefore

$$[p^k][p^l] = [p^k p^l]$$

So $C^*_{\equiv}$ is closed with respect to concatenation. Defined concatenation is also associative operation on $C^*_{\equiv}$. For any $[p^\alpha], [p^\beta], [p^\gamma] \in C^*_{\equiv}$

$$([p^\alpha][p^\beta])[p^\gamma] = [p^\alpha p^\beta][p^\gamma] =$$

$$[(p^\alpha p^\beta)p^\gamma] = (p^\alpha(p^\beta p^\gamma)] =$$

$$[p^\alpha]([p^\beta p^\gamma]) = [p^\alpha]([p^\beta][p^\gamma])$$

Finally the equivalence class of $p^0 = [0]$ serves as the identity since

$$[p^\alpha][p^0] = [p^{\alpha 0}] = [p^\alpha]$$

□

Proposition 3.3. For the class of character controlled networks $C^*_{\equiv}$ is a group.

Proof: In this case concatenation operation is composition of permutations. Since $C^*_{\equiv}$ is a monoid we only need to show the existence of inverses. For any sequence $p^\alpha = p_1 p_2 \dots p_\alpha$ the inverse $(p^\alpha)^{-1} = p_\alpha^{-1} p_{\alpha-1}^{-1} \dots p_1^{-1}$. So the inverse for $[p^\alpha]$ is $[(p^\alpha)^{-1}]$. □

Proposition 3.4. If two control sequences p^α and p^β are defined on C^* then

1. p^α and p^β are equivalent if and only if $\sigma(p^\alpha) = \sigma(p^\beta)$.
2. If $p^\alpha \equiv p^\beta$ and $p^{\alpha_1} \equiv p^{\beta_1}$ then $p^\alpha p^{\alpha_1} \equiv p^\beta p^{\beta_1}$ where concatenation represents functional compositions.

Proof: If $p^\alpha \equiv p^\beta$ then $\phi^\alpha(x, p^\alpha) = \phi^\beta(x, p^\beta)$ for $\forall x \in X$. So it is true for I, the identity permutation and thus $\sigma(p^\alpha) = \sigma(p^\beta)$. In the opposite direction from

$$\sigma(p^\alpha) = \sigma(p^\beta) = \sigma(Ip^\alpha) = \sigma(Ip^\beta)$$

and the claim follows from proposition 3.1. □

Corollary 3.1.

1. Sequence p^k connects x to y if and only if $\sigma(p^k) = x^{-1}y$
2. If p^k and p^m are sequences connecting x to y and y to x respectively, then $\sigma(p^m) = \sigma((p^k)^{-1})$

We can have several possible sets C which can serve as control sets. Certainly it would be best to use the smallest one. For that reason we introduce the concept of the *base* of a network.

Definition 3.7. A set of control symbols $\{c_1, c_2, \ldots, c_n\}$ from C is called *independent* in a network Γ if

$$c_{\pi(1)}c_{\pi(2)}\cdots c_{\pi(n)} \neq [0] = I \text{ for any } \pi \in \Sigma_n$$

Definition 3.8. A set $B = \{c_1, c_2, \ldots, c_n\}$ of independent control symbols from C, such that any control sequence from C^* can be expressed as a sequence of elements of B is called the base of the network.

Parallel paths play a crucial role in investigation of conflict free communication as well in measuring fault tolerance of interconnection networks. By providing alternative routes it is possible to lower congestion and concentration of traffic on the network. Characterization of parallel paths has been studied for different types of networks [30, 38, 27, 29]. Finite automata approach is particular well suited for defining and describing node disjoint routes.

Definition 3.9. Two control sequences, p^k and p^l are said to be disjoint and the paths they define parallel if and only if $\phi^k(x, p^k) = \phi^l(x, p^l)$ and $\phi^\alpha(x, p^\alpha) \neq \phi^\beta(x, p^\beta)$ for $\forall \alpha < k, \beta < l$ and $\forall x \in C$.

Very important question in network control is how to construct and identify control sequences capable of creating parallel paths between a given set of nodes. An answer to this question for a class of Omega networks is given in [29]. The following theorem gives another solution which is applicable to any bit-controlled interconnection network.

Theorem 3.1. [20] Let $P = \{\langle s(i), d(i)\rangle | i \in H\}$ where H is a set of indexes, be the set of desired connections from given input nodes $s(i)$ to output nodes $d(i)$. Let t_{ij} denotes the relative distances between two source nodes $s(i)$ and $s(j)$. Also let p^α and p^β be control sequences with signatures $\sigma(p^\alpha) = s(i) +_2 d(i)$ and $\sigma(p^\beta) = s(j) +_2 d(j)$. If $t_{ij} \neq \sigma(p^a) +_2 \sigma(p^b)$ for all integers $a, b \leq max(\alpha, \beta)$ then it is possible to define conflict free routing for P.

Proof: Let control sequences p^α and p^β define routes for pairs $\langle s(i), d(i)\rangle$ and $\langle s(j), d(j)\rangle$ respectively. For two paths to be disjoint the following must hold

$$\phi^a(s(i), p^a) \neq \phi^b(s(j), p^b) \text{ for } \forall a, b \leq max(\alpha, \beta)$$

where p^a is a substring of p^α and p^b substring of p^β. Expanding ϕ and rearranging terms results in

$$s(i) + s(j) \neq \sigma(p^a) +_2 \sigma(p^b)$$

□

Corollary 3.2. Let p^k be a control sequence which routes $s(i)$ to $d(i)$ and $s(j)$ to $d(j)$, i.e. $\sigma(p^k) = s(i) + d(i) = s(j) + d(j)$. If $\sigma(p^\alpha) \neq t_{ij} = s(i) + s(j)$ for $\forall \alpha < k$ then all pairs of distinct source nodes with the same relative distance t_{ij} which are not part of any other paths, can be connected to their respective destinations by p^k thru parallel paths.

Proof: Follows from the previous theorem. □

We have similar result for character controlled networks.

Proposition 3.5. Let $P = \{\langle s(i), d(i)\rangle | i \in H\}$ where H is a set of indexes, be the set of desired connections from given input nodes $s(i)$ to output nodes $d(i)$. Let r_{ij} be a sequence defining a path between two source nodes $s(i)$ and $s(j)$. Also let p^α and q^β be control sequences describing paths from $s(i)$ to $d(i)$ and $s(j)$ and $d(j)$ respectively. If $\sigma(t_{ij}q^c) \neq \sigma(p^c)$ for all integers $c \leq max(\alpha, \beta)$ then it is possible to define conflict free routing for P.

Proof: Let $\sigma(r_{ij}q^c) = \sigma(p^c)$ for some $c \leq max(\alpha, \beta)$. This means that $r_{i,j}q^c p^{c-1}$ is a control sequence which is a ring. It also means that there are a routings from

source nodes which enter the same intermediate node after c steps therefore causing a conflict. □

4 Ring Connected Star Graph

4.1 Topological Properties

In this chapter we concentrate on a new topology called the Ring Connected Star Graph. We need first some properties of permutations.

Definition 4.1. A permutation which can be represented as a single cycle of two elements is called a *transposition* .

The importance of transpositions follows from the fact that any cycle can be represented as a product of transpositions. It follows from the following simple result:

$$(a_1, a_2, \ldots, a_k) = (a_1, a_2)(a_1, a_3)\ldots(a_1, a_k)$$

As is well known every permutation can be represented as a product of disjoint cycles. Cycles sequences of symbols (in our case we use numbers) such that each symbol maps into the next symbol in the cycle. The last elements maps in the first. Cycles that are made of single elements are usually disregarded. So for example $(1,3,5)(2,6)$ is a representation of the following permutation:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 6 & 5 & 4 & 1 & 2 \end{pmatrix}$$

We will denote the length of a cycle c as $|c|$.

Note also that each transposition can be further represented using transpositions having 1 as one of two symbols. It follows from the next equality:

$$(a, b) = (1, a)(1, b)(1, a)$$

Lemma 4.1. If a permutation of n elements can be represented as a product of r disjoint cycles then it can be represented as succession of $n - r$ transpositions.

Proof: Since each cycle of length k is representable as a product of $k - 1$ transpositions the total number must be

$$\sum_{i=1}^{r}(|c_i| - 1) = n - r$$

□

Definition 4.2. The order n Ring Connected Star Graph (RCS_n) is an undirected graph made of n! nodes labeled with permutations of n symbols $1, 2, \ldots, n$. There is an edge between two vertices if and only if their labels differ in the first and one other position or if one label can be obtained from the other by cyclic rotation.

We can see easily that RCS_n is a Cayley graph of the symmetric group Σ_n with the following set of generating permutations

$$\mathcal{S} = \{(1,2),(1,3),\ldots,(1,n),(1,2,\ldots,n),(n,1,2,\ldots,n-1)\}$$

In this model we see vertices of the graph as elements of the symmetric group while edges represent multiplications of these elements by the members of the generating set. We will denote edges in RCS_n as $e_2, e_3, \ldots, e_n$ where e_k stands for $(1, k)$. These edges are called *exchange* edges. The other type of edges are $(1, 2, \ldots, n)$ denoted as s and $(n, 1, 2, \ldots, n-1)$ written as t. We name s and t as left and right *shifts*. We will also be using terms *shuffles* or *rotations* for s and t edges. See Figure 3..

Using finite automata approach from the previous chapter this network can be described as a rank n automation over the alphabet $L = X = \{1, 2, \ldots, n\}$. N is made of $n!$ elements and we denote elements of Q as all n-tuples of letters in L where

each letter appears only once. So the elements of N are all permutations of the set L.The control set is $C = \{e_2, e_3, \ldots, e_n, s, t\}\}$.

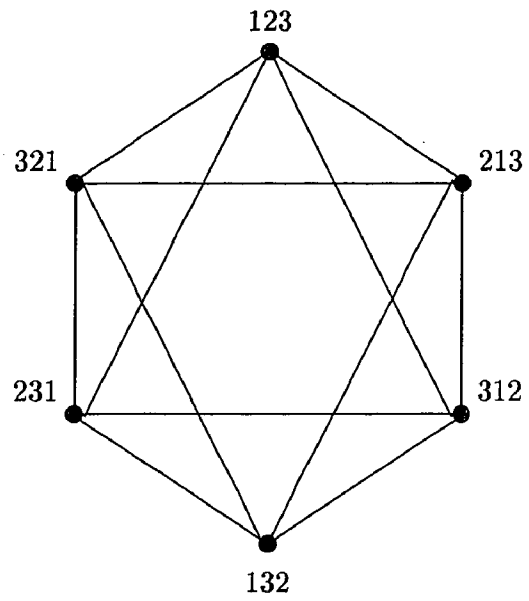

Figure 3: RCS_3

The transition function is

$$\delta(p, \alpha) = p\alpha \text{ for } \forall p \in N, \forall \alpha \in C$$

and $p\alpha$ is ordinary composition of permutations.

The degree of RCS_n is obviously $n + 1$.

A special case of RCS is the Star graph network. This is obtained from the Ring Connected Star Graph by removing shuffle edges.

Definition 4.3. The n-star graph [1], denoted by S_n, is undirected graph made of $n!$ vertices labeled with the $n!$ permutations of n symbols $1, 2, \ldots, n$. There is an edge between any two vertices, if and only if their labels differ only in the first and in one other position.

Definition 4.4. The complete n-star graph, denoted by CS_n, is undirected graph made of vertices labeled with the $n!$ permutations of n symbols $1, 2, \ldots, n$. There is an edge between any two vertices, if and only if their labels differ in exactly two position.

Both topologies are at the same time Cayley graphs and therefore graphs of the symmetric group. The n-star graph has as a set of generators the set of all transposition of the form $(1, j)$ where $j \in \{2, 3, \ldots, n\}$. Similarly the generating set for CS_n is $\{(i, j) | i, j \in \{1, 2, \ldots, n\}\}$.

Both the star graph and complete star graph are furthermore strongly hierarchical [1] and thus recursively definable. So S_n can be represented as n copies of S_{n-1} graphs.

There are in fact several ways in which we can describe these subgraphs. First, we can create subgraphs by fixing one of the positions in the node labels with a particular number. We denote such substars as $S_{n-1,i,r}$ where r is a position $(2 \leq n \leq n)$ and i is possible value $(1 \leq i \leq n)$. So $S_{n-1,i,r}$ is induced subgraph with symbol i fixed on the position r. If we consider only the symbol in the last place we can omit the position writing only $S_{n-1,r}$ for a substar with the symbol r fixed in the last place. Another way is to partition S_n into $S_{n-1,i,1}, S_{n-1,i,2}, \ldots, S_{n-1,1,n}$.

[1] The name comes from star transitional trees, from which the graphs can be derived. In literature the term Star graph is often used for what we call the Spoke graph.

However with $i = 1$ the $S_{n-1,1,1}$ is a collection of $(n-1)!$ isolated vertices. If we limit our attention to only those labels in which a particular symbol is fixed in a specific position, other than the first, then these vertices make a $(n-1)$-star $S_{n-1,i,i}$. Since there are n symbols and $n-1$ positions, it follows that any n-star contains $n(n-1)$ copies of S_{n-1} graphs.

Proposition 4.1. For a given n-star network S_n there are

$$\binom{n-1}{n-k}\frac{n!}{k!}$$

k-star substars in S_n where $2 \leq k \leq n$.

Proof: We obtain k-stars by selecting and fixing $n-k$ symbols in the label alphabet. There are $\binom{n-1}{n-k}$ ways to choose positions and $\frac{n!}{k!}$ selections for symbols. Therefore the number of S_k is

$$\binom{n-1}{n-k}\frac{n!}{k!}$$

□

Proposition 4.2. The number of edges in S_n is given recursively by

$$E_n = nE_{n-1} + \frac{n(n-1)!}{2} = nE_{n-1} + \frac{n!}{2}$$

Proof: In any Star graph of order n we have n Star subgraphs of order $n-1$. Each node in S_{n-1} is connected to $(n-1)!$ other vertices counted twice. □

Corollary 4.1. The number of edges in S_n is equal to

$$E_n = n!\left(\frac{n-1}{2}\right)$$

Proof: It follows from the solution of the recursive equation in the proposition 4.2. We start with $E_2 = 1$. □

Lemma 4.2. The Star graph is a bipartite graph.

Proof: We define a mapping from Σ_n into Σ_n as $\pi : p\alpha = q$ where $p, q \in \Sigma_n$ and $\alpha \in \{e_2, e_3, \ldots, e_n\}$. If p is an even permutation then q will be an odd one and vice versa. So all nodes are divisible into two equal groups. One made of even and the other of odd permutations. □

Proposition 4.3. Two substar graphs of a RCS_n are connected with $3(n-2)!$ edges.

Proof: As was proved above any two substar graphs S_{n-1} are connected with $(n-2)!$ edges. In RCS_n each node has two more edges, that is, $2(n-1)!$ additional edges in each S_{n-1}. This edges are distributed among $n-1$ remaining substars giving $2(n-1)!/2(n-1)$ links between any two of them. So the total is $3(n-2)!$. □

Proposition 4.4. The number of edges in the RCS_n is

$$G_n = E_{S_n} + n!$$

where E_{S_n} is the number of edges in the Star graph S_n.

Proof: We have two shuffle edges for every of n! nodes counted twice. □

Corollary 4.2. The number of edges in RCS_n is equal to

$$G_n = n!\left(\frac{n+1}{2}\right)$$

An important property of any graph which is used as a network is how efficiently it can be laid out. We can measure that with the number of edges that must intersect. Let suppose that in an embedding of a graph on a surface an edge must intersect with another edge in a point which is not a vertex. Thus we must have a "bridge" or "handle" to avoid the intersection. The number of available bridges on a given surface is called the *genus of the surface.* Similarly we define the *genus of a graph* as the minimum of all genuses of all surfaces on which the graph can be embedded. We will denote the genus of a graph G as $\gamma(G)$. We also define the *girth of a graph* G (denoted $g(G)$), as the length of the shortest cycle in G. The edges of a graph divide the surface on which it is placed, into connected surfaces which we call regions. We only consider surfaces that have *2-cell embedding* property. A region is called a 2-cell if any closed simple curve embedded inside the region can be collapsed to a single point.

Lemma 4.3. [24] The girth of the Star graph S_n of order $n \geq 3$ is

$$g(S_n) = 6$$

Proof: S_n is bipartite and therefore there are no cycles of odd length. S_3 which is contained in any $S_n, n > 2$ is a cycle of length 6. A cycle of length 2 would imply multiple edges between nodes. Finally by enumerating all possible control sequences of length 4 we can see that there are no cycles of length 4. We must have two pairs of exchanges in order to restore displaced elements. So we have tree possible orderings:

1. $e_i e_i e_j e_j$

2. $e_i e_j e_j e_i$

3. $e_i e_j e_i e_j$

Each of the cases would result either in multiple edges or in a path which is not a cycle. □

Lemma 4.4. [24] The number of regions in the Star Graph of order n is given by:

$$r = n! \left(\frac{n-1}{6}\right)$$

Proof: The complete proof of this lemma can be found on [24]. Basically we can prove that any Star Graph can be seen as a collection of S_3 graphs, as was indicated informally at the beginning of this section, forming a kind of honeycomb. Also each edge belongs to no more than two regions. Since the number of edges in S_n is $n! \left(\frac{n-1}{2}\right)$ and each S_3 is a hexagon the results follows immediately. □

Lemma 4.5. The girth of the Ring Connected Star graph of order $n \geq 3$ is

$$g(S_n) = min(4, n)$$

where $min(a, b)$ is the smaller of numbers a, b

Proof: Each RCS_n contains a S_n. Therefore $g(RCS_n) \leq g(S_n) = 6$. Also it must be $g(RCS_n) > 2$ or we would have multiple edges between nodes. Any cycle made just of exchange steps would, at the same time, created a cycle for the embedded S_n. Since we don't have a cycle on S_n with the length less than 6 one must use shuffles in any closed path that would be shorter than $\gamma(S_n)$. We begin by demonstrating that we have no closed paths of length 3 when $n > 3$. Since $s^n = t^n = I$ any sequence made of $s = t^{-1}$ and t of length 3 must be non identity and is not a cycle. A path made of two exchanges and a shuffle will not create a

cycle. If it would be cycle that would imply that s or t can be presented as a product of two exchanges. For example if se_ie_j is a cycle it follows that

$$se_ie_j = I \text{ or } s = e_j^{-1}e_i^{-1} = e_je_i$$

and that is impossible. Finally we have a case of control sequences made of two shuffles and one exchange. If this was a cycle we would have an exchange that can be expressed as a product of two shuffles. From

$$s^\alpha e_i s^\beta = I \text{ where } \alpha, \beta \in \{1, -1\}$$

follows

$$e_i = s^{-\beta}s^{-\alpha}$$

If $\alpha = -\beta$ it would mean that $e_i = I$ and if α and β have same signs that would result in $e_i = s^2$ or $e_i = t^2$ and this is not true for any i.

