INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

ANALYSIS OF WORMHOLE ROUTINGS IN CAYLEY GRAPHS OF PERMUTATION GROUPS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirement for the

degree of

DOCTOR OF PHILOSOPHY

By

SUNG CHUL BOO

Norman, Oklahoma

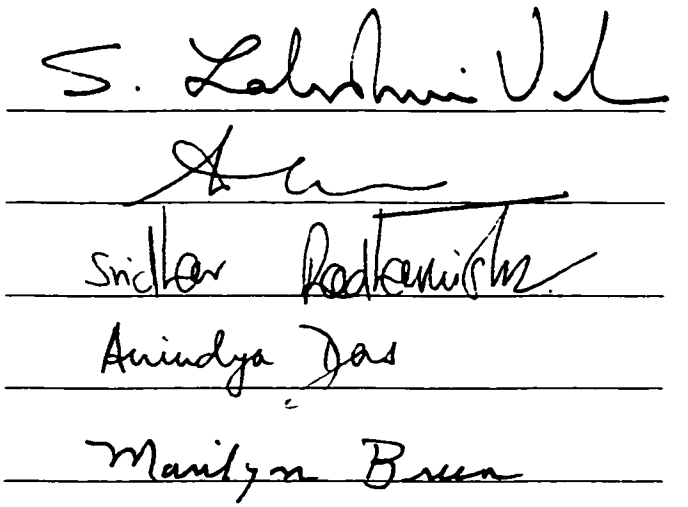
1999

ANALYSIS OF WORMHOLE ROUTINGS IN CAYLEY GRAPHS OF PERMUTATION GROUPS

A Dissertation APPROVED FOR THE

SCHOOL OF COMPUTER SCIENCE

BY



ACKNOWLEDGEMENT

I would like to express my appreciation to Dr. S. Lakshmivarahan, my advisor, for his support and encouragement during my school years at the University of Oklahoma. Special thanks to the Dr. S. Dhall for his valuable discussion sessions, and to the other members of my committee, Dr. Radhakrishnan, Dr.Das in the School of Computer Science and Dr. Breen in the Department of Mathematics.

I also want to thank to Dr. Boppana, the University of Texas at San Antonio, for sharing his simulator, which provided the tools of our simulation study. Mr. Chen and Mr. Hegde gave help in the early stages of this work too.

This dissertation would not been completed without support of the Korean Air Force, and I am deeply grateful for their generous support. Finally, I would like to thank my late father, Kyebong Boo, and my mother, Kyunghee Song, for their inspiration and support, and my wife, Yunsuk, and my two boys, Steve and Young, for their love.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

ABSTRACT

Over a decade, a new class of switching technology, called wormhole routing, has been investigated in the multicomputer interconnection network field. Several classes of wormhole routing algorithms have been proposed. Most of the algorithms have been centered on the traditional *binary hypercube*, *k-ary n-cube mesh*, and *torus* networks. In the design of a wormhole routing algorithm, deadlock avoidance scheme is the main concern. Recently, new classes of networks called Cayley graphs of permutation groups are considered very promising alternatives. Although proposed Cayley networks have superior topological properties over the traditional network topologies, the design of the deadlock-free wormhole routing algorithm in these networks is not simple. In this dissertation, we investigate deadlock free wormhole routing algorithms in the several classes of Cayley networks, such as *complete-transposition* and *star* networks. We evaluate several classes of routing algorithms on these networks, and compare the performance of each algorithm to the simulation study. Also, the performances of these networks are compared to the traditional networks. Through extensive simulation we found that adaptive algorithm outperformed deterministic algorithm in general with more virtual channels. On the network performance comparison, the *complete transposition* network showed the best performance among similar sized networks, and the *binary hypercube* performed better compared to the *star* graph.

CHAPTER 1

INTRODUCTION

For the past decades, the interest of communication performance in massively parallel computers (MPC) has been investigated in several directions such as network topology, switching technology, and routing algorithms. Several research efforts of network topology led to the design of an efficient interconnection network, which connects processors to memories or processors among themselves. Massively parallel computers, in general, may be loosely classified into two groups: (1) multicomputers with shared memory organization and (2) multicomputers with non-shared or distributed memory organization. In a shared memory organization, an array of processors and a common bank of memory units are connected through a fast bus or through many types of multistage dynamic interconnection networks. In this organization, the processors communicate by writing onto and reading from the common or shared memory. Cray X/Y-MP, DEC GIGAswitch, NEC Cenju-3, IBM RP3, and IBM SP [DuNiYa1997][LaDh1990] are some examples of this type of organization. In a distributed memory organization, on the other hand, an assemblage of processors, with each of the processors having its own local memory, are connected in *static* or *fixed* interconnection networks. Examples of this type of organization include the Intel Paragon, MIT J-Machine, Cray T3D/E, and, nCUBE, etc. From the point of view of communication on an interconnection network, switching technique determines when and how processors are communicating each other by means of several resources.

Traditionally, *circuit switching* from the telephone communication field and *packet switching* from the computer communication fields were the most popular techniques in the first generation of parallel computers. The routing algorithm is very closely coupled with switching technology. A routing algorithm decides a path of message delivery in the network and provides arbitration among the competing resources.

1. 1 Interconnection networks

We will concentrate on the distributed memory organization architecture of multicomputers. Interconnection networks for distributed memory multicomputers are usually modeled by the underlying graph of network where the *nodes* denote the *processors* and the *edges* denote the communication lines between the processors. We will use the terms interconnection networks and graphs interchangeably in this dissertation. In the network, each node is a complete computer with its own memory, a local processor, and other functional units. The communication lines (or links) connect processors through the router. A router is connected to a processor through the internal channels or links. Therefore, each processor can execute its own task independently without the burden of the communication task since a router solely deals with a communication task. Generic processor architecture in a multicomputer is illustrated in Figure 1-1. The new generation of the multicomputer interconnection network uses a *direct* network such that each node has a point-to-point or direct connection to some of the other nodes.

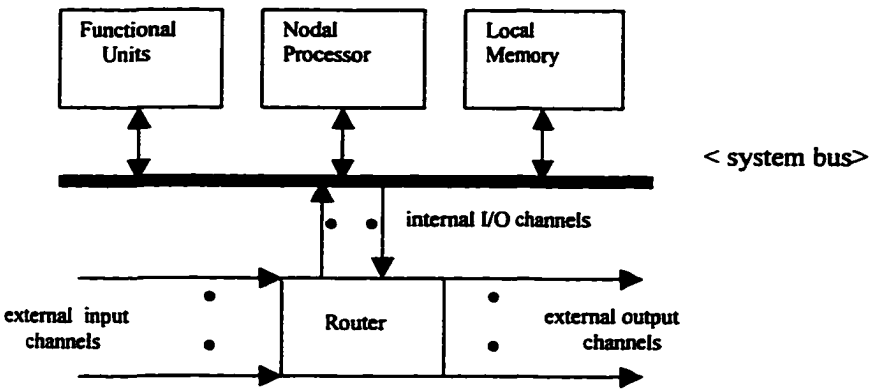


Figure 1-1. A generic node architecture and router connections.

The implementation of distributed parallel computer interconnection network topologies has been centered on *binary hypercube*, *k-ary n-cube torus* or *mesh* networks due to their simplicity of construction and scalability of networks without disruption of network properties [BoCh1996][DaAo1993][DaSe1987][GrPiBe1994]. A new representation of interconnection network called Cayley graph based on finite group theory has been investigated for over a decade. These networks are believed to have better performance in terms of *degree*, *diameter*, *connectivity*, and *fault tolerance* against to the number of processors. In the literature, various families of Cayley graphs of permutation groups are proposed as very promising network topologies for an interconnection network with its salient features such as *symmetry*, *recursive scalability*, and *efficient routing*, etc, [AkKr1989][Misi1991] [LaJwDh1993]. Among the conventional network topologies, a *binary hypercube* is considered as a standard by which to compare other topologies. Because a binary hypercube has very simple and

abundant topological properties, it provides very good criteria to analyzes the performance of other types of network topologies.

1. 2 Switching schemes

Given a topology of the network, there are four basic classes of switching schemes: *circuit* switching, *packet* switching, *virtual cut-through*, and *wormhole routing*. The switching techniques determine when and how the internal switches are set to connect from the router inputs to the outputs and the time at which the message components may be transferred along the routing path. These techniques are tightly coupled with a *flow control* mechanism for the synchronized transfer of units of information between routers. The flow control protocol of a network determines how resources are allocated and how message collisions over resources are resolved. Switching techniques differ in the relationship between the size of the unit of transmission and message flow control. Recently wormhole routing has become one of the most popular switching techniques for the new generation of parallel computers since it reduces the effect of the path length compared to packet switching in computer communication field [DaSe1987] [NiMc1993]. Also, its use of resources is much less than *virtual cut-through* switching [LeKl1979]. In wormhole routing, a message is divided into a sequence of small indivisible units called *flits*. Only the *head* flit contains the source/destination information needed in the routing, and all the other flits follow the header much like cars in a train; hence, the name wormhole routing. Therefore, when the header flit advances along the network path, the remaining flits are transmitted in a

pipeline fashion. If the header flit encounters a busy channel, all the flits remain inside the network until that channel becomes available instead of storing whole message into one local memory as in packet switching or virtual cut-through switching. However, the spreading of flits in the network may cause a deadlock if the routing algorithm does not have avoidance or prevention schemes. For dealing with deadlock problems, researchers have taken one of the two approaches. First, deadlock free routing algorithms have been proposed by restricting routing freedom with multiplexing a physical channel into several virtual channels [DaSe1987]. Second, deadlock detection and recovery schemes were introduced with the traditional *time stamp* strategy [AnPi1995]. An example of generic wormhole router architecture is illustrated in Figure 1-2.

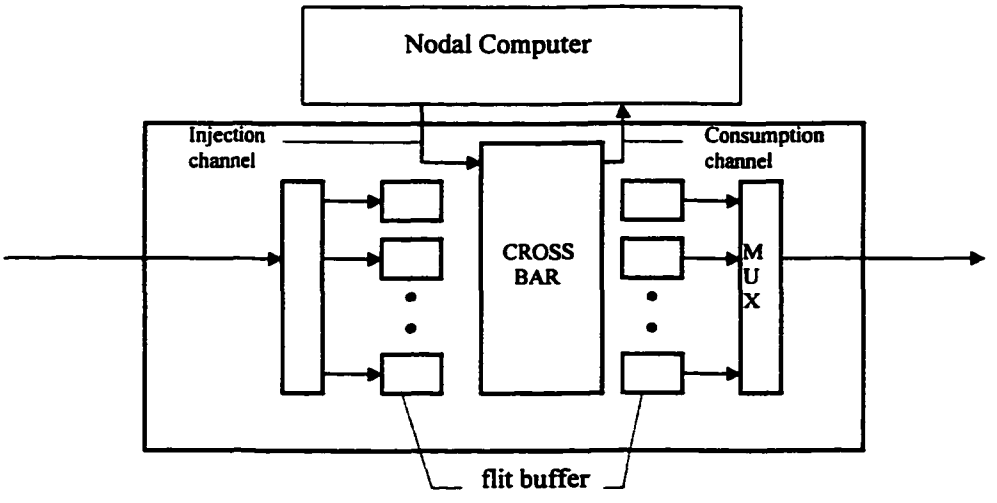


Figure 1-2. A generic wormhole routing router architecture.

1. 3 Routing algorithms

Given the network topology and switching scheme, there are three types of routing algorithms in terms of how the routing paths are determined. In deterministic routing, the routing path of a message is determined by its source and destination address. Thus, the routing paths are fixed before a head-flit departs the source node. Adaptive routing algorithms use information about network traffic and/or channel status to avoid congested or faulty regions of the network. In randomized routing algorithms, routing paths are determined randomly by the random generator. A routing algorithm is said to be *minimal* if the routing messages take the shortest paths to their destination. A minimal and fully adaptive routing algorithm allows a message to be routed via any one of the shortest paths between its source and destination. In *non-minimal* routing, a message can take the longer path instead of the shortest path, but the message should be guaranteed that it is delivered to its destination.

1. 4 Objectives

In this dissertation, we seek deadlock free routing algorithms in the family of Cayley networks of permutation groups based on the wormhole routing switching architecture. Several schemes of deadlock-free routing algorithms have been proposed and evaluated mainly on the *binary hypercube*, *k-ary n-cube, torus* and *mesh*. We extended these deadlock free routing algorithms to new classes of Cayley graphs and evaluated the performance of each algorithm through the simulation study. Also, the performance of the new class of networks is compared to traditional networks against

similar size of nodes. In our research on wormhole routing algorithms in Cayley graphs of permutation groups, our main concerns are performance sensitivity with routing freedom and the use of resources (flit buffer, virtual channel, etc.) for deadlock freedom with various traffic patterns and loads.

1.5 Organization of this Dissertation

This dissertation is organized as follows. In Chapter 2, we present the basic definitions and mathematical backgrounds of interconnection networks. Also, the properties of the Cayley networks of interest including *star* and *complete transposition* graphs and the issues of wormhole routing are included. In Chapter 3, we present the framework of wormhole routing algorithm characteristics and deadlock free conditions. Then we describe five classes of the deadlock free algorithms in *complete transposition* and *star* graphs. Empirical performance evaluations of routing algorithms, based on extensive simulation and comparisons of several networks, are given in Chapter 4. Chapter 5 contains comments and concluding remarks. Some of the results in this dissertation have already been published and are contained in [BoChLaDh1998].

CHAPTER 2

MATHEMATICAL PRELIMINARIES

In this chapter, we describe the basic mathematical tools and definitions of group theory needed for building interconnection networks. Also, we introduce various switching technologies and their characteristics with the model of message transmission time. There are several considerations leading to the choice of a network in the development of parallel computers on a commercial basis. The simplest of these include the *degree*, *diameter*, *regularity* and *parallel paths* between a pair of vertices in the graph. Here, a new representation of interconnection networks called Cayley graphs, based on finite group theory, is investigated. For the purpose of comparison, the basic concepts of switching technologies and interesting issues in wormhole routing are introduced.

The basic definitions from the graph theory are given in section 2.1. In section 2.2, we introduce Group theory. Several classes of the Cayley graph of permutation groups are presented in section 2.3. The comparisons of the switching technologies are presented in section 2.4. Several issues leading to the design of wormhole routing algorithms are given in the section 2.5. Section 2.6 includes the models of communication in the multicomputers and Section 2.5 contains a summary of this chapter.

2.1 DEFINITIONS AND CONCEPTS FROM GRAPH THEORY

Interconnection Networks have been modeled by a direct graph $G = (V, C)$, where the vertices of the V represent the set of *processing nodes*, and the *edges* of the graph C represent the set of communication links.

- Degree

The number of links connecting a node to other neighboring nodes is called *degree*. In a direct graph, the *out-degree* of a node is the number of links leaving it, and the *in-degree* of a node is the number of links entering it. A graph is called *regular* if all the nodes have the same degree.

- Neighbor and Bipartite

If (u, v) is a link in a graph $G = (V, C)$, where $u, v \in V$ and $(u, v) \in C$, then we say that node v is adjacent to node u or node v is a neighbor of u or vice versa. A bipartite graph is an undirected graph in which V can be partitioned into two sets V_1 and V_2 such that $(u, v) \in C$ or $(v, u) \in C$ implies either $u \in V_1, v \in V_2$ or $u \in V_1, v \in V_2$.

- Path, Diameter, Cycle

A *path* of length k from a node u to a node u' in a graph $G = (V, C)$ is a sequence $\langle v_0, v_1, v_2, \ldots, v_k \rangle$ of nodes such that $u = v_0$, $u' = v_k$, and $(v_{i-1}, v_i) \in C$ for $i = 1, 2, \ldots, k$. The length of a *path* is the number of links in the path. A path is *simple* if all nodes in the path are distinct. In a simple path, the length of the path is called the *distance* between u

to u'. Diameter is the maximum distance between two nodes in the graph. In a direct graph, a path $<v_0, v_1, v_2, \ldots, v_k>$ forms a cycle if $v_0 = v_k$ and the path contains at least one edge. A *hamiltonian path* is a simple path such that the path visits every vertex exactly once. If a graph contains a *hamiltonian cycle*, the graph said to be a *hamiltonian graph*. A graph with no cycles is called *acyclic*.

2.2 GROUP THEORY

- **Groups**

Let S be a nonempty set and '$\cdot$' be a binary operation and be called the *product*, then we define a group $(S, \cdot)$ with following conditions [BeBl1996]:

(i) Closure: For all $a, b \in G$ the element $a \cdot b$ is a uniquely defined element of S.

(ii) Associativity: For all $a, b, c \in S$, we have $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.

(iii) Identity: There exists an *identity* element $e \in S$ such that $e \cdot a = a$ and $a \cdot e = a$ for all $a \in S$.

(iv) Inverse: For each $a \in S$ there exists an inverse element $a^{-1} \in S$ such that $a \cdot a^{-1} = e$ and $a^{-1} \cdot a = e$.

A group is said to be *abelian* if $a \cdot b = b \cdot a$ *(Commutative)* for all $a, b \in S$. The set of all integers under addition is an abelian group and is denoted by $(\mathbf{Z}, +)$.

- **Subgroups**

Let S be a group, a subset H of a group S is a *subgroup* if and only if:

(i) $a, b \in H \rightarrow a \cdot b \in H$.

(ii) $e \in H$, where e is the identity of S.

(iii) $a \in H \rightarrow a^{-1} \in H$.

- **Group of integers modulo *n***

Let n be a positive integer. The set $\mathbf{Z}_n$ of integer modulo n is an abelian group under *addition* of congruence classes. The group $\mathbf{Z}_n$ is finite and $|\mathbf{Z}_n| = n$. An example of $\mathbf{Z}_4$ addition table is given in Table 2-1.

TABLE 2-1. Addition in $\mathbf{Z}_4$.

	0	**1**	**2**	**3**
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

- **Direct product**

Let G_1 and G_2 be groups. The set of all ordered pairs (x_1, x_2) such that $x_1 \in G_1$ and $x_2 \in G_2$ is called the direct product of G_1 and G_2, denoted by $G_1 \times G_2$.

Let $< n > = \{ 1, 2, \ldots, n\}$ and let

$$p = \begin{pmatrix} 1 & 2 & \ldots & i & \ldots & n \\ p_1 & p_2 & \ldots & p_i & \ldots & p_n \end{pmatrix}$$

with $p_i \in < n >$ and $p_i \neq p_j$ for $i \neq j$, that is , p is a *permutation* of $< n >$, where p_i denotes the object at position i. For simplicity in notation, we use the standard one-dimensional array representation, that is, we write p as:

$$p = p_1\ p_2\ \ldots\ p_n \tag{2. 1}$$

Let S_n denote the set of all permutations over $< n >$. Let $p, q \in S_n$. Define an associative binary operation '$\cdot$' (called the product) as $(p \cdot q)_x = p(q_x)$, that is, '$\cdot$' denotes the usual(right to left) composition of functions. $(S_n, \cdot)$ forms a group called the symmetric permutation group. In general, $p \in S_n$ can be expressed as the product of k disjoint cycles and l invariants:

$$p = c_1\ c_2\ \ldots\ c_k\ e_1\ e_2\ \ldots\ e_l \tag{2. 2}$$

where $c_i = (a_{i1}\ a_{i2}\ \ldots\ c_{it_j})$, $t_j \geq 2$, $a_{ij} \in < n >$ are distinct for $1 \leq j \leq t_j$ and $1 \leq i \leq k$ and $p_{ei} = e_j$ for $i \leq j \leq l$. For example, a permutation $p = 312465$ can be expressed as the product of cycles and invariants as:

$$p = (1\ 3\ 2)\ (5\ 6)\ (4). \tag{2. 3}$$

For convenience, the invariants are deleted when p is expressed by its cycle structure such that $p = (1\ 3\ 2)\ (5\ 6)$.

Thus, if $|c_i|$ denotes the length of the cycle, it follows that:

$$n = l + \sum_{i=1}^{k} |c_i| \qquad (2.4)$$

The first term is the number of invariants, and the second term represents total number of objects placed inside of the cycles.

Cycles of length two are called *transpositions* and play a very basic role in the study of permutation groups. Let $T_{ij} = (i, j)$ denote the transposition which is a permutation that swaps the objects at position i and j. Clearly, $T_{ij} = T_{ij}^{-1}$ and $T_{ij} \cdot T_{kl} = T_{kl} \cdot T_{ij}$ where $i \neq j \neq k \neq l$. Any cycle in the form of (2, 3) can be expressed as the product of the transpositions. Let $c = (a_1\ a_2 \cdots a_t)$ be a cycle of length t. Then it can be verified that

$$c = T_{a_1 a_t} \cdot T_{a_1 a_{t-1}} \cdots T_{a_1 a_3} \cdot T_{a_1 a_2}.$$

In a permutation p, if $p_j < p_i$ for $i < j$, then p_j is said to constitute an *inversion* in p. As an example in p = 13452, the pairs 3 and 2, 4 and 2, 5 and 2 are the three inversions. A permutation is said to be an *odd* or *even* permutation if the number of inversions in p is odd or even. Transpositions are naturally odd permutations since they swap a low position object for a high position object.

Let $\Omega \subseteq S_n$. Define for $k \geq 2$

$$\Omega^k = \Omega^{k-1} \cdot \Omega = \{p \cdot q \mid p \in \Omega^{k-1}, q \in \Omega\}$$

where $\Omega^0 = \{ I \}$. If Ω is such that

$$S_n = \cup_{k \geq 0} \Omega^k,$$

then the set Ω is called the generator of S_n.

2.3 CAYLEY GRAPHS OF PERMUTATION GROUPS

We define Cayley graph of permutation groups as follows: Let Γ be any finite group with identity I and let Ω be a finite set of generators for Γ such that:

(a) if $g \in \Omega$, then $g^{-1} \in \Omega$ and

(b) (b) $I \notin \Omega$. Given (Γ, Ω), define a *Cayley* graph $G = (V, E)$ where $V=\Gamma$ and E is defined as follows;

$$E = \{(x, y)_g \mid x, y \in V \text{ and } y = x \cdot g\}.$$

The two directed edges $(x,y)_g$ and $(y, x)_{g^{-1}}$ are viewed as an <u>*undirected*</u> edge connecting x and y. Since Ω is a generator set for Γ, clearly G is a <u>*connected*</u>, <u>*uniform*</u>, <u>*simple*</u> graph of degree $|\Omega|$.

2.3.1 Complete transposition graphs (CT_n)

Consider the finite permutation graph S_n of all $n!$ permutations of n objects under the normal product operation. Let $\Omega_{CT} = \{ (i, j) \mid 1 \leq i < j \leq n \}$. The *complete transposition graph* (CT_n) is defined as the Cayley graph of (S_n, Ω_{CT}). Clearly for $n \geq 3$, CT_n is a *regular* or *uniform* graph of degree $\frac{n(n-1)}{2}$, order n!, diameter n-1, and the number of edges $\frac{n(n-1)n!}{4}$. This graph was introduced in [LaJwDh1993] and has been extensively studied from the point of view of packet routing. The routing in Cayley graph is a sorting process based on a greedy strategy. The routing process from node q to r in CT_n is the same as finding the routing path from $q^{-1} \bullet r$ to I [LaJwDh1993]. As an example in CT_7, let p = (1 4 3) (2 5) (6 7) and I = 123456. One of the routing paths from p to I is as follows:

$$p = 4513276 \xrightarrow[(1\ \ 4)]{} 351427 \xrightarrow[(1\ \ 3)]{} 1534276 \xrightarrow[(2\ \ 5)]{} 1234576 \xrightarrow[(6\ \ 7)]{} 1234567 = \mathrm{I}$$

From the above example, it is obvious that the sorting step cycle c_i in the form of (2, 2) for $1 \leq i \leq k$ is $| c_i | - 1$ steps. Thus, the shortest distance between p to I is defined as follows:

$$(\mathrm{I}, p) = \sum_{i=1}^{k} (| c_i | - 1) = n - k - l \qquad (2.5)$$

Examples of CT_3 and CT_4 are illustrated in Figure 2-1. It is shown that CT_n simultaneously contains many well-known Cayley Graphs such as *Star graph, bubblesort graph, and binary hypercube* as subgraphs.

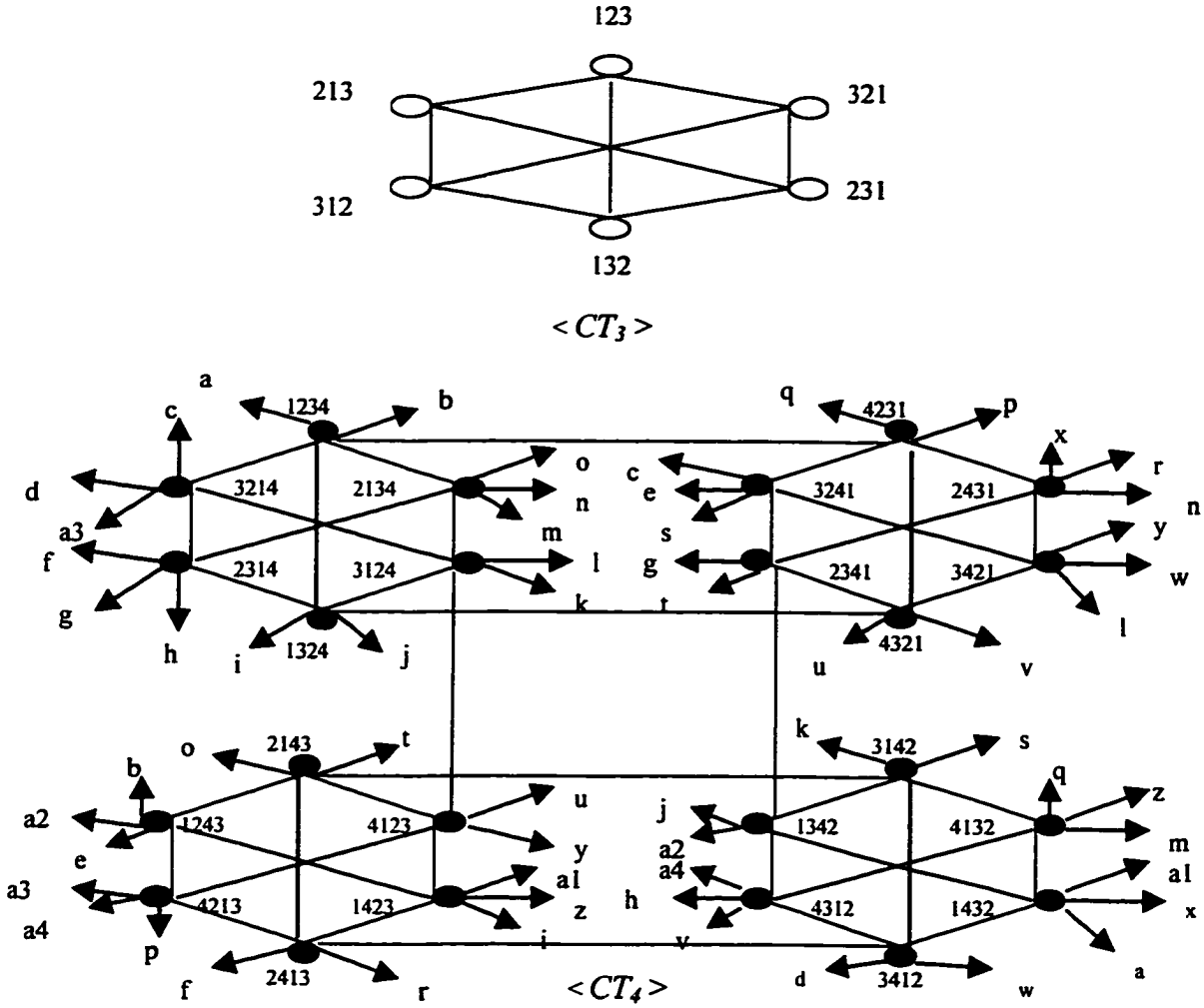


Figure 2-1. Examples of CT_3, CT_4.

LEMMA 2.3.1: CT_n is *bipartite* and *hamiltonian* [Jwo1991].

PROOF: Since a node can be obtained from its adjacent node by applying the corresponding generator in Ω_{CT} and all the generators are transpositions, it can be seen that any two neighboring nodes in CT_n are in opposite parity. Hence we can partition the

network nodes evenly into two groups, even parity nodes and odd parity nodes as the identity node I is even parity. It follows that the graph is bipartite. Finding the hamiltonian cycle in CT_n is simple. Since CT_n is constructed with n copies of CT_{n-1}, once we find the hamiltonian cycle of CT_{n-1}, the connection of the n separated cycles is straightforward.

2.3.2 Star graphs (ST_n)

Let $\Omega_{ST} = \{(1, j) \mid 2 \le j \le n\}$. The *Star* graph ($ST_n$) is defined as the Cayley graph of (S_n, Ω_{ST}). ST_n is also a regular and connected graph of degree $n-1$, order $n!$, diameter $\lfloor 3(n-1) / 2 \rfloor$ and the number of edges $\frac{(n-1)n!}{2}$. The distance between any pair of u to v in ST_n can be expressed by the distance from I to p, where p is the product of $u^{-1} \cdot v$ and the form of (2. 1).

$$d(\mathrm{I}, p) = \begin{cases} n + k - l & \text{if } p_1 = 1 \\ n + k - l - 2 & \text{otherwise} \end{cases} \qquad (2.6)$$

For example, the distance between 23415(u) and 41253(v) in ST_5 is calculated as follows; $u^{-1} = 41235$, $u^{-1}v = 34152 = (1\ 3)\ (2\ 4\ 5) = 5 + 2 - 2 = 5$.

An attractive feature of these two classes of graphs is that it has a small diameter over the number of nodes and has a great deal of symmetry much like the *hypercube*. Examples of ST_n are illustrated in Figure 2-2.

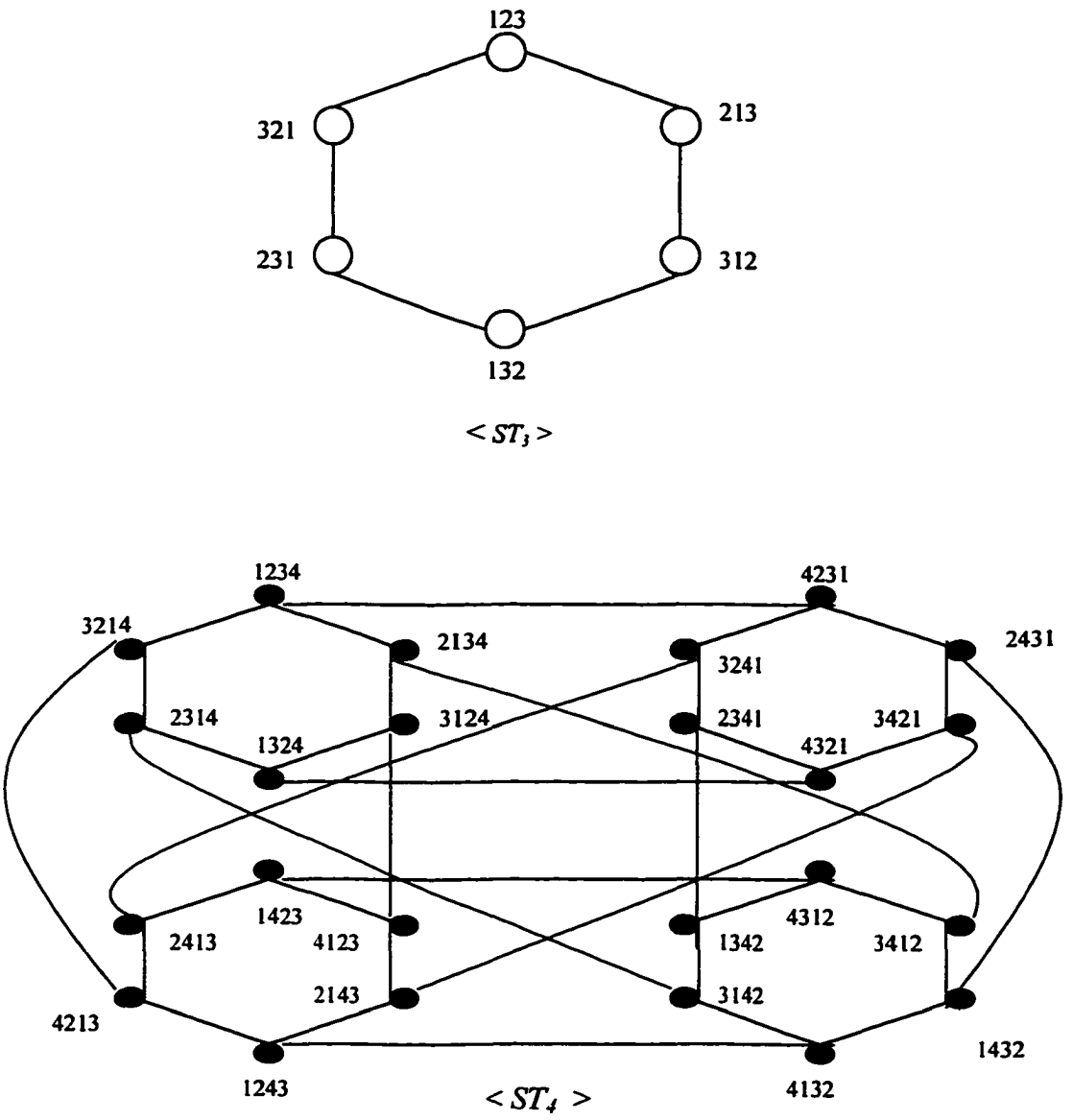


Figure 2-2. Examples of ST_3, ST_4.

LEMMA 2.3.2 The star graph(ST_n) is *bipartite* and *hamiltonian* [Jwo1991].

PROOF: The proof of this lemma is similar to that of LEMMA 2.3.1.

The extensive analysis of topological properties in *star* graph can be found in [AkKr1989][LaJwDh1993][Misi1991].

2.3.3 Binary Hypercubes (BC_n)

Formally, a *binary hypercube* has 2^n nodes, each node has a unique n-bit binary address. For any pair of nodes, the shortest distance is calculated by *hamming distance* [LaDh1990]. We can construct n-dimensional hypercubes with Cayley graph of permutation group. Let C_{2n} be a subgroup of S_{2n} such that each element of C_{2n} is a product of the transpositions of the form $T_{(2i-1)2i}$, where $1 \leq i \leq n$. It can be verified that $\Omega_{BC} = \{ (2i\text{-}1\ 2i) \mid 1 \leq i \leq n\}$ generates C_{2n}, where $| C_{2n} | = 2^n$.

Let $p = p_1\ p_2\ \ldots\ p_{2n} \in C_{2n}$. Let $\sum_2 = \{0, 1\}$ and let $(\sum_2)^n$ denote the set of all binary strings of length n. Let $x = x_1\ x_2\ \ldots\ x_{2n} \in (\sum_2)^n$. Define a one to one and onto function f: $C_{2n} \rightarrow (\sum_2)^n$ such that:

$$x = f(p) \tag{2.7}$$

$$x_i = \begin{cases} 0 & \text{if} \quad p_{2i} > p_{2i-1} \\ 1 & \text{if} \quad p_{2i} < p_{2i-1} \end{cases} \tag{2.8}$$

That is, $(\sum_2)^n$ is a binary encoding of the subgroup C_{2n} generated by Ω_{BC}. An example of BC_3 is illustrated in Figure 2-3.

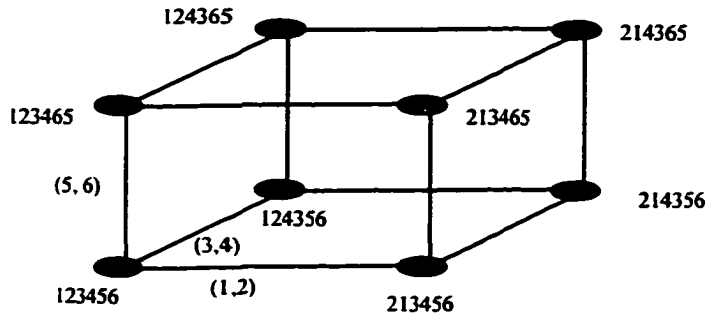


Figure 2-3. An example of BC_3.

2.3.4 *K-ary n-cube* (Torus)

Traditionally, a *k*-ary *n*-cube torus is a network with k^n nodes. The dimensions of the n-dimensional torus will be referred to as $X_{n-1}, \ldots, X_0$. Each node of the torus will be denoted by a tuple $(x_{n-1},\ldots,x_0)$, with $0 \le x_i < k$ for all $0 \le i < n$, and will be connected to nodes $(x_{n-1},\cdots, x_{i+1},(x_i+1) \bmod k, x_{i-1},\cdots,x_0)$, and $(x_{n-1},\cdots, x_{i+1},(x_i-1) \bmod k, x_{i-1},\cdots,x_0)$, for all $0 \le i < n$. The link connecting nodes $(x_{n-1},\cdots, k\text{-}1,\cdots,x_0)$ and $(x_{n-1},\cdots, 0,\cdots,x_0)$ along the dimension X_i is called a *wrap-around* link. Without the wrap-around links, the graph is called *k-ary n-cube mesh* or *grid*. Also, torus graphs can be expressed by product network.

Let $< k > = \{0, 1, \ldots, i, \ldots k\text{-}1\}$. Let G_1 be a graph $G_1 = (V_1, E_1)$ where $V_1 = < k >$,

$E_1 = (i, j)$ for $i, j \in V_1$ and $j = i + 1 \bmod k$ or $j = i - 1 \bmod k$.

Then define direct product graph of $Q_2^k = G_1 \times G_1 = (V, E)$, where $V = V_1 \times V_2 = \{(i, j) \mid i \in V_1 = < k >, j \in V_2 = < k >\}$

$E = \{((x, u)(y, v)) \mid \text{if } x = y \text{ then } (u, v) \in E_1, \text{ if } u = v \text{ then } (x, y) \in E_2\}$

Then a *k*-ary *n*-cube network can be expressed direct product graph of

$Q_n^k = G_1 \times G_2 \ldots \times G_n = (V, E)$, where $V = V_1 \times V_2 \times \ldots V_n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in V_i$ and $1 \le i \le n\}$

$$E = \{(x, y) \mid x, y \in V \text{ and } y = (a_1, \cdots, a_{i-1}, (a_i + 1) \bmod k, a_{i+1}, \cdots, a_n) \text{ or}$$

$$y = (a_1, \cdots, a_{i-1}, (a_i - 1) \bmod k, a_{i+1}, \cdots, a_n)\}.$$

Indeed, torus network is a product of rings and mesh network is a product of lines. Figure 2-4 illustrated Q_2^4 network.

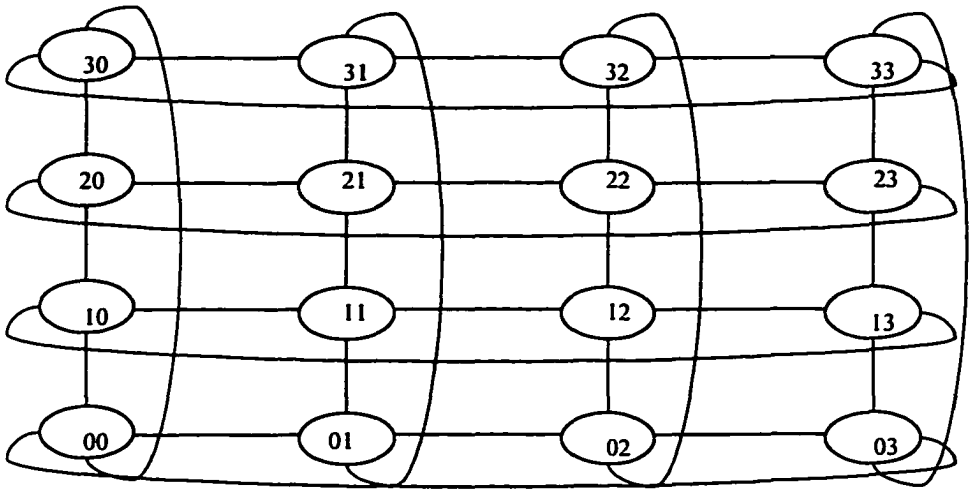


Figure 2-4. Example of 4-ary 2-cube torus (Q_2^4) network.

In Table 2-2, we compared the family of Cayley graphs of interest against to the size, degree, and diameter of the networks.

Table 2 – 2. Comparison of network properties.

Networks	size	degree	diameter
complete transposition (CT_n)	$n!$	$n(n-1)/2$	$n-1$
star (ST_n)	$n!$	$n-1$	$\lfloor 3(n-1)/2 \rfloor$
binary n-cube (BC_n)	2^n	n	n
k-ary n-cube torus (Q_n^k)	k^n	$2n$	$\lfloor k/2 \rfloor n$

2.4 SWITCHING TECHNIQUES

We will overview the characteristics of various popular switching techniques and evaluate the performance parameters of switching technologies.

2.4.1 Circuit Switching

To decrease the amount of time spent transmitting messages, a traditional *circuit switching* scheme is adopted from a telephone switching method [Bene1965]. In circuit switching, a physical circuit is constructed between the source and destination nodes by the control head of message during the circuit establishment phase. In the message transmission phase, a message is transmitted to the destination along the circuit in pipeline fashion. During this phase, the channels assigned to the circuit are reserved exclusively for the circuit; hence, there is no need for buffering at the intermediate nodes. In the circuit termination phase, the circuit is torn down from the destination as the tail of the message is delivered to its destination. Therefore, the message transmission time is

not proportional to the distance of the path while the circuit is established except for a few propagation delay times. However, circuit switching has some drawbacks on the performance of communication time in parallel computers: 1) It may suffer from delay of establishing a circuit between the source and destination if one of the channels needed to form the circuit is used by the other circuit. 2) Even after the message leaves the source the circuit, it remains until whole message is delivered to the destination node. This feature decreases the utilization of the channels by limiting access to channels which are not being used for transmission. The network latency for circuit switching can be computed following form:

$$T_{circuit\text{-}switching} = (L_c / B)D + L/B,$$

where L_c is the length of the control header transmitted to establish the circuit. If $L_c << L$, the distance D has a negligible effect to the network latency.

2.4.2 Packet Switching

First generation parallel machines have used *packet switching* mode (also, called *store and forward*) borrowed from the computer network community. In packet switching, a message is divided into several packets and a packet is the smallest unit of transmission in the network with its own routing information. Usually, the size of the packet is fixed and contains several bytes of header, data part, and a single byte CRC trailer [McBo1994]. To deliver a packet to the destination, a source node forwards the packet to a neighboring node (called an intermediate node) when the output channel is available. Therefore, each whole packet is stored in the intermediate node and forwarded

when an output channel to the next node is available. Since packet switching transmits messages in fix-size pieces, it greatly simplifies the management of intermediate storage and flow control of queuing. However, the spreading of the packets in the network may cause a sequencing burden unless all the packets are delivered to the destination sequentially. From the parallel computer view, packet switching has several drawbacks: 1) Each intermediate node must store the packet at the memory space before forwarding, so it requires a large memory space (at least a packet size) in the router. 2) Network latency is proportional to the distance between the source and the destination nodes since packets are completely stored in the intermediate node. The network latency is computed as follows:

$$T_{packet\text{-}switching} = (L / B)\, D, \quad L: \text{ packet length.}$$

Thus, distance D is the dominant factor to the latency of packet switching.

2.4.3 Virtual Cut-Through

One of the combined versions of packet and circuit switching is *virtual cut-through* [KeKl1979]. In virtual cut-through, a packet is constructed with a packet header and a data part. Before the data part is transferred, a packet header is first delivered to an intermediate node. When the header arrives in an intermediate node and its outgoing channel is free, it continues its journey to its destination. Then the data part immediately follows the header path. Thus, the header does not need to be stored in an intermediate node before being transmitted. The buffering of the packet occurs when the header meets a busy channel. So, the transmission of a packet to the destination is in a pipeline fashion

without congestion. Therefore, each node requires at least a packet size memory buffer to hold a blocked packet. From the side of memory space requirements, it is very similar to packet switching. However, network latency could be decreased in sparse traffic loads. The network latency without blocking time can be computed by the following form:

$$T_{virtual\ cut\text{-}through} = (L_h/B)D + L/B,$$

where L_h: length of the header field.

When L >> L_h, the second term, L/B, dominates, and the distance D will produce a negligible effect on the network latency.

2.4.4 Wormhole Routing

Wormhole routing also uses a virtual cut-through approach for switching. Instead of storing the blocked packet completely in an intermediate node and then transmitting it to the next node, wormhole routing does not hold whole message, rather all the blocked parts of message remain inside the network. Only the portions of the message are buffered at each intermediate node along the transmission path. A flit is the smallest unit of information that a queue or channel can accept or refuse. In *Cray T3D*, flit consists of 148 bits [ScTh1994]. Due to channel bandwidth constraints, transferring a single flit between neighboring nodes may require multiple physical channel cycles. A *phit* is the unit of information that a channel can deliver in a single step or cycle. After a router examines the incoming header flit(s) of a message, it selects the next channel based on the routing algorithm and availability of outgoing channel. As flits are forwarded, the flits

of message are spreading out across the channels which form a routing path between the source and destination. It is possible for the first flit of a message to arrive at the destination node before the last flit of a message leaves the source node. As the header advances along the specified routing path, the remaining flits follow in a pipeline fashion since all other flits have no routing information except the header flit. Because of this, a strict ordering of the flits must be maintained. Once the relative portion of the flits is scrambled, it would be impossible to reconstruct the message. If the header flit encounters a channel already in use, it is blocked in buffer (called *flit buffer*) until the channel becomes available. Rather than buffering the remaining flits in a specific place by removing them from the network channel, they remain in flit buffers along the established route. The network latency for wormhole routing is computed as follows:

$$T_{wormhole} = (L_f / B)D + L/B,$$

where L_f: the length of each flit,

B: channel bandwidth,

If $L_f << L$, the path length D will not significantly affect the network latency unless it is very large. The main advantages of wormhole switching are: 1) low memory requirement in routers (only require a flit size buffer for each incoming and outgoing channel), 2) pipeline data movement in the absence of contention, and 3) high utilization of the channel. The effects of wormhole switching on individual messages can be highly unpredictable. Since the memory requirements are low, contention in the network can substantially increase the latency of a message in parts of the network. This characteristic induces the following issues for reliable router design.

2.5 DEADLOCK, LIVELOCK, AND STARVATION

In wormhole routing, the spreading of flits in the network, and the pipelining of message flow may cause severe problems unless the routing algorithm has all the problem prevention or avoidance schemes. The main disadvantage of wormhole routing is channel congestion since the blocked flits are not relinquished from the communication channels. Also, this feature induces the *deadlock* problem since the blocked flits are holding channels (corresponding flit buffers) while competing available channels, so no flits can advance until one of the channels among the deadlock involved channels is released. In a deadlock situation, all the involved messages are requesting resources held by other messages while holding the resources requested by these messages. A different situation arises when some messages are not able to reach their destination after leaving their source although they are never permanently blocked. This situation is known as *livelock*. It may happen when the routing algorithm allows the non-minimal path and does not have prevention schemes. Also, a message may be permanently excluded to proceed, even if the requested resource is available for a time, then other messages may have priority against this message. This is called starvation, and the message never has a chance to proceed to the destination. The prevention schemes of livelock and starvation are simpler than deadlock prevention. To prevent livelock, the routing algorithm may provide only a minimal path to messages, then every step taken by the messages makes one step closer to the destinations and messages are finally delivered to their destinations in the last step. When non-minimal paths are allowed, livelock can be prevented by limiting the number of misrouting operations, too. To deal with the starvation problem,

researchers proposed some fair resource allocation policies such as *first in first service* or *round-robin* allocation policy such that each of the arriving messages has an equal chance of proceeding by the policy [DuNiYa1997]. However, the handling of deadlock is not straightforward. There are three strategies to deal with deadlock: *deadlock prevention*, *deadlock avoidance*, and *deadlock recovery* [DuNiYa1997] [AnPi1995]. In the prevention scheme, resources are granted to a message in such a way that a request never leads to a deadlock. It can be achieved by reserving all the required resources before starting message transmission. In the avoidance scheme, a resource is granted to a message only if the resulting global state is safe. A common technique is to put all the resources in order, and resource granting is strictly restricted by this order. In the deadlock recovery scheme, deadlock is allowed. The deadlock recovery scheme is only activated when deadlock occurs. This scheme naturally leads to the deadlock detection strategy [AnPi1995]. In Figure 2-5, a possible deadlock situation is presented in a four nodes unidirectional ring network. Each node has a message to send an opposite side of node such as node 0 sends a message to node 2, node 1 is to node3 and vice verse. If all the nodes transmit the message simultaneously, all of the head-flits can be placed in the input buffers of neighboring nodes according to channel direction. At the next time step, all of the headflits should be blocked since every output buffer requested by the headflits already are occupied by headflits at a previous step. In this configuration, no message releases the output flit buffer at all. Consequently, all of the messages are blocked forever unless some actions should be taken for breaking these cyclic resource requirements. Since deadlocks occur due to the lack of resources, the deadlock avoidance scheme is

realized by developing a deadlock free routing algorithm with necessary or sufficient number of resources.