Now we show that for $n > 3$ there is a cycle of length 4. The following control sequence is a cycle:

$$g = te_nse_2$$

$$g = (1, n, n-1, \ldots, 2)(1, n)(1, 2, \ldots, n)(1, 2) = I$$

We can see that 1 is mapped to n, then n to 1 then again to 2 and finally back to 1. For any other symbol $k \neq 1$, n is transferred to $k-1$ by t. The next steps leaves $k-1$ in the same place and shuffle s moves it back to k. The last exchange has no effect. Similarly n is mapped to itself. Whence $g = 1$ is a cycle of length 4. □

The proof of the following well known result can be found, for example, in [43]

Theorem 4.1. *Generalized Euler's Formula* If $G = (V, E)$ is a connected graph

with 2-cell embedding on a surface of genus g and r regions then

$$|V| - |E| + r = 2 - 2g$$

Let assume that that the girth of graph G is $\gamma(G) = g$. The boundary of each region contains at least g edges because g is the length of the minimal cycle. Every edge also belongs to the boundaries of at most d regions, where $d \geq 2$, so it follows that $gr \leq d|E|$. Substituting this in Euler's formula we prove:

Theorem 4.2. [13] Let $G = (V, E)$ be a connected graph then

$$\gamma(G) \geq \frac{|E|\left(1 - \frac{d}{g}\right) - |V|}{2} + 1$$

Proposition 4.5. Let $RCS_n = (V, E)$ be a Ring Connected Star Graph of order n. Then

$$\gamma(RCS_n) \geq n!\left(\frac{n-3}{8}\right) + 1$$

Proof: Using the fact that $|E| = n!\left(\frac{n+1}{2}\right)$, $|V| = n!$,$\gamma(RCS_n) = 4$ $d \geq 2$ and replacing in the formula from Theorem 4.2 we obtain the result. □

Proposition 4.7. The genus of the Ring Connected Star Graph satisfies the following inequalities:

$$n!\left(\frac{n-3}{8}\right) + 1 \leq \gamma(RCS_n) \leq 1 + n!\left(\frac{n-1}{6}\right)$$

Proof: The left side follows from proposition 4.5. We prove here the right side. We know that each RCS_n contains a Star graph of order n and so it has all regions

of that graph. If r denotes the number of regions in RCS_n then $r \geq n!\left(\frac{n-1}{6}\right)$ we get from Euler's formula

$$2\gamma(RCS_n) \leq 2 - |V| + |E| - n!\left(\frac{n-1}{6}\right)$$

Substituting values for $|V| = n!$ and $|E| = n!\left(\frac{n+1}{2}\right)$ and simplifying we obtain

$$\gamma(RCS_n) \leq 1 + n!\left(\frac{n-1}{6}\right)$$

□

4.2 Characterization of RCS_n and the Star Graph

Next we concentrate on characterization of the Star graph and the Ring Connected Star graph. In order to define conditions for a graph to be a Star graph we need a technique developed for embedding of graphs on compact orientable 2-manifolds (see [16, 43, 44]).

Let $G = (V, E)$ be a connected graph such that the set of nodes $V = \{v_1, v_2, \ldots, v_n\}$. We denote $V_i = \{v_k \in V | (v_i, v_k) \in E\}$. Let $\pi_i : V_i \rightarrow V_i$ be a cyclic permutation of length equal to the size of $|V_i| = n_i$. Then there is a one-to-one correspondence between 2-cell imbeddings of G and choices for π_i. The proof of the following theorem can be found in [43]. It states the relation between 2-cell embeddings of G (on all possible surfaces) and the n-tuples $(\pi_1, \pi_2, \ldots, \pi_n)$ of cyclic permutations.

Theorem 4.3. *The Rotational Embedding Scheme* [43] Each choice of $(\pi_1, \pi_2, \ldots, \pi_n)$ determines a 2-cell embedding $G(M)$ of G in a surface M, such that there is an orientation (say counterclockwise) on M which induces a cyclic ordering of the edges (v_i, v_k) at v_i in which the immediate successor to (v_i, v_k) is the edge $(v_k, v_{\pi_i(v_k)})$, $i = 1, 2, \ldots, n$. In fact, given $(\pi_1, \pi_2, \ldots, \pi_n)$ there is an algorithm which produces the determined embedding. Conversely, given a 2-cell embedding $G(M)$ in a surface M with a given orientation, there is a corresponding $(\pi_1, \pi_2, \ldots, \pi_n)$ determining that embedding.

As an example, consider the embedding of a Star graph of order 3 on Euclidean plane (Figure 4.2.). Here $V = (1, 2, 3, 4, 5, 6)$ and assuming counterclockwise orientation

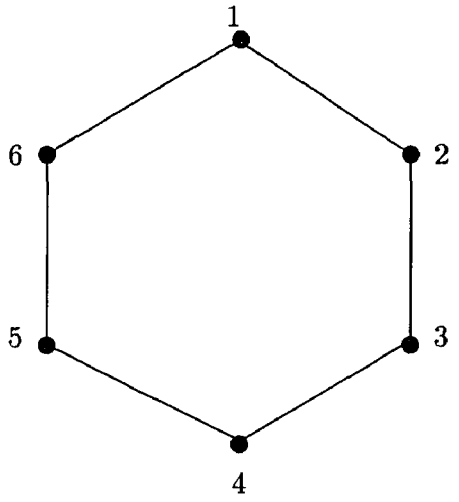

Figure 4: Embedding of S_3 in E_2

$$\pi_1 = (6,2),\ \pi_2 = (1,3)$$

$$\pi_3 = (6,2),\ \pi_4 = (1,3)$$

$$\pi_5 = (6,2),\ \pi_6 = (1,3)$$

It is obvious that π_i are cyclic ($(6,2) = (2,6)$). π_1 tells us also that we can leave the node 1 to nodes 6 and 2 (in this order) or that the immediate successor of $(1,6)$ must be $(6, v_{\pi_2(1)}) = (6,5)$.

Let $D = \{(a,b)|(a,b) \in E\}$ and define a map

$$\Pi : D \to D \text{ as } \Pi(a,b) = (b, \pi_b(a))$$

This mapping is one-to-one and, thus, is a permutation of E. Being a permutation Π can be expressed as a product of disjoint cycles. We name each of this cycles a Π-

orbit. In this way Π-orbits partition the set E. This implies that Π-orbits describe the boundaries of all 2-cell regions in this particular embedding $G(M)$. If we begin at vertex v_i and proceed along (v_i, v_j) to v_j, the next edge in a counterclockwise direction is $(v_j, v_{\pi_j(i)})$. Going further in the same way we must finally arrive back to v_i thru some edge v_m, v_i. The path we made is a 2-cell region corresponding to this Π-orbit.

We will term the length of cycles in permutation Π as the length of a π-orbit. Using approach developed in [24] we can prove:

Proposition 4.8. In the Star graph of order n, the length of Π-orbits is equal 6.

Proof: Since S_n is bipartite we can split its set of nodes into two equal sets. One is made of even permutations and we call it E and the other is composed of odd permutations which we denote O. Now we have to define tuple $(\pi_1, \pi_2, \ldots, \pi_{n!})$. We have two cases:

1. If $v_i \in E$ then
$$\pi_i = (v_i e_2, v_i e_3, \ldots, v_i e_n)$$
2. If $v_i \in O$ then
$$\pi_i = (v_i e_n, v_i e_{n-1}, \ldots, v_i e_2)$$

According to Rotational Embedding Scheme there is 2-cell embedding of S_n defined by above $n!$-tuple. Now starting with $v_i \in E$ we move to v_j along an edge defined with some e_k. So $v_j = v_i e_k$ and $2 \leq k \leq n$. Also $v_j \in O$. To leave v_j we must use e_{k-1} to get to some $v_l = v_j e_{k-1}$. $v_l \in E$ again. To continue we have to use $e_{k-1+1} = e_k$ to leave v_l. It is obvious that in doing this further we will be using only

two exchanges, that is e_k and e_{k-1}. They can change only symbols in the first, k-th and $k-1$-th places. So we are permuting only three symbols and our domain is a subset of label which is isomorphic to S_3 (see Figure 4.2). If we start with a vertex $v_i \in O$ we will again reach the starting node after completing a cycle isomorphic to S_3. As all Π-orbits define all regions, this shows that all regions are hexagons. Since S_3 is a cycle of length 6 we see that the length of all Π-orbits is 6. □

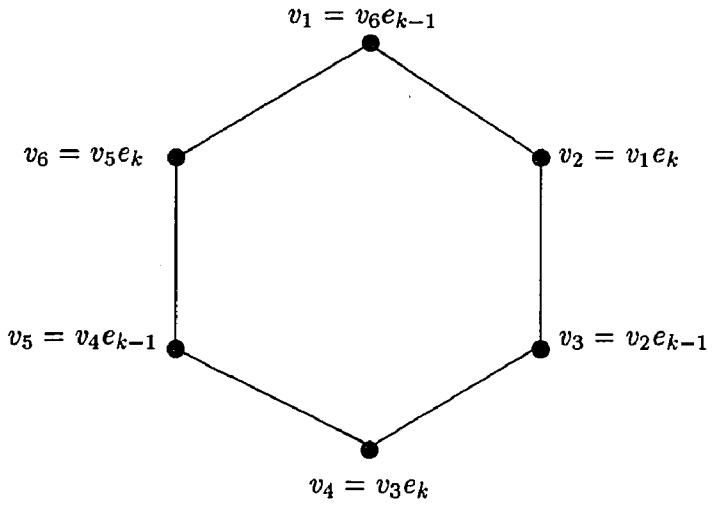

Figure 5: A Π-orbit in the Star Graph

Corollary 4.3. A given connected graph $G = (V, E)$ is a Star graph of order n if:

1. The size of G is $n!$.
2. The degree of each node is $n - 1$.
3. G is bipartite.

4. The length of any Π-orbit is equal to 6. (Or, in other words, any region is a hexagon)

Proof: If G is a Star graph S_n then all conditions follow from the definition and previous propositions. □

Characterization of the Star and Ring Connected Star graphs. Corollary 4.3 defines necessary condition for a graph to be the Star graph. It is possible to verify whether a given graph is a Star graph using the following procedure. First the size of the graph must be $n!$ and the degree of each node must be $n-1$. To confirm that Π-orbits are of length 6 we will create an embedding that will demonstrate whether that the graph is a Star graph. At the same time we will create a labeling scheme for this graph. We choose an arbitrary node and label it as identity $I = (12\ldots n)$. This node must be connected to $n-1$ different neighboring nodes. We can choose one of them and label it with Ie_2. Now going counterclockwise we label remaining nodes $Ie_3, Ie_4, \ldots, Ie_n$. Let us term those $n-1$ nodes as stage 1 nodes. For each of them we can continue in a similar way. For example let us consider $p = Ie_2$ vertex. It also has $n-1$ neighbors one of which is identity $I = pe_2$. Starting with this edge we again label nodes but this time in clockwise direction as $pe_n, pe_{n-1}, \ldots, pe_3$. We do the same for all stage 1 labels. So for stages that are even number the labeling is performed in counterclockwise direction and for odd stages in clockwise direction. By the construction it is obvious that edges belonging to even and odd stages must be in different partitions of the graph. Thus there are no edges between nodes of the same stage or nodes with same stage parity. In other words our labeling follows description of $\pi_1, \pi_2, \ldots, \pi_{n!}$ from the previous theorem. By the definition any Π-orbit has the form $p = e_k e_{k-1} e_k e_{k-1} \ldots$. If each sequence

p forms a cycle from any node of the graph then the given graph can be the Star graph otherwise it is not. It is not difficult to see that the time complexity of this procedure for a graph of size $N = n!$ is $T_{s_n} = O(N) = O(n!)$.

Using this result we can characterize the Ring Connected Star graphs. Given a graph $G = (V, E)$ we first try to embed a Star graph in it. For a graph of size n and degree equal $n + 1$ we have to choose an initial node that we label as identity. Then we have to select $n - 1$ nodes connected to the initial node and test if we can create an embedding of S_n. It can be done in $\binom{n}{2}$ ways. If we have a S_n embedded all that we have to check is if edges that were left outside of S_n link nodes p, q such that $ps = q$ or $pt = q$. If this is true we have proved that the graph is RCS_n.

4.3 Routing on RCS_n and Star graphs

Since both RCS_n and S_n are vertex symmetric in all routing problems it is enough to consider an arbitrary node and the special node labeled with the identity permutation.

Theorem 4.4. [2, 15]If X is a node of an n-star, labeled with permutation P then the minimum length distance (MLD) between X and the node Y labeled with identity permutation I is given as:

$$d(X,Y) = \begin{cases} c+m & \text{if 1 is in first position in X} \\ c+m-2 & \text{if 1 is not in first position} \end{cases} \quad (1)$$

where c denotes the number of X cycles of length at least two and m is the number of symbols in these cycles.

Proof: Finding a MLD path between the node labeled with identity permutation and an arbitrary node is tantamount to optimal sorting of the given label using only transposition of $(1,j), 2 \leq j \leq n$ type. We can do this by repeatedly applying one of the following two steps until we reach identity.

- If symbol 1 is in the first position we exchange it with any other symbol not in its proper place.
- If symbol $k \neq 1$ is in the first position we replace it with the symbol in position k.

The second transformation reduces by one the number of displaced symbols and the length of one of the cycles which has the size greater than two because, the moved

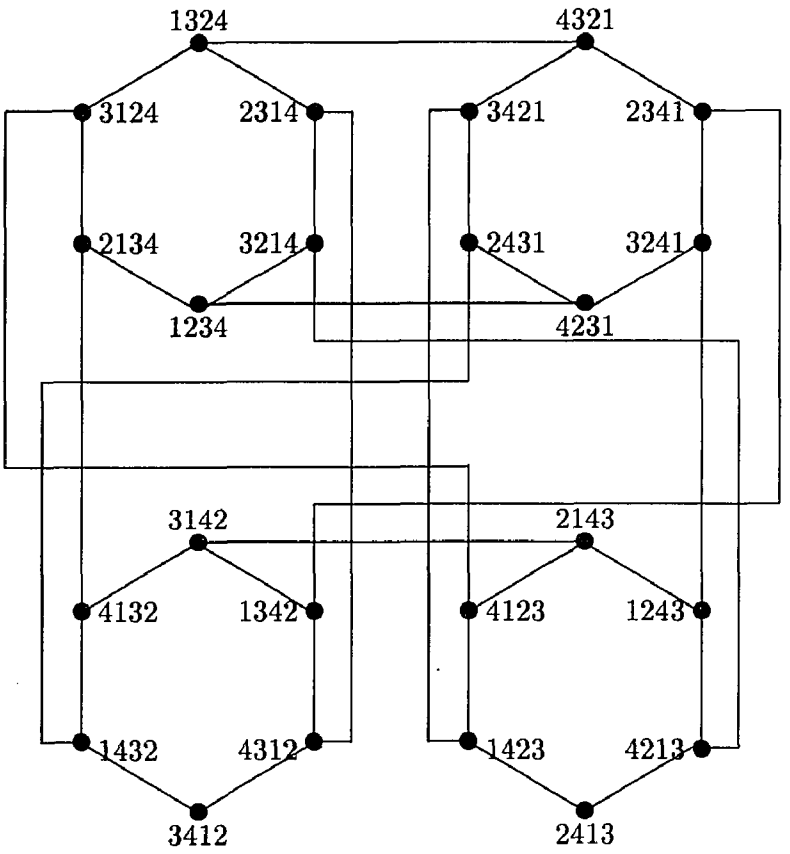
1324
4321
3124
2314
3421
2341
2134
3214
2431
3241
1234
4231
3142
2143
4132
1342
4123
1243
1432
4312
1423
4213
3412
2413

Figure 6: S_4

symbol belongs to that cycle. The first transformation increases the size of one of the cycles by one. Since our goal is to reduce the length of all cycles to one, we need to do at least the number of steps equal to the number of cycles of size greater than 1 plus the number of symbols in those cycles. However, if 1 was not originally in the first position the minimal number of transformation is reduced by 2 because we don't have to move symbol 1 from the first position and then return back. □

Corollary 4.4. [2] The diameter of an n-star graph is $\lfloor \frac{3}{2}(n-1) \rfloor$.

Proof: We get the maximal length path when we maximize the values for c and m. If two labels P_1 and P_2 have the same number of symbols that are in their proper positions and also P_1 has 1 in its first position then it is obvious that the distance of P_1 must be greater than the distance of P_2. So let us consider only labels with 1 in its first position. So for odd n we have $c = (n-1)/2$ and $m = n-1$ so $d = \frac{3}{2}(n-1)$. For even n, $c = \lfloor (n-1)/2 \rfloor$ and $m = n-2$ and therefore $d = (n-1)/2 + n - 2 = \frac{3}{2}(n-1) - 1/2$. □

We describe a simple routing algorithm for S_n.

Let p be a permutation. We can therefore represent it as a product of transpositions such that the first element in all of them is 1. We call such representation *normal.* A normal representation is called reduced if it doesn't contain two same transpositions next to each other. In other words in a reduced normal form we remove all pairs $(1,a)(1,a)$ from the representation because they form a loop.

Proposition 4.9. For a given node with a label p, the routing for the minimal length path is given by reduced representation of p.

Proof: Let assume that p can be represented as a product of r cycles.

$$p = c_1 c_2 \ldots, c_r \text{ where } |c_i| > 1$$

We consider two cases. The symbol 1 is in one of the cycles. In this case 1 is not in the first position of p. As above we can first depict p as a product of $n - r$ transpositions and then replace each transposition $(a, b), a, b \neq 1$ with a product of tree transpositions with 1 in the first place. (i.e. $(a, b) = (1, a)(1, b)(1, a)$). For each cycle $(a_1, a_2 \ldots, a_p)$ which doesn't contain 1 we will get a reduced sequence in normal form of length $p+1$. This is because we can remove $p-3$ pairs of redundant transpositions $(1, a_1)(1, a_1)$ each transposition coming from the end and from the beginning of consecutive transpositions. In the cycle c that contains 1 the number of transpositions in reduced normal form is $|c| - 1$. Let assume that the first cycle contain 1. Then the total number of transpositions in the reduced normal form is given by:

$$|c_1| - 1 + \sum_{i=2}^{r}(|c_i| + 1) = \sum_{i=1}^{r}|c_i| + r - 2$$

If the symbol 1 is not a member of any cycle c_i then the number of transpositions is

$$\sum_{i=1}^{r}(|c_i| + 1) = \sum_{i=1}^{r}|c_i| + r$$

This shows that the number of transpositions in the reduced normal form is equal to the minimal length distance of the node p. □

Now we extend some of the results for the Star graph for the case of Complete Star graphs.