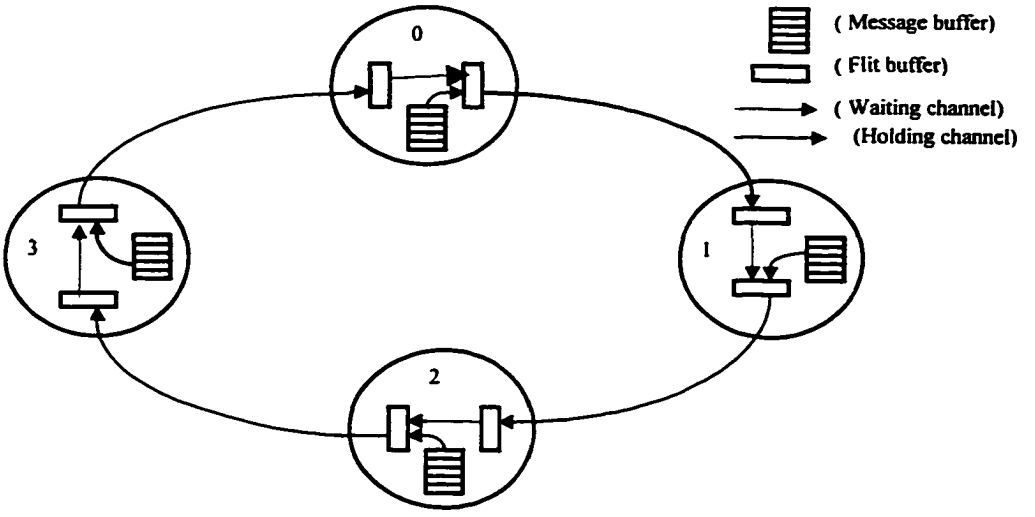


Figure 2-5. An example of deadlock in a wormhole routed *ring* network.

2.6 COMMUNICATION MODELS

In parallel and distributed scientific computing, data must be distributed periodically in such a way that all processors can be kept busy performing useful tasks. Because a static network does not physically share memory, the nodes in parallel computers must distribute data or programs by means of messages passing through the communications network. Algorithm designers have identified the need for different modes for parallel communication in a network. One of the basic communication modes is *point-to-point* communication, called *unicast*. In the unicast mode, only a single source

and a single destination node are involved on a message passing. A *multicast (or collective)* communication service is one in which the same message is delivered from a source node to an arbitrary number of destination nodes. *Broadcast*ing, which involves all nodes in network, is a special case of multicast. The following communication modes are possible communication aspects required in parallel computer application algorithm.

- Single-node broadcast: a node sends a message to all other nodes.
- Single-node scatter(*private communication*): a node sends different packets to every node in the network.
- Multinode broadcast: all nodes simultaneously perform their own broadcast.
- Total exchange(*gossiping*) is similar to multinode broadcast, except that all the messages are different.
- Single node accumulate: dual operation to single node broadcasting (*gather*).
- Multinode accumulate: dual operation to multibroadcasting.
- Permutation routing: distinct nodes send the message to distinct destinations.

Although, almost of all the above communication modes can be implemented by the *unicast* mode, this method creates too much unnecessary traffic and is likely to cause message delay, network congestion, and even deadlock. Recently, multicast algorithms in parallel computer have been extensively studied and proposed in [McTsRo1995][RoMcCh1995]. In a wormhole routing scheme, however, multicasting

communication could be inefficient since the intermediate node can not forward duplicated messages while waiting for one of the busy channels [McTsRo1995].

2.7 SUMMARY

In this Chapter, we introduce the concepts from the graph theory, group theory, and several classes of Cayley networks of interest. Also, we present switching technologies and their comparison. The first parallel machines generally used packet or circuit switching technology for interprocessor communication. Recently, second generation parallel machines adopt the wormhole routing scheme as the mechanism of choice. Finally, we introduced the issues related to wormhole routing such as deadlock, livelock, and starvation, and identified several modes of communication in the interconnection network. Most of the materials in this chapter are classic and for additional references refer to [AkKr1989][LaJwDh1993] for the Cayley graph, and to [Bene1965][KeKl1979] [DuNiYa1997] for switching technologies and router designs.

CHAPTER 3

ROUTING ALGORITHMS

The routing algorithms in a router based on conventional topologies have been investigated by a number of researchers and several issues of routing algorithms have been studied for decades. In this chapter, we extend the well known *deadlock-free* wormhole routing algorithms in *hypercube* and *torus* to CT_n and ST_n based on *unicast* or *point-to-point* communication. Routing algorithms are classified into different ways by the policy of establishing a routing path between the source and destination. A *deterministic* algorithm always provides a fixed path between a source and destination pair. In *adaptive* routing, a routing path is determined by the dynamic traffic condition of the network. From the point of view of the path length, the routing algorithms can be classified into *minimal* (shortest) and *non-minimal*. For the number of alternative paths, *fully* adaptive and *partially adaptive* algorithms are other classifications that must be taken into account.

In Section 3.1 we present our assumptions for developing routing algorithms based on wormhole routing switching scheme. More deadlock conditions and handling issues in wormhole routing are presented in Section 3.2. We derive deterministic algorithm in a family of the Cayley graphs in Section 3.3. Section 3.4 includes several classes of adaptive routing algorithms and deadlock-free conditions. The summary of the chapter is in Section 3.5.

3. 1 ASSUMPTIONS

We make assumptions used in this study of the routing algorithms throughout the dissertation.

1) A message arriving at its destination node is eventually consumed in finite time.
2) A node can generate messages of arbitrary length destined for any other node at any rate.
3) Once a queue accepts the first flit of a message, it must accept the remainder of the message before accepting any flits from another message. A message may occupy several channels simultaneously.
4) A queue can not hold flits belonging to different messages. So, two or more messages can not coexist in the same queue simultaneously.
5) A routing path taken by a message depends on its destination and on the status of the output channels (free or busy). At a given node, a deterministic routing function provides an output channel based on the current node and destination node labels, while an adaptive routing supplies a set of output channels based on the routing policy. A selection from this set is made based on the status of the output channels at the current node. This selection is performed in such a way that a free channel (if any) is supplied. If all the output channels are busy, the message header will be routed again until it reserves a channel.
6) When several messages are waiting for a free output channel, they are routed by a round-robin scheduling policy, thus preventing starvation.

7) The routing function may allow messages to follow a non-minimal path based on deadlock free conditions.
8) Livelock can be prevented by minimal routing or by limiting the direction of the message flow.

3.2 DEADLOCK AVOIDANCE IN WORMHOLE ROUTING

Deadlock handling in wormhole routing is one of the key issues in developing routing algorithm. Deadlock occurs when messages traveling in the communication system develop dependency loops among themselves that prevent further movement [LiHa91]. One way to solve the deadlock problem is to allow the preemption of messages involved in a potential deadlock situation. Preempted messages can be rerouted or discarded. This scheme is mainly used in the computer communication network since the performance of network latency is less sensitive than a parallel computer and the switching scheme is packet switching. In wormhole routing interconnection network, however, preemption of the flit may scramble message structures. So, the message may be lost in the network, and it severely affects the reliability of a parallel machine. Thus, researchers have focused on developing deadlock free routing algorithms with the schemes of breaking dependency loops conditions. To break the dependency loops, one of the methods used is to limit the direction of messages transmission so that dependency loops do not exist in the interconnection networks. However, this approach is not applicable to all the topologies such as a *ring* network with unidirectional channels between a pair of nodes in Figure 2-5. Rather, researchers have proposed the concepts of

virtual channel implementation using multiplexing technology, which shares the bandwidth of a physical channel with several virtual channels.

3. 2. 1 Channel dependency graph

In [DaAo1993], the authors proposed *Channel Dependency Graph* (*CDG*) to express of the dependency relationship between channels. The channel dependency graph is a directed graph, $D = G\ (C,\ E)$. The vertices of D are the channels of the interconnection network I. The arcs of D are the pairs of channels $(c_i,\ c_j)$ such that there is a direct channel dependency from c_i to c_j, where $c_i,\ c_j \in C$. The example of a channel dependency graph based on Figure 2-5 is illustrated in Figure 3-1. Here, we label the channels as c_i, where i is the source of channel, and the message is labeled m_j, where j is the destination of the message. Routing is performed by a minimal path through unidirectional channels. Thus, message m_2 can send a flit (headflit) to node 1's input buffer, m_3 to node 2's input buffer, and so on.

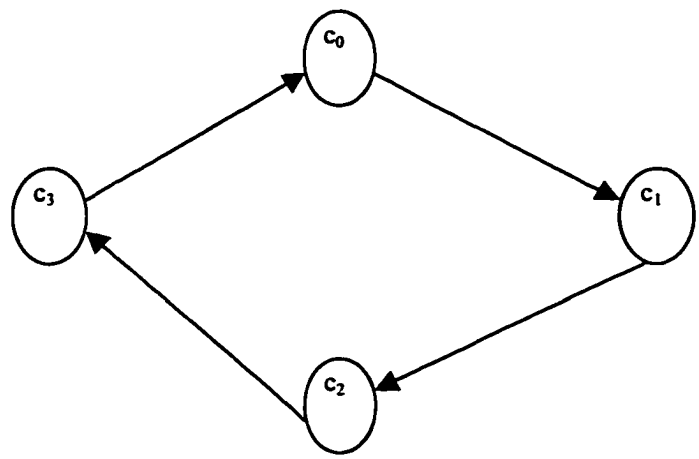


Figure 3-1. Channel dependency graph from the Figure 2- 5.

As a consequence of the resource limitation (here channels or flit buffers), no message can advance more than one step, and the messages have to wait for the next channels simultaneously. In the channels dependency graph, each node represents holding channel by a message, and each arc represents a waiting situation by a message. Therefore, in deadlock configuration, channel dependency graph forms a cycle.

3.1. 2 Virtual channel

Since deadlock occurs in the dependency loop condition of the channel requesting relationships, breaking the dependency loop is one of the deadlock avoidance schemes in wormhole routing algorithms. To construct acyclic channel dependency graph in the algorithm, virtual channel solution is the most popular approach in the community. Because of the limitation of resources, especially the shortage of channels between a pair of nodes, the possibility of deadlock is increased among the several messages competing in one channel. The virtual channel is the logical abstraction that shares the physical channel's bandwidth. Multiple virtual channels are multiplexed on a single physical channel by *frequency division multiplexing* (FDM) or *time division multiplexing* (TDM). In the wormhole router architecture, each virtual channel has its own flit buffers (input, output), control, and data path. Also, a physical network can be divided into multiple disjoint logical networks. Also, multiple virtual channels are useful to increase the degree of connectivity of networks and adaptability of routing algorithms. The flow control of the message among the virtual channels in the router is performed at two levels: (1) Virtual channel assignment is made at the message level with the routing algorithm, (2) physical channel bandwidth is allocated at the flit level with multiplexor. An example of

conventional architecture with four virtual channels sharing a physical channel is illustrated in Figure 3-2 [NiMc1993]. A dedicated single-bit control (Request / Acknowledge) wire exists between the input virtual channel and the output virtual channel of two adjacent nodes. The scheduler multiplexes data from the virtual channels over the physical channel. A fair scheduling policy (here the round robin) can be used. To preserve the bandwidth of the physical channel, only those virtual channels that have a nonempty flit buffer at the sending side and an unfilled flit buffer at the receiving side may participate in the scheduling decision. However, this virtual channel approach may cause some drawbacks. When the number of virtual channel increases, the complexity of scheduling may increase and it may cause a severe delay in network latency. Also, the sharing of a single physical channel's bandwidth may increase latency of the message.

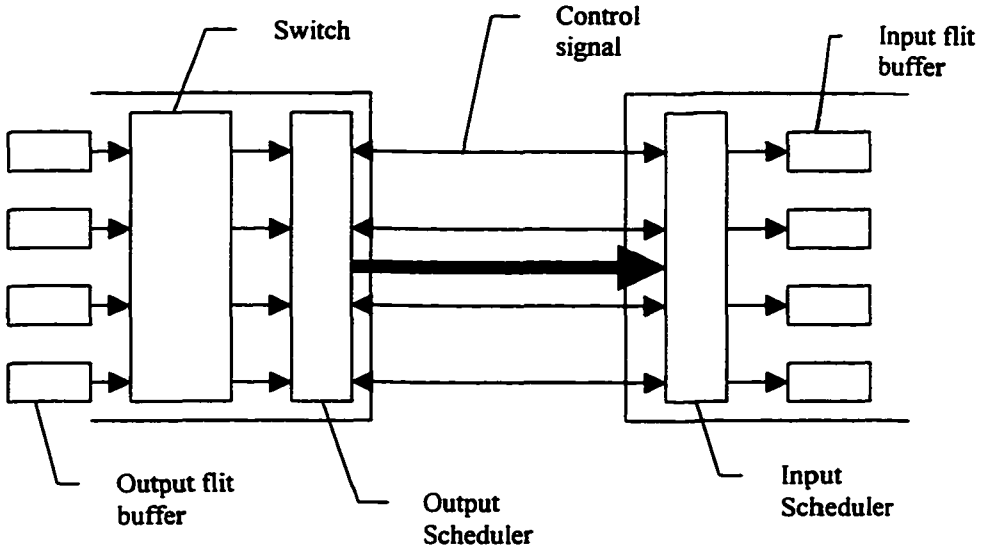


Figure 3-2. Four *virtual channels* share an unidirectional physical channel.

Now, we add one more virtual channel to Figure 3-1, and label the virtual channel c_{ij}, where i is the virtual channel class, j is the source node label of channel. If the j is 0, channels are low virtual channels, and if j is 1, they are high virtual channels. Then we can have total order of virtual channels according to their subscripts; $c_{00} < c_{01} < c_{02} < c_{03} < c_{10} < c_{11} < c_{12} < c_{13}$. Consider routing is performed following; (1) Messages at a node that are numbered less than their destination node are routed on the higher channels, (2) messages at the node that are numbered greater than their destination node are routed on the low channel. Based on the above routing algorithm, messages are followed in the strictly increasing order of virtual channels and there is no cycle in the channel dependency graph D. Here, virtual channels c_{00} and c_{13} are not used. Figure 3-3 illustrated the network and the channel dependency graph between the messages. In this configuration, a message from node 0 holds channel c_{10} at the first hop and waits for channel c_{11}. A message from node 1 holds channel c_{11} and waits for channel c_{12}. Then all the messages from node 1 and node 2 request high channels. However, a message from node 2 takes channel c_{02} and waits for channel c_{03} which is held by a message from node 3. Then a message from the node 3, which holds channel c_{03} requests high channel c_{10} for the next hop. The holding and waiting relationships for each message is represented by the channel dependency graph in (*b*). In (*c*), we combined each of the message channel dependency graphs into one graph based on the requesting order. Then, the message from node 2 (m_2) arrives at the destination first and is followed by the messages from node 1(m_2), node 0 (m_0), and node 3 (m_3). Thus, the combined channel dependency graph is

acyclic. Finally, all of the messages can be destined in a finite time and deadlock does not occur in this configuration and the routing algorithms.

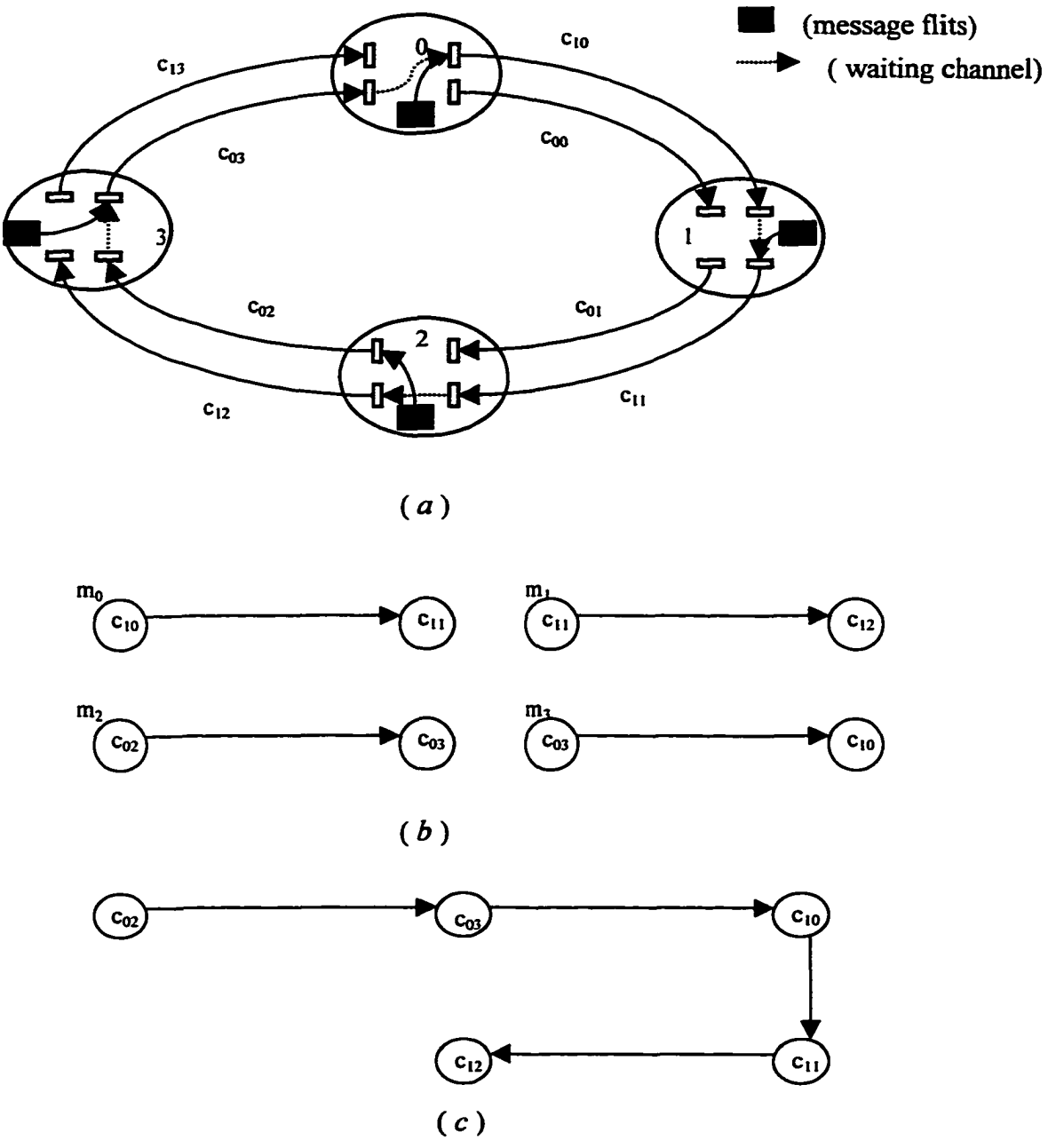


Figure 3-3. Illustration of the ring network with two virtual channels and message flows in Figure 2-5 (*a*), and its channel dependency for each message (*b*), and combined channel dependency graph for four messages(*c*).

3. 3 DETERMINISTIC ROUTING

Deterministic routing algorithms establish a routing path between a pair of nodes as a function of the destination and the current node labeling system. When deterministic routing is implemented in a distributed way, the routing algorithm supplies a single routing choice at each intermediate node based on current and destination addresses. In the process of routing decision, channel status is not considered whether the output channel is to be used or not. This simple characteristic of deciding the next neighbor node is called absolute addressing [Chie1993], since the routing decision is not effected by any other parameters except current and destination addresses. However, for the freedom of deadlock, the routing decision would be followed by certain rules to break a possible cyclic condition in a channel dependency graph as a consequence of the routing decision. Following definitions describe the deterministic routing function and configurations of the deadlock situation in the channel dependency graph and we follow the notations in [DaSa1987].

Definition 3.3.1 A deterministic routing function R: $C \times V \rightarrow C$ maps the current channel c_c and destination node n_d to the next channel c_n on the route from c_c to n_d , $R(c_c, n_d) = c_n$.

Definition 3.3.2 A channel dependency graph D for a given interconnection network $G = (V, C)$ and routing function R, is a directed graph, $D = G\ (C, E)$. The vertices of D are channels of G. Edges are the pairs of channels connected by R:

$$E = \{(\ c_i,\ c_j) \mid R(c_i,\ n) = c_j,\ \text{for some } n \in V\}.$$

Channels are not allowed to route to themselves, so there is no 1-cycles (self-loops) in D.

Definition 3.3.3 A *configuration* is the assignment of a list of nodes to each queue. The number of flits in the queue for channel c_i will be denoted $size(c_i)$. Associated with each channel, c_i, is a queue with capacity $cap(c_i)$. A configuration is legal if:

$$\forall\, c_i \in C,\ size(c_i) \leq cap(c_i)$$

That is, a flit can not be accepted in the queue beyond capacity of queue such that the amount of the flits in the queue are less than or equal to its capacity.

Definition 3.3.4 If the first flit in the queue for channel c_i is destined for node n_d then $head(c_i) = n_d$. A *deadlock configuration* for a routing function R is a nonempty *legal* configuration of channel queues:

$$\forall\, c_i \in C,\ (head(c_i) \neq n_d \text{ and } c_j = R(c_i, n) \Rightarrow size(c_j) = cap(c_j))$$

That is, if a messages head doesn't arrive destination yet and the next intermediate channel, provided by the routing function R, is full, then it implies that all the involved queues are occupied by the message flits and there is no room for incoming flits. Thus, no flit can advance further, so deadlock occurs.

Corollary 1. A deterministic routing algorithm for interconnection network G is deadlock free if and only if there are no cycles in the channel dependency graph [DaSe1987].

PROOF: $\Rightarrow$) Suppose a network has a cycle in the channel dependency graph D. This cycle must be a cycle of two or more by the definition of the network, there is no self loop such that the channel in the dependency graph does not request itself. Thus, one can construct a deadlocked configuration based on definition 3.3.3 and definition 3.3.4 by filling the queues of each channel in the cycle with flits destined for a node two channels away, if the first channel of the route is along the cycle. See the Figure 2-5 for this configuration.

$\Leftarrow$) Suppose a network has no cycle in channel dependency graph D. Since D is acyclic, one can assign a total order to the channels of C so that if $(c_i, c_j) \in E$ then $c_i > c_j$. Consider the least channel in this order with a full queue c_l. Every channel c_n that c_l feeds is less than c_l and thus does not exist in the channel dependency graph. That is, the channels of c_n should be destinations and the destined flits are immediately removed from the network. Thus, no flit in the queue for c_l is blocked and one does not have deadlock.

3. 3. 1 Dimension order algorithm

The most classic way of determining the routing path up to now is looking at a routing table established prior to message injection (called *source routing*) in packet switching and circuit switching networks. However, in the regular networks, the use of routing function (called *distributed routing*) as a scheme to determine the routing path is more efficient and saving memory space since the network symmetries do not require routing tables, rather they determine the routing path based on the node labels. Dally

[DaSe1987] developed the deterministic deadlock-free wormhole routing algorithms in *torus, cube connected cycle*, and *shuffle-exchange* networks based on the *e-cube* algorithm in *binary hypercube*. According to this algorithm, the routing processes are same as following the monotonically increasing or decreasing order of dimensions from the node labeling systems in these networks. For the avoidance of deadlock, they introduced the virtual channel control schemes in these networks. However, if base network has very well defined dimension concept from the node labeling system, then we can derive deadlock free algorithm without virtual channel. For example, in *binary hypercube*, messages are routed through the node, which the least (or most) significant position of the object in current node and the destination node label is matched but they were different right before being transmitted. Then the head flit will be one step closer to the destination node on each of the steps and the number of different objects in the current node label may decrease by one between current and destination. One of the good examples of *e-cube* algorithms without virtual channel control is illustrated in Figure 3-4. A message from node 000 to 111 is routed to 001 in the first step, in which the least significant object of the node label is matched to the destination. The message takes a channel to 011 as the second step which the second position of the label is matched to the destination label. Finally, the message takes a channel to destination from node 011, in which the leftmost bit of the label is matched to destination address. The dotted lines also show the message path from 111 to 000 by the dimension order algorithm. In this routing algorithm, the choice of intermediate node is governed by the direction of correcting current node label to the destination label.

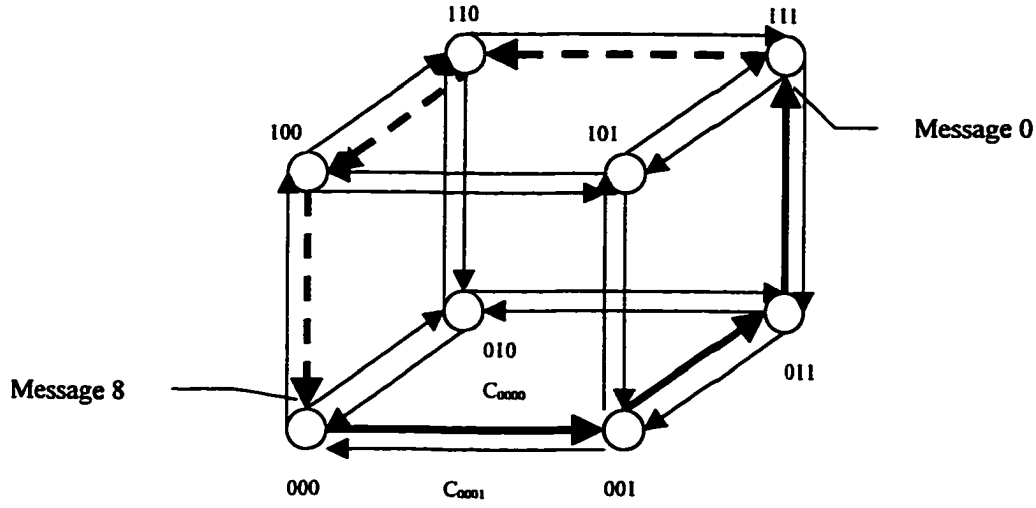


Figure 3-4. Examples of message routing paths by the *e-cube* routing algorithm in *binary hypercube*.

For the presenting of deadlock freedom, we label the channels as C_{ij}, where i stands for the dimension of the channel direction where $0 \leq i \leq n\text{-}1$ and j refers to the channel source node label. Then we can order all the channels in a n-dimensional binary hypercube based on the lexicographical order of channel label suffixes. So, the total order of channels in a 3-dimensional hypercube as follows:

$$C_{0000} < C_{0001} < C_{0010}, \ldots\ldots\ldots, C_{2110} < C_{2111}.$$

The channel dependency graphs by the *e-cube* routing algorithm on above hypercube network example are illustrated in Figure 3-5.

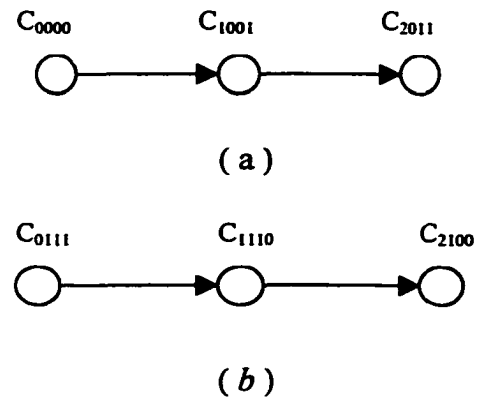


Figure 3-5. Channel dependency graphs for message 0 (a), message 8 (b).

In Figure 3-5, all the channels taken by the messages follow the increasing order of channel order and no messages can take a lower channel right after taking a high order channel. Therefore, the *e-cube* algorithm does not create any cycles in the channel dependency graph and the algorithm guarantees that it is free of deadlock with only physical channels. The extension of the *e-cube* algorithm is applied to the *torus* network with two virtual channels, a *mesh* network (or called *XY* routing) without virtual channels, and a *cube connected network* with three virtual channels.

- **Application to CT_n**

The extension of the *e-cube* algorithm in a binary hypercube to CT_n is simple. The concept of dimension in node labels is derived from the position of objects between the source and the destination labels. Since the generators of the CT_n can swap any of the

two objects, the routing is a procedure similar to sorting the current node label to the destination address one object at a time.

Algorithm 1: *Dimension order algorithm in CTn*
/* Input: *current address* $p = p_1 p_2 p_3 \ldots p_n$
destination address $q = q_1 q_2 q_3 \ldots q_n$ */

1. *If* $(p == q)$ sends the message to the local processor and exit;
2. Find the *left most position* i in q, where $q_i \neq p_i$ and $q_k = p_k$ for $k < i$
3. Find a *position* of the object j in p, where $p_j = q_i$ and $i \neq j$.
4. *If* the corresponding channel with (i, j) is available
   send the message via that channel and exit
   else return to step 2.

A dimension order routing algorithm in CT_n routes the messages in a strictly monotonic order on the label permutation (from *left* to *right*) from the current node. For example, a routing path from 12345 to 43521 on CT_5 based on a dimension ordering algorithm, is determined as follows:

$$12345 \xrightarrow[(1,4)]{} 42315 \xrightarrow[(2,3)]{} 43215 \xrightarrow[(3,5)]{} 43512 \xrightarrow[(4,5)]{} 43521$$

By observing the above example, we can find that the first elements of transposition are listed by increasing order on the way to the destination. So, the channel

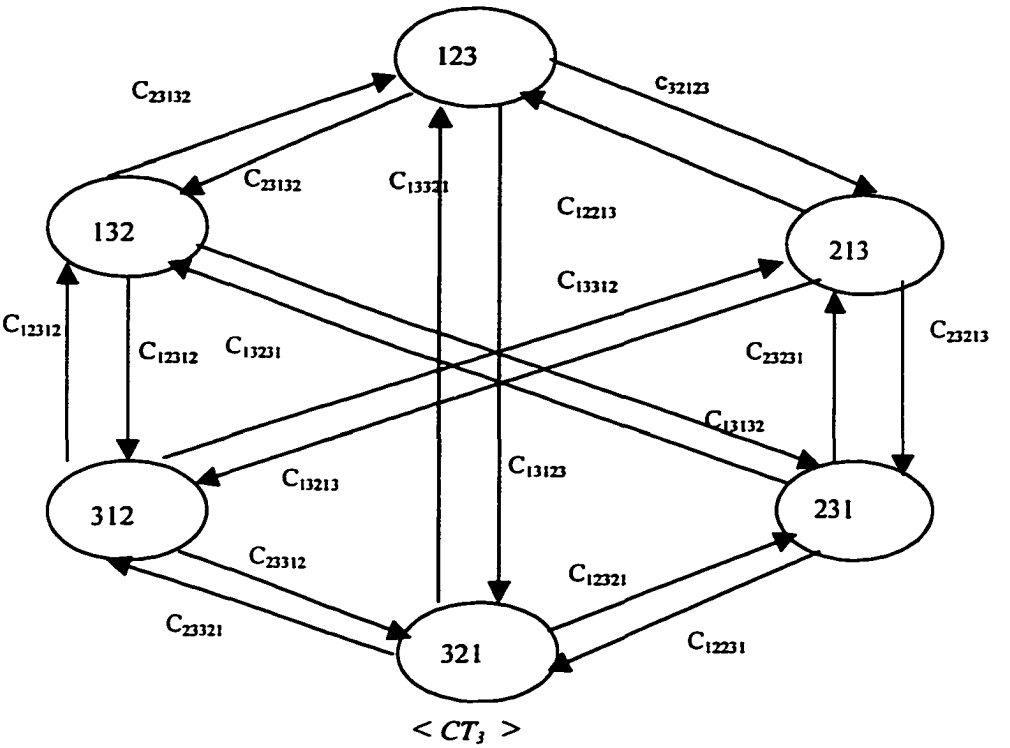
dependency graph in the dimension order routing algorithm does not have any cycle by the decreasing order of channels. Therefore, we can summarize above observations by the following.

LEMMA 3.3.1 Channel dependency graph of *dimension order routing algorithm* in CT_n is *acyclic*.

PROOF: Let the channel label be C_{ijn}, where i and j are the channels corresponding to the transposition in the generator ($(i, j) \in \Omega_{CT}$), n be the node label of the channels' source ($n \in V$). Then we can sort all the channels by increasing lexicographical order such that:

$$C_{121234\ldots n} < C_{131234\ldots n} < C_{233234\ldots n} \cdots \cdots < C_{n(n-1)n(n-1)\ldots 1}$$

An example of the channel label and lexicographical order of the CT_3 is illustrated in Figure 3-6. Since messages are routed from the increasing order of the total channel ordering, no message takes a low order channel after taking a high order channel. In the above example of the routing path from node 12345 to 43521, we can see the generator selection is monotonic increasing order of lexicographical channel ordering. Therefore, there are no cycles in the channel dependency graph.



$< CT_3 >$

Order	Channel	Order	Channel	Order	Channel
0	C_{12123}	6	C_{12213}	12	C_{12312}
1	C_{13123}	7	C_{13213}	13	C_{13312}
2	C_{23123}	8	C_{23213}	14	C_{23312}
3	C_{12132}	9	C_{12231}	15	C_{12321}
4	C_{13132}	10	C_{13231}	16	C_{13321}
5	C_{23132}	11	C_{23231}	17	C_{23321}

Figure 3-6. An example of CT_3 with channel labels and channel ordering.

THEOREM 3.3.2 The dimension order routing algorithm of CT_n is deadlock free with only physical channels.

PROOF: From the LEMMA 3.3.1, channel dependency graph is acyclic, and from the corollary 1, the routing algorithm is deadlock free and the routing algorithm naturally does not require virtual channels to break the cycles in channel dependency.

- **Application to Star(ST_n) networks**

The extension of a dimension order routing algorithm in Star networks is not simple. Since the generator of the star network only swaps the first position of an object and the rest of the positions, swapping certain objects may take at least two steps and it may cause cyclic dependency between channels without virtual channels. For example, a routing path from 12345 to 54321 on ST_5 would be as follows:

$$12345 \xrightarrow[(1,5)]{} 52341 \xrightarrow[(1,2)]{} 25341 \xrightarrow[(1,4)]{} 45321 \xrightarrow[(1,2)]{} 54321$$

From the above example, the list of the generators is not following a monotonic order of routing channels since the generator (1, 2) is used after (1, 4) is being used and it is a violation of a rule to follow ordering of the channels. So, it could produce cyclic conditions in a channel dependency graph unless the algorithm is deadlock free with the addition of more channels. One of the possible approaches is that by supplying a sufficient number of virtual channels to a physical channel derive deadlock free algorithm with these virtual channels. However, this approach will take more resources (virtual channels and flit buffers), and we will derive adaptive routing algorithm for ST_n using the virtual channels in the next section.

3.4 ADAPTIVE ROUTING

Adaptive routing has been proposed as a means of improving network throughput and performance robustness. In adaptive routing, message routing paths are determined by the dynamic situation of network traffic. By using an adaptive routing scheme, the utilization of the channels could be increased since the congested traffic may distribute to unused channels. In adaptive routing, algorithms could have two functions for the decision of the output channel: routing function and selection function. A routing function supplies a set of output channels based on the current node and the destination node. A selection of next intermediate node from this set is made by the selection function, based on the status of output channels at the current node. This selection is performed in such a way that a free channel (if any) is supplied by the selection function, otherwise routing messages have to be blocked in the flit buffers for the empty channels.

Definition 3.4.1 Let F be the set of valid channel status, F = {*free*, *busy*}.

Definition 3.4.2: A selection function $S: P(C \times F) \rightarrow C$ selects a free output channel (if any) from the set supplied by the routing function R. The selection can be random or based on priorities.

Thus, adaptive routing algorithms are able to follow alternative paths (if any) instead of waiting a busy channel, and these algorithms increase routing flexibility by the expense of more complex and slower routing procedure than deterministic algorithm. Adaptive

routing algorithms can be classified into *fully adaptive* and *partially adaptive* routings based on the selection of routing path. A fully adaptive algorithm can select any of the intermediate nodes as a next hop for a routing message. Therefore, a routing path taken by a message in fully adaptive algorithm would be one of a path among the all the shortest paths between the source and the destination. A partially adaptive routing algorithm is one where the selection of an output channel is restricted to the subset of all the intermediate nodes. Thus, its selection of the path could be one of the paths within a subset of the all the paths to the destination. It should be noted that although all the physical paths are available, routing algorithm may restrict the use of a certain class of virtual channels in order to avoid deadlock.

3.4.1 Negative-hop algorithm

In [BoCh1994], the authors proposed frameworks of fully adaptive minimal routing algorithms, called a *negative-hop* scheme, in several regular networks. Originally, their ideas came from the *buffer reservation(BR)* technique [Gunt1981] [Gopa1985] of packet switching(or Store-and-Forward) networks. For the deadlock avoidance in a packet switching network, a buffer pool in each node is partitioned into several classes, and the incoming messages are placed in the buffer full according to the hops, which is the number of routing length until that node. The allocation of the buffer classes to a messages is the main point in routing algorithm. The minimum number of buffers sufficient to guarantee deadlock freedom is the great concern in our research. To derive wormhole routing algorithm from the packet switching algorithm, we modified the way of buffer control to virtual channel control. In packet switching, buffers in the router are

the main resources for which messages compete, while communication channels are in wormhole routing. The basic idea of wormhole routing from the buffer reservation technique is from the splitting each physical channel into as many virtual channels as there were buffer classes in the packet switching. Thus, a physical network can be divided into several virtual networks with its own virtual channel classes. For deadlock free routing, messages are routed among the virtual networks based on given virtual channel classes. The process of designing a wormhole routing algorithm, R, from the Store-and-Forward algorithm, SF, consists of two steps: specification of C, the set of virtual channels, R, and the routing function from $C \times N$ to C.

(1) Let $b_1, \ldots, b_m$ be classes of buffers occupied by messages before reaching their destinations in the Store-and-Forward algorithm. Then, for the wormhole routing algorithm, on each physical channel in the network, we provide virtual channels of classes $c_1, \ldots, c_m$ and corresponding flit buffers.

(2) Let $(b_1, y, b_2) \in$ SF, a hop from buffer b_1 to b_2 by a message destined to y in the Store-and-Forward routing. Then, (c', y, c_1), $(c_1, y, c'') \in R$, where Class(b_1) = Class(c_1), Channel(c_1) = Channel(b_1, b_2), c' is any virtual channel simulated for any buffer and the physical channel combination used by the message to reach b_1, and c'' is any virtual channel simulated for any buffer and physical channel combination used by the message after reaching b_2. If (b_1, y, b_2) is the first hop of the message in the Store-and-Forward routing,

then c' is the injection channel of node b_1, If (b_1, y, b_2) is the last hop of the message in the Store-and-Forward routing, then c'' is the consumption channel of the node of b_2.

Suppose a message M is routed from x to y using buffers $b_1, ..., b_t$ for hops 1, ..., t-1; b_1 is the buffer occupied at injection and b_t is the buffer occupied at consumption. Therefore, (b_1, y, b_2), (b_2, y, b_3) , ..., $(b_{t-1}, y, b_t) \in SF$. Then, (inj_x, y, c_1) , ..., (b_{t-2}, y, c_{t-1}), $(c_{t-1}, y, cons_y) \in R$, such that $Class(c_i) = Class(b_i)$, $0 < i < t$, and $Channel(c_i) = \text{Channel}(b_i, b_{i+1})$; inj_x is the injection channel in node x and $cons_y$ is the consumption channel in node y. Figure 3-7 illustrated a hops in Store-and-Forward and its application to a wormhole routing hop.

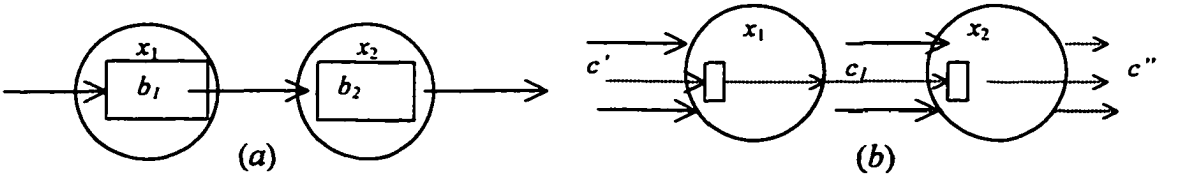


Figure 3-7. Illustration of hops in a wormhole algorithm (*b*)constructed from a store-and-forward algorithm (*a*).

In Figure 3-7, (*a*) illustrates the hop by a message from packet buffer b_1 to b_2 in Store-and-Forward routing, and (*b*) illustrates the version of the wormhole routing for the same message based on the Store-and-Forward routing in (*a*). The flit buffer used in node x_1 is the dedicated flit buffer c' and the flit buffer used in node x_2 is the dedicated flit buffer to

c_1. The dotted lines indicate additional virtual channels simulated on each physical channel. Also, the virtual channels c' and c'' are dependent on the hops taken before arriving at node x_1 and after arriving at node x_2, respectively. The derivation of wormhole routing from the Store-and-Forward routing induces following properties.

LEMMA 3.4.1 Wormhole routing R is free of deadlock if S is free of deadlock and satisfies any one of the following conditions [BoCh1996]:

(a) a message always acquires buffers not used by it before,

(b) a message does not revisit a node immediately after leaving it, or

(c) a message never visits the same node twice.

COROLLARY 2: If *SF* (Store-and-forward algorithm) ensures that message acquire buffers in the greater than order, >, of some partial order on b, then the wormhole routing R is deadlock free.

PROOF: Since *SF* allocates buffers to messages as per an anti-symmetric relation, no message reuses a buffer, and S is deadlock free from the LEMMA 3.4.1.

In the negative-hop store-and-forward algorithm [Gopa1985], the network is partitioned into several subsets, such that no subset contains two adjacent nodes. If C is the number of subsets, the subsets are labeled 0, 1,...,C-1, and nodes in subset i are labeled with color value i.

Definition 3.4.3 A hop is a *negative hop* if a message is delivered from a high color node to a low color node in one time step; otherwise, it is a nonnegative hop.

A message occupied a buffer class b_i at an intermediate node if and only if the message taken exactly i negative hops to reach that intermediate node. If H is the maximum hops taken by a message and C is the number of the colors, then the maximum number of negative hops that can be taken by a message is:

$$H_N = \lceil H(C-1)/C \rceil \qquad (3.4.1.1)$$

Gopal proved that the number of sufficient *buffer classes* for deadlock freedom in a Store and Forward switching network is $H_N + 1$ classes [Gopa1985]. For the version of the wormhole routing, the number of sufficient virtual channel classes required by negative-hop algorithm is defined as:

$$1 + \left\lceil \frac{(C-1)(H-1)}{C} \right\rceil \qquad (3.4.1.2)$$

Thus, the number of sufficient virtual channels per physical channel in wormhole routing is a function of diameter and the number of colors from a graph coloring problem. For the lesser number of subsets, a bipartite network is a good application of negative-hop algorithm since it requires only two colors.

Since CT_n and ST_n are bipartite graphs from LEMMA 2.3.1 and LEMMA 2.3.2, we can partition the networks into two subsets such that no subsets contains two adjacent nodes, and each node in the subset has the same color value such as 0 or 1. Then the negative-hop is from the color value 1 node to the 0 node, and the non-negative hop is vice-versa. The assignment of the virtual channel classes occurs only after a message takes a negative-hop.

Algorithm 2: Negative-hop algorithm

Initially, *current channel class* = 0, *current-host* = source of the message, and all the nodes have coloring value (1 or 0)

1. If (current address = = destination address) send message to a local processor and exit.
2. If color of the current-host is 0 or the colors of the previous-host and current-host match, then increment current-class by one.
3. Find a neighbor set in the shortest path to the destination, by applying all the generators.
4. Select any neighbor node in the set as the next-host.
5. Reserve the virtual channel of class current-class.
6. If the virtual channel is available,

   set previous-host <- current-host, current-host <- next-host,

   route the message;

   otherwise, go to step 4.

LEMMA 3.4.2 The channel dependency graph by the negative-hop algorithm is acyclic.

PROOF: To prove an acyclic condition, we label each of the virtual channels as $C_{i,j,p,q,}$ where i is the class of virtual channel, j denotes the color value of the channel source node, p and q represent input and output channels node's labels, respectively. Then the following relationships exist between the two consecutive hops $C_{a,b,p,q}$ and $C_{c,d,q,s}$ by the algorithm:

(1) $a < c$ **or**

(2) $a = c$ **and** $b < d$

The relationship (1) tells that the hop from p to q was a negative-hop and that the virtual channel class is incremented right after the negative-hop. The relationship (2) tells that the hop from p to q was a non-negative hop and q to s is a negative-hop since q has color value of **1**, the color values of p and s should be valued **0** by the definition of the graph coloring. Then we can list all the channels taken or requested by a message, by lexicographic ascending order of virtual class, and by node coloring value. Since messages are routed through the partially ascending order of virtual channel suffixes, no message waits in a channel where the suffixes of the wait channel is less than the previous channel. Therefore, by Corollary 2, there is no cycle in channel dependency graph. In Figure 3-8, we presented the assignment of virtual channels classes into a message based on the negative-hop algorithm.

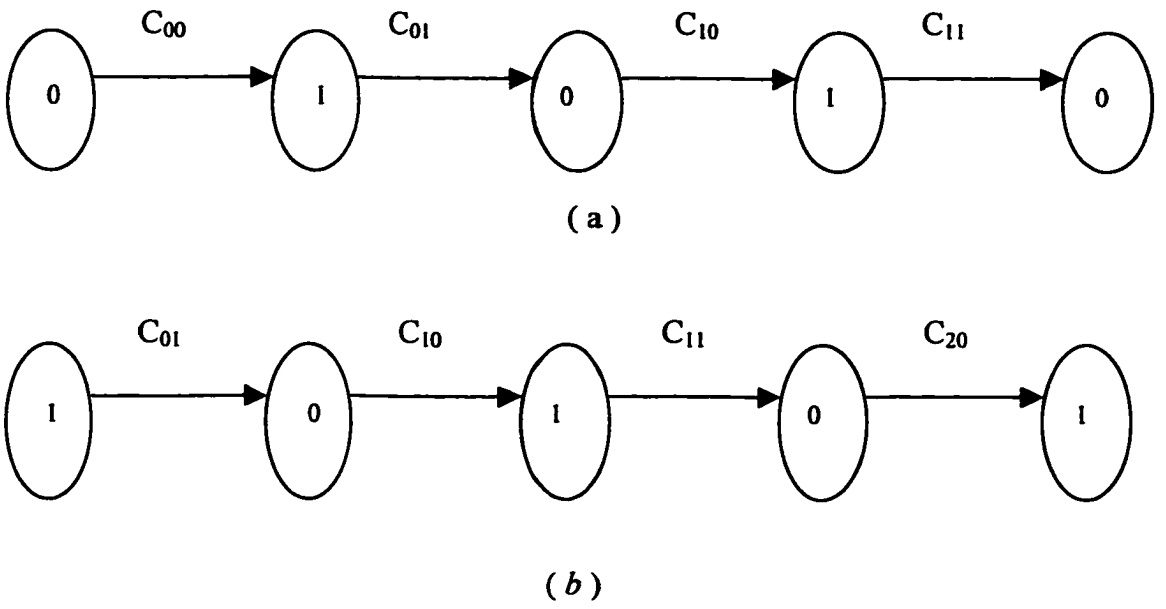


Figure 3-8. Virtual channel assignment based on the Negative-hop routing algorithm with distance four messages.

The number of negative hops a message is taken in (a) is two, but the virtual channels are used in only 2 classes. Though the last hop is negative, it is finally delivered to the local processor by our assumption and the algorithm does not involve assigning the virtual class. In (b), however, the usage of the virtual channels is three and its negative hop is two. Since the message start from the node color value 1, it requires one more virtual channel over configuration (a). Therefore, the maximum number of negative hops in bipartite networks is almost half the diameter and the sufficient number of the virtual channels per physical channel is defined as one more to the maximum negative hops.

THEOREM 3.4.1 The negative-hop algorithm is deadlock free with $\left\lfloor \frac{n-1}{2} \right\rfloor + 1$ virtual channels in CT_n, $\left\lfloor \frac{\left\lfloor \frac{3(n-1)}{2} \right\rfloor}{2} \right\rfloor + 1$ in ST_n.

PROOF: Since the channel dependency graph of the negative-hop algorithm is acyclic by LEMMA 3.4.2, and from Corollary 2, there is no deadlock. Also, from the table 2-3, the diameter of the CT_n is n-1, and from the property of (3.4.1.2) the sufficient number of virtual channels is computed to $\left\lfloor \frac{n-1}{2} \right\rfloor + 1$ for CT_n. Likewise, for ST_n, we can prove the theorem with $\left\lfloor \frac{\left\lfloor \frac{3(n-1)}{2} \right\rfloor}{2} \right\rfloor + 1$ virtual channels.

3.4.2 Disrupt-hop Algorithm

In the Cayley family of network, the properties of the generator set have a very important role in determining network degree, diameter, and scalability. Mišic[Misi1992] introduced wormhole routing algorithm in ST_n and proved that the algorithm is deadlock free with n-1 virtual channels. The algorithm is a result of a detailed analysis of generator properties in ST_n. The algorithm can apply to the deterministic when the routing path is determined by the source routing, which is the routing path that is decided before the message injection. However, the use of resources is the same as an adaptive algorithm. So we will concentrate on deriving an adaptive routing algorithm with distributing routing here. To apply the algorithm, generators (or corresponding channels) are ordered by transposition indices. For the deadlock freedom, the assignment of virtual channels to a message is determined by the ranks of generator order. Since the routing path in Cayley graphs of a permutation group is same as the sorting a source node permutation to a

destination permutation, we can represent the path as a collection of corresponding generators. Let the minimal routing path, Π_m, which is a list product of generators, and let the transposition (1, 2) to g_1, (1, 3) to g_2, (1, n) to g_{n-1}, and so on. Then, in ST_n, $\Pi_m = g_i,..g_j,..g_k$ where $1 \le i, j, k \le n-1$, and $g_i\, g_j\, g_k \in \Omega_{ST}$. We introduce a partial ordering of generators in the following:

$$g_{n-1} > g_{n-2} > \ldots > g_2 > g_1. \qquad (3.4.2.1)$$

For example, the generator order in ST_6 is (1, 6) > (1, 5) > (1, 4) > (1, 3) > (1, 2).