Proposition 4.10. If X is a node of an complete n-star, labeled with permutation P where P is a sequence of cycles

$$P = C_1 C_2 \ldots C_k$$

each cycle C_i having c_i symbols, then the minimum length distance (MLD) between X and the node Y labeled with identity permutation I is given as:

$$d(X,Y) = \sum_{i=1}^{k}(c_i - 1)$$

Proof: The MLD path is the path we have to take to sort the permutation P. In each step we can place one of the symbols directly in its position. Such a transformation reduces the size of one cycle by 1. We continue with the same cycle until its size is 1. The last step will however place two symbols in their right places at the same time. Thus to sort a cycle of a length c we need to do exactly $c - 1$ transposition. □

Corollary 4.5. The diameter of complete star graph CS_n is $n - 1$.

Proof: Follows from the proposition above as the maximal possible length of a path. □

Using some of the results we have proved so far we can now investigate the diameter of the Ring Connected Star Graph. We say that two symbols of a permutation are in *correct relative positions* if there is a transformation made of shifts of the same type (left or right) which will place them in their correct or home positions. We will denote the home distance of a symbol x in permutation p relative to shuffle y as $d_{p,y}(x)$ where y is s or t. If there is no ambiguity subscript denoting the permutation or shuffle can be omitted. For example in $p = (7,4,6,3,2,8,1,5)$ symbols 1,4 and 8 are in correct relative positions since two right shuffles would put them in the first,forth and the eighth place, so $d_{p,s}(1) = d_{p,s}(4) = d_{p,s}(8) = 2$. We will call the smallest number of shifts of the same type that places a symbol in its right place as *home distance* for a particular shuffle transformation. In the example

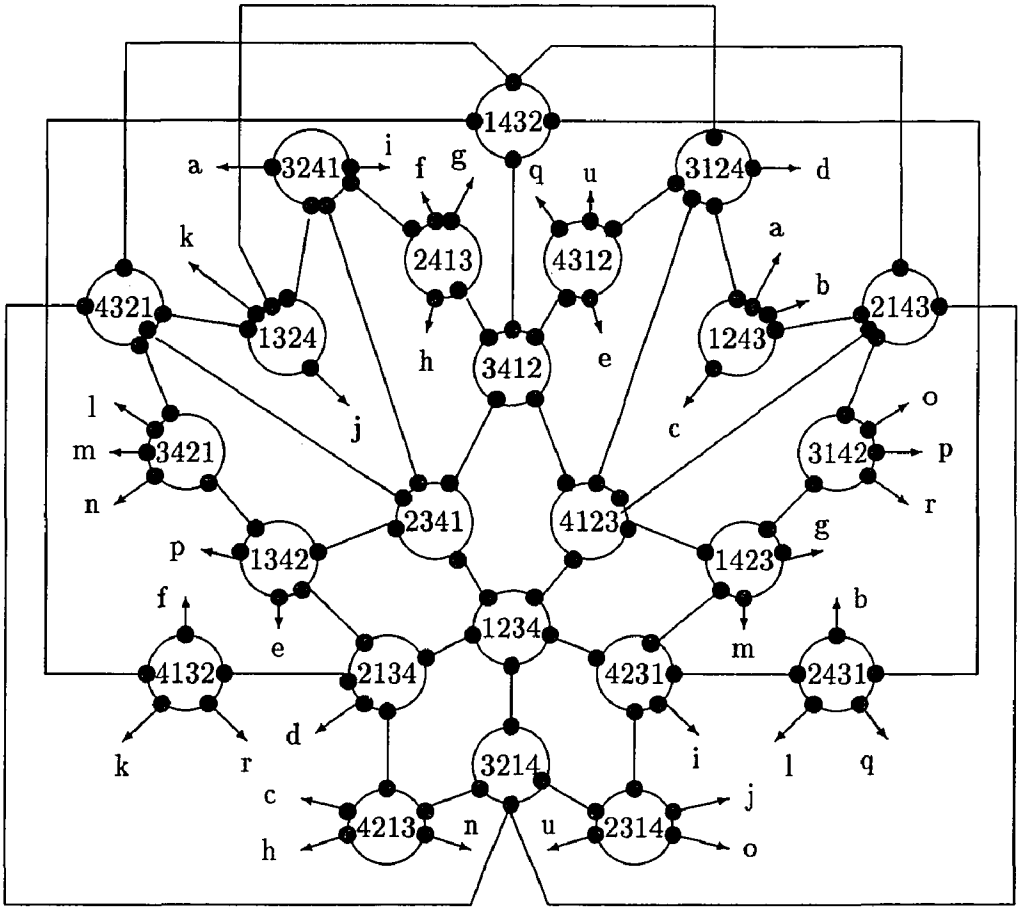

Figure 7: RCS_4

above, using shuffle s, the home distance for 7 is 2 and for symbol 2 is 3. A symbol already in its place has home distance equal zero. It is obvious from the definition that home distance of a symbol is a number between 1 and $n-1$. It is also true that $d_{p,s}(x) = n - d_{p,t}(x)$. Since home distances of all symbols that are in correct relative positions are same, we can talk about home distance of a sequence of symbols in correct relative position (CRP). That is equal to the home distance of any symbol in the sequence. If a is a such sequence of length k we say that a is k-CRP sequence.

We need first the following results.

Lemma 4.6. Any permutation which can be represented as a product of cycles of length greater than 1 must have at least two elements which are in correct relative positions.

Proof: Since the length of all cycles is greater than 1 it means that the home distance of all symbols is greater than 0. We have to consider in this case only transformations made of shuffles in the same direction so possible values are in the set $\{1, 2, \ldots, n-1\}$ We have n symbols however, and thus at least two of them must have the same home distances. □

Lemma 4.7. In any 2-cycle the sum of home distances is equal to n.

Proof: Let us consider a cycle (a, b) where $a < b$. Since a is mapped into b and b into a the whole permutation can be represented as

$$\begin{pmatrix} \ldots & a & \ldots & b & \ldots \\ \ldots & b & \ldots & a & \ldots \end{pmatrix}$$

where dots represent other symbols. It follows that

$$d(b) = n - d(a)$$

so

$$d(a) + d(b) = n$$

□

Lemma 4.8. In any 3-cycle the sum of home distances is n or $2n$.

Proof: We consider a 3-cycle of three symbols a,b and c. Here we have two possibilities. One for (a, b, c) and the other for (a, c, b). In a way similar to the previous lemma we have

$$d(b) + d(c) = n - d(a)$$

for the first case and

$$n - d(b) + n - d(a) = d(c)$$

in the second case. From this follows the claim.□

Proposition 4.11. For any k-cycle in a permutation of n elements the sum of home distances of all symbols is a multiple of n. That is

$$n \leq \sum_{i=1}^{k} d(s_i) \leq (k-1)n$$

Proof: Induction on the length of the cycle. □

Corollary 4.6. The sum of home distances of all symbols in a permutation of n elements is a multiple of n.

$$0 \leq \sum_{i=1}^{k} d(s_i) \leq (m-c)n$$

Here c is the number of cycles of length greater than 1 and m is the number of symbols in those cycles.

Lemma 4.9. There is a routing in RCS_n which sorts any permutation in $\lceil \frac{3}{2}(n-2) \rceil$ steps.

Proof: The following algorithm can sort any permutation. The idea is to form two sequences of numbers such that elements of each are in the correct relative positions to each other. For example $(4,5,1,2,3)$ is made of two ordered sequences $4,5$ and $1,2,3$ while $(4,5,3,1,2)$ is not. It is possible that one of the sequences is empty. Then using shift transformations and applying it on the shorter sequence we obtain the identity permutation.

Algorithm 4.1. input: permutation p

1. We compute the distance from the identity node using formula from Theorem 4.4. If it is less than $\lceil \frac{3}{2}(n-2) \rceil$ we apply the procedure from the theorem otherwise we proceed with the next step.

2. If p is a made of two ordered sequences (one might be empty) go to the last step.

3. Let assume that the number k is in the first position. We find the longest sequence (not necessarily continuous) of symbols that are in correct relative positions. If we have more than one sequence of the same length, we select the one with smaller distance. Let the correct relative position of k regarding the sequence be i_k.

4. If $i_k \neq 1 \bmod n$ we exchange k and the number in position i_k else we have to do shifts. We choose s or t, depending which one will bring a symbol not in the sequence to the first position sooner.If we have more than one choice we use shuffle which will keep home distance of our sequence smaller.

5. go to step 2.

6. Perform shifts (s if the first is smaller, t if the second is shorter) on the shorter of two sequences for the number of times equal to the size of the sequence.

Assume that the length of the longest sequence of elements in correct relative positions is k where $1 \leq k \leq n$. The first step of the algorithm will put at least $k+1$ symbols in their correct relative positions. In each next step at least one more number is placed in the right position within one of the two sequences. The last step will position two symbols in their right places. Ii is possible that we have to perform a shift after each exchange except after the last one. However if we have to do a shift it means that the last exchange positioned correctly at least two more symbols. In the worst case our k-CRP is positioned at the beginning and the end of the permutation. In this case we need to perform at most $\lceil k/2 \rceil$ shuffles before we can start with exchanges.

So the maximal number of transformations we need to create sequences is

$$n - k - 1 + \lceil k/2 \rceil$$

In order to complete routing we have to perform one shift for each symbol in the shorter sequence. This can be done in no more than $\lfloor n/2 \rfloor$ steps. So the total number of steps has the upper bound

$$n/2 + n - 1 - k + \lceil k/2 \rceil = \frac{3}{2}(n-2) + 2 - k + \lceil k/2 \rceil$$

for even n and

$$(n-1)/2 + n - k - 1 + \lceil k/2 \rceil = \frac{3}{2}(n-2) + 1/2(3 - 2k + 2\lceil k/2 \rceil)$$

if n is an odd number.

Now the upper bound in both cases is reached for $k = 1$ and is in fact equal to the diameter of S_n. Hence if the routing for the given permutation is less than the diameter of S_n we apply standard routing in the Star graph with the step 1. If the limit $\frac{3}{2}(n-1)$ is reached, it can happen only in the following cases. If n is odd and the permutation has symbol 1 in the first position and $(n-1)/2$ 2-cycles or one 3-cycle and $(n-1)/2-3$ 2-cycles. We can assume that symbol 1 is not part of k-CRP. Namely in the former case if 1 is in the sequence and $k = 1$ there is only one possible permutation made of 2-cycles. It is described in the proposition 4.12 together with a routing that satisfies the claim. In the later case we have two 3-cycles. If we assume that $k = 1$ it means that all symbols in 2-cycles have different distances from $\{1, 2, \ldots, n-1\}$ all satisfying lemma 4.7. So for two remaining 3-cycles we have three pairs of numbers, say, $x_1, y_1, x_2, y_2, x_3, y_3$ having the following property:

$$x_1 + y_1 = n$$

$$x_2 + y_2 = n$$

$$x_3 + y_3 = n$$

It is impossible to divide this set and to form two subsets of size three which comply with lemma 4.8. So again $k > 1$ and all symbols in cycles have distances greater than 0, and thus 1 is not in the longest CRP cycle. However, since symbol 1 is in the first place we don't have to do $\lceil k/2 \rceil$ shifts and therefore

$$\frac{1}{2}(3 - k + \lceil k/2 \rceil) < 1/2$$

and so for odd n the the claim of the proposition is always true. If n is an even number then the diameter of S_n is reached if the permutation has symbol 1 in the first position and one 3-cycle and $(n-3)/2$ cycles of length 2 or $n/2$ cycles of length

2. In the first case it follows from lemma 4.6 that there must be at least two elements with the same home distances. So the length of the longest sequence of numbers in correct relative positions is not less than 2, i.e. $k > 1$. For the second case let us assume that the length k of any sequence of symbols in relative correct positions is equal 1. We have one 3-cycle and $\frac{n-2}{2}$ 2-cycles plus symbol 1 which is at home. Since elements in 2-cycles satisfy condition $d(a) + d(b) = n$ there are $2(n/2 - 1)$ possible pairs of different distances for their symbols. No symbol in a 2-cycle can have distance equal to $n/2$ because it would mean that both elements in the cycle have the same distance and would imply $k = 2$. So assuming that all elements belonging to 2-cycles have different home distances it follows that for the 3-cycle we have available only 3 different values $n/2$ and a pair x, y such that $x + y = n$. But then $n/2 + x + y$ is not equal neither to n nor to $2n$ and that is impossible by lemma 4.8 Thus we must have at least two elements with the same home distances. So in both cases $k > 1$. If $k > 3$ then $2 - k + \lceil k/2 \rceil \leq 0$ and the claim of the preposition is true. To complete the proof we need to consider cases when $k = 2$ and $k = 3$. For permutations with 1 in the first place it follows as in the proof for odd n. Symbol 1 is not in k-CRP cycle and the distance of this cycle must be greater than 0 because all symbols different from 1 are in 2 or 3 cycles, therefore must have home distances greater than 0. In the remaining case the permutation is made of 2-cycles. So if the selected k-CRP cycle would cause shuffles without exchanges we can select and use the following k-CRP cycle. Each element of the original k-CRP cycle is a part of a 2-cycle. So a sequence made of corresponding elements from 2-cycles forms another k-CRP. The home distance of this sequence is the same as the distance of the first one relative to the shuffle of opposite direction. At the same time the new sequence would not cause shuffles without exchanges.

Whence the upper bound of steps for any routing is $\lceil \frac{3}{2}(n-2) \rceil$. □

Proposition 4.12. The diameter of n-Ring Connected Star Graph is equal to

$$\left\lceil \frac{3}{2}(n-2) \right\rceil$$

Proof: We show that there is a permutation that requires precisely that number of steps to sort. It is $p = (1, n, n-1, \ldots, 3, 2)$. Let us consider first the case when n is odd. Then the following sequence of generators sorts the permutation:

$$e_3 s e_5 s \ldots e_n s s^{(n-1)/2-1}$$

For even n we have the following path:

$$e_3 s e_5 s \ldots e_{n-1} s s^{(n-3)/2}$$

So in this case the length of the MLD path is given as

$$d = n - 2 + (n-2)/2 = \frac{3}{2}(n-2) \text{ for n even}$$

and

$$d = n - 1 + (n-3)/2 = \frac{3}{2}(n-2) + 1/2 \text{ for n odd}$$

Now to complete the proof we have to show that the given permutation can not be routed in a smaller number of steps. Observe that we have to use both exchanges and shuffles. Shuffles alone can not sort $(1, n, n-1, \ldots, 2)$ and if we use only exchanges then it is reduced to routing on the Star graph. The sorting of this permutation on S_n doesn't have smaller time complexity than in this proposition. We first have to remove symbol 1 from the first position so that we can exchange and move other symbols to their correct positions. To position remaining $n-1$ symbols without

bringing 1 again to the first place we need at least $\lfloor \frac{3}{2}(n-2) \rfloor$ steps. That is in the case when we only use exchanges as in the Star graph. Since shuffles do not change relative positions we can use them only to create sequences of numbers in correct relative positions and that amounts to foregoing algorithm. Then at the end we have to shuffle or exchange the symbol 1 again. Otherwise, if we try to create ordered sequences then the number of steps is as was shown above. □

Using previous result we can find a condition for having shifts in minimal length paths. Let c be the number of cycles of length greater than 2 in a permutation p of n symbols. We set k to be the length of the longest RCP sequence and I the number of symbols with home distance equal zero.

Proposition 4.13. Let p be a permutation of n symbols. A MLD path must contain shuffle steps if

1. $c + k - (i-1) > \lfloor n/2 \rfloor$ when $p(1) = 1$
2. $c + k - (i+1) > \lfloor n/2 \rfloor$ when $p(1) \neq 1$

Proof: A MLD path made of exchanges will be longer than the diameter in the first case

$$c + n - i > \frac{3}{2}(n-2) + 2 - k \text{ for even n}$$

and

$$c + n - i > \frac{3}{2}(n-2) + \frac{3}{2} - k \text{ for odd n}$$

from which it follows

$$c + k - i + 1 > n/2 \text{ for even n}$$

or

$$c + k - i + 1 > (n-1)/2 \text{ for odd n}$$

Similarly in the second case. □

One possible way to handle MLD paths is to provide processing elements with tables, stored in the local memories, with information about optimal paths between nodes. There are two possible approaches for this method. One is *central control* and the other is *distributed control* way of controlling routing. In the former approach a particular node (for example the node labeled with identity permutation) contains in its local memory tables and control sequences necessary for routing. Those control sequences can be used by other processing elements with some modifications. In the later case, which is more convenient for long distance communication and distributed systems, each node creates its own tables and control sequences. In either way, tables used for routing can be created in the form of a tree. The root is the current PE. From the root we have $n + 1$ branches representing the nodes with the distance from the root equal one. Continuing from all children labels, and without adding processing elements already in the tree, we can build the whole table. Finding a MLD to a particular node amounts to finding destination's node label in the tree. The path to the label is the minimal length path for that PE. The following result is foundation for central control routing.

Proposition 4.14. Let assume that control sequence $p^k = a_1 a_2 \ldots a_k$ defines a minimal length path from identity to a node yx^{-1}. Then p^k is MLD between nodes x and y.

Proof: By the definition of p^k

$$\sigma(p^k) = yx^{-1}$$

and so is $x\sigma(p^k) = y$ or $xp^k = y$. Let assume that p^k is not the minimal length path between node x and y. Let q^l where $l < k$ be the MLD path. Since $xq^l = y$ it follows that $q^l = yx^{-1}$ or $\sigma(q^l) = yx^{-1}$ which contradict assumption that p^k is the minimal distance routing between identity and yx^{-1} □

Thus a path between nodes x and y is identical to a path from I to yx^{-1} where I is identity. So we need to create the routing tree only for the node labeled with identity and store it in all other processing elements. All needed routing, then, can be calculated from it using the previous proposition. The following algorithm creates the MLD tree for a node.

Algorithm 4.2. *RoutingTree*(p)

Input is the root node label p and the output is the distance tree $T(p)$ for this node.

The first step is initialization of an internal associative array D of size $n!$. It is indexed by node labels. The value stored for each index is called stage and is initially set to 0.

1. Create a tree T with root p
2. Set $stage = 0$
3. For each node $currentNode$ in T with $D(currentNode) = stage$ do
4. For $i = 1$ to $n - 2$ do

   $newLabel = currentLabele_i$.

   If $D(newLabel) > 0$ or $newLabel = p$ next i.

else $D(newLabel) = stage + 1$ and make it a child node of $currentNode$

Repeat last two steps for s and t.