Definition 3.4.2.1: A *commutative product* is a list of generators of which the first generator is appended as a last generator. Otherwise, we call the list of generators an ordinary product. In ordinary product, each of the generators is distinct in the list. For example, product of (1,2)(1,3)(1,4) or $g_1g_2g_3$, is an ordinary product, and (1,2)(1,3)(1,2) or $g_1g_2g_1$ is a commutative product since the first generator (1,2) or g_1 is appended as a last product.

LEMMA 3.4.3 If some product of generators in a ST_n consists of two products of distinct sets of generators $\Pi_m = \Pi_a \bullet \Pi_b$, and if any of Π_a or Π_b is commutative, then $\Pi_m = \Pi_a \bullet \Pi_b$ or $\Pi_m = \Pi_b \bullet \Pi_a$.

For example, if a routing path from 234561 to 654321 in ST_6 is in the following form:

$$234561 \xrightarrow[(1,5)]{} 634521 \xrightarrow[(1,2)]{} 364521 \xrightarrow[(1,4)]{} 564321 \xrightarrow[(1,2)]{} 654321,$$

then another routing path is in the following forms:

$$234561 \xrightarrow[(1,2)]{} 324561 \xrightarrow[(1,4)]{} 524361 \xrightarrow[(1,2)]{} 254361 \xrightarrow[(1,5)]{} 654321.$$

LEMMA 3.4.4 Every product of an arbitrary number of generators in a star graph of order n can be reduced to a product of $k \geq 1$ products of distinct sets of generators:

$$\Pi_m = \Pi_1 . \Pi_2 ... \Pi_i ... \Pi_k ,$$

in which at most, one product is ordinary and all others are commutative [Misi1991].

LEMMA 3.4.5 The maximal number of k of the product in a minimal product from LEMMA 3.4.4 is $(n-1)/2$ if n is odd, and $(n-2)/2 + 1$ if n is even.

Definition 3.4.2.2 *Disrupt-hop* is a hop when a message takes a low rank channel right after it took high rank channel.

In the routing algorithm, a message can select any of the neighbors as a routing node within a subset in which is a collection of neighboring nodes, which lie in the shortest paths to the destination. Thus, the algorithm takes any one of the shortest paths between the source and the destination based on the dynamic condition of the traffic. For the deadlock freedom, however, the control of the virtual channel should be careful on the routing path. The virtual channel classes in each hop are determined by the maximum number of disrupt hops in which a message is taken immediately before the routing

decision. The key point of the disrupt-hop algorithm is finding the maximum number of disrupt-hops in the routing path. We will introduce the algorithm first and show that the number of sufficient virtual channels for deadlock freedom in ST_n and CT_n. The algorithm assigns a virtual class of the channel according to the ranks of the output generators, which are corresponding output channels. Thus, a message may not use more than one virtual channel when all the participating generators are listed by a partially increasing order, or it may use all the classes of the virtual channels when all the participating generators are listed in a decreasing manner. For example, the partial shortest paths and channel assignments between 12345 to 23451 in CT_5 are listed in Figure 3-9.

Algorithm 3: Disrupt-hop Algorithm

Initially *current_class = 0, current_rank = 0*

1. If (current address = = destination address) then send a message to local processor and exit;
2. Find all the candidate neighbors by applying all the generators and distance computing function;
3. Select any neighbor node from the candidates and find the channel rank by property (3.4.2.1)
4. If (current rank > neighbor rank) then *reserve_class = current_class + 1;*
5. Reserve the neighboring channel with *reserve_class class;*
6. If the *reserve_class* is available then
   current_class:= reserve_class, route the message
   Else go to step 3.

$$12345 \xrightarrow[(1,5)]{C_0} 52341 \xrightarrow[(1,4)]{C_1} 42351 \xrightarrow[(1,3)]{C_2} 32451 \xrightarrow[(1,2)]{C_3} 23451$$

$$\xrightarrow[(1,2)]{C_0} 21345 \xrightarrow[(2,3)]{C_0} 23145 \xrightarrow[(3,4)]{C_0} 23415 \xrightarrow[(4,5)]{C_0} 23451$$

$$\xrightarrow[(3,4)]{C_0} 12435 \xrightarrow[(2,4)]{C_1} 13425 \xrightarrow[(1,5)]{C_2} 53421 \xrightarrow[(1,4)]{C_1} 23451$$

Figure 3-9. Partial list of shortest paths and virtual channel assignments from node 12345 to node 23451 based on a negative-hop algorithm.

LEMMA 3.4.6 The channel dependency graphs in CT_n and ST_n based on a disrupt-hop algorithm, are acyclic.

PROOF: Let the virtual channel label be $VC_{i,j,s,d}$, where i stands for a class of virtual channel, j stands for the rank of a generator, and s, d stand for the channel source and destination, respectively. Let any two consecutive channels be $VC_{a,b,p,q}$ and $VC_{c,d,q,r.}$, then there exists the following relationships among the any two consecutive hops.

(1) $a = c$ *and* $b < d$ **or**

(2) $a < c$ *and* $b > d$

In relationship (1), a message takes a non-disrupt hop on q to s immediately after taking p to q hop. Thus, the class of the virtual channel is the same as the previous hop and the rank of channel is greater than the previous channel. In relationship (2), however, a message takes a disrupt-hop q to s right after a p to q hop. Then we can list the order of

the virtual channels by the lexicographical order of the virtual channel indices. Therefore, a message does not take lower rank of a virtual channel immediately after taking the high rank of virtual channel, and a message takes a monotonic increasing order of the channels. Thus the lemma follows.

THEOREM 3.4.2. A disrupt-hop algorithm is deadlock free with n-1 virtual channels per physical channel in ST_n and CT_n.

PROOF: It is obvious in CT_n with n-1 virtual channel. Since the diameter of CT_n is n-1 and all the lists of the generators in a minimal path consist of ordinary products, a message could take all the disrupt hops except the first hop (all the messages take a virtual channel class 0 as a first hop). Thus the maximum number of generator disruption is n–2 and the algorithm needs n-1 virtual channels per physical channel. For the proof of the ST_n, we need to find the maximum number of the disruptions in the ST_n diameter. The maximum number of order disruptions between generators in a commutative or an ordinary product with p distinct generators is p–1 since all the generators are distinct except for the last generator, in which first and last generators are same, in a commutative product. Also, the maximum number of disruptions among the generators between k products is n-1-k by the LEMMA 3.4.5. Thus, the overall number of generators in minimal routing is n-1+k-1, if the product has one ordinary product and k-1 commutative products. Also, when there are k commutative products and no ordinary product, it consists of n-1+ k generators. Thus a maximum number of disruptions of ordering

between the generators is the sum of the maximum number of disruptions among generators in k products and of a maximum number of order disruptions between the products. So, the maximum number of disruptions in the longest path can be calculated as follows:

Maximum number of disruptions among the generators in k products: $n - 1 - k$

Maximum number of disruptions among the k products: $k - 1$

Total maximum number of disruptions: $(n-1-k)+(k-1) = n - 2$

Since every message takes the virtual channel 0 as a first hop, we need $n-1$ virtual channels to take care of $n-2$ disruptions in ST_n.

3.4.3 *-Channel algorithm

We have concentrated on a sufficient number of virtual channels for deadlock freedom in adaptive algorithms. However, the increase of the virtual channels per physical channel also increases the cost and node delay time considerably in networks with adaptive algorithms, and its effects might cause delay by the adoption of adaptive algorithms in commercial fields [Chie1993]. Since the number of sufficient virtual channels in negative-hop or disrupt-hop is the function of the diameter, the number will be increased when the network size increases. Duato [Duat1993][Duat1995] proposed innovative ideas of an adaptive wormhole routing algorithm design framework and proved the necessary and sufficient conditions of deadlock freedom with a fixed number of virtual channels. Based on Duato theories, several researchers proposed the applications, called a *(star)-channel algorithm, on various networks and showed

performance metrics on the binary hypercubes, meshes [SuSh1995], and torus [GrPi1994] networks. The key point of his idea is to divide the network into two groups: an adaptive network and escape networks. Messages are allowed to route using any of the adaptive network channels, but if any of the channels in the adaptive network is busy, a message may be routed in the escape networks in which channel dependency graph based on the routing sub-function is acyclic. In conventional networks, such as hypercube, meshes, and tori, a dimension order deterministic algorithm is used for routing sub-function. Since the channel dependency graph of deterministic algorithms is acyclic, the algorithm guarantees the delivery of messages to the destination without deadlock situation and without an adaptive scheme. To represent the relationship between the adaptive channels and the deterministic channels, we add the following definitions for the purpose of introducing a new algorithm on our target networks.

Definition 3.4.3.1 An adaptive routing function is $R: N \times N \rightarrow P(C)$, where $P(C)$ is the power set of C, and supplies a set of alternative output channels to send a message from the current node n_c to the destination node n_d, such that $R(n_c, n_d) = \{c_1, c_2, \ldots, c_p\}$, where $\forall\ n \in N$ and $c_1, c_2, \ldots, c_p$ are just an enumeration of arbitrary channels which are supplied by the routing function R. The routing function used in the previous section as $R: C \times N \rightarrow P(C)$ is not applicable here.

Definition 3.4.3.2 A routing sub-function R_1 for a given routing function R is a routing function that supplies a subset of the channels supplied by R such that $R_1(x, y) \subseteq R(x, y)$ $\forall x,y \in N$.

Definition 3.4.3.3 An *extended channel dependency* graph D_E for a given interconnection G and routing sub-function R_1 of a routing function R, is a directed graph, $D_E = G(C_1, E_E)$. The vertices of D_E are channels supplied by the routing sub-function R_1 for some destinations. The arcs of D_E are the pairs of channels (c_i, c_j) such that there is a dependency from c_i to c_j directly, indirectly, direct-crossly, or indirect-crossly [Duat1995].

Algorithm 4: *-channel algorithm on CT_n

1. If (current address == destination address) send a message to local processor and Exit.
2. Find all the neighbors using the generator set and distance computing function.
3. If any one of the neighbors is available then
   route a message through the adaptive channel
   else
   find a neighbor using **Algorithm 1** (*dimension order algorithm*);
   if the neighbor is empty then
   route the message through the escape channel; else go to step 2;

THEOREM 3.4.3. A *connected* and adaptive routing function R for an interconnection network G is deadlock free if there exists a routing sub-function R_1 that is

connected and has no cycles in its extended channel dependency graph D_E [Duat1995][Duat1993].

A message can take any of the adaptive channels if its flit buffer is empty. The *-channel algorithm does not allow it to wait for an adaptive channel, occupied by another message. Instead of waiting for an adaptive channel until it is empty, the algorithm finds an escape channel, supplied by the routing sub-function (dimension order algorithm here). Then route the message through the escape channel if that channel is empty, otherwise the routing algorithm waits for adaptive channels or an escape channel until any of them is empty. Thus, a message is routed through any available channels, supplied by the routing function or routing sub-function. One of the favorable points of this algorithm is that the number of virtual channels per physical channels is not dependent on the size of the network. If the routing sub-function, usually a deterministic algorithm, requires n classes of virtual channels, the sufficient number of virtual channels for adaptive routing is $n + 1$, one more virtual channel is solely dedicated to adaptive routing. In Figure 3-10, the routing paths between node 1234 to 2341 in CT_4 are illustrated with adaptive and escape channels. In the escape channel direction, another adaptive channel exists concurrently, but the figure did not illustrate the adaptive channel on the escape channel.

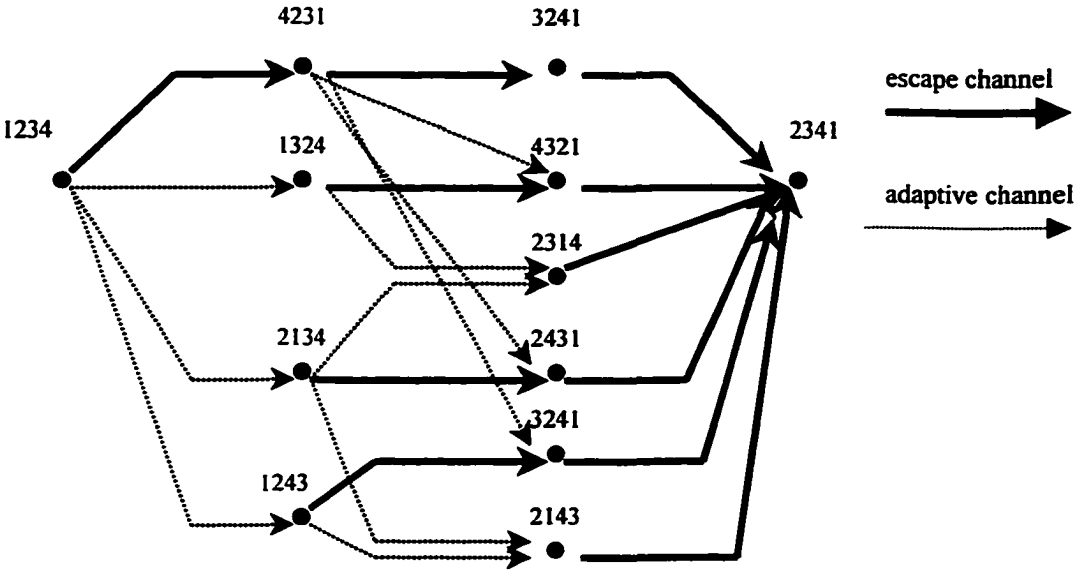


Figure 3-10. Illustration of adaptive and escape channels in the shortest paths between node 1234 and 2341.

LEMMA 3.4.7 The extended channel dependency graphs (D_E) in CT_n base on *-channel algorithm are acyclic.

PROOF: Since *-channel algorithm uses a dimension order routing algorithm as a routing sub-function, the extended channel dependency graph is acyclic from the LEMMA 3.3.1, even if some of channels CDG in LEMMA 3.3.1 are missing in the D_E. In Figure 3-11. One of the possible channel dependency graphs is illustrated with some dependencies with the adaptive channels. Even though channels c_i and c_j are not connected directly in the extended channel dependency graph, it is connected indirectly by the adaptive channels. Thus, the extended channel dependency graph by the routing sub-function is acyclic.

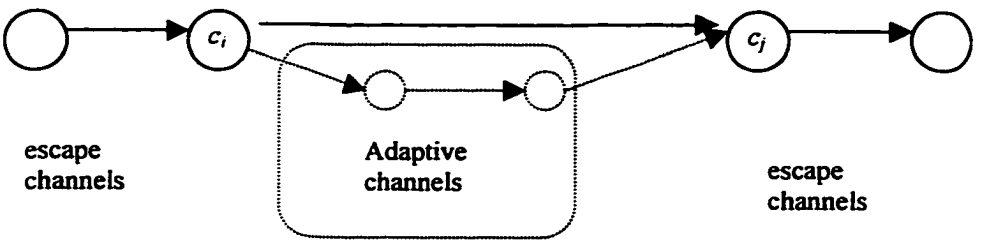


Figure 3-11. An extended channel dependency graph by routing sub-function R_1.

LEMMA 3.4.8 Channel dependency graphs (CDG) by the *-Channel algorithm are acyclic in CT_n.

PROOF: We prove the LEMMA by contradiction. Suppose there is a cyclic condition in the channel dependency graph. By the routing algorithm, a message head with an adaptive channel does not wait for an adaptive channel, which is occupied by another message, rather it requests an escape channel by the routing sub-function. Then we prove at least one of the channels in cyclic condition in the channel dependency graph is in the escape network in which the extended channel dependency graph is acyclic. In figure 3-12, we illustrated two cases of channel dependency graphs, one has cyclic dependency and the other has acyclic dependency. Figure 3-12 (a) showed the channel dependency is cyclic. Since adaptive channels do not wait for a busy channel, all the channels should be escape channel in this figure or one of the channels is an adaptive channel. However, escape channels do not make cyclic condition by the algorithm 1. Thus, this cyclic

dependency is impossible by the dimension order routing algorithm which is the routing sub-function R_1. Therefore, channel a in Figure 3-12(a) is adaptive channel. In Figure 3-12 (b), if message 4 with channel d does not wait for an adaptive channel, which is already occupied message 1, it requests escape channel e by the routing sub-function. Thus, the channel dependency graph should be acyclic, and it contradicts our assumption.

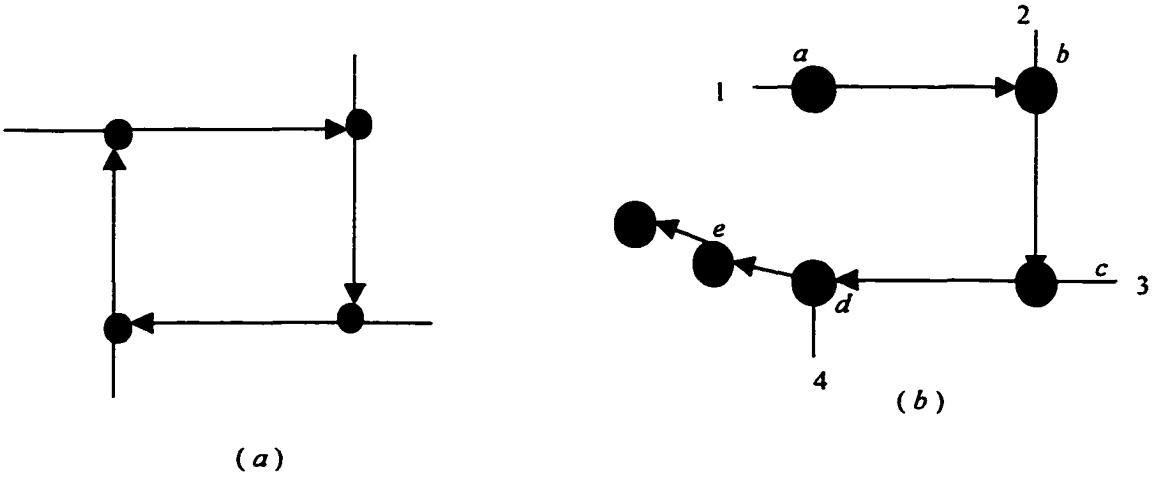


Figure 3-12. Channel dependency graphs with cyclic dependency (a), and acyclic dependency graph between channels (b).

THEOREM 3.4.5 *-channel algorithm in CT_n is deadlock free with two virtual channels.

PROOF: The theorem follows from the LEMMA 3.4.7, and LEMMA 3.4.8.

3.4.4 Non-minimal partially adaptive algorithm

A routing algorithm is partially adaptive when a message is routed through only a subset of all the shortest paths. Unlike a fully adaptive routing, the set of candidates of the next neighbors is limited based on the network directions. The main point of this algorithm is in finding a *hamiltonian path* and ordering all the nodes based on the sequence from the start node to end node. In the hamiltonian path, all the nodes have at least one higher rank neighbor and one lower rank neighbor except the start node, which has only higher rank neighbors, and the end node, which has only lower rank neighbors. Then we can split the network into two sub-networks. The Up-path network consists of channels, in which the direction of the channel is from a low order rank node to a high order rank node. The Down-path network is defined in the opposite direction. Both of the two networks have no cycles since all channel directions are same in one of the networks. To avoid deadlock, a message can take any of the channels in the UP-path when one of the channels on the shortest path is exists whether it is available or not. The Down-path network can be taken when no Up-channel exists in the candidates set. Once a message is routed to the Down-path network, it should be remained inside of the Down-path network. If there is no shortest path channel in the Down-path network, a message is routed through the neighbor in which the rank is one less than the current node. Then a message is finally delivered to the destination inside of the Down-path network and the restriction of roll back to Up-path network prevent cyclic condition of channel dependency graph. Figure 3-13 illustrates the examples of the UP-path network and the Down-path network in CT_3 without any cycles in the networks.

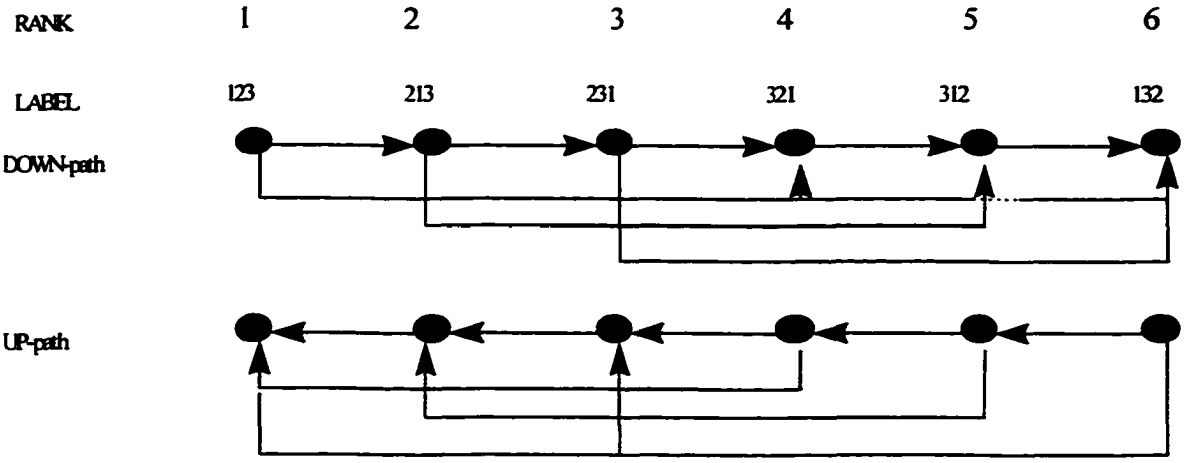


Figure 3-13. UP-path and DOWN-path networks in CT_3 based on the *hamiltonian* path node rank.

Algorithm 5: (UD-path algorithm)

/* Initially message_TAG = None */

1. If (current address = = destination address) send a message to local processor and exit.
2. If ((message_TAG == None) or (message_TAG = = UP) then
   - 2.1 Find all the neighbors that lie in the shortest path
   - 2.2 If any one of the candidate neighbor rank is greater than current rank then{ Send a message through one of the UP_path; Message_TAG = UP }
   - 2.3 Else{ send a message through the one of the DOWN-path channel; message_TAG = DOWN }
3. Else /* message_TAG = DOWN */{
   - 3.1 Find all the neighbors in which the rank of the node is less than current
   - 3.2 If there exists any neighbor, send a message to that neighbor else send a message through the neighbor, which the node rank is one less than the current node.

LEMMA 3.4.9 Channel dependency graph of UD-path algorithm is acyclic.

PROOF: Since UP-path and DOWN-path networks, based on the hamiltonian path, have no cycles in the network, and messages are routed from the UP-path to the DOWN-path or DOWN-path only, there is no message flow from the DOWN-path to UP-path. Thus, any message can't take a path from the DOWN-path channel to the UP-path channel. Therefore, the condition of the cycle in the channel dependency graph is denied.

THEOREM 3.4.6 The UD-path algorithm is deadlock free and non-minimal adaptive without virtual channels in CT_n, and ST_n respectively.

PROOF: From the LEMMA 3.4.9, channel dependency is acyclic and a message is finally delivered to the destination either to the UP-network or to the DOWN-network based on the directions that a message may take. Since a network can be partitioned into two sub-networks and a message does not take UP-path network right after taking the DOWN-path network, algorithm does not require virtual channels for deadlock freedom.

3. 5 SUMMARY

In this chapter, we extended the existing deadlock free routing algorithms to CT_n and ST_n with various resource requirements. Main issues in this chapter are centered on (a) how we avoid the cyclic dependency condition in the channel dependency graph, and (b) how we minimize the resource requirement, especially the number of virtual channels

per physical channel for deadlock freedom. To avoid cyclic channel dependency graph, one of the approaches is that messages are routed in the predetermined directions by the source and destination labels in the deterministic routing. In the adaptive routing, the physical network is partitioned into several logical networks with virtual channels, and by removing the dependency between the logical networks, routing algorithm guarantees deadlock freedom. Thus, the minimization of the logical networks is the key point for deriving cost effective routing algorithms. A deterministic routing algorithm based on the *e-cube* scheme in binary hypercube is proposed for CT_n. Also, for the tolerance of severe traffic congestion, we proposed various adaptive routing algorithms either minimal or non-minimal bases. We conclude by providing a comparison of these routing algorithms in the Cayley family of the networks based on the function of the size of the network, and the number of sufficient virtual channels for deadlock freedom in Table 3-1 and Table 3-2.

TABLE 3-1: Applicability of routing algorithms to the networks

Algorithms Networks	Dimension Order	*-channel	Negative-hop	Disrupt-hop	UD-path
Hypercube(BC_n)	Y	Y	Y	Y	Y
Torus (Q_n^k)	Y	Y	Y	?	Y
Star (ST_n)	Y	Y	Y	Y	Y
Complete Trans. (CT_n)	Y	Y	Y	Y	Y

TABLE 3-2: Virtual channel requirements on routing algorithms of networks.

Algorithms / Networks	Dimension Order	*-Channel	Negative-hop	Disrupt-hop	UD_path
Hypercube (BC_n)	1	2	$\left\lfloor \frac{n}{2} \right\rfloor + 1$	n	1
Torus (Q_n^k)	2	3	$\left\lfloor \frac{\left\lfloor \frac{k}{2} \right\rfloor n}{2} \right\rfloor + 1$	?	1
Star (ST_n)	n-1	n	$\left\lfloor \frac{\left\lfloor \frac{3(n-1)}{2} \right\rfloor}{2} \right\rfloor + 1$	n-1	1
Complete Trans. (CT_n)	1	2	$\left\lfloor \frac{n-1}{2} \right\rfloor +$	n-1	1

? : Unknown

CHAPTER 4

PERFORMANCE EVALUATIONS

In this chapter, we investigate the performance metrics and simulation environment for evaluation of the algorithms presented in Chapter 3 on several classes of networks. We present the performance comparisons from the extensive simulation study. The extraction of the analytical model in the wormhole routing is difficult to develop. Since it is hard to capture the model of dynamic situation of the network, message traffic distribution and the different probability of channel usage by the algorithms are unpredictable [Yo1997]. Therefore, we evaluate the various routing algorithms by empirical simulation study. The simulator used was originally built for *mesh* and *torus* networks by Boppana and Chalasani. We modified their simulator to accommodate various Cayley networks of permutation groups and revised some of the internal structure of the router model for better observation of the performance. The simulator uses a register-transfer level to compare the performance of the algorithms with various resource parameter settings.

In section 4.1, we introduce our simulation model and the main performance parameters used to compare the routing algorithms. Section 4.2 contains performance plots and analysis several compatible networks. In section 4.3 we analyze the performance of the networks among the very similar sizes. Section 4.4 is the summary.

4.1 SIMULATION ENVIRONMENT

For the simulation of the wormhole routing algorithms, we built logical interconnection networks using a set of C++ classes supporting: network topologies, communication patterns, data collection routines, and routing policies. A separate input file contains a specification of the simulation: types of network, internal structure of the router, the data collection interval, the number of resources installed in the router, and some random number seeds. An example of a simulation input file is illustrated in Figure 4-1. As soon as the simulator is executed, it builds a network based on the information in the input file. The internal structure of the router on each node is determined at this stage.

```
{
dimen:              7
network:            hypercube
processor:          wormhole
router:             star2
traffic:            uniform
min_dumps:          3
max_dumps:          10
inject_prob:        arr.rate
msgs_per_dump:      20000
warmup_cycles:      2000
interdump_cycles:           0
report_every:               0
link:               duallink
node_delay_for_headflits:   1
node_delay_for_dataflits:   1
link_delay:         1
flow_delay:         1

msgsize_in_flits:   20
consume_buffer_length:      8
consume_every_ncycles:      1
messages_per_consume:       8

slots_per_class0:           1
slots_per_class1:           1
lanes_per_class0:           1
lanes_per_class1:           1
max_injection_buffers:      inject.size
wh_lanes_per_cycle:         2
wh_queuelength:             4
wh_flowthresh:              2
dynamic_multiplex:          ON
reset_lanes_every_ncycles:  4

message_latency_duration:message_life_cycle
dumping_method:waiting_dump

% Seeds for random numbers, one set for each
dump
random_seeds0:  8833 0795 5065 7169 4239
2958 6949 8317 7628
random_seeds1:  9352 4164 8593 2682 1457
4995 7100 7095 0859
}
```

Figure 4-1. An example of simulation input file.

4.1.1 Performance parameters

The most important performance metrics in interconnection network performance are *latency* and *throughput*. Latency is the time span taken by a message from message generation to delivering a message to a destination. Since latency is measured through the lifetime of the message in the network, we can further split the time as *transmission time* , *set-up time* in the router , and *waiting time* in the network. Transmission time refers to the duration of flit delivery between a pair of nodes. Set-up time is the time taken by the router for flow control and the duration of the routing decision. The waiting time of a message is measured by total time that a message is blocked in the router by other messages due to the contention of the channels or buffers on the way to its destination. We fixed transmission and setup time to a unit cycle such that either a deterministic or an adaptive algorithm takes the same time for a routing decision. However, this assumption is not true for real routers since adaptive routing takes more routing decision time than deterministic. For the more detail router cost model based on specific CMOS gate array technology can be found in [Chie1993]. The author suggested that adaptive routers are generally significantly more complex, so the flow control time of the adaptive router is longer than for the deterministic router. In this dissertation, we fixed router flow time on deterministic and adaptive algorithms for the simplicity of simulation. The *throughput* of a network is the number of messages delivered per unit of the time. Besides the latency and throughput, utilization of a physical channel or a virtual channel would be valuable performance parameters for the observing the maximum use of resources. We measured the average network latency and throughput based on the given traffic loads separately. In

Chaos Normal Form (CNF) presentation of performance [DuNiYa1997], two plots are used to evaluate the performance of the interconnection networks. In one plot, latency is displayed by the function of the applied loads, and another plot displays the throughput as a function of the applied loads. The reason we use two plots is that interpretation of the graph is not simple in wormhole routing over the saturation point when we use only one plot by the function of the network's capacity to latency. For example, the latency on or over the network saturation point may increase when the utilization of the channel is decreasing. This graph does not represent real performance characteristics of the network beyond the network saturation point.. If we use another graph to plot the throughput, the performance variation above the saturation point is visible.

4.1.2 Workload models

To evaluate and simulate the interconnection network, real workload models or traffic patterns are needed. However, the extraction of a real workload model is difficult due to the differences from one architecture to another, and from one application to another. Thus, researchers have proposed synthetic workload models for comparison purposes. The workload model is basically defined by three parameters: *distribution of destination*, *injection rate*, and *message length*. The distribution of destination can be categorized to *uniform* (random) or *non-uniform* traffics. In uniform distribution, each of the nodes has an equal probability to be selected as a message destination by random number generation. In non-uniform distribution, the destination of the message in a source node is usually fixed for the duration of the simulation. *Bit-reversal, dimension-*

reversal, perfect-shuffle, butterfly, matrix transpose, and complement traffics are among the non-uniform distribution patterns used in simulations. In our simulation, we used the uniform traffic loads only since the distribution represents the behavior of nodes that are equally likely. The injection rate is usually the same for all the nodes. Among the popular distributions, *poison* and *exponential* distributions are used as an injection rate in each of the nodes. We used exponential distribution as the message injection rate in our simulation. The message length in our simulation is fixed to 20 flits including one headflit. However, the message length may be variable for the study of the sensitivity of network performance by the different message lengths.

4.1.3 Router model

In wormhole routing, the buffer organization in the router is one of the critical features that needs to be considered for designing an efficient router. In traditional wormhole router, each virtual channel has *dedicated buffers*. Thus, incoming flits are placed in the input buffer, and outgoing flits are placed in the output buffer after switching arbitration with one large switching crossbar. This scheme severely increases the number of flit buffers when a routing algorithm uses a large number of virtual channels or a network has a high degree of nodes. For example, if the sufficient number of virtual channels per physical channel is p then the total number of input flit buffers is computed by $2mp$ where m is the number of nodes. In a *2-dimensional torus* network with **-channel* algorithm, each node requires 2 x 4 x 3 = 24 input flit buffers since the

degree is 4 and a **-channel* algorithm requires 3 virtual channels per physical channel. Also, the same number of flit buffers is required for outbound flits. To reduce the burden of the large amount of the buffer space, researchers examined the virtual channel placement alternatives in a cross bar switch, and evaluated their effectiveness [BoDa1997][BoCh1996]. According to their reports, the placement of flit buffers inside the two cross bar switches significantly increased the performance of message latency over the traditional input/output dedicated buffer router systems. We placed the flit buffers between the two cross bar switches, and these flit buffers were shared by the input/output channels simultaneously and called *centralized buffering* scheme. Since flit buffers are shared by incoming and outgoing flits, direct deadlock could occur between incoming flit and injection flit unless direct deadlock prevention scheme existed. To prevent this direct deadlock, exclusive flit buffers are placed for injected flits. Thus, injected flits are placed in the exclusive flit buffers only before they leave the source node. We fixed the number of exclusive flit buffers as the same number of node degrees.

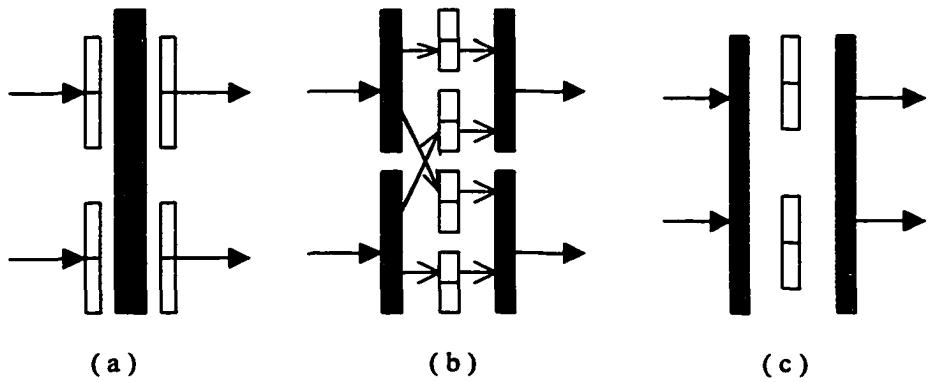


Figure 4-2. Router buffering schemes. (a) Dedicated 2 x 2 flit buffer switch, (b) 2 x 2 middle-buffered switch, and (c) 2 x 2 centralized flit buffer switch

This scheme is also different from a *middle buffering* system [BoDa1997]. In a middle buffer system, all the input and output flit buffers are placed between two cross bar switches. The total number of flit buffers is same as a dedicated system and only the placement is different. In a centralized system, only one flit buffer is used for both incoming and outgoing channels. Thus, we can decrease the number of flit buffer. Figure 4-2 illustrates three types of buffering schemes. The size of the flit buffer (depth of buffer) is another factor for consideration in implementation. Theoretically, one flit size buffer per virtual channel is enough in wormhole routing, but the limited size of the flit buffer may bring network saturation point with a very small amount of traffic, so that the network capacity is decreased. If the size of the flit buffer is the same as the message length, the switching technology could be the same as the virtual cut-through because one flit buffer can hold a whole message. The router architecture used in our simulation is illustrated in Figure 4-3. In the figure, we fixed the buffer size to four flit buffers in each virtual class. Then the maximum number of flits in a queue, that a virtual class buffer can hold, is four.

In Figure 4-3, the flow control in the centralized organization is done in cross bar 1. So, when a header flit arrives, say, from Node 1 to Node 2, it is allocated a central buffer by establishing a connection through cross bar 1 of Node 2 or is refused connection. The headerflit is retained in the Node 1 for a few cycles, by which time rejection of the header, if it occurs, will be known.

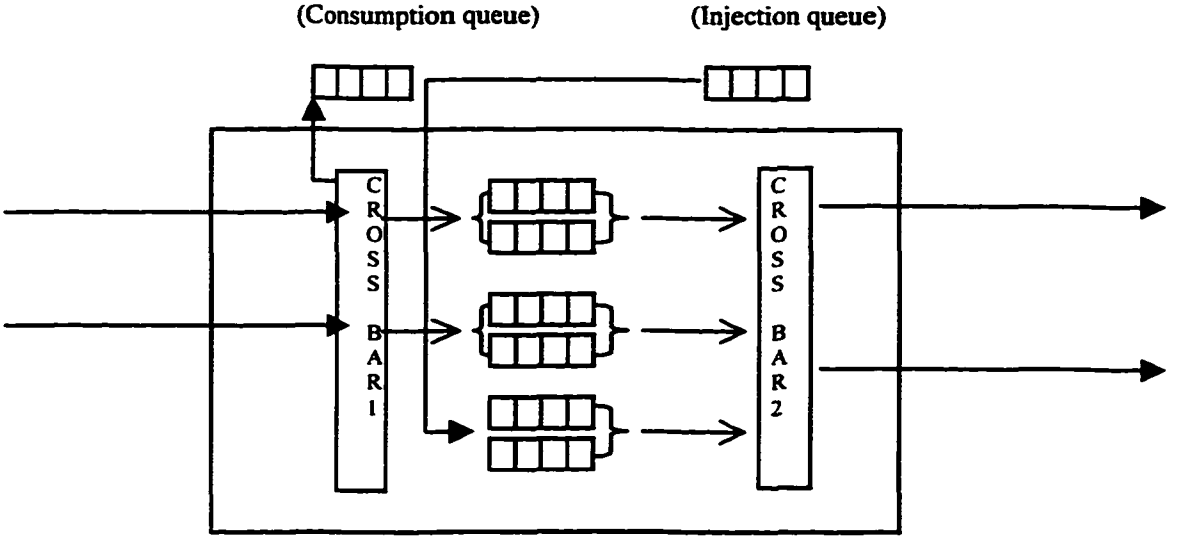


Figure 4-3. An example of centralized flit buffer router architecture with buffer length four.

Once the connection is established, the allocated central flit buffer acts as a dedicated flit buffer to that virtual channel, and the transit of data flits is similar to that of dedicated flit buffer implementation. The cross bar delay in the dedicated buffer organization is eliminated in the centralized buffer organization. The virtual channel controller is implemented using Cross bar 2. Once a flit buffer is allocated with that virtual channel, it remains associated with that virtual channel until it is released. Therefore, a multiplexer placed between the input channels and buffers in the logical organization is set once at the time of setting up the path. Flits coming on a physical channel are available at all the flit buffers allocated to them and an appropriate flit buffer accepts its corresponding flit. This is similar to storing a flit in one of the appropriate flit

buffers associated with the physical channel in the dedicated buffer organization. Therefore, the connection from an input virtual channel to an output virtual channel takes more time, and data flits go through two smaller cross bars instead of one large cross bar in a dedicated flit buffer organization. But the centralized organization with buffers between the crossbars lends itself easily to pipelining, thereby avoiding an increase in clock cycle time. Since the centralized organization has a longer data path, the router delay for a message increases compared to the dedication organization, when the number of buffers is kept the same.

4.1.4 Simulations

We simulated the five classes of routing algorithms in several different topologies with a sufficient number of virtual channels and a centralized flit buffer organization. Messages are generated at each node randomly with a geometric distribution of inter-arrival time, and message length is fixed at 20 flits. An example of the headflit class object is illustrated in Figure 4-4. A headflit is a derived type of class, which includes a public flit object class, and all the information of the routing and virtual channel classes.

The uniform random generator was used for the decision of a message destination. For each of the simulation steps, message generators produce 20,000 messages and wait for all the messages that are destined for their destination. To exclude unstable initial traffic, we did not collect performance data until 2,000 (10%) messages were destined. For simplicity of simulation, we assumed that the node delay is fixed as a

unit time (one flit time). Also, message queuing delay in the injection queue was included for the latency of the message. The selection functions of the adaptive algorithms are fixed so that the first candidate node in the array is checked first for availability, then the next candidate is checked and so on. To control the flow of the injection traffic, we designed the injection queue large enough so that all the messages generated during the simulation will not be rejected due to the lack of the injection queue or the congestion of the network.

```
class flit_obj
{
public:
        flit_obj  *next;                    // For chaining flits
        flit_types          type;           // Head flit, data flit, tail flit, etc
        shint               msgsrc;         // Processor who generated message
        shint               msgdst;         // Processor who will consume message
        uint32              gtime;          // Time message was generated
        shint               flits_left;     // Flits left after this flit in the msg
        shint               dst_lane;       // For wormhole routing lane support
};
class headflit_obj: public flit_obj
{
public:
        shint               host;           // Current host of message
        shint               lasthost;       // Last host of message
        boolean             first_arrival;  // Whether the message has been routed.
        shint               hops;           // Number of hops msg has taken
        misroute            misrouted;      // Misrouted status
        ntuple              src_ntuple;     // Source of msg as an ntuple
        ntuple              dst_ntuple;     // Destination of msg as an an tuple
        shint               size_in_flits;  // Size of message in flits
        shint               msg_dimen;      // dimension in which the message is blocked
        msg_directions      msg_dir;        // types of links to be used in the dimension
        fring_directions    fring_dir;      // Message direction in an fring
        shint    choice_nbrs[MAX_LINKS_PER_PROC+1]; //set of candidate neighbors
        shint    neighbor_decision;
        shint    rclass;                    // Current routing class
        void     *rptr;
};
```

Figure 4-4. Internal components of flit and derived type of headflit.

The multiple flit buffers in the router accept a headflit when all the buffers, here four buffers, are empty. Thus, there is no case where one flit buffer holds different message flits simultaneously. The performance data of each message is collected when a message is delivered to its destination, and the statistics of simulation were computed after all the 20,000 messages were destined. This simulation continued until the following conditions were satisfied for the reliability of the performance data: (1) simulation should be performed at least three times, (2) after three rounds of the simulation, if the standard deviation between the average latency and throughput in each of the simulation is converged to 0.05, (3) the count of simulation is reached to ten, such that the standard deviation does not converge within the range. For all the results in this paper, 95%-confidence intervals are ±5% of the respective values reported.

4.2 PERFORMANCE COMPARISONS

We first simulated CT_4, ST_4, BC_5, and *4-ary 2-cube* torus networks based on the routing algorithms that the number of virtual channels is not greater than their diameter. Theoretically, all the algorithms presented in Chapter 3 can apply to any network with a considerable number of virtual channels. However we limited our simulation study of algorithms to only the algorithms which require a moderate number of virtual channels so that the number of virtual channels used is not over the dimension of the networks.

4.2.1 Complete Transposition Networks

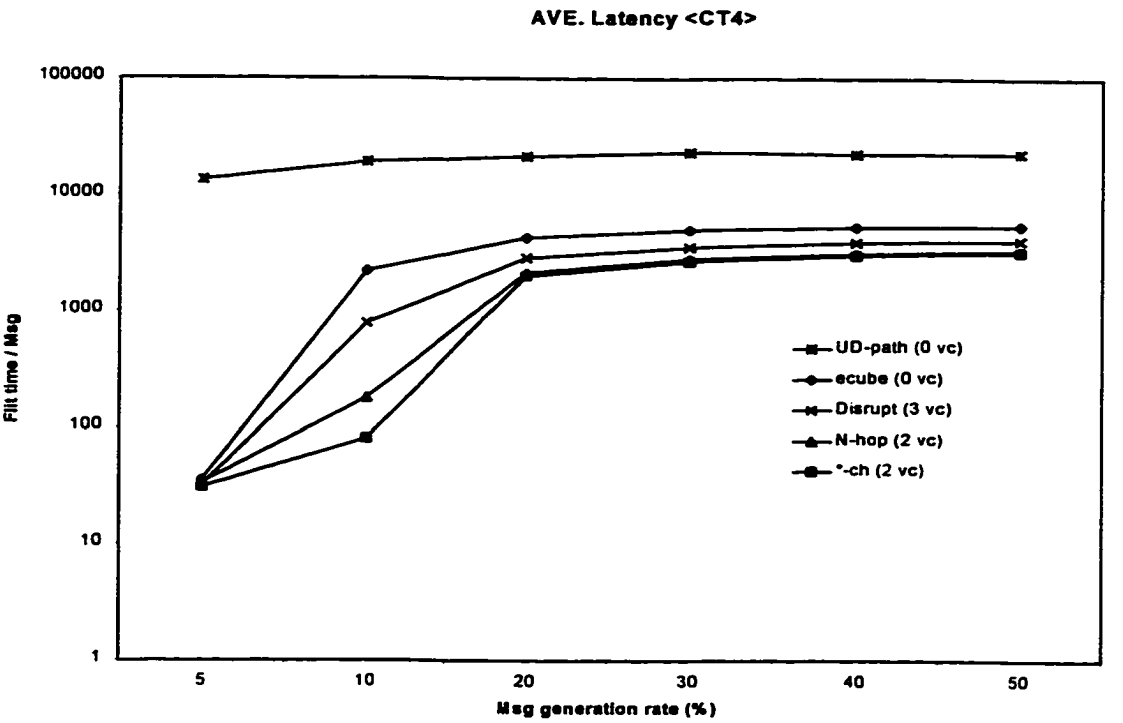
In the simulation of the complete transposition networks, we built CT_4 and CT_5 with all the five classes of algorithms with a sufficient number of virtual channels given by chapter 3. The size of the networks and the number of virtual channels are illustrated in Table 4-1.

Table 4-1. Simulation parameters of *complete transposition* networks.

Network		CT_4	CT_5
# of Processor		24	120
Degree		6	10
Virtual channels	Dimension order	1	1
	Negative-hop	2	3
	Disrupt-hop	3	4
	*-Channel	2	2
	UD-path	1	1

In the simulation of the CT_4 network, adaptive algorithms outperformed the deterministic algorithm except the UD-path algorithm. Among the adaptive algorithms, *-channel algorithm latency outperformed any other adaptive algorithms with only two virtual channels. At very low traffic loads, less than 10%, most of the deterministic and fully adaptive algorithms showed very close performance curves. At the 10% traffic load, we can easily distinguish the performance sensitivity of the algorithms and effect of the additional virtual channels. The UD-path algorithm, partially adaptive and non-minimal, showed the worst performance among the algorithms. Intuitively, this situation can be

anticipated by the unbalanced traffic distribution to the UP-path or the Down-path networks in the algorithm. Since a message in the UP-path network can't proceed if any one of channels exists in the UP-path network, the message can't take the DOWN-path channel, even if it is one of the shortest path neighbors. Thus, when there are other choices in the DOWN-path network, the message has to wait for a channel in the UP-path network rather than taking the DOWN-path channel. Though the algorithm in CT_4 is minimal, it does not show better performance over the deterministic algorithm. The performance graphs of routing algorithms in CT_4 are Figure 4-5. In the simulation of the CT_5, a negative-hop algorithm with *three* virtual channels showed best performance over the saturation point at around 20% traffic loads. Though the *-channel algorithm showed better latency and throughput over the negative-hop algorithm at less populated traffics, adding one more virtual channel increased performance about 10% with respect to latency and throughput. Also, it delays the saturation point such that it can hold more traffic than the other algorithms. Thus, the curve of the negative-hop algorithm in Figure 4-6 shows that it crosses the *-channel algorithm curve between 10% and 20%. Though the disrupt-hop algorithm used four virtual channels same with the diameter as the CT_5, it was not advantageous for the performance since the use of virtual channels is tightly related selection function in the algorithm. Thus, we confirm that the selection function does not affect to the deadlock but it affects the performance of routing algorithms [FeGr1991]. The performance plots of the CT_5 illustrated in Figure 4-6.



Throughput CT4

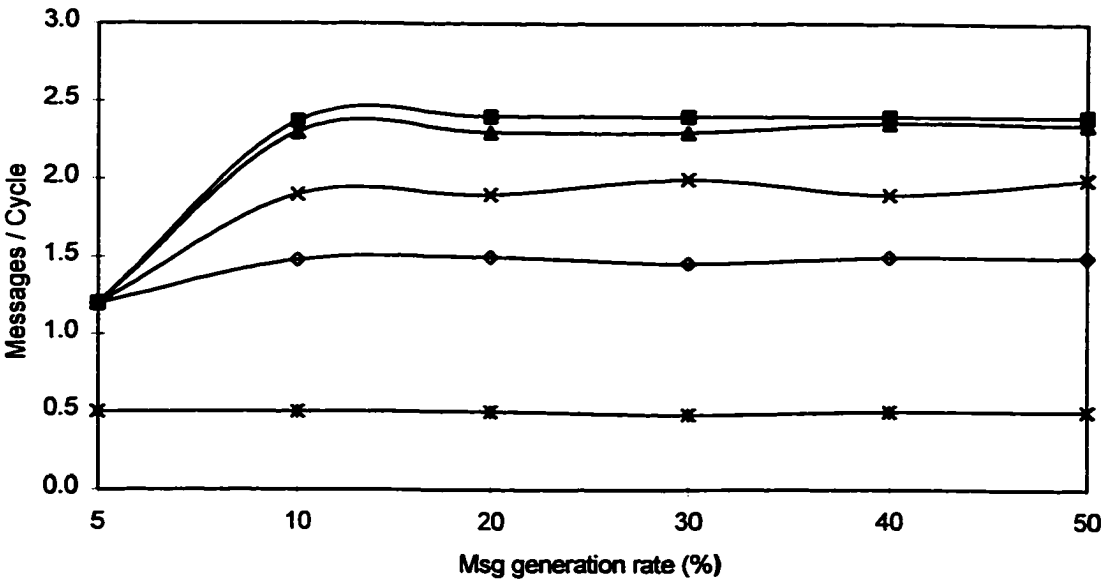


Figure 4-5. Performance plots of algorithms in CT_4.