End for.

5. $stage = stage + 1$

6. End For each.

7. return T

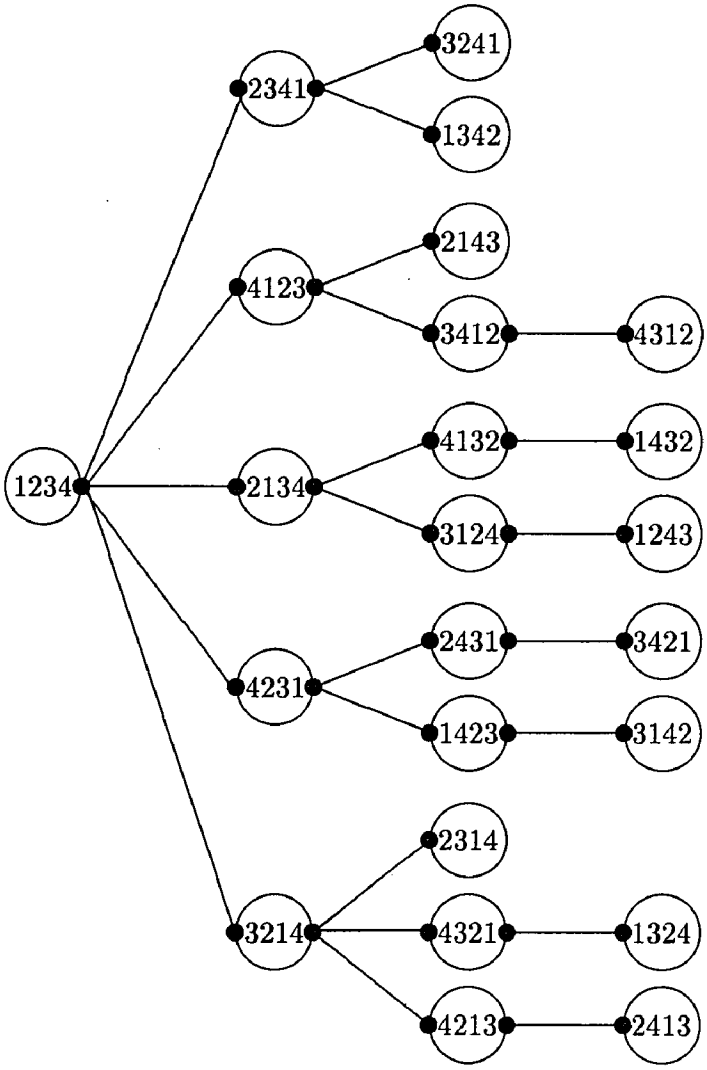

Figure 8: The distance tree for RCS_4

5 Multistage Ring Connected Star Graphs

In this section we address properties and implementations of multistage Ring Connected Star Graphs ($MRCS$). This is also a model in which a topology based on the Star graph is implemented in the form of a multistage interconnection network. The addition of shift operation is crucial for this implementation (which is also the essential difference between Star graphs and RCSs).

In addition to terminologies introduced before we designate identity exchange as e_1. This is the transformation that "replaces" the symbol in the first position with the symbol in that position. In other words e_1 doesn't change the permutation. We begin with the following result.

Proposition 5.1. For any two labels represented by permutations p_s and p_d there is a routing which can be expressed in the following standard form

$$\alpha = \left(\prod_{i=1,k\in I}^{n-1} e_{i_k}^i s \right) s$$

where $I = \{1, 2, \ldots, n\}$ such that $p_s\alpha = p_d$

Proof: Let assume that the source label is $p_s = (s_1, s_2, \ldots, s_n)$ and that the destination node is $p_d = (d_1, d_2, \ldots, d_n)$. In the first step we exchange symbol s_1 with the symbol d_1. If $s_1 = d_1$ we use e_1. After this step we are at the node $(d_1, s_2', \ldots, s_n')$. Here s_i' denotes possible changes caused by the previous exchange.

$$s_i' = \begin{cases} s_1 & \text{if } s_i = d_1 \\ s_i & \text{otherwise} \end{cases} \qquad (1)$$

The exchange operation will then be followed by shift s and the result is $(s_2', \ldots, s_n', d_1)$.

In the next step we replace s'_i with d_2 giving $(d_2, s''_2 \ldots s''_n d_1)$ where again

$$s''_i = \begin{cases} s'_2 & \text{if } s'_i = d_2 \\ s'_i & \text{otherwise} \end{cases} \tag{2}$$

Now shift transforms this into $(s''_2 \ldots s''_n d_1 d_2)$. So each stage is made of an exchange followed by a shift. After $n-2$ pairs of exchanges and shuffles the source is transformed to $(s_x, s_y, d_1, d_2, \ldots, d_{n-2})$ where one of s_x, s_y is d_{n-1} and the other d_n. So the final exchange and two shuffles will result in the destination node. □

Using this result we can create a $n-1$ stage multistage network implementation of the RCS_n network. Each stage is made of shuffle lines which implement s transformations and switching boxes which perform exchanges e_i. Each switching box has n inputs and n outputs as depicted in Figure 9. Let inputs and outputs be labeled as $i_1, i_2, \ldots, i_n$ and $o_1, o_2, \ldots, o_n$ respectively. Each input (output) label is a permutation of n symbols and $i_j = o_j$ for all $i \in \{1, 2, \ldots, n\}$. The input (output) number k is obtained by a shift s from the input (or output) number $k-1$. The last shift results in the first label. The boxes may be in two legitimate states. The first is *straight* when i_k is directly linked to o_i. The second state is *swap* where for a pair of labels i_k, i_j the label i_k is connected to o_j and i_j to o_k. All other inputs are linked to corresponding outputs. Each stage contains $(n-1)!$ switching boxes. From each processing element we have n exchange lines $e_1, e_2, \ldots, e_n$. If the label of the PE is p then line e_j is linked to the input i_j if and only if $pe_j = i_j$. The outputs are linked to processing elements, that have the same labels. That is, each o_j is connected only to the PE element with label o_j.

In this way at each stage one exchange (including e_1) and one shuffle on each input are performed. From the previous result it follows then that $n-1$ stages are enough to enable any point to point routing.

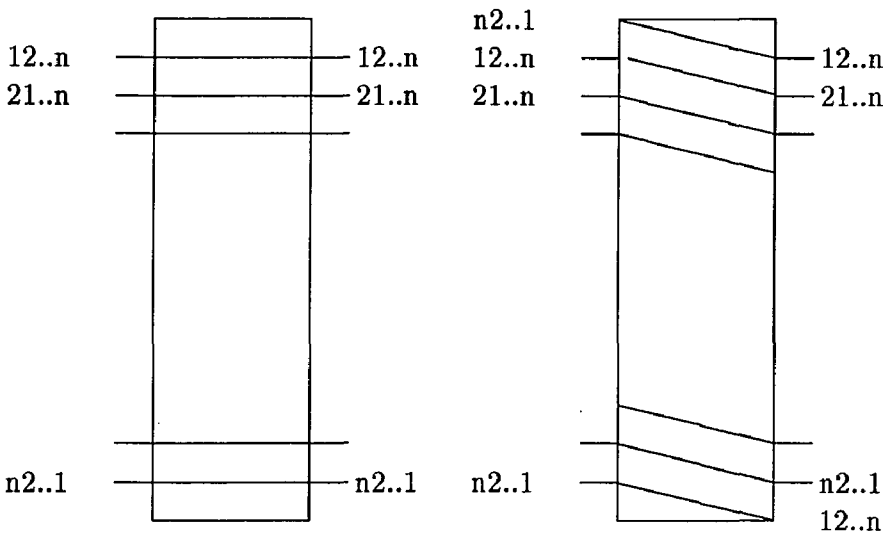

Figure 9: $n \times n$ Switching elements: Straight and Exchange

It is possible to implement this network using only classical 2 by 2 switching elements (see Figure 10.). To prove this we need the next result.

Proposition 5.2. For any two labels represented by permutations p_s and p_d there is a routing which can be expressed in the following standard form

$$\alpha = s \prod_{i=1,k\in I}^{n-1} e_{i_k}^i s$$

where $I = \{1, 2, \ldots, n\}$ such that $p_s\alpha = p_d$

Proof: Let assume that the source label is $p_s = (s_1, s_2, \ldots, s_n)$ and that the destination node is $p_d = (d_1, d_2, \ldots, d_n)$. In the first step we do a shift which results in $(s_2, s_3, \ldots, s_1)$. Then we exchange symbol s_2 with the symbol d_2. If $s_2 = d_2$ we use e_1. After this step we are at the node $(d_2, s_2', \ldots, s_n')$. Here s_i' denotes possible

changes caused by the previous exchange.

$$s'_i = \begin{cases} s_1 & \text{if } s_i = d_2 \\ s_i & \text{otherwise} \end{cases} \tag{3}$$

Next we shift using s and the result is $(s'_2, \ldots, s'_n, d_2)$. In the next step we replace s'_2 with d_3 giving $(d_2, s''_2 \ldots s''_n d_2)$ where again

$$s''_i = \begin{cases} s'_2 & \text{if } s'_i = d_2 \\ s'_i & \text{otherwise} \end{cases} \tag{4}$$

Now shift transforms this into $(s''_2 \ldots s''_n d_2 d_3)$. So each stage is made of an exchange followed by a shift. After $n-2$ pairs of exchanges and shuffles the source is transformed to $(s_x, s_y, d_2, d_3, \ldots, d_{n-2})$ where one of s_x, s_y is d_1 and the other d_n. So the final exchange and a shuffle will result in the destination node. □

We will call the routing described in the previous proposition *canonical routing.*

It is possible to implement the Ring Connected Star graph using a multistage network based on 2×2 switching elements. (See Figure 12.) The stages are made of shuffles followed by exchanges which are implemented using switching boxes. We need again $n-1$ stages plus the final one which is made of only shuffle steps. Similarly, as before, we have $n-1$ shuffle lines starting from each processing element i.e., a line for each possible exchange. Let input and output labels in a switching box be i_1, i_2 and o_1, o_2 respectively. Labels are permutations of n symbols such that $i_1 = o_1$ and $i_2 = o_2$. Furthermore $i_2 = i_1 e_j$ for some $1 < j \leq n$. In this way a swap implements an exchange while straight results in e_1.

Complexity Measure. One way to measure the complexity of a multistage network is to compute the number of switching elements as a function of the the size

of the network.

Proposition 5.3. The number of switching element in a multistage RCS_n is $O(n!(n-1)^2)$.

Proof: From each node we have $n-1$ shift lines that correspond to all possible exchanges. However, each box is shared by two nodes so the number of switching elements in each stage is $\frac{n!(n-1)}{2}$. Since we have $n-1$ stages the result follows. □

When we compare the total number of switching elements in a multistage Shuffle Exchange network of size N, which is $\frac{NlnN}{2}$, we see that the number in multistage RCS_n with a similar size is smaller. It is asymptotically sublogarithmic in its size. That is, if the size of RCS_n is $N = n!$ then the number of elements in all stages is $O(N \ln \ln N)$.

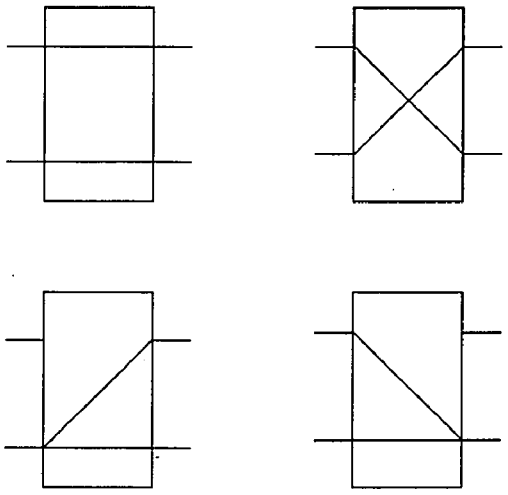

Figure 10: 2×2 Switching Elements: Straight,Exchange, Lower and Upper Broadcast

Control Strategy. The destination label can be used as tag for canonical routing. So each packet contains the label of the destination node initially. In each step of the path this label is modified by transformation performed in the step. This shows the current modification of the label in its transformation from the source into the destination label. An interchange box in the network at stage k, need only examine the position of the destination symbol d_k. If it is not in the first position we need to apply exchanges otherwise straights. The control unit of each switching element needs to perform the following simple algorithm.

Algorithm 5.1. *SetSwitchingBox*(k)

Input: Destination node $dest[n]$ as an array of n symbols

Current modification $current[n]$

1. pos = position of $[dest[k+1]]$ in current
2. If $pos = 1$ set to straight

   else set to e_{pos}

For example canonical routing from the node labeled with $p = (123456)$ to the destination $q = (362154)$ is given by the following sequence where $\rightarrow$ and $\hookrightarrow$ depict exchanges and shifts respectively.

1. $123456 \hookrightarrow 234561$
2. $234561 \rightarrow 634521 \hookrightarrow 345216$
3. $345216 \rightarrow 245316 \hookrightarrow 453162$

4. $453162 \rightarrow 153462 \hookrightarrow 534621$

5. $534621 \rightarrow 534621 \hookrightarrow 346215$

6. $346215 \rightarrow 436215 \hookrightarrow 362154$

Since topologies of all stages are equal we can also devise an implementation which shortens the average length of paths. The idea is to use only one stage of switching elements which are then connected back to corresponding processing elements. In this way if the destination label is reached before all stages are completed we do not have to continue any farther. This makes routing more optimal than on the complete multistage RCS_n networks. An example for RCS_3 is depicted on Figure 11. In this implementation each processing element is connected, as before, to $n-1$ switching boxes. If the label of the processing element is p then it is linked to input ports of switching elements with labels ps. Output ports correspond to $pse_1, pse_2, \ldots, pse_n$ and are linked back to PEs with the same labels, i.e. $pse_1, pse_2, \ldots, pse_n$. The path is determined as before, using canonical routing. At each pass thru the processing elements the current label is compared to the destination label. As soon as those two labels are equal the destination is reached.

We can distinguish two types of stage control. In the case of *box level control* we assume that all switching boxes have independent control and that each one can be set to straight or exchange independently from any other switching element in the network. In *stage level control* all switching boxes belonging to a single stage have to be set in the same way. So they must be all straight or all exchanges. Obviously if we have conflict free routing in stage level control setting we have conflict free routing in the box level control setting.

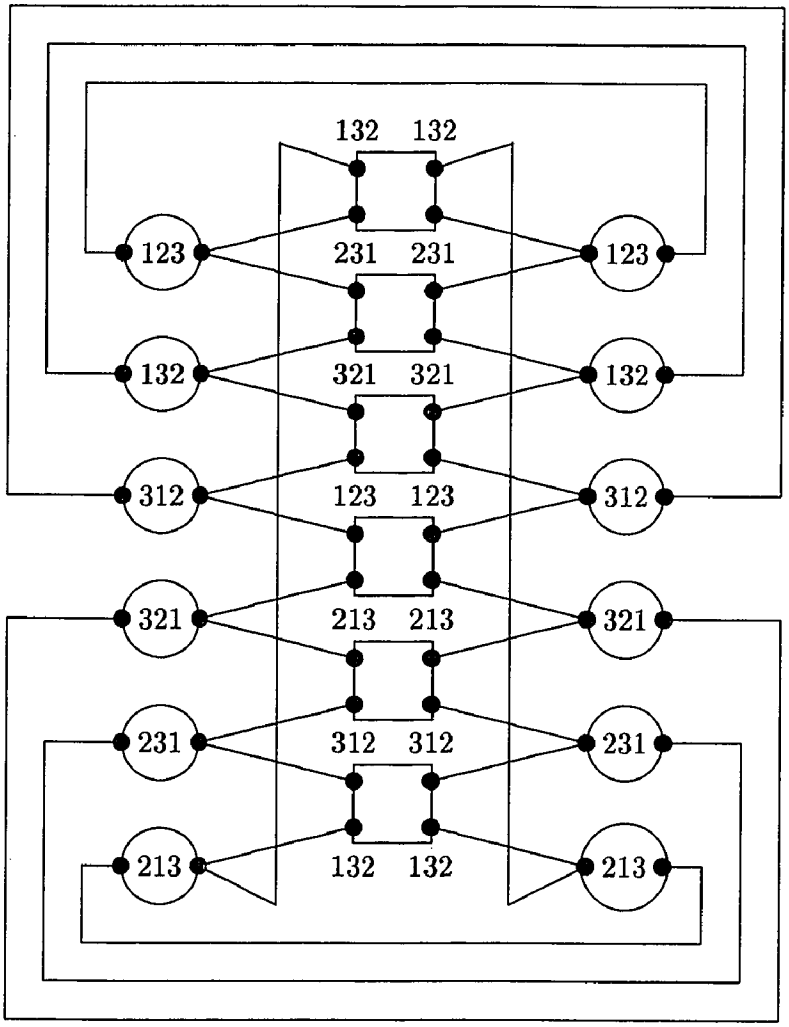

Figure 11: Optimized version of $MRCS_3$.

Let again $p = (s_1 s_2 \dots s_n)$,$p' = (s'_1 s'_2 \dots s'_n)$ be source nodes and $q = (d_1 d_2 \dots d_n)$ and $q' = (d'_1 d'_2 \dots d'_n)$ be the respective destinations. Each packet in its path from a source to a destination enters intermediate stages. The labels of these nodes can be seen as modification of the source label at the action of exchanges and shuffles. Let $\partial_j(d_i)$ denotes the distance of the destination symbol d_i from the beginning of the modified source label p at the stage j. In the case when we talk about source labels we will omit the index. For example if $s_1 = d_4$ then $\partial(d_4) = 0$ or if $s_i = d_j$ then $\partial(d_j) = i - 1$. So if $p = (132546)$ and $q = (523164)$ $\delta(1) = 0$ or $\delta(5) = 3$.

We define $\dot{+}$ and $\dot{-}$ as addition and subtraction relative to n. That is

$$i \dot{+} 1 = i + 1 \text{ for } 1 \leq i \leq n - 1$$

and $n \dot{+} 1 = 1$. Similarly

$$i \dot{-} 1 = i - 1 \text{ for } 2 \leq i \leq n$$

and $1 \dot{-} 1 = n$.