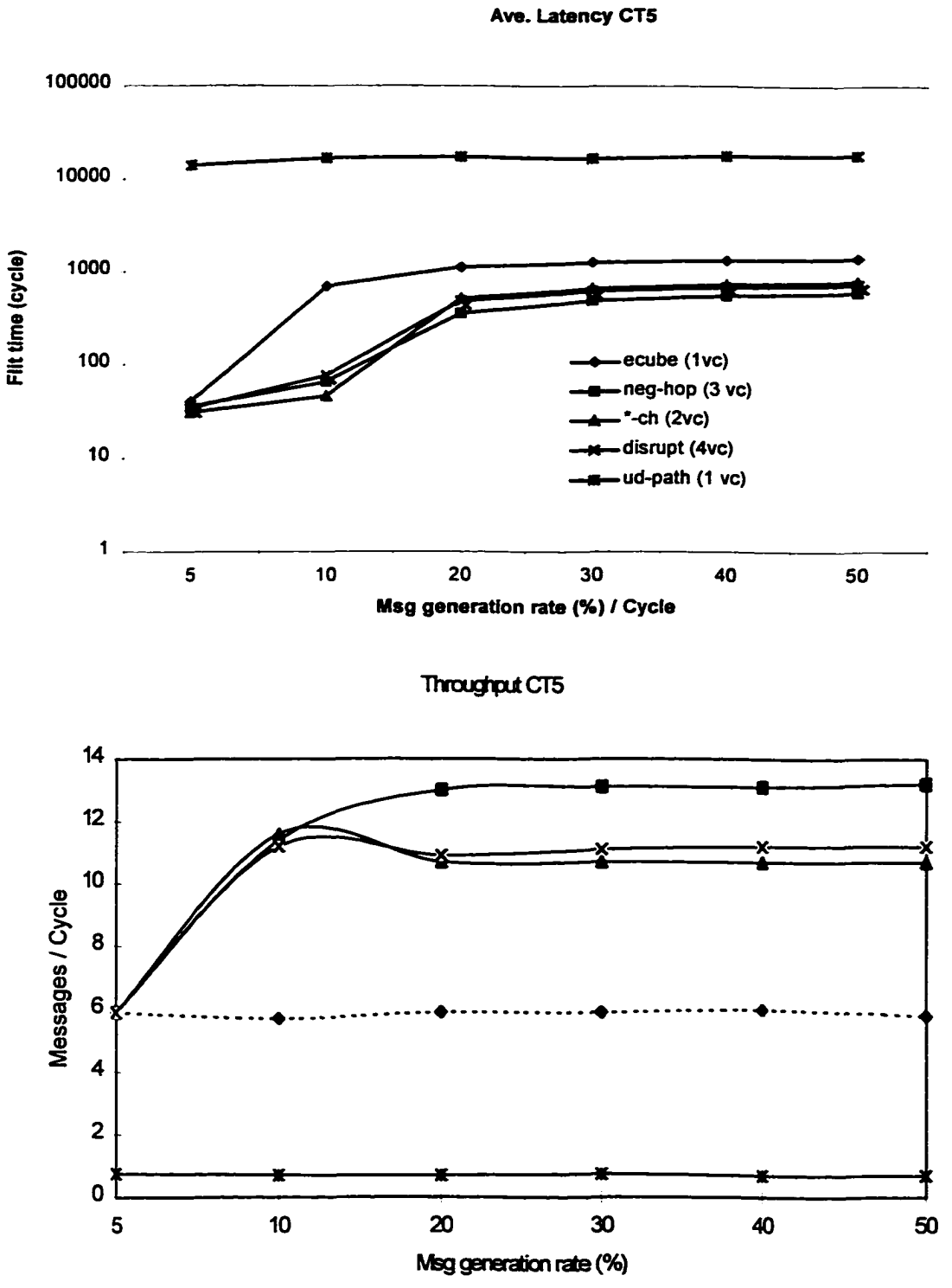


Figure 4-6. Performance plots of algorithms in CT_5.

4.2.2 Star Networks

For the performance comparison in the Star networks, we simulated ST_4, and ST_5 networks with negative-hop, disrupt-hop, and UD-path algorithms. Since these algorithms are not confined to specific network topologies, we can apply the simulator in CT_n to ST_n with a modification of the distance computing function by property of (2. 3). The simulation parameters of the Star networks are illustrated in Table 4-2.

Table 4-2. Simulation environment of Star networks.

Network		ST_4	ST_5
# of Processors		24	120
Degree		4	6
Diameter		4	6
Virtual channels	Negative-hop	3	4
	Disrupt-hop	3	4
	UD-path	1	1

The result of the simulation shows that the negative-hop algorithm is still favorable over the disrupt-hop algorithm, but the performance difference is less than 2%. Both algorithms use an equal number of virtual channels and the networks are saturated at a very early stage of the simulation in ST_4 and ST_5. In both cases, the networks are saturated between the 5% to 10 % ranges. Throughput in ST_4 is almost the same as either the negative-hop or the disrupt-hop algorithm. However, in the ST_5 simulation, the negative-hop throughput is placed over the disrupt-hop algorithm. Therefore, the negative-hop algorithm may provide performance gain over the disrupt-hop algorithm in larger networks. Figure 4-7 and Figure 4-8 show the performance of star networks with three classes of algorithms. Like in CT_n, the performance curve of UD-path algorithms do

not show any interesting results because they are non-minimal and some messages take longer paths rather than taking shortest paths, available.

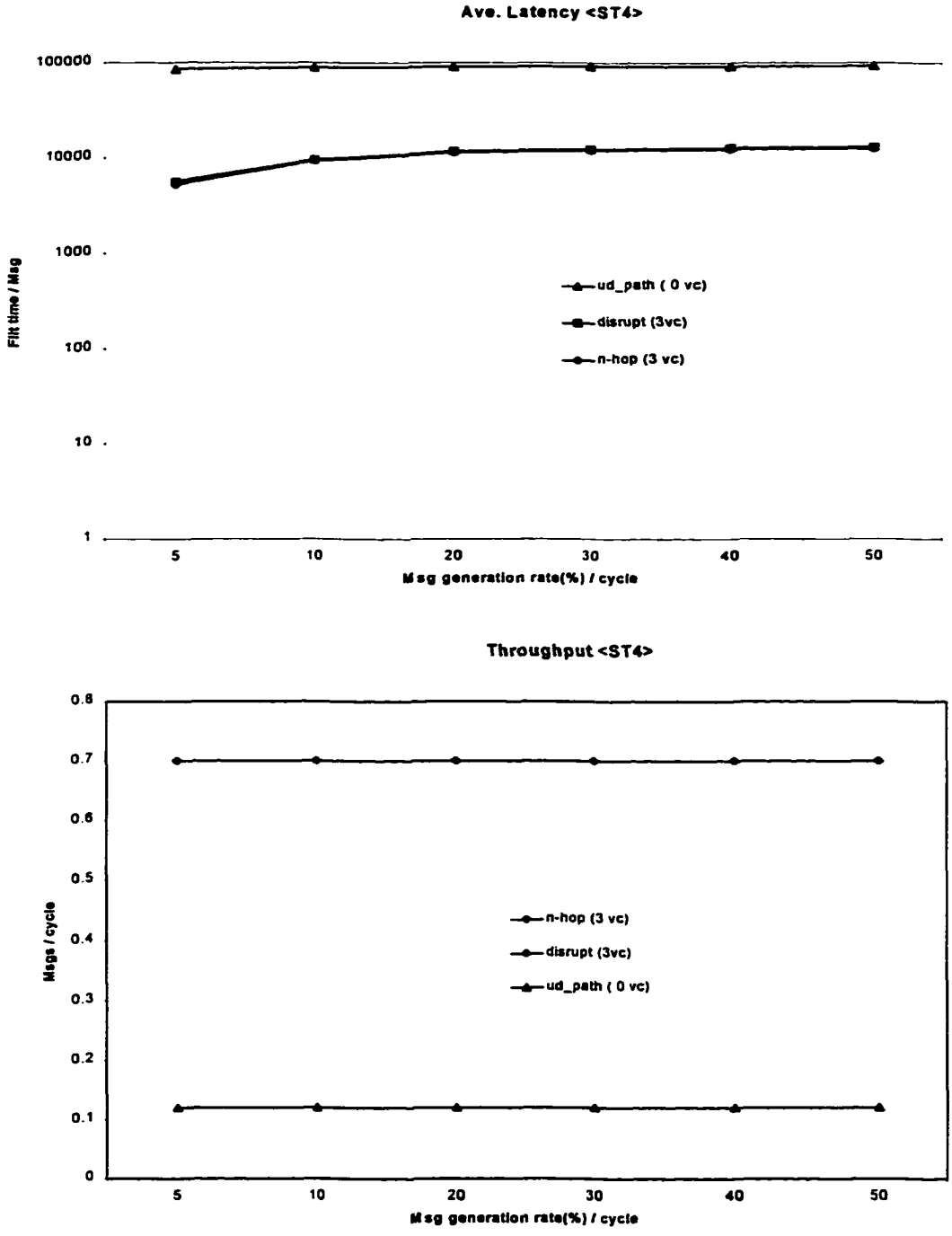


Figure 4-7. Performance plots of algorithms in ST_4.

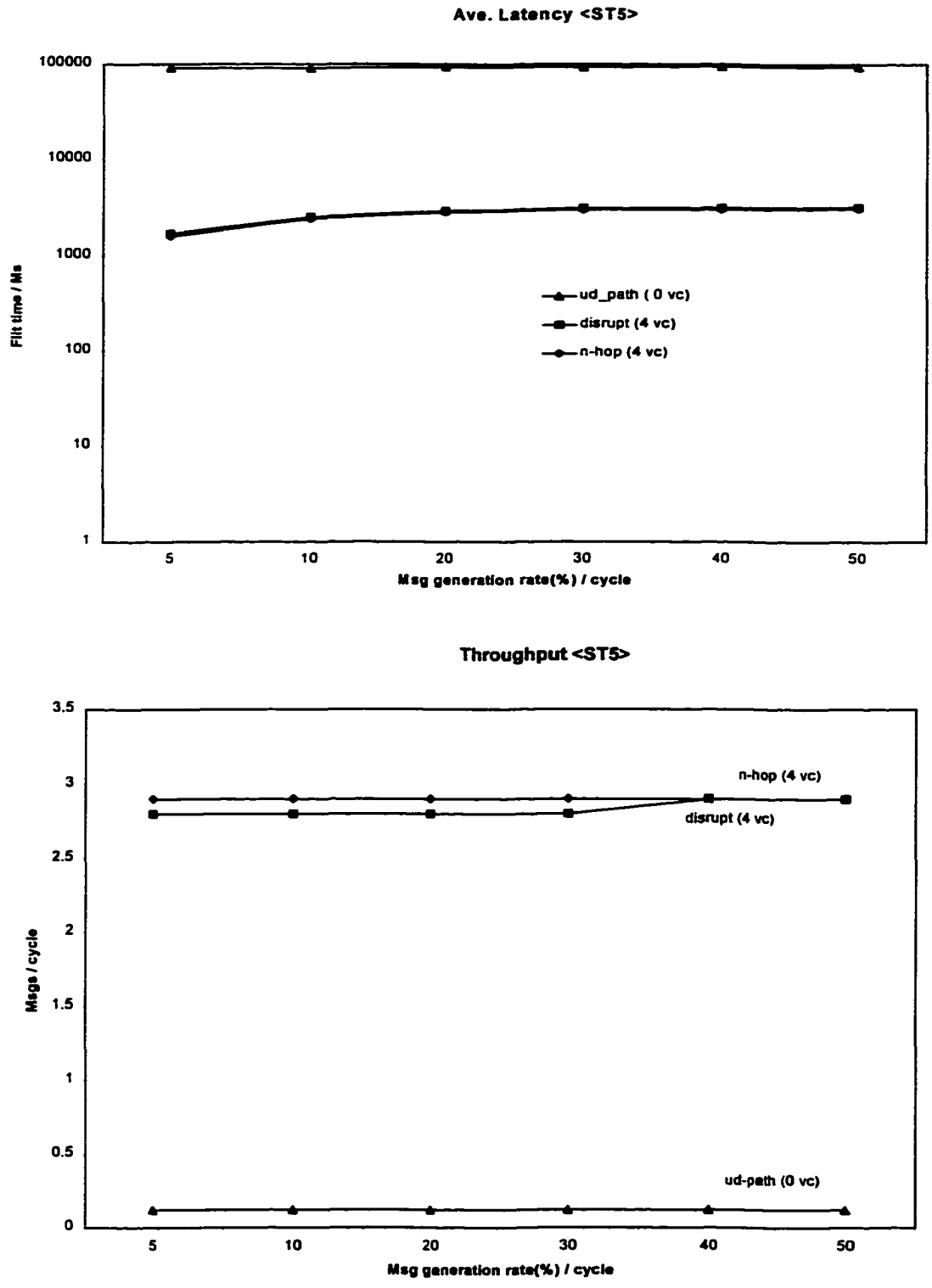


Figure 4-8. Performance plots of algorithms in ST_5.

4.2.3 Binary Hypercubes

Our simulation extended to binary hypercube networks for the validity of algorithm performance on different topologies. Since our routing algorithms are originally based on application to the binary hypercubes, we can apply four classes of algorithms to BC_5 and BC_7 networks so that the sizes of the networks are very similar to our previous simulations for CT_n or ST_n. The simulation parameters are illustrated in Table 4-3. The simulation results are very similar to CT_n networks, though the usage of virtual channels is different between the *-channel and the negative-hop algorithms. At 5% message generation rates, the *-channel algorithm with two virtual channels outperformed the negative-hop algorithm with three virtual channels either BC_5 or BC_7. With increased message generation rates, however, the negative-hop algorithm outperformed the *-channel algorithm either in average latency or in throughput. Also, both the deterministic algorithm and the UD-path algorithm saturated at a very early stage of the simulation at even less than the 5% message generation rates.

Table 4-3. Simulation parameters of *binary hypercube* networks.

Network		BC_5	BC_7
# of Processors		32	128
Degree		5	7
Diameter		5	7
Virtual channels	e-cube	1	1
	*-channel	2	2
	Negative-hop	3	4
	UD-path	1	1

Figure 4-9 and Figure 4-10 present the performance plots of algorithms in BC_5 and BC_7, respectively. In both performance plots, the negative-hop algorithm shows the best performance curves on or over the network saturation points with more resources.

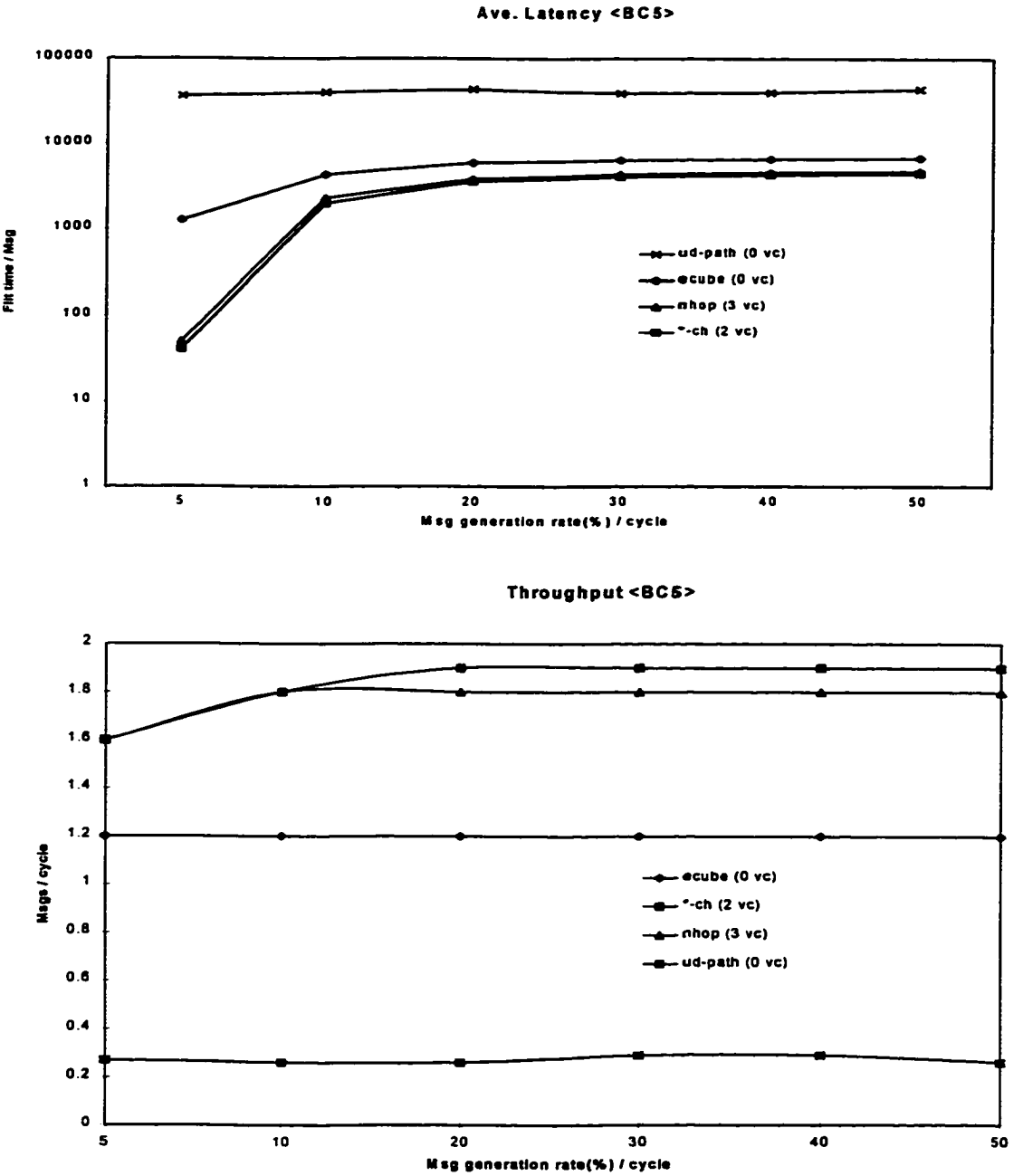


Figure 4-9. Performance plots of algorithms in BC_5.

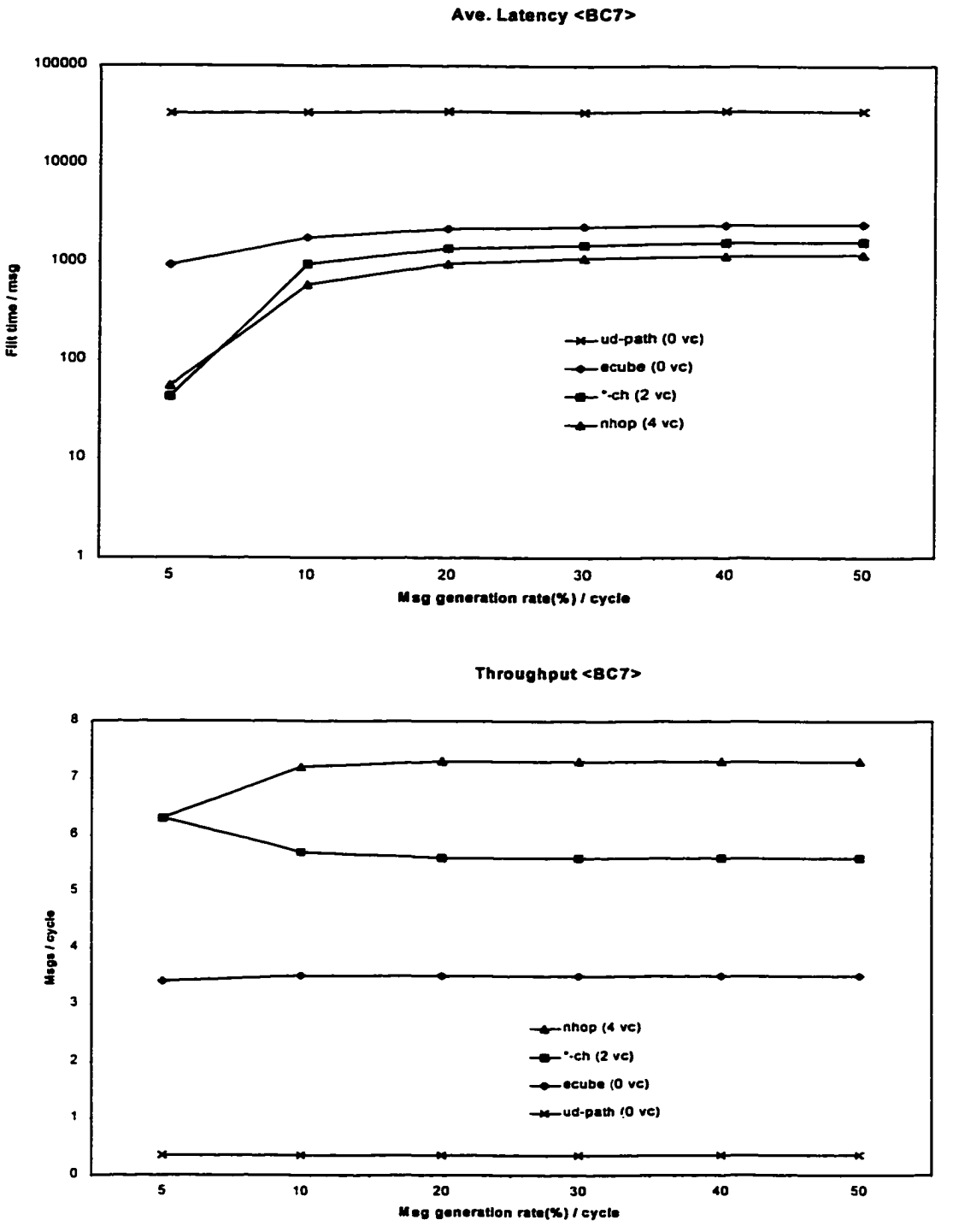


Figure 4-10. Performance plots of algorithms in BC_7.

4.2.4 Torus networks

The simulation of the torus network was performed with four classes of algorithm: dimension-order, *-channel, negative-hop, and UD-path algorithm. Each of the algorithms was evaluated with a minimum number of sufficient virtual channels. For the simplicity of the simulation, we constructed on only 2-dimensional torus networks. The simulation parameters illustrated in Table 4-4 are based on the algorithms. The results on the torus networks show that the networks are saturated at very light traffic generation rates at about 10% traffic loads. The plots of the throughputs are maintained at almost the same levels regardless of the traffic generation rates except for the negative-hop algorithm in 11x11 torus network with six virtual channels. In the 5-ary 2-cube torus network, the *-channel algorithm with 3 virtual channels outperformed the negative-hop algorithm with the same virtual channels, with the same result in CT_4 and BC_5, the previous simulation. However, in the large network (11-ary 2-cube), the negative-hop algorithm outperformed the *-channels algorithm at all the rates of message generation. Even the ratio of the throughput between the negative-hop and the *-channels algorithms increased from 10% in (5-ary 2-cube) to 70% (11-ary 2-cube).

Table 4-4. Simulation parameters of *torus* networks.