The relationship of distances in a multistage RCS_n is given by the following set of recursive equations.

$$\partial_1(d_k) = \partial(d_k) \dot{+} 1$$

$$\partial_{i+1}(d_k) = \begin{cases} \partial_i(d_{i+1}) \dot{-} 1 & \text{if } \partial_i(d_k) = 0 \\ \partial_i(d_k) \dot{-} 1 & \text{otherwise} \end{cases} \tag{5}$$

Proposition 5.4. If $\partial(d_k) \geq k - 1, \partial(d'_k) \geq k - 1, k \geq 2$ then stage level control conflict occurs at stage k if and only if

$$\partial(d_k) = k - 1 \text{ and } \partial(d'_k) > k - 1$$

Proof: In each stage before k, $\partial(d_k), \partial(d'_k)$ are reduced by 1. Since distances are at least $k-1$ symbols d_k, d'_k will never be a part of exchange. If $\partial(d_k) = k-1$ when we reach the stage k it will have value 0 but it will be $\partial(d_k) > 0$ so we need to set stage to straight for the first path and to exchange $e_{\partial(d_k)-k+1}$ for the second. In the other direction conflict means that we have to apply both settings or that $\partial(d_k) - k + 1 = 0$ and $\partial(d'_k) > k-1$. □

5.1 Point To Point Connections

In a multi stage RCS_n network conflicts can occur in two cases. First in a switching element if we have to apply exchange and straight at the same time. We call this *exchange conflicts.* The second case is when two lines from different exchange boxes, lead to the same processing element at the same time. These we name *shuffle conflicts.* Let assume that we have to connect nodes $p_1 = (s_1 s_2 \dots s_n)$ and $p_2 = (s'_1 s'_2 \dots s'_n)$ to destinations $q_1 = (d_1 d_2 \dots d_n)$ and $q_2 = (d'_1 d'_2 \dots d'_n)$

Proposition 5.5. Canonical routing can cause exchange conflicts in the first stage if and only if $s_2 = d'_2 = d_2 \neq s'_2$.

Proof: Let assume that we have an exchange conflict in the first stage. Let i_u and i_l denote upper and lower input labels of the switching element. Input label in a switching element of the first stage must have the following format assuming that p_1 and p_2 were source labels.

$$i_u = (s_2 s_3 \dots s_n s_1)$$

and

$$i_l = (s'_2 s'_3 \dots s'_n s'_1)$$

We have a conflict if we have to perform on one label e_1 and exchange defined in the switching element, say e_j where $2 \leq j \leq n$, on the other. Thus $s'_2 = s_j \neq d_2$ and $s_2 = s'_j = d'_2 = d_2$. In the other direction if $s_2 = d'_2 = d_2 \neq s'_2$ then in the first stage i_u needs straight setup of the switching element while i_l requires an exchange. Therefore we have a conflict.□

Proposition 5.6. Sufficient condition for conflict free canonical routing in stages 2 to $n-1$, for box level control, is that destination labels differ in all symbols in positions greater or equal to 2 i.e. if $d_i \neq d'_i$ for $2 \leq i \leq n-1$.

Proof: Let assume that conflict has occurred at stage k. If it is a exchange conflict it means that in the same switching elements we have inputs at labels

$$i_u = (t_1 t_2 \ldots d_2 d_3 \ldots d_k)$$

and

$$i_l = (t'_1 t'_2 \ldots d'_2 d'_3 \ldots d'_k)$$

where d_i and d'_j denote already formed sequences of destination symbols and let $d_i \neq d'_i$ for $2 \leq i \leq k$. But by the definitions of the multistage network the switching element performs either e_1 or e_j for some $2 \leq j \leq n$ so i_u and i_l can differ in at most two symbols one of which is in the first position. Since we have at least two destination symbols in each label (because $k \geq 2$) at least one pair of them must be equal and this contradicts the assumption. If we have a shuffle conflict then the output labels of the previous switching elements were equal. Since in a canonical routing it results in placing of k-th destination symbol in the first place it means that $d_k = d'_k$ so for a conflict we must have at least one pair of equal corresponding symbols in destination label. □

Corollary 5.1. Sufficient conditions for conflict free routing on the multi stage RCS_n with box level control are:

1. $s_2 \neq d_2, s'_2 \neq d'_2$ or $s_2 = d_2, s'_2 = d'_2$
2. $d_i \neq d'_i$ for $\forall i$ such that $2 \leq i \leq n-1$

Proof: Follows directly from the previous two propositions. □

Proposition 5.7. Canonical routing defines unique mapping from any element of the set of all source labels to any element of the set of all destination labels.

Proof: In the first stage before d_2 symbol is placed at the beginning of the string we can use any of n exchanges $e_1, e_2, \ldots, e_n$. However in stage k we have positioned symbol $d_2, d_3 \ldots, d_{k-1}$ at the end and so we can apply only one of $e_1, e_2, \ldots, e_{n-k+1}$. In the last stage only applicable options are e_1 or e_2. Thus the total number of different labels that can be obtained starting from a single node is

$$n \times (n-1) \times (n-2) \ldots \times 2 = n!$$

Since by proposition 5.2 there is a canonical path from any node to any other node, it follows that all $n!$ destinations must be different. □

Unlike classical multistage networks, like the Generalized Cube [39] or Omega, where there is only a single possible path between any two nodes, with RCS_n we may have more than one rout from most pairs of sources and destinations. Namely, if we consider all possible exchanges in all stages then the total number of destinations for each source node is n^{n-1}. This number is greater than $n!$ and thus some destinations must have multiple paths. Certainly all paths from a particular source node to some destination node meet in the last shuffle stage. However in the stages before the last shuffle we can have multiple disjoint paths. For example, there are two parallel paths from node (123) into (123) see Figure 13. One is the canonical path:

$$123 \hookrightarrow 231 \rightarrow 231 \hookrightarrow 312 \rightarrow$$

$$312 \hookrightarrow 123$$

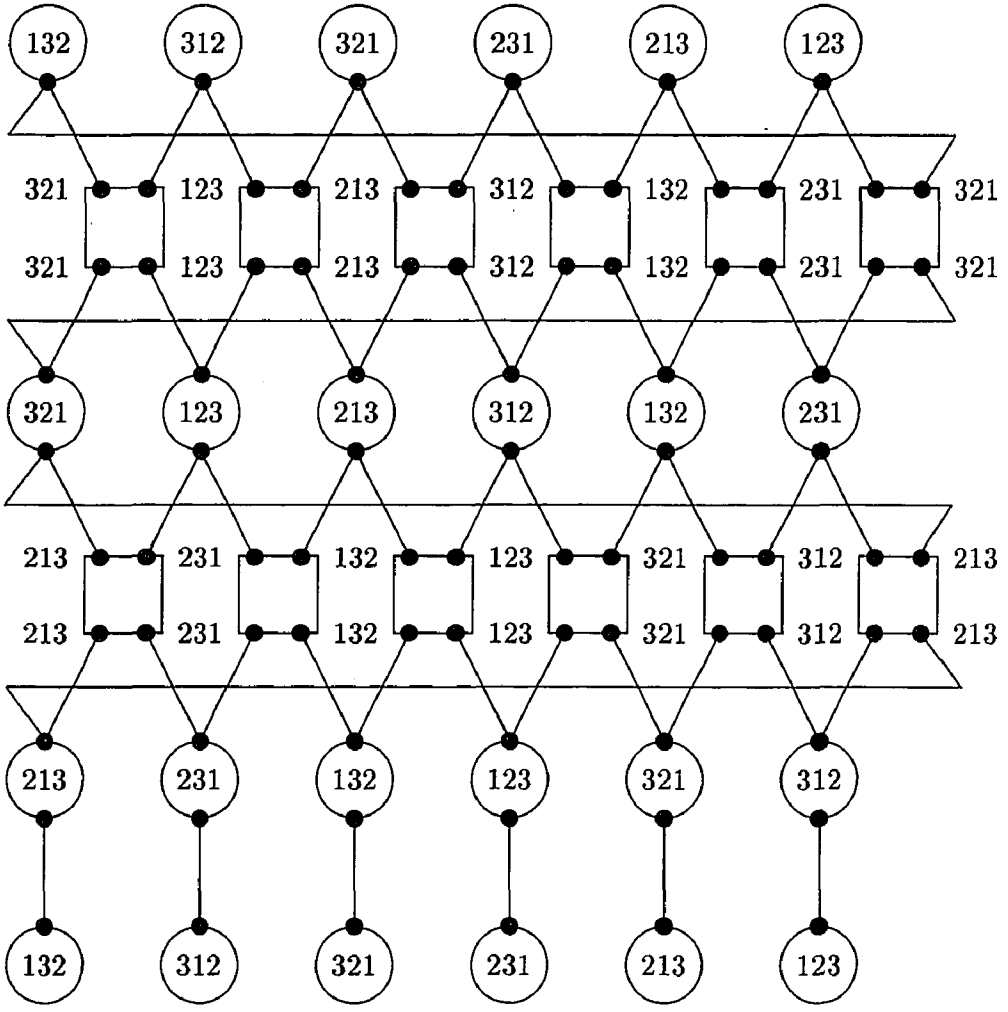

Figure 12: Multistage RCS_3: Circles represent processors and boxes are switching elements. Lines from processing elements to switching elements depict shifts.

The other is given by the following sequence:

$$123 \hookrightarrow 231 \rightarrow 321 \hookrightarrow 321 \rightarrow$$

$$213 \hookrightarrow 312 \rightarrow 123$$

On the other hand there is only one path from (123) to (321) and so not all pairs of source destination labels have possible disjoint paths.

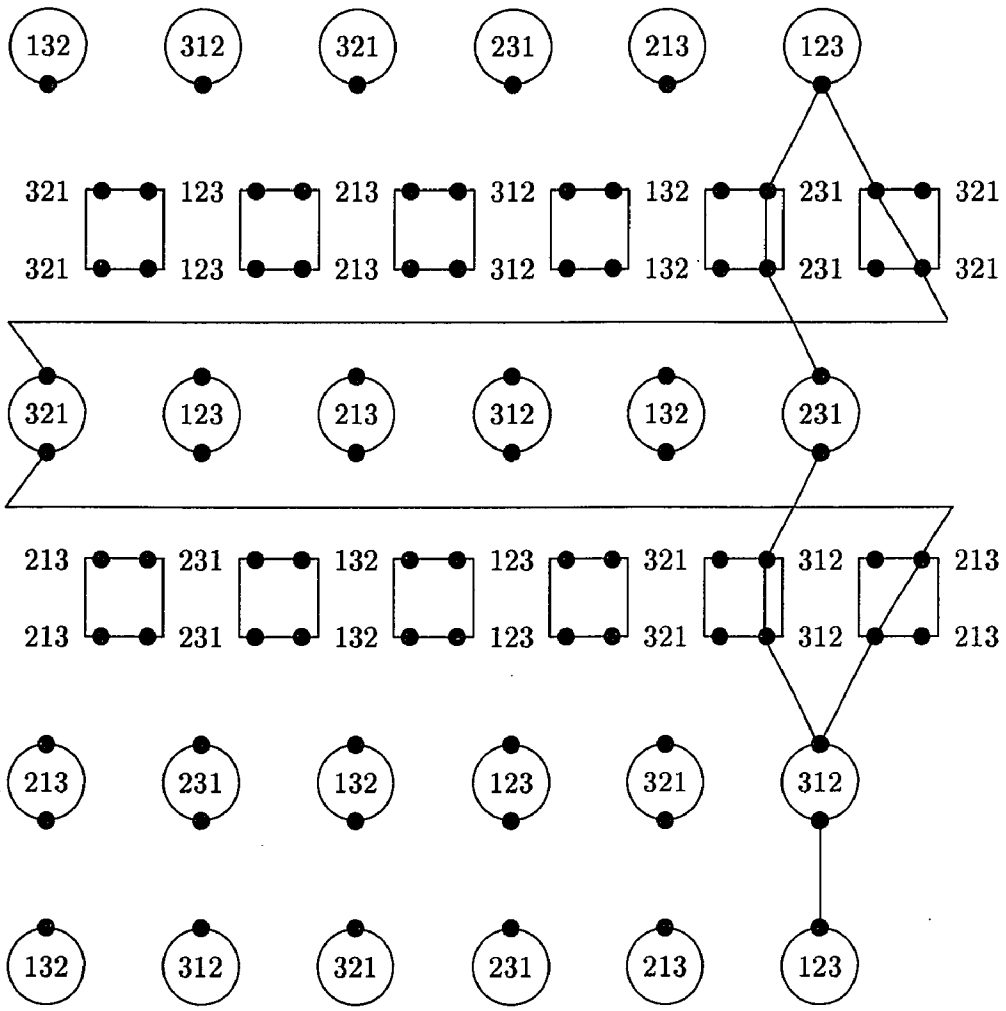

Figure 13: Multistage RCS_3: two paths in routing from (123) to (123)

5.2 Permutations and Broadcasts

In this section we investigate permutations connections. That is one to one mappings from the set of all nodes Σ_n into itself. We can also consider *partial permutation* that is mappings of subsets of Σ into themselves.

Proposition 5.8. Let p_i denote source nodes and q_i corresponding destination nodes where $1 \leq i \leq n!$ in a multistage RCS_n network. If α is a permutation from Σ_n and $p_i\alpha = q_i$ for $\forall i$ then there is a conflict free canonical routing from all source nodes to destination nodes.

Proof: Let assume that the canonical routing from an arbitrary node p_i to its destination q_i is given by

$$\rho = se_{i_1}se_{i_2}\ldots e_{i_{n-1}}s$$

that means that $p_i\rho = q_i$ or since $p_i\alpha = q_i$ it follows $\rho = \alpha$ thus ρ defines a canonical routing for all pairs of corresponding source and destination nodes. Furthermore any component of the control sequence ρ maps Σ onto itself in every stage. It is therefore impossible for any two nodes in any of the stages to be directed to the same destination and cause a conflict. This is true for both box and stage level controls. □

Definition 5.1. Let $P = \{\langle p_i, q_i\rangle | i \in I\}$ where I is a set of indexes from $\{1, 2\ldots, n!\}$. p_i and q_i represent pairs of corresponding source and destination labels. Assume that $\alpha \in \Sigma_n$. Then $\alpha P = \{\langle \alpha p_i, \alpha q_i\rangle | i \in I\}$. If there is a conflict free routing for P on the multistage RCS_n we denote it as $RCS_n \uparrow P$.

Proposition 5.9. Assume that P is a set of source and destination labels as

defined above. If for any q_i, q_j such that $i \neq j, i, j \in I$ we have that $q_i[k] \neq q_j[k]$ for all $k \in \{2, 3, \ldots, n\}$ then $RCS_n \uparrow P$ and $RCS_n \uparrow \alpha P$ for any $\alpha \in \Sigma_n$.

Proof: If $q_i[k] \neq q_j[k]$ for all $k \in \{1, 2, \ldots, n\}$ then by proposition 5.6 we have a conflict free routing for P, i.e. $RCS_n \uparrow P$. Also it must be that $\alpha q_i[k] \neq \alpha q_j[k]$ and thus $RCS_n \uparrow \alpha P$.□

Corollary 5.2. If P satisfies conditions of the previous proposition then

1. $RCS_n \uparrow P'$ where $P' = \{\langle \alpha p_i, q_i \rangle | i \in I, \alpha \in \Sigma_n\}$.
2. $RCS_n \uparrow P''$ where $P'' = \{\langle p_i, \alpha q_i \rangle | i \in I, \alpha \in \Sigma_n\}$.

This result can not be extended to any set of source destination labels that have a conflict free routing. For example, let $P = \{\langle (2134), (3124) \rangle, \langle (4123), (3241) \rangle\}$. There is a canonical routing such that $RCS_4 \uparrow P$. However, $RCS_4 \not\uparrow e_2 P$ because there is a conflict in the first stage.

Definition 5.2. Let P be a set of source destination pairs in RCS_n. The extension P' of P on $RCS_{n+\nu}$, where $\nu > 0$ is $P' = \{\langle p'_i, q'_i \rangle | i \in I, \}$ where $p_i[k] = p'_i[k]$ and $q_i[k] = q'_i[k]$ for all $1 \leq k \leq n$ and $p'_i[k] = q'_i[k] = k$ for $n < k \leq \nu$.

Proposition 5.10. If $RCS_n \uparrow P$ then $RCS_{n+\nu} \uparrow P'$ for $\forall \nu > 0$ where P' is an extension of P.

Proof: For the first $n-1$ stages in $RCS_{n+\nu}$ we use the same routing as in RCS_n except that e_i of path in RCS_n will be replaced with $e_{i+\nu}$ whenever $stage + i > n$. For the remaining ν stages all switching boxes are set to straight, that is we use e_1. □

Example 5.1. Routing in RCS_4 from $(3,1,2,4)$ to $(4,2,3,1)$ is represented with the following path $\alpha = se_2se_3se_2s$. In RCS_6 we have an extension of that routing to a path from $(3,1,2,4,5,6)$ to $(4,2,3,1,5,6)$ given by $se_2se_{3+2}se_{2+2}se_1se_1s$.

The previous proposition and example demonstrate scalability of the multistage Ring Connected Star Graph network.

Proposition 5.11. If p_1 and p_2 are two source nodes and q_1 and q_2 are their corresponding destination labels then if $q_1p_1^{-1} = q_2p_2^{-1}$ we have a conflict free canonical routing for both pairs simultaneously with identical control sequences.

Proof: By assumption $q_1p_1^{-1} = q_2p_2^{-1} = \rho$ for some $\rho \in \Sigma_n$. So

$$p_1\rho = q_1 \text{ and } p_2\rho = q_2$$

Now from the proposition 5.8 it follows that for the pairs $\langle p_1, q_1\rangle$ and $\langle p_2, q_2\rangle$ we have node disjoint paths. □

Proposition 5.12. Necessary condition for conflict free canonical routing, with stage level control, of a set of labels p_i, $i \in S$ to a set of destination nodes q_i,$i \in D$ where $|D| = |S|$ is that there is a bijection $\sigma : S \to D$ which is either identity, i.e. $\sigma(p_i) = p_i$ for all $i \in D$ or it never maps a node to itself, so $\sigma(p_i) \neq p_i\ \forall i \in D$.

Proof: If we have identity mapping, that is, if each node routes to itself the result follows from the previous proposition. If we have both identity and non identity mappings then we must have a conflict. To wit, control sequences for identical transfers contain only shifts and e_1 control vectors. Any non identity mapping must contain at least on exchange e_j different of e_1. In that stage we have, therefore, a stage level conflict between e_1 and e_j. □

Proposition 5.13. Let $p_1 = (s_1 s_2 \dots s_n)$ and $p_2 = (s'_1 s'_2 \dots s'_n)$ be source and $q_1 = (d_1 d_2 \dots d_n)$ and $q_2 = (d'_1 d'_2 \dots d'_n)$ be their corresponding destination labels then if for every $i \in \{1, 2, \dots, n\}$

$$\partial(d_i) = \partial(d'_i)$$

we have a conflict free canonical routing for both pairs simultaneously and the control sequences are identical.

Proof: The condition of the proposal remains true after shifts, that is if $s_i = d_k$ and $s'_i = d'_k$ for some $i, k \in \{1, 2, \dots, n\}$. This remains true after both labels have been shifted in the same direction. On the other hand at stage l it is true that there is j such that $s_j = d_l$ and $s'_j = d'_l$. So at this stage exchange e_j has to be performed on both paths. At each stage the exchange components of the control sequences are identical. Thus the control sequences describing paths are identical and routing is conflict free by proposition 5.8. □

Broadcasting is a case of one-to many mapping. This occurs when both lower and upper states of interchange boxes are used in a path or when more than one shuffle line is used. The later case means that more than one exchange is applied. Again we can consider stage and box level control for broadcasting. In the later case all involved switching elements must be set to exchange or all to broadcast setting in order to avoid stage level conflicts.

Proposition 5.14. There is a canonical, box controlled, broadcasting path from any node into any subset of nodes.

Proof: The only case in which we can have a conflict in box controlled routing is when the same switching box has to be set to straight and switch at the same time.

However this is equivalent to broadcast setting. Thus we have to determine paths for all destination nodes separately and set to broadcast those switching elements where we have to use both settings. □

If we provide queues for the switching elements of all stages, we can extend the previous result for multi-node broadcasting. That is if packets arriving in a processing element enter a waiting queue from which they are dispatched one at a time, we can have broadcast from a set of nodes into another set of nodes such that each source node sends data to all nodes in the set of destination labels.

5.3 Partitioning the Multistage Ring Connected Star Graph

As was defined earlier, the partitionability of an interconnection network is the ability to divide the network into independent subnetworks of different sizes. Also each subnetwork must have all of the interconnection characteristics and capabilities of a complete network of the same type built to be the same size.

Proposition 5.15. The Multistage Ring Connected Star graph of order n ($MRCS_n$) can be partitioned into n independent subnetworks $MRCS_{n-1}$ of size $n-1$.

Proof: The key to partitioning is the choice of the input and output nodes belonging to each stage of subnetworks. By the definition, all labels of the source nodes in $MRCS_n$ are permutations of n symbols. We need to divide these labels between n subnetworks $MRCS^i_{n-1}$ $i \in \{1,2,\ldots,n\}$. In a permutation of n symbols we can fix any of them and consider only the remaining elements. For example, we have $(n-1)!$ permutation with the symbol n in any of n possible positions. Therefore we can choose all permutation with the symbol n fixed in position i to form the i-th $MRCS^i_{n-1}$. So $MRCS_n$ is divided into n subnetworks such that $MRCS^i_{n-1}$ contains initially processing elements with labels that have the symbol n in the position i. After shift steps the fixed symbol, that is n, will be moved for one position to the left. So the first stage of $MRCS^i_{n-1}$ is made of label with n in the position $i-1$. In this stage e_{i-1} lines are ignored. In the stage i symbol n will be brought to the first position. Therefore, in this stage we have to use only e_1, i.e. straight, in the setting for the switching elements. The shift in the next stage $(i+1)$ is going to move n back to the end of labels. This also takes care of an additional stage which is inherited from $MRCS_n$ ($MRCS_n$ has one stage more than $MRCS_{n-1}$). In all other stages, in $MRCS_{n-1}$, assuming that n is in place

j, we ignore shift lines that lead to to the switching elements which implement e_j exchange. For a particular $MRCS^i_{n-1}$ it means that in stage k we ignore shift lines leading to e_{i-k} switching box. Here subtraction in the index is relative to n, i.e. $i-k = n+i-k$ when $i < k$. There can be no conflict between any two subnetworks $MRCS_{n-1}$. At the beginning any two of subnetworks have the symbol n in different places in their ports labels. In each stage shifts will move n for one position in all $MRCS^i_{n-1}$. So the labels of processing elements belonging to different subnetworks will remain different in every stage since we do not use exchanges that could move symbols n.□

Example 5.2. In a multistage network $MRCS_4$ we can form 4 subnetworks $MRCS_3$ consisting of the following processing elements:

$$MRCS^1_3 = \{4123, 4132, 4213, 4231, 4312, 4321\}$$

and the first stage is set to straight. This partition contains the following elements $(1234, 1324, 2134, 2314, 3124, 3214)$ in the first stage.In the stage two lines leading to e_3 type switching elements are not used and the set of processing elements is made of $\{2341, 3241, 1342, 3142, 1243, 2143\}$ Similarly, in the stage three we do not use e_2 switching lines and processing elements are $\{3412, 2413, 3421, 1423, 2431, 1432\}$. Remaining subnetworks are formed in the same way.

$MRCS^2_3 = \{1423, 1432, 2413, 2431, 3412, 3421\}$ stage two is set to straight,

$MRCS^3_3 = \{1243, 1342, 2143, 2341, 3142, 3241\}$ stage three is set to straight,

and

$MRCS^4_3 = \{1234, 1324, 2134, 2314, 3124, 3214\}$ stage four is set to straight.

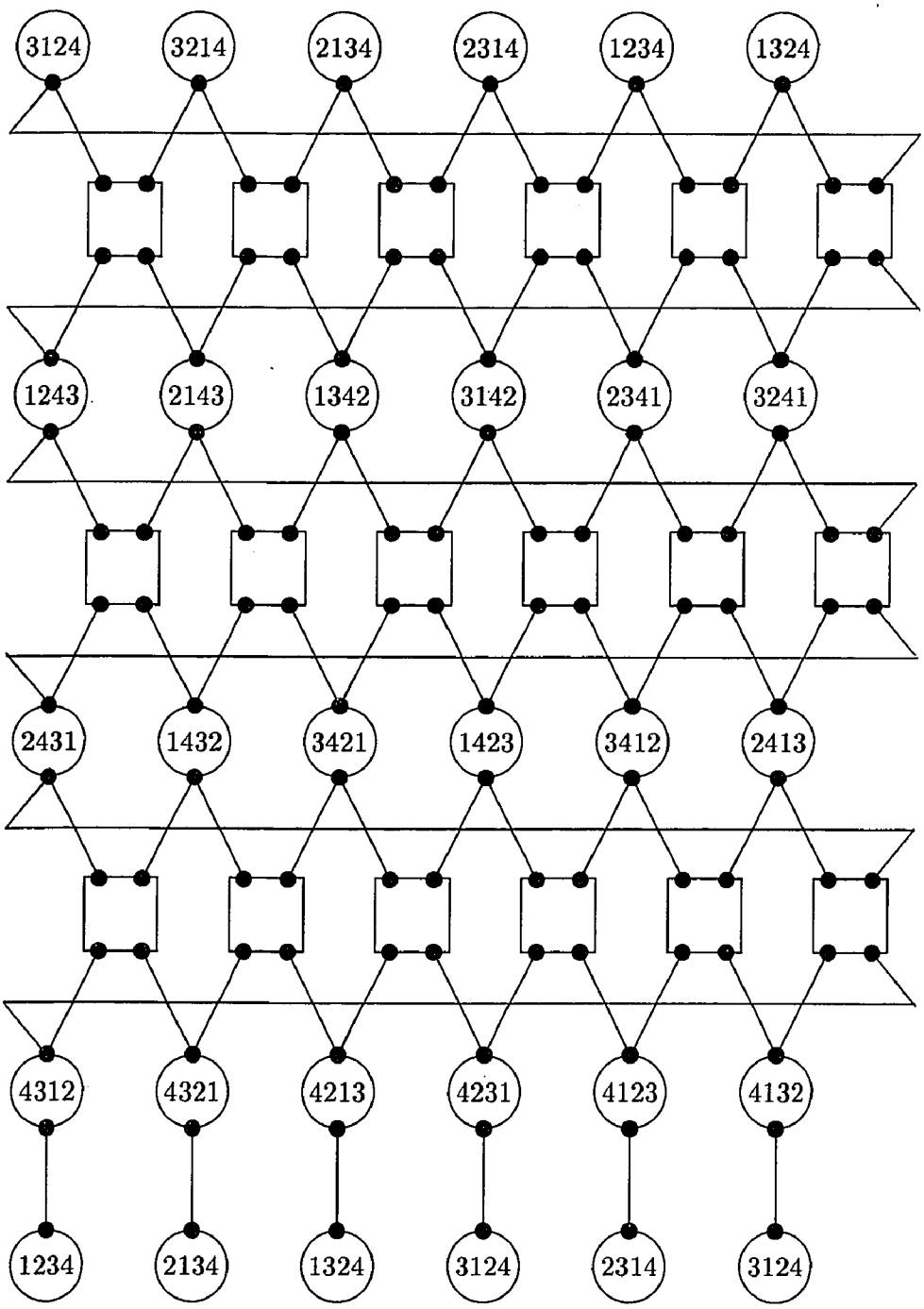

Figure 14: Partition $MRCS_3^4$.

6 Containment Properties

In this section we will study containment properties of Ring Connected Star Graphs. We derive several results related to emulation and embedding of other topologies in RCS_n. Approach introduced in chapter 2. and based on Finite Automata will prove particularly useful.

Let (X, C, ϕ) be a network and $p^k = a_1 a_2 \dots a_k$ be, as before, a control sequence of length k defined over C. We say that p^k is *linear* if it forms a path without cycles. That is for any $x \in X$ and any $1 \leq l < j \leq k$, $x a_l a_{l+1} \dots, a_j \neq x$. p^k is called *circular* if it forms a closed path. In other words if $p'^k = a_2 a_3 \dots a_{k-1}$ is linear and for some $x \in X$ $xp^k = x$ then p^k is circular. Of special interest are *node-independent* control sequences. They produce the same result regardless of the node on which they were applied. Any algorithm based on node-independent sequences is referred as node-independent.

6.1 Ring Connected Star graphs and Linear Arrays

Proposition 6.1. Let (X, C, ϕ) be any RCS_n network and $p^k = a_1 a_2 \dots a_k$ be a control sequence. Then

1. p^k is linear if and only if $\sigma(a_1 a_2 \dots a_i) \neq \sigma(a_1 a_2 \dots a_j)$ for any $2 \leq i < j \leq k-1$.

2. Any linear sequence is node independent.

Proof: (1) If p^k is linear then it contains no cycles and thus no equivalent subsequences. So $\sigma(a_1a_2\ldots a_i) \neq \sigma(a_1a_2\ldots a_j)$ for any $2 \leq i,j \leq k-1$. In the other direction equality of signatures of two subsequences would imply that they are equivalent. Since they are of different lengths and share all but $|i-j|$ control vectors that would mean that the sequence $a_{i+1}a_{i+2}\ldots a_j$ forms a ring.

(2) Let assume that p^k is not node-independent, that is there is some node $x \in X$, $x \neq I$ such that xp^k is not linear. That means that some $i \neq j$ and $i,j \leq k$

$$\phi(xa_1a_2\ldots a_i) = \phi(a_1a_2\ldots a_j)$$

or

$$\sigma(xa_1a_2\ldots a_i) = \sigma(a_1a_2\ldots a_j)$$

which is impossible. □

Proposition 6.2. Let p be a linear sequence. If it is not of the form $p = \beta^{-1}\alpha\beta$, then any p^m where $m < order(p)$ is also a linear sequence.

Proof: We prove by assuming the opposite. Let $p \neq \beta^{-1}\alpha\beta$ but p^m, where m is less than the order of p, has a loop. That means that we have sequences a, b, c and d such that

$$p = ab = cd \text{ and } bp^kc = I \text{ where } k < m \text{ I is identity}$$

From $bp^kc = I$ it follows

$$c = p^{-k}b^{-1} = b^{-1}a^{-1}p^{k-1}b^{-1}$$

Since c is the starting subsequence of p and it ends in the subsequence b. b^{-1} at one and and b on the other end can not overlap in more than one element. If

$b = b_1 b_2 \ldots b_l$ then $b^{-1} = b_l^{-1} b_{l-1}^{-1} \ldots b_1^{-1}$ if they had more than one common element they would have to be equal. That is impossible because p is linear and can not contain subsequences of successive same elements. If they share $b_1 = b^{-1}$ we can assume that

$$\beta = b b_1^{-1} \text{ and } \beta^{-1} = b^{-1} b_1$$

This contradicts the assumption that $p \neq \beta^{-1} \alpha \beta$. □

The following algorithm creates a control sequence which defines a linear path of length $k \leq n$ in a RCS_n. Certainly the length of any linear sequence can not be grater than the total number of edges in RCS_n which is $\frac{n!(n+1)}{2}$. The set of control vectors is $C = \{c_1, c_2, \ldots, c_\alpha\}$ and $c_1 = e_2, c_2 = e_3, \ldots, c_\alpha = e_n$

Algorithm 6.1. Linear(k) Input: length $k \leq (n-1)^2 + n - 2$

Output: Linear control sequence p of length k

1. $p = c_1$
2. If $k \leq \alpha$ then

   For $i = 1$ to k do $p = pc_i$ end for

   Else $m = k - \lfloor \frac{k}{\alpha} \rfloor (n-1)$

   For $i = 1$ to $\lfloor \frac{k}{\alpha} \rfloor$ do

   For $i = 1$ to α do $p = pc_i$ end for

   end for
3. If $m > 0$ then

   For $j = 1$ to m do

For $l = 1$ to α $p = pc_l$ end for

end for

4. Output p

Proof: In the Ring Connected Star Graph we have following relations: $e_2e_3 \ldots e_n = s$, $s^n = I$ and $s^k \neq I, \forall k < n$. Thus concatenation of $n-1$ paths s plus $e_2e_3 \ldots e_{n-1}$ creates a path of length $(n-1)^2 + n - 2$ which can not be a cycle. □

Example 6.1. A linear path of length 13 in RCS_5 created by the algorithm is

$$p^{13} = e_2e_3e_4e_5e_2e_3e_4e_5e_2e_3e_4e_5e_2$$

A linear path of length $n! - 1$, in the Ring Connected Star Graph of order n, can be created with the following recursive procedure.

Algorithm 6.2. Input: integer $n > 2$, length $l < n! - 1$

output: P_l a linear sequence of length l for RCS_n.

1. For $l \leq 4$ paths are e_2, e_2e_3, $e_2e_3e_2$ and $(e_2e_3)^2$.
2. $P_3 = (e_2e_3)^2e_2$ and $Q(P_3) = (e_3e_2)^2e_3$
3. For $i = 4$ to n do
4. $P_i = (P_{i-1}e_iQ(P_{i-1}))^{i-1}P_{i-1}e_iP(P_{i-1})$.

   If $Length(P_i) \geq l$ then exit for loop
5. return P_i with last $Length(p_i) - l$ components removed

Proof: (Outline) The proof is based on the fact that any RCS_n contains the S_n. The Star graph of order n can be built recursively of n copies of S_{n-1}. We can create the path starting with a node in an arbitrary substar S_3 using only e_2 and e_3. This path visits all nodes of S_3 (S_3 is a hexagon) in, say, clockwise direction but without returning to the starting node. This forms the component P_3 in the algorithm above. In the node reached thru the action of P_3 we add e_4 which leads to another S_3. Here we form another path that goes thru all nodes but in opposite direction, say, counterclockwise. This is what $Q(P)$ is. We can continue in this way creating a linear path that visits all nodes in some S_k by making linear paths in all $S_{k-1} \subset S_k$. From the last node in that path we move using e_k to another S_k. Inductively, the path will cover all nodes in substars $S_5, S_6 \ldots$ until we reach the desired length or when the length is $n! - 1$. □

Example 6.2. A linear sequence of length $n! - 1$ in RCS_4 is

$$P_4 = [(e_2e_3)^2e_2]e_4[(e_3e_2)^2e_3]e_4[(e_2e_3)^2e_2]e_4[(e_3e_2)^2e_3]$$

Theorem 6.1. [20] There is a Hamiltonian path in any RCS_n graph. Thus, the Ring Connected Star Graph is a Hamiltonian graph.

Proof: The proof follows from the previous algorithm when the length is set to $n!$. The last step is e_n which makes $H_n = P_n e_n$ a cycle and a Hamiltonian path. □

6.2 RCS and Ring graphs

The Ring Connected Star Graph also contains a rich collection of rings as subnetworks.

Proposition 6.3. Let p^k and q^r be two linear sequences defined in a network $RCS_n = (X, C, \phi)$. The following sequences are node independent and form rings.

1. $p^k(q^r)^{-1}$ (concatenation of p^k and $(q^r)^{-1}$) if and only if $\sigma(p^k) = \sigma(q^r)$.

2. $(p^k)^{m_p}$ where m_p is the order of permutation $\sigma(p^k)$ if $p_k \neq \beta^{-1}\alpha\beta$.

Proof: (1) Since $x(p^k(q^r)^{-1}) = (xp^k)(q^r)^{-1} = x$ when $\sigma(p^k) = \sigma(q^r)$ it follows that $\sigma(p^k(q^r)^{-1}) = I$.

(2) Since each linear sequence defines a non identity permutation the order of that permutation in the group Σ_n is greater than 1. Let it be m. Since $\sigma((p^k)^m) = I$ we have a ring of length km by the proposition 6.2. □

We define a relation $\sim$ on the set of all permutations P by the rule

$$x \sim y \text{ iff } xs^k = y \text{ for some } 0 \leq k \leq n$$

The relation $\sim$ is an equivalence relation.

Definition 6.1. For a given permutation p on n symbols, we define an *orbit* of p as $O_p = \{ps^k | 1 \leq k \leq n\}$. The number of different elements defines the size of an orbit.

Orbits of P are equivalence classes of the relation $\sim$. The size of each orbit is equal n.

Lemma 6.1. There are $(n-1)!$ orbits in a RCS_n graph. Furthermore each element of an orbit O_p belongs to one and only one of the star graphs $S_{n-1,k}$.

Proof: Let $S_{n-1,k}$ be the star graph that has the symbol k in the last position of all its labels. Let us start with a label $p = (i_1, i_2, \ldots, i_n)$ that belongs to S_{n-1,i_n}. Since each shift transformation will make the penultimate symbol of a permutation the last symbol of the result we see that shifts moves p into $S_{n-1,i_{n-1}}$. Thus, each element of an orbit O_p belongs to a different S_{n-1} graph. □

In this way each orbit visits all star graphs S_{n-1} and the set of all orbits forms a collection of disjoint rings that connect all Star graphs S_{n-1}.

Any Ring Connected Star Graph of order n can be viewed as a network made of n copies of $(n-1)$-Star graphs. Another way is to consider orbits i.e. n-rings as subnetworks. In this case we have a network built of $(n-1)!$ disjoint n-rings.

Proposition 6.4. There are $(n-1)!$ different ways to form n-rings in a RCS_n graph.