Network		5-ary 2-cube	11-ary 2-cube
# of Processors		25	121
Degree		4	4
Diameter		4	10
Virtual channels	e-cube	2	2
	*-channel	3	3
	Negative-hop	3	6
	UD-path	1	1

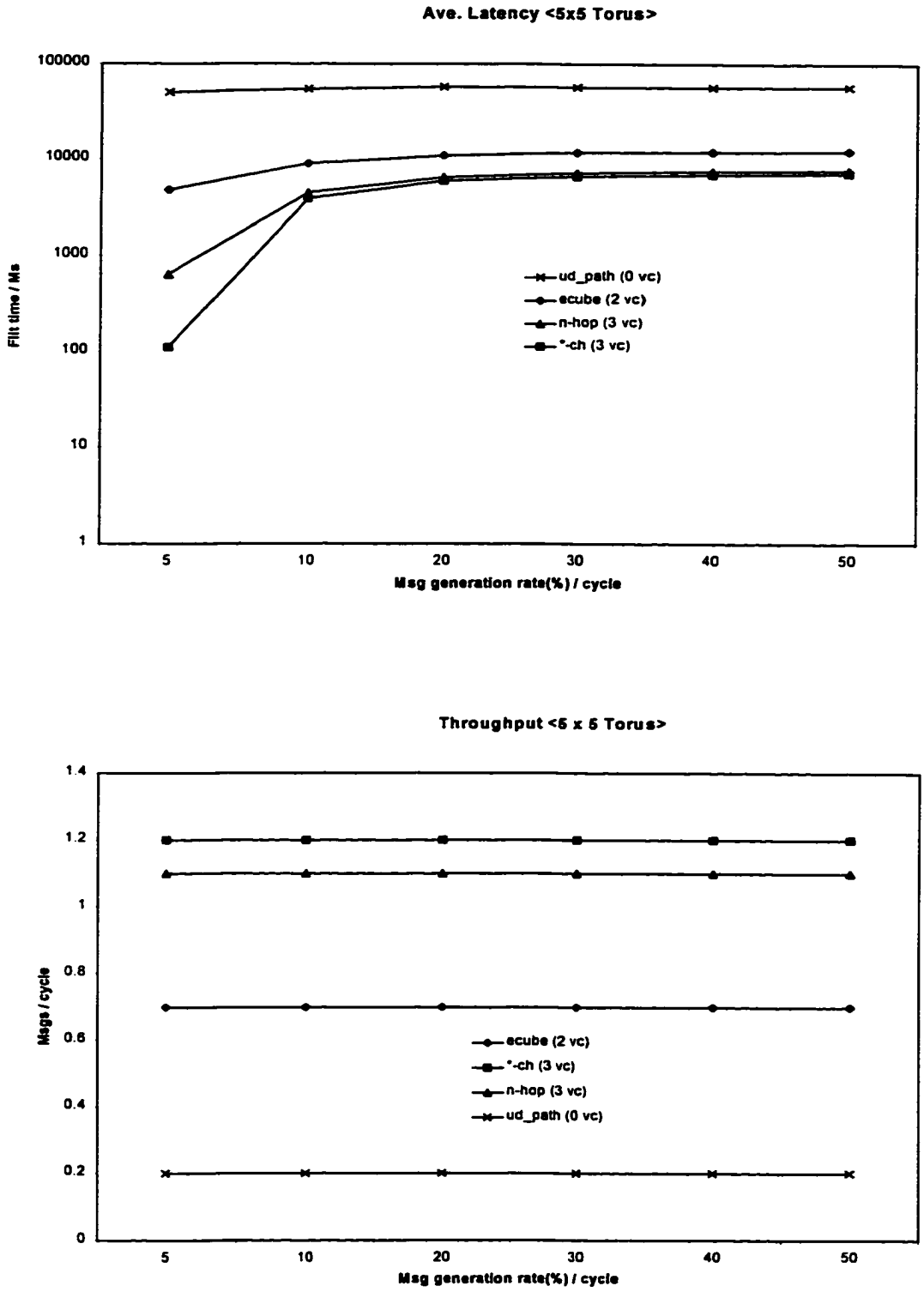


Figure 4-11. Performance plots of algorithms in 4-ary 2-cube.

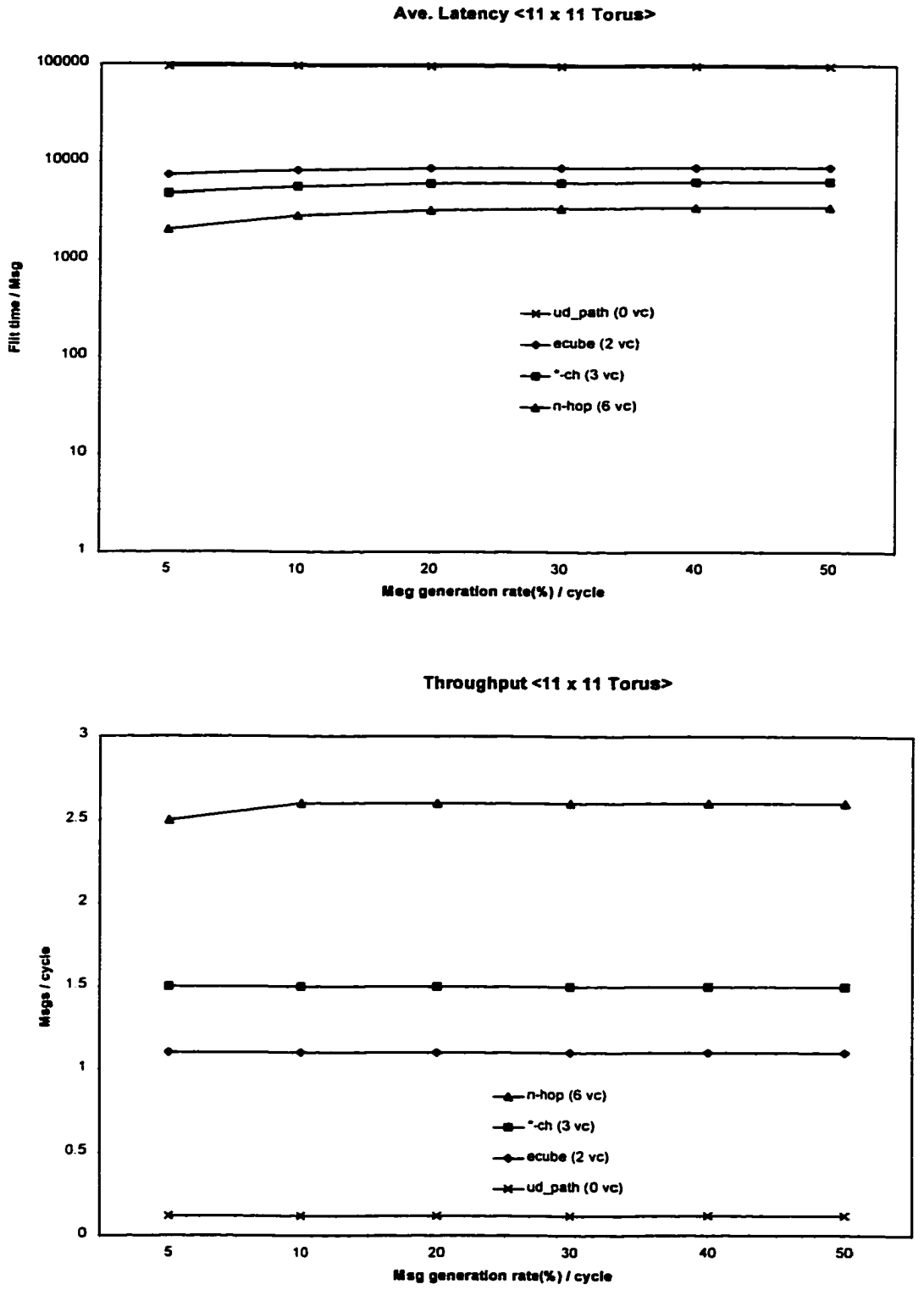


Figure 4-12. Performance plots of algorithms in 11-ary 2-cube.

4.3 EVALUATION OF NETWORKS

The performance comparison between CT_4, ST_4, and a 5-dimensional hypercube is illustrated in Figure 4-13 and CT_5, ST_5, and the 7-dimensional *binary hypercube* is illustrated in Figure 4-14 with several classes of routing algorithms. Even though absolute comparison of performance networks is not possible due to the different topological properties such as degree, diameter, etc, it may be worthwhile to compare performance with networks of similar size. Among the networks, CT_n outperformed the other networks with respect to either deterministic or adaptive algorithms. These results are already expected since CT_n has the largest degree and smallest diameter among the interesting networks. For the ST_n, two fully adaptive minimal algorithms (negative-hop and disrupt-hop) showed the worst performance among the three networks both in latency and in throughput. Furthermore, the ST_n network saturated at very low traffic loads with more virtual channels than the *-channel algorithm in hypercube network. Thus, we confirmed that there exists a trade off to choose interconnection network. Given the number of processors, we need to carefully consider the optimal point between communication performance gain and the cost of physical channels. In our simulation, the *binary hypercube* still could be a favorable choice over the ST_n, and CT_n would be the best alternative with respect to routing flexibility and fast message latency time.

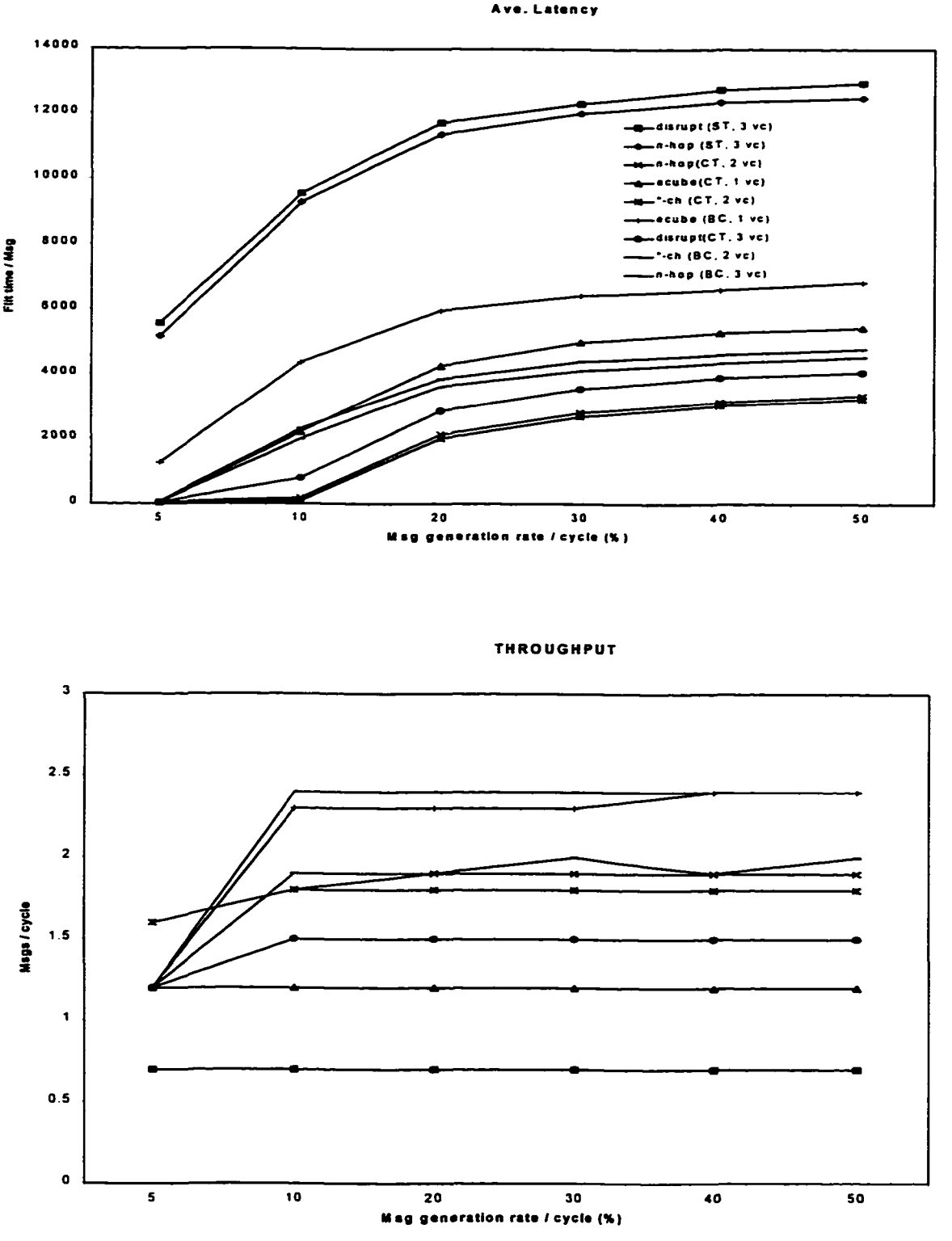


Figure 4-13. Performance plots of networks with CT_4, ST_4, and BC_5.

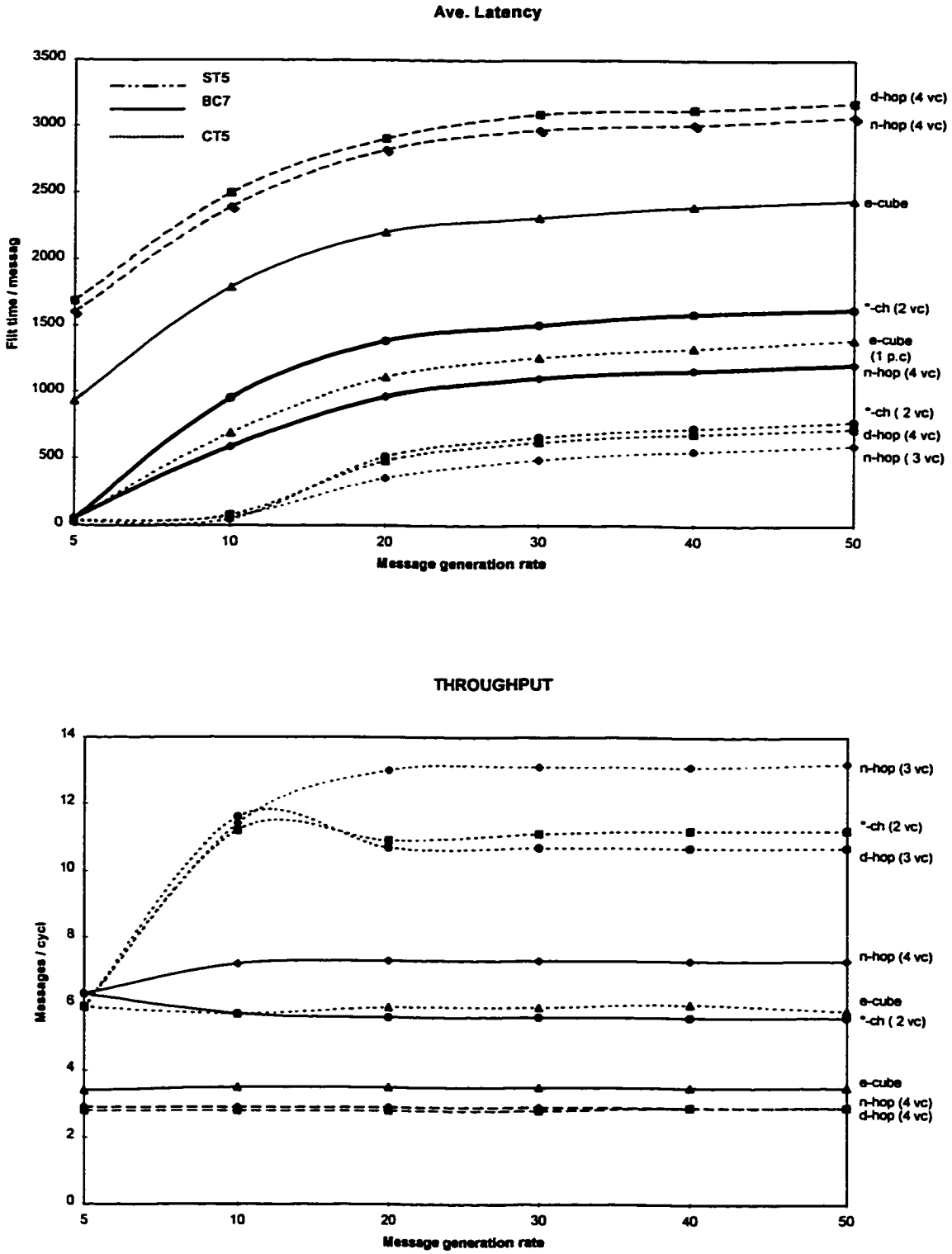


Figure 4-14. Performance plots of networks with CT_5, ST_5, and BC_7.

4.4 SUMMARY

We investigated the performance of routing algorithms in four classes of networks by the simulation based on the centralized flit buffer system. Our simulation focused on the performance of message delivery based on the uniform traffic pattern and synthesized traffic generation rate per cycle. The simulation results showed that fully adaptive and minimal routing algorithms outperformed deterministic routing algorithm with more virtual channels. Also, a deterministic algorithm outperformed a non-minimal partially adaptive algorithm with only a physical channel in both algorithms. Among the adaptive routing algorithms, the *-channels algorithm showed the best performance at sparse traffic loads over the adaptive routing algorithms, but the negative-hop algorithm outperformed the *-channel algorithm on or above the network saturation point of traffic. In the disrupt-hop algorithm on the CT_n or the ST_n, the algorithm uses more virtual channels than negative-hop algorithm but does not show better performance than the negative-hop algorithm. For the comparison of the networks, CT_n showed the best overall performance over the ST_n and the BC_n networks. Also, the binary hypercubes showed better performance than the star networks.

CHAPTER 5

CONCLUSIONS

In this dissertation, we have investigated various wormhole routing algorithms in the Cayley networks, which are considered very promising alternatives for distributed memory interconnection networks. We introduced the characteristics of wormhole routing schemes, and compared the effect of their technology over traditional switching schemes. In the wormhole routing algorithm, a deadlock avoidance scheme is the key point for reliable message transmission between processors. Since once any one of the flits of message is lost in the network, the message can't be delivered or retransmitted in the wormhole routing network, the traditional preemption resource control scheme for deadlock avoidance is not applicable to wormhole routing. In wormhole routing, deadlocks are avoided by the clever control of resources or by limiting selection of the routing path. In developing wormhole routing algorithms, transmission channels and flit buffers are the main resources in the router. In the deterministic routing algorithm, message flow directions are strictly limited by the given routing function, and there is no flexibility for routing decisions due to the traffic congestion. However, in the adaptive routing, messages can take any of the shortest paths on the way to the destinations with control of the virtual channels that share the physical channel's bandwidth.

In chapter 2, we introduced the basic framework for constructing Cayley graphs of interconnection networks and switching schemes in the router design. Recently, designs of interconnection networks based on group theory were considered very promising alternatives in substituting for traditional topologies. We showed that traditional binary hypercube and k-ary n-cube torus networks could be built by permutation group theory with a different node labeling system. Also, we focused on the issues of deadlock, livelock, and starvation in a wormhole routing interconnection network.

In chapter 3, we introduced the basic assumptions and tools of wormhole routing for developing deadlock free routing algorithms. We proposed a deterministic routing algorithm in a *complete transposition* network with physical channels between a pair of nodes. For the adaptive routing algorithm, we proposed three classes of deadlock free algorithms: negative-hop, disrupt-hop, and *-channel. For the purpose of comparing the performance of various routing algorithms, we presented a non-minimal partially adaptive algorithm called the UD-path algorithm based on *hamiltonian* path.

In chapter 4, we analyzed the performance of the five classes of routing algorithms by the simulation study. In our simulation, adaptive algorithms outperformed deterministic algorithms. In the comparisons between minimal deterministic and a non-minimal partially adaptive scheme, deterministic algorithm outperformed non-minimal partially adaptive algorithm with same simulation parameters. Among the minimal adaptive routing algorithms, the *-channel algorithm showed better performance over the

negative-hop and the disrupt-hop algorithm when the traffic is sparse. However, over the network saturation point of the traffic, the negative-hop scheme showed the best alternative among the other adaptive algorithms. Finally, the performances of the networks are analyzed among the various network topologies of similar size. Among these networks, the complete transposition network showed better performances over the binary hypercube and the star networks. Also, simulation study shows that binary hypercube networks are still favorable topologies over the star networks in communication performance prospective. Here, we list a number of further issues for research of wormhole routing algorithms based on the Cayley network.

(1) A sensitivity study of routing algorithms with real application of traffic patterns.

(2) A model of router delay based on the routing algorithms would be of interest.

(3) The effect of the virtual channels on limited bandwidth channels.

(4) Develop routing algorithms for another family of Cayley graphs of permutation groups and alternating groups such as *bubblesort* graph, *pancake* graph, *modified bubblesort* graph, and supertoroid graph.

REFERENCES

[AkKr1989] S. Akers and B. Krishnamurthy, "A Group-Theoretic Model for Symmetric Interconnection Networks", *IEEE Transactions on Computers*, vol. 38, no. 4, pp. 555-566. Apr. 1989.

[AnPi1995] Anjan V. and T. Pinkston, " An Efficient, Fully Adaptive Deadlock Recovery Scheme: DISHA", *Proceedings of ISCA'95*, pp. 201-210, Santa Margherita Ligure, Italy, 1995.

[BeBl1996] J. Beachy and W. Blair, Abstract Algebra Second Edition, Waveland Press, Inc., Prospect Height, Illinois, 1996.

[Bene1965] V. E. Benes, Mathematical theory of connecting networks and telephone traffic, Academic Press, New York, 1965.

[Bigg1974] N. L. Biggs, Algebric Graph Theory, Cambridge University Press, Cambridge, 1974.

[BoCh1996] R. Boppana and S. Chalasani. "Framework for Designing Deadlock-Free Wormhole Routing Algorithms", *IEEE Transactions on Parallel and Distributed System*, vol. 7, no. 2, pp. 169-183, Feb. 1996.

[BoDa1997] Y. Boura and C. Das, "Performance Analysis of Buffering Schemes in Wormhole Routers", *IEEE Transactions on Computers*, vol. 46. no. 6, pp. 687-694, June 1997.

[BoChLaDh1998] S. Boo, W. Chen, S. Lakshmivarahan, and S. Dhall, "Analysis of Wormhole Routing in Complete Transposition Graphs: A Simulation Study", *Proceedings of High Performance Computing 1998*, Boston, MA, pp. 174-179, Mar. 1998.

[Chie1993] A. Chien, "A Cost and Speed Model for k-ary n-cube Wormhole Routers", *Proceedings of Hot Interconnects '93*, Palo Alto, CA, Aug. 1993.

[DaAo1993] W. Dally and H. Aoki, " Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels", *IEEE Transactions on Parallel and Distributed System*, vol. 4, no. 4, pp. 466-475, Apr. 1993.

[Dall1992] W, Dally, "Virtual-Channel Flow Control", *IEEE Transactions on Parallel and Distributed System*, vol. 3, no. 2, pp. 194-205, Mar. 1992.

[DaSe1987] W. Dally and C. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks", *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547-553, May 1987.

[Duat1995] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks", *IEEE Transactions on Parallel and Distributed System*, vol. 6, no. 10, Oct. 1995.

[Duat1993] J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks", *IEEE Transactions on Parallel and Distributed System*, vol. 4, no. 12, pp. 1320-1331, Dec. 1993.

[DuNiYa1997] J. Duato, L. Ni and S. Yalamanchili, INTERCONNECTION NETWORKS: An Engineering Approach, IEEE Computer Society Press, 1997.

[FeGr1991] S. Felperin, L. Gravano, G. Pifarre, and J. Sanz, "Routing Techiques Massively Parallel Communication" *Proceedings of the IEEE*, vol. 79, no. 4, pp. 488-503, Apr. 1991.

[GiNi1994] C. Glass and L. Ni, "The Turn Model for Adaptive Routing", *Journal of ACM*, vol. 41, no. 5, pp. 874-902, Sep. 1994.

[Gopal1985] I. Gopal, "Prevention of Store-and-Forward Deadlock in Computer Networks", *IEEE Transactions on Communications*, vol. Com-33, no. 12, pp. 1258-1264, Dec. 1985.

[GrPiBeSal1994] L. Gravano, G. Pifarre, P. Berman, and J. Sanz, "Adaptive Deadlock- and Livelock-Free Routing With all Minimal Paths in Torus Networks", *IEEE Transactions on Parallel and Distributed System*, vol. 5, no. 12, pp. 1233-1252. Dec. 1994.

[Gunt1981] K. Gunther, "Prevention of Deadlocks in Packet-Switched Data Transport Systems", *IEEE Transactions on Communications*, vol. Com-29, No. 4, pp. 512-424, Apr. 1981.

[HuKl1996] P. Hu and L. Kleinrock, " A Simple Host Deflection Scheme for High-Speed LANs Using Wormhole Routing", *Proceedings Intl' Conference on Network Protocols*, pp. 124-130, Columbus, Ohio, 1996.

[Jwo1991] J-S. Jwo, "Analysis of Interconnection Networks Based on Cayley Graphs related to Permutation Groups", PhD thesis, University of Oklahoma, School of Computer Science, 1991.

[KeKl1979] P. Kermani and L. Kleinrock, " Virtual Cut-Through : A New Computer Communication Switching Technique", *Computer Networks*, vol. 3, pp. 267-286, 1979.

[LaJwDh1993] S. Lakshmivarahan, J. Jwo and S. K. Dhall, " Symmetry in interconnection networks based on Cayley graphs of permutation groups: A survey", *Parallel Computing* 19, pp. 361-407, 1993.

[LaDh1990] S. Lakshmivarahan and S. K. Dhall, Analysis and Design of Parallel Algorithms, McGraw Hill, New York, 1990.

[LiHa1991] D. Lindar and J. Harden, "An Adaptive and Fault Tolerant Wormhole Routing Strategy for k-ary n-cubes", *IEEE Transactions on Computers*, vol. 40, no. 1, Jan. 1991.

[McTsRo1995] P. K. McKinley, Y. J. Tsai, and D. F. Robinson, "Collective communication in Wormhole –Routed Massively Parallel Computers", *IEEE Computer*, vol. 28, no.12, pp. 39-50, Dec. 1995.

[McBo1994] N.R. McKenzie, K.Bolding, et al, "Cranium