Proof: Each element of any substar S_{n-1} can initiate a n-ring as its orbit. Since the size of S_{n-1} is $(n-1)!$ the result follows. □

Another type of n-rings can be created using both shuffles and exchanges. Using the following result

$$(1,2)(1,3)\ldots(1,n) = (1,2,\ldots,n)$$

we get

$$(1,2)(1,3)\ldots(1,n)(1,2,\ldots,n)^{-1} = I \text{ where I is identity permutation}$$

Again since permutations form a group and we can apply cancelation it follows that two such rings must be identical or disjoint.

This type of n-rings connects $n-1$ elements of a single substar $S_{n-1,k}$ with an element of another substar $S_{n-1,l}$

6.3 RCSs and Spoke graphs

Proposition 6.5. An n-cube can be embedded in any CS_{2n} complete star graph or $H_n \subseteq CS_{2n}$.

Proof: Because of symmetry we can start with the node labeled $(123\ldots n)$. The following set of transpositions generates the n-cube.

$$S = \{(1,2),(3,4),\ldots,(2n-1,2n)\}$$

Each node is connected to exactly n other nodes defined with transpositions in S. We can assume that each two symbols in a label represent a code for a digit. The complement of such a digit is defined as a transposition of its components. That is the complement of (a,b) is (b,a). Each pair of directly connected nodes forms a graph which is isomorphic to the graph of the cyclic group of order 2, C_2. Therefore the the group of the graph can be viewed as a direct product of n copies of C_2

$$C_2 \times C_2 \times \ldots C_2$$

a group of order 2^n and isomorphic to H_n. □

Proposition 6.6. There are $(2n)!/2^n$ different ways of embedding the n-cube into CS_{2n}.

Proof: The graph of the n-cube, H_n, is a subgraph of S_{2n}. Therefore, the group of H_n is isomorphic to a subgroup of Σ_{2n} and its order is 2^n. By Lagrange's theorem

there are $(2n)!/2^n$ cosets or copies of that subgroup and thus that many disjoint embeddings in CS_{2n}. □

Definition 6.2. n-Spoke graph (Sp_n) is a graph made of a single central node which is directly connected to $n-1$ remaining nodes. So Sp_n has n vertices and $n-1$ edges connecting the central vertex to $n-1$ peripheral nodes.

Proposition 6.7. There is a decomposition of the S_n into a set of $(n-1)!$ Sp_n graphs such that all nodes of $S_{n-1,n}$ are peripheral nodes of spoke graphs. Also no two nodes of $S_{n-1,n}$ belong to the same spoke graph.

Proof: From the definition of $S_{n-1,n}$ we know that all transpositions $e_i, 1 < i \leq n-1$ connect the nodes within the substar. Only e_n will connect a node to a node in another substar. If a node p has symbol k in its first position e_n will define a link to a node in $S_{n-1,k}$. So for any label p in $S_{n-1,n}$ e_n defines a mapping

$$\phi : p \rightarrow pe_n$$

Our claim is that the set of images of ϕ applied to nodes of $S_{n-1,n}$ can be used as central nodes of $(n-1)!$ disjoint spoke graphs. It is obvious that any two images of different nodes must be different. To form spoke graphs around each image, observe that is enough to consider links defined by all exchanges $e_i, 1 < i \leq n$. We only need to show that no two spoke graphs have a common peripheral vertex. That is for any $p, q \in S_{n-1,n}$ and any $1 < i, j \leq n$

$$pe_ne_i \neq qe_ne_j$$

We consider two cases, when p and q have the same symbol in the first position and when they don't. In the first case images of p and q under ϕ belong to the same

substar, say $S_{n-1,k}$. Let assume that $\phi(p)e_i = \phi(q)e_j$ for some i,j. Then

$$(n,\ldots,k)e_i = (n,\ldots,k)e_j$$

and this is possible only if $i = j$ so

$$pe_ne_i = qe_ne_i \text{ or } p = q$$

In the second case images belong to different substars and consequently only transposition e_n can map them into the same vertex. But again it would imply that $p = q$ which is a contradiction. □

Corollary 6.1. Any star graph S_n can contains $(n-1)!$ disjoint n-spoke graphs.

We will describe now an algorithm for finding and distributing of Maximal element on S_n and RCS_n networks. The method will be based on decomposition of star graphs into disjoint spoke graphs. We will assume that initially every node has stored a number in its local memory. At any time step any two connected nodes can simultaneously exchange and compare their respective numbers. At the end of the step both node contain the bigger of the two numbers. On the other hand a node can broadcast its number to only one of its neighbors in a single time step.

Algorithm 6.3. For Finding the Maximal Element in RCS_k or S_k

1. Set n to k.
2. If $n = 2$ exchange and compare elements and go to the last step.
3. In first $n-1$ steps the central node of every spoke graph compares and perhaps exchange values with each peripheral vertex, one at a time. At the end of this phase each central vertex contains the maximal element of its spoke graph.

4. In this step each central node sends its stored number along a line that connects it with $S_{n-1,n}$. That is e_n. If the number is grater than the number stored in the destination node that number is replaced.

5. At this stage nodes of $S_{n-1,n}$ contain maximal elements of all spoke graphs.(Now we continue recursively on $S_{n-1,n}$ starting with $S_{n-2,n-1}$) So we set $n = n - 1$ and go to the second step.

6. when we obtain maximal value in all nodes of $S_{n-1,n}$, that number is sent back thru e_n lines to all central nodes of spoke graphs. In the last step central nodes of all spokes broadcast this value to all peripheral elements.

Time complexity of the Algorithm 6.3. If we denote time complexity of the algorithm on S_n as T_n it is easy to see that we get the following relation:

$$T_n = n - 1 + 1 + T_{n-1} + 1 + n - 2 = 2n - 1 + T_{n-1}$$

Using $T_2 = 1$ we can find that

$$T_n = \frac{(2n)(2n+1)}{2} - n(n+1) - 3 \text{ for } n > 1$$

or

$$T_n = n^2 - 3$$

Thus, we have an algorithm which is almost logarithmic in the size of the network.

6.4 Ring Connected Star graphs and Trees

Several broadcasting algorithms, that is, embedding of spanning trees were proposed for the Star graph [1, 15, 31]. Using a greedy algorithm it is possible to create a

spanning tree with the optimal height of $\lfloor 3/2(n-1) \rfloor$. Another algorithm using recursion for the single node broadcast was developed in [31]. Its time complexity is $\frac{n(n-1)}{2}$. By transforming the Star graph into the domain of $n \log n$ permutation networks [1] one can obtain single node broadcasting with the time complexity equal to $3(n \log n - n/2)$.

Using a simple greedy algorithm it is possible to have single node broadcast in time complexity that is equal to the diameter of RCS. Also a simple congestion free single node broadcasting algorithm can be designed similar to one defined in the previous chapter for finding of the maximal element.

Algorithm 6.4. Single Node Broadcast or Embedding of a spanning Tree.

1. For a given RCS_n or S_n we select a substar, say, $S_{n-1,n}$. Then using for example greedy algorithm from [15] we create a tree Tr_{n-1} for $S_{n-1,n}$.
2. In the next step using e_n exchange we can attach to each vertex of Tr_{n-1} the corresponding spoke graph Sp_n.

Time Complexity. The time complexity of this construction is equal to time complexity for Tr_{n-1} plus 2 (for all spoke graphs), that is

$$T_n = \lfloor 3/2(n-2) \rfloor + 2$$

It is possible to modify the algorithm by recursively applying the same procedure to spoke graphs of $S_{n-1,n}$ etc. If T_n denotes the time complexity for the size n, we get the following relation:

$$T_n = 2 + T_n = 2(n-2) + T_2$$

Since $T_2 = 1$ the solution is $Tn = 2n - 3$. This time complexity is higher than in the procedure outlined before but it is still logarithmic to the size of the graph ($n!$). □

The following algorithm can be used for embedding of complete binary trees in RCS_n. We can embed trees having height up to $n - 1$. Addition is relative to n, i.e. $n + 1 = 2$.

Algorithm 6.5. Btree(n,h)

Input: n and height h

Output: control sequences capable of mapping the tree into RCS_n

1. $Tree(1) = [e_2, e_3]$ and $TreeCompNumb(1) = 2$
2. $j = 2$
3. While($j \leq h$ and $j \leq n - 1$) do
4. For $i = 1$ to $TreeCompNumb(j - 1)$ do

   $TreeCompNumb(j) = 0$

   find the last control vector in Tree(j-1)(i) and its index l

   create two new components by concatenating e_{l+1} and e_{l+2}

   If a new component is already in the tree, use s or t

   $TreeCompNumb(j) = TreeCompNumb(j) + 2$

   end for
5. $j = j + 1$

6. end while

7. output $Tree(h)$

For example the algorithm produces the following sequences when applied on RCS_4 (see Figure 15.).

The first tree is $Tree(1) = [e_2, e_3]$. In the next step the algorithm creates $Tree(2) = [e_2e_3, e_2e_4, e_3e_4, e_3e_2]$. The last step produces

$$Tree(3) = [e_2e_3e_4, e_2e_3e_2, e_2e_4e_2, e_2e_4e_3, e_3e_4e_2, e_3e_4e_3, e_3e_2t, e_3e_2e_4]$$

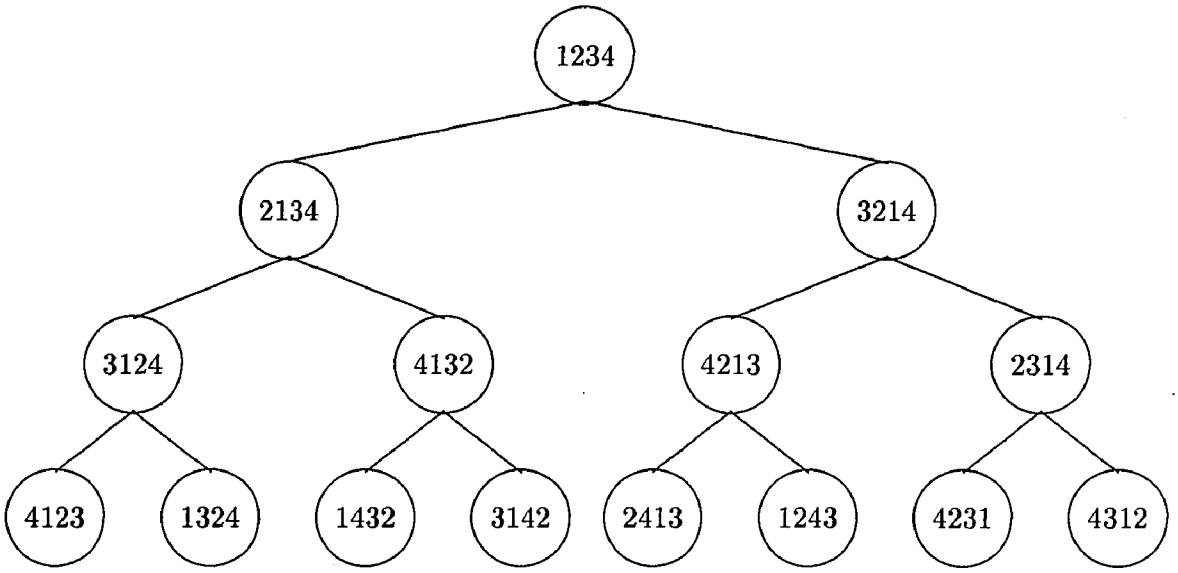

Figure 15: Complete Binary tree in RCS_4

7 Fault tolerance

In this section we investigate some additional properties of the Ring Connected Star Graphs and similar topologies which are related to their fault tolerance. Failures are divided into two groups. In the first group are cases of failures of processing elements. This is a case when one or more nodes are disabled and can not be used as intermediate vertices of paths. In the second group are failures of links or lines that connect nodes. Here one or more edges are disconnected or not perform correctly. Processing elements connected with faulty lines will remain operational as long as there is at least one undamaged line left. Thus, the failures in the first group can be considered as a special case of failures of the second type. To wit, we can view cases in the first group as situations in which all edges that connect one node to remaining PEs are not working.

The main issues that we are interested here is first how to measure fault tolerance of a network. How many nodes or lines can be removed and still maintain the network operational. We also investigate backup paths in a case of failure for a particular routing. Also it is possible to consider fault tolerance regarding to subnetworks like arrays, rings or substars.

One possible indicator for fault tolerance of a network is existence of parallel paths between two processing elements.

Let assume that we are given a path, preferably a MLD path, between two processing elements. The path can be given by a control sequence $p_k = c_1 c_2 \dots c_k$ where $c_i \in \{e_2, e_2, \dots, e_n, s, t\}$.

Proposition 7.1. Any path p_k between two nodes in RCS_n that has a single node failure can be replaced with a path p_l where $l \leq k + 4$.

Proof: A single node failure (excluding source and destination nodes) means that a subsequence of length 2 in p_k is not functioning. In other words we have to replace some $c_i c_{i+1}$ such that $1 < i < k$. We consider all possible cases. Each control vector is an exchange or a shuffle. So cases are

1. $c_i c_{i+1} = se_x$ In this case we use the following identity

$$se_x e_y e_x t = e_{x+1} e_{y+1} e_{x+1}$$

Since $e_x e_y e_x$ just exchanges symbols in positions x, y the identity above claims that we get the same result if we shift a permutation to the right for one place, exchange two symbols and shift back to the left as if we just exchange symbols at positions $x + 1, y + 1$. From here it follows that

$$se_x = e_{x+1} e_{y+1} e_{x+1} t^{-1} e_x e_y$$

So the unoperational sequence of the length 2 is replaced with a backup sequence of length not greater than 6, so the increase in the length is at most 4.

2. $c_i c_{i+1} = e_x e_y$ Here we again use the aforementioned identity to obtain $e_x e_y = s^{-1} e_{x+1} e_{y+1} e_{x+1} t^{-1} e_x$

3. $c_i c_{i+1} = e_x t$ can be replaced in the same manner as above.

4. Cases 1. and 3. in opposite order can be done in the same way using identity

$$te_x e_y e_x s = e_{x+1} e_{y+1} e_{x+1}$$

In the special case when $c_i c_{i+1} = te_n$ or $c_i c_{i+1} = e_n s$ we can use the RCS_n girth identity $te_n = e_2 s$ to create alternative path of the same length.

□

The complete proofs for the following two results can be found in [15]. They can also be proved using combinatorial arguments as in the proofs of the results related to MLD paths in chapter 4.

Theorem 7.1. [15] If A and B are two nodes of n-star graph labeled with permutations P_1 and P_2 having the same first symbol then there are $n-1$ parallel paths between A and B. m of these paths are of minimal distance $c+m$ the remaining $n-m-1$ are of length $c+m+2$.

Theorem 7.2. [15] If A and B are two nodes of n-star graph labeled with permutations P_1 and P_2 having different first symbols then there are $n-1$ parallel paths between A and B. c of these paths are of minimal length $c+m-2$, $m-c-1$ paths are of length $c+m$ and the remaining $n-m$ are of length $c+m+2$.

Proof: (outline only, for both theorems) In both theorems routing from A to B is constructed in a series of steps. In each step a symbol is placed in its correct i.e. destination positions or is moved to the first place for subsequent placement. Parallel paths are formed by all possible ways to build the destination label from the source label. It obviously depends on the number of cycles and the number of elements in them. For example if $A = (43215)$ and $B = (12345)$ we see that $A = (1,4)(2,3)$ we can have four paths of minimal lengths depending on which cycle and which element we start with. Additional paths which have the length equal $c+m+2$ are obtained by first disturbing a symbol already in its correct place

different from 1 (there are $n - m - 1$ of them) by moving it from its place to the first position. Misplaced symbols say $2, 3, \ldots$ are then corrected in this order. Finally the disturbed symbol is placed back to its correct position. (See example following proposition 7.3)□

Proposition 7.2. If A and B are two nodes of complete n-star graph labeled with permutations P_1 and P_2 then the number of parallel paths between A and B is equal $M = \sum_{c_i>2} c_i + \sum_{c_i \leq 2}(c_i - 1)$

Proof: Because of symmetry we can assume that the label P_1 is the identity permutation. We can see every MLD path as a sequence of steps. In each step we perform one transposition in one of the cycles. Each cycle C of a length $c > 2$ can contribute up to c different initial steps. That is to say, we can put one of c symbols in its right position in the first move. If $c = 1$ there is only one possible initial step. Thus we can not have more than $M = \sum_{c_i>2} c_i + \sum_{c_i \leq 2}(c_i - 1)$ different parallel paths because there can be only M different first steps. In the second step, for each of M branches we have to select a cycle and a symbol in it to put in the right position. The number of cycles left can be one less than in the previous step providing that we have had a cycle of the length 2. The number that can be positioned in each branch is also one less than in the prior move. But it leaves enough symbols to be put in the right position in each of the branches so that no one of them have same symbols in the correct positions. Since all branches are different in the prior step any positioning will again yield a different branches. Thus we can continue building all M of the parallel paths and it follows that M is the number of them □

Proposition 7.3. In the Ring Connected Star graph of order n, there are $n + 1$ parallel paths between any two nodes. Two of them are of length no greater than

$2n-1$ and the remaining $n-1$ are of length less or equal to $m+c+2$ where c and m are, as before, the number of cycles and the number of elements in them.

Proof: Since each each RCS_n contains the Star graph of order n, we can use theorems 7.1 and 7.2 to create $n-1$ parallel paths between any two processing elements A and B. This paths must be of length which is not greater than $m+c+2$. These paths also use all exchange links that lead to or from nodes A and B. They are also made only of exchanges. The remaining two parallel paths must start and end with shuffles. Let $A=(s_1,s_2\ldots,s_n)$ and $B=(d_1,d_2,\ldots,d_n)$. In the first case we start with shuffle s. We then place the symbol d_2 in the first place if it is not there already. After that we do s again. We continue in this way by placing d_3 in the first place and then shifting it on the end. At the end we should have the symbol d_n in the first position so that a single shift s will result in the destination label.

The other routing is built around shuffle t. It starts with a t shift, then we place d_n in the first position, shift with t again, place d_{n-1} and so on. This will create the destination by building a sequence of strings $(d_n\ldots),(d_{n-1}d_n\ldots),\ldots,(d_2,d_3\ldots,d_n,d_1)$. The last step is as above a shift, in this case t. The maximal numbers of steps in both routing described above is made of n shifts and up to $n-1$ exchanges.

In all steps of those routings labels will be different from any label created only by exchanges. Using exchanges we gradually place symbols one after another into their correct positions while using shuffles as above symbols will be shifted for one or more place from their destinations. Only in the very last move are they shifted to their destination. We prove this formally for the routing that uses shuffle s, for t the proof is similar. We start from A and create two parallel paths. One using only exchanges, following rules and steps of theorems 7.1 and 7.2, and the other

using shifts s as described earlier. We name the first path, *e-path* and the later s-path. In the first step of routing in RCS_n we perform an exchange which changes places of only two symbols while shifts moves all symbols so the nodes must be different. Let us consider only the second $\lfloor n/2 \rfloor$ symbols. That is $s_{\lfloor n/2 \rfloor}, \ldots, s_n, s_1$ which is what we have after s is performed. At least one of those symbols must be in different position compared to the same symbol in label produced using only exchanges. Now we do exchanges in both routings. They may result in at most one symbol in corresponding labels (one in e-path and the other in s-path) to be in the same positions. From now every two exchanges in the e-path will place at least one element in its correct position. This corresponds to a shift followed by an exchange in the s-path. Since we consider only the block of $\lfloor n/2 \rfloor$ rightmost symbols shifts and exchanges can place no more than one symbol to be equal in both labels for e and s paths. Also each shift moves all elements and therefore would make corresponding labels different. The subsequent exchange can correct this but not for more than one element. Therefore the only way to create equal labels in both paths is to use exchanges in in the e-path to place symbols in such a way, that when a block (possibly modified by exchanges), that we consider, is finally shifted to the leftmost half of the label, all corresponding symbols will be equal. However this is impossible by the nature of exchanges in the e-path. Those exchanges, beginning with the second, place symbols in the their right positions. If we restrict action only to the left side half of the label it would result in $d_2, d_3, \ldots, d_{\lfloor n/2 \rfloor}$ symbols in correct places. On the other hand, as defined earlier, s-paths places those very same symbols in the right side half of its labels. So we can't have two same labels in the first $2\lfloor n/2 \rfloor$ steps. In any step after the e-path will have to place a symbol d_i for $i > \lfloor n/2 \rfloor$ in its correct place, that is, in the the right side section which in the s-path doesn't contain a single symbol which is at its home. The number of symbols

in which corresponding labels in both path differ must only grow afterward. Thus s-path is another parallel path. □

For example consider all parallel paths between $A = (132546)$ and $B = (123456)$ First using only exchanges we obtain the following 5 paths. Four of them correspond to MLD-paths in S_6. We depict an exchange as $\rightarrow$ and shuffles using $\hookrightarrow$.

1.

$$132546 \rightarrow 231546 \rightarrow 321546 \rightarrow 123546 \rightarrow 423516 \rightarrow 523416 \rightarrow 123456$$

2.

$$132546 \rightarrow 312546 \rightarrow 213546 \rightarrow 413526 \rightarrow 513426 \rightarrow 213456 \rightarrow 123456$$

3.

$$132546 \rightarrow 432516 \rightarrow 532416 \rightarrow 132456 \rightarrow 231456 \rightarrow 321456 \rightarrow 123456$$

4.

$$132546 \rightarrow 532146 \rightarrow 432156 \rightarrow 234156 \rightarrow 324156 \rightarrow 423156 \rightarrow 123456$$

5.

$$132546 \rightarrow 632541 \rightarrow 236541 \rightarrow 326541 \rightarrow 623541 \rightarrow 423561$$
$$\rightarrow 523461 \rightarrow 623451 \rightarrow 123456$$

6. And the following two paths use shifts:

$$132546 \hookrightarrow 325461 \rightarrow 235461 \hookrightarrow 354612 \hookrightarrow 546123 \rightarrow 456123$$
$$\hookrightarrow 561234 \hookrightarrow 612345 \hookrightarrow 123456$$

7.

$$132546 \hookrightarrow 613254 \rightarrow 163254 \hookrightarrow 563214 \hookrightarrow 456321 \hookrightarrow 145632$$

$$\rightarrow 345612 \hookrightarrow 234561 \hookrightarrow 123456$$

Ring Connected Star Graphs are also maximally fault tolerant. In chapter 3. we proved that any RCS_n contain $\binom{n-1}{n-k}\frac{n!}{k!}$ k-Star subgraphs.

Definition 7.1. A graph is *m k-star graph fault tolerant* if there remains at least one functional k-star graph after we remove m nodes from the set of vertices.

Proposition 7.4. The RCS_n is at least $\frac{n!}{k!}$ k-star graph tolerant.

Proof: We have an operational k-star graph left providing no more than $\binom{n-1}{n-k}$ k-star graphs have a single faulty node. Whence RCS_n is at least

$$\frac{\binom{n-1}{n-k}\frac{n!}{k!}}{\binom{n-1}{n-k}} = \frac{n!}{k!}$$

k-star fault tolerant. □

Definition 7.2. We say that a graph is *k n-ring fault tolerant* if there remains at least one n-ring when we remove k nodes from the set of vertices.

Proposition 7.5. The RCS_n graph is $(n-1)! - 1$ n-ring fault tolerant.

Proof: Since every element of an arbitrary substar $S_{n-1,k}$ is connected thru its orbit to all other substars it is necessary to remove all nodes from a single substars to make the network n-ring fault tolerant. As long as there is a node left on that substar, we still have one orbit available and thus an n-ring. □

Proposition 7.6. The RCS_n graph contains an n-ring as long as it has undamaged substar $S_{n-1,k}$ and single node of another substar that has symbol k in the positions 1 or $n-1$.

Proof: Let us denote the node outside of the substar $S_{n-1,k}$ as P. P is linked to a node inside $S_{n-1,k}$ using shuffle s. This mapping would move the symbol k to the last position in Ps. We can now apply exchanges $e_i, 1 < i \leq n$ one after another in increasing order of i. The last exchange e_n will be connected to P. □

Definition 7.3. A graph $G = (N, E)$ is termed *n-connected* if for any two nodes $p, q \in N$, they remain connected if no more than $n-1$ nodes including all associated links, are deleted from $N - \{p, q\}$.

Proposition 7.7. The RCS_n $(n+1)$-connected.

Proof: First it is obvious that RCS_n can not be more than $(n+1)$-connected. This is because we can always disconnect any regular graph by removing the number of nodes equal to its fault tolerance. If we remove up to n nodes from RCS_n the graph remains connected and therefore there is a path between any two remaining vertices. □

8 Concluding Remarks and Open Problems

8.1 The RCS_n and the Hypercube

The cube is a very popular topology because of its many attractive properties. It connects a large number (2^n) processors, has a low degree and diameter (n). It is vertex and edge symmetric. Routing algorithms are simple and optimal. Any cube can be recursively decomposed in cubes of lower dimensions. It possess a multitude of paths between vertices thus having a high fault tolerance. It also has little or no degradation of performance in a presence of tolerable number of faulty PEs. In fact, even with a high number of failures when the network becomes disconnected we still have large segments that are operational. Its fault tolerance is $n-1$ while the fault diameter is $n+1$.

Comparison of basic properties

Parameter	n-cube	n-star	RCS_n
Size	$N = 2^n$	$N = n!$	$N = n!$
Node degree	n	$n-1$	$n+1$
Diameter	n	$\lfloor 3/2(n-1) \rfloor$	$\lceil 3/2(n-2) \rceil$
Average distance	$1/2n$	$n + 2/n + \sum_{i=1}^{n} 1/i - 4$	
Fault tolerance	n	n-1	n

However when we compare hypercubes and Ring Connected Star Graphs and star graphs that have a similar number of nodes it is possible to discern that these graphs are in many aspects superior to the hypercube. In table 8.1. we compare

some basic properties of the Cube and the Star graph.

In table 8.2 we show comparison of the actual values for cubes and star graph of a similar size. For example the n-cube connects 2^n vertices, has a degree and a diameter of size which is logarithmic in the size of the cube. The Ring Connected Star Graph, on the other hand, has a degree and a diameter which are *sub-logarithmic* in its size.

Comparison of n-cubes RCS and n-stars

Network	Size	Degree	Diameter	Avg.Distance
5-star	120	4	6	3.7
7-cube	128	7	7	3.5
RCS_5	128	6	4	≤ 3.7
9-cube	512	9	9	4.5
6-star	720	5	7	4.8
RCS_6	720	7	6	≤ 4.8
10-cube	1024	10	10	5
7-star	5040	6	9	5.9
RCS_7	5040	8	8	≤ 5.9
12-cube	4096	12	12	6

Thus, not only asymptotically, but even from a practical point of view, it is evident from Table 8.2, that with its smaller degrees and diameters the Ring Connected Star Graph is superior. Finally, when we compare genuses of the Star network and the hypercube with close number of nodes, we can see that the former is less than

the later [24]. Though both are large numbers, it indicates nevertheless that the star should have a more efficient layout than the hypercube.

8.2 Results and topics for further research

In this thesis we have introduced a new topology, the Ring Connected Star Graph, and investigated its static and dynamic properties. The diameter of the network was shown to be smaller than that for the hypercube or the Star graph of comparable size. We have found an upper limit for the genus of the Ring Connected Star Graph and the necessary condition for a graph to be a Star graph network. Several multi-stage versions of this topology were also described and analyzed. Routing strategies for the network were proposed for both single stage and multi stage implementations. Sufficient conditions for conflict free routing in the multi stage implementation is determined. We have examined also conflict free routing in the case of a broad class of permutations and for single node broadcasting. It was shown how to partition any multistage Ring Connected Star Graph into a collection of independent sub-networks of the same type. Several new algorithms are also described. One which determines the maximal element for both the Ring Connected Star Graph and the Star graph, and another for single node broadcasting. The time complexity of both procedures is logarithmic in the size of the network. With a small constant increase in the degree this network represents an improvement compared to the Star graphs and hypercubes of the same size. It has also a greater number of parallel paths and therefore has higher fault tolerance than the Star graph. Since RCS_n contains S_n graph any algorithm for the Star graph can be applied automatically. We can view RCS also as a network of disjoint rings. They can be used to improve communication between substars of RCS_n. It is also possible to consider rings as separate networks. Mapping and routing from one ring into another is particularly easy and straightforward. So in a case of failure on any of the rings any computation can be transferred quickly to any of the remaining rings. Finally we can see RCS_n (as well

as S_n) as a collection of disjoint Spoke graphs. The spoke graphs can be connected in at least two ways. One is using a single substar S_{n-1} which connects one of the peripheral vertices from each of Spoke graphs. Another way would be using rings to connect central nodes.

Multistage implementation of RCS_n which was introduced in this thesis posses some very interesting features and compares favorably to some standard multistage networks. It can be viewed as an enhancement to multistage Shuffle Exchange networks.

All this shows, we believe, that the Ring Connected Star Graph has a very rich structure which deserves additional research.

In the group of problems related to topological properties of the network we have the question of the number of regions in the graph of order n. This is equivalent to determining the exact value for the genus. The second problem is the sufficient condition for a graph or in a more general context for a subgraph to be RCS graph. Our conjecture is that the necessary condition proved in this thesis is also the sufficient condition. This question is also closely related to the problem of embedding of the Ring Connected Star Graph in other topologies and graphs.

Though several procedures for routing were described in this thesis, the optimal point to point routing is still an open question. Also we still lack a metric that could be used to measure the distance between nodes in the network. Such a metric would indicate if a routing is getting closer to its destination and therefore would facilitate MLD routing. Another approach to optimal routing is farther investigation of types of control vectors that we need in MLD paths. For some pairs of labels optimal MLD paths are made only of exchanges, while for some other we have to use a

combination of shifts and exchanges or only control sequences made of shifts. The problem is to find conditions when each type of control vectors must be used (or must not) in the sequence for MLD paths.

Multistage Ring Connected Star graphs have also some very interesting problems. In some cases we can have multiple paths between nodes and in some cases there is only one, the canonical path. In fact it looks that in $MRCS_n$ for each source label there are exactly n destination that have only one possible path. The remaining $n! - n$ destination have at least two possible routings. It would be very important if we could determine the distribution of those parallel paths. The second problem in this group is the question of sufficient condition for conflict free routing.

The Ring Connected Star Graph appear to be highly fault tolerant and very convenient as a network of other topologies like Stars, Rings or Spoke networks. It looks that it might be possible to embed meshes into RCS_n with a relatively small node dilation. We also need more efficient sorting and broadcasting algorithms. Proposed algorithms do not take full advantage of all properties of RCS_n. Also, due to its similarities to the ordinary Shuffle Exchange, RCS may be more efficient than the Star graph in sorting, for example, of bitonic sequences [41].

References

[1] Akers, S.B. and Krishnamurthy,B. "A group-theoretic model for symmetric interconnection networks" IEEE Tran. on Comp. 1989 Vol.38,No 4, 555-566

[2] Akers,S.B. Horel,D. and Krishnamurthy,B. "The Star graph: An attractive alternative to the n-cube" Proc. Int. Conf. Parallel Processing, pp. 393-400,1987

[3] Akers,S.B. and Krishnamurthy,B. "The Fault Tolerance of Star Graphs" Proc. 2nd Int. Conf. Supercomputing, vol III,pp. 270-276,1987

[4] Akers,S.B. and Krishnamurthy,B. "On Group Graphs and Their Fault Tolerance" Proc. IEEE Tran. on Comp. Vol. 36,No.7,1987

[5] Annexstein,F. Baumslag,M. and Rosenberg,A.L. "Group action graphs and parallel architectures" SIAM Journal on Comp. 1990 Vol. 19,No. 3,544-569

[6] Bavel,Z. *Introduction to the theory of automata* Prentice-Hall,1983

[7] Baumslag,M. *Cayley Networks: A group theoretic approach to the design and analysis of parallel networks* Doctoral dissertation,City University of New York, 1991.

[8] Beineke, L.W and Harary, F. "The genus of the n-cube", Canadian J. Math. vol. 17. pp 494-496, 1965

[9] Benes, V.E. *Mathematical Theory of Connecting Networks*, Academic Press, New York, 1965.

[10] Biggs,N.L. *Algebraic Graph Theory* Cambridge, Cambridge University Press,1974

[11] Bruck, J. Cypher, R. and Soroker,D. "Embedding Cube-Connected Cycles Graphs into Faulty Hypercubes", IEEE Trans. Comp. Vol. 43,NO. 10,1994

[12] Cayley, A. "The theory of groups: a graphical representation", Amer. J. Math. 1,pp 174-176, 1879

[13] Chartrand,G. and Lesniak,L. *Graphs and Digraphs*, Pacific Grove,California,Wadsworth & Brooks,1986

[14] Coxeter,H.S.M. and Moser,W.O.J. *Generators and Relations for Discrete Groups* Springer-Verlag,Berlin,3rd edition,1972.

[15] Day,K. and Tripathi, A. "A Comparative Study of Topological Properties of Hypercubes and Star Graphs" IEEE Trans. on Parallel and Distributed Systems, Vol.5,No 1, 1994

[16] Edmonds, D. "A combinatorial representation for polyhedral surfaces", Notices Amer. Math. Soc. 7, 1960

[17] Feng,T.Y. "A Survey of Interconnection Networks", IEEE Computer,pp. 12-27. Dec. 1981.

[18] Flynn,M.J. "Very High-Speed Computing Systems" Proc.IEEE vol.14,pp.12-27,1966.

[19] Ghozati,S.A. "A New Methodology for the Design and the Analysis of a class of Interconnection Networks", To appear in: Journal of C/EE,1994.

[20] Ghozati,S.A. "High Speed Communication Networks", IASTED, Reliability, Quality Control and Risk Assessment, ACTA Press,p.163-167, Oct. 1993.

[21] Ghozati,S.A. "Topological design of Computer Networks", AMSE Periodical, Vol.22, No.2,p19-27,1994.

[22] Ghozati,S.A. and F.W.Chen,"Communication Algebra for Design and Analysis for Interconnection Networks", AMSE Periodical, Vol.31,No.1 p.55-64,1994.

[23] Hayes,J.P. *Computer architecture and organization* McGraw-Hill,1988.

[24] Hoelzeman, D.A. Bettayeb S. On Genus of Star Graph , IEEE Trans. on Comp. Vol. 43, No. 6, 1994.

[25] Hoffman,C. *Group Theoretic Algorithms and Graph Isomorphism.* New York,Springer-Verlag,1982

[26] Hsu, D. Frank and Wei, S.L. David "Permutation Routing and Sorting on Directed de Bruijn Networks", ICCP, 1995

[27] Jwo,J.S.,Lakshmivarahan,S. and Dhall,S.K "Characterization of node disjoint(parallel) paths in star graphs",

[28] Krishnamoorthy,M.S. and Krishnamurthy,B. "Fault Diameter of interconnection Networks" Comp. and Math. with Applications,19xx

[29] Lawrie, D.H. "Access and alignment of data in an array processor", IEEE Trans. Comput., Vol. C-24, pp. 1145-1155, Dec. 1975

[30] Madabhushi, S.V.R., Lakshmivarahan, S., Dhall, S.K., "A note on Orthogonal graphs" in 5th. Int. Parallel Processing Symp. 1991, pp. 432-437.

[31] Misic,J. and Jovanovic,Z. "Communication Aspects of the Star Graph Interconnection Network" IEEE Trans. on Parallel and Distr. Systems, Vol. 5, No. 7, 1994

[32] Palis,M., Rajasekaran, S. and Wei, D.S.L. "Packet routing and PRAM Emulation on Star Graphs and Leveled Networks", Journal of Parallel and Distributed Computing,vol. 20, no. 2, 1994,pp. 145-157

[33] Peterson, J.L., "Petri nets", Comput. Surveys, Vol. 9, pp. 223-252, Sept. 1977.

[34] Quinn,M.J. *Parallel Computing; Theory and Practice* McGraw-Hill,Inc. 1994

[35] Roberts,F. *Applied Combinatorics* Prentince-Hall,1984

[36] Ringel, G. "Über die kombinatorishe Probleme am n-dimensionalen Würfel und Würfelgitter", Abh. Math. Sem. Univ. Hamburg, vol. 10. pp 10-19,1955

[37] Santoro,N. and Khatib,R. "Labeling and implicit routing in networks", Comp.Journal,Vol.28, No. 1,pp. 5-8,1985

[38] Scherson, I.D. "Orthogonal graphs and the analysis and construction of a class of multistage interconnection networks" in Proc. Int. Conf. Parallel Processing, Vol. 1, pp. 380-387,1990

[39] Siegel,J.H. *Interconnection Networks for Large-Scale Parallel Processing* Lexington Books,Lexington,1985

[40] Starke,P. *Abstract automata* North-Holland,1972

[41] Stone,H.S. Parallel Processing with the Perfect Shuffle, IEEE Trans. on Comp.,Vol. C-20, No. 2, 1971

[42] Thomson Leighton,F. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes* Morgan Kaufman Publishers, 1992

[43] White, Arthur *Graphs, Groups and Surfaces* North-Holland, 1984

[44] Youngs,J.W.T. "Minimal imbeddings and the genus of a graph", J. Math. Mech. 12,pp 303-315, 1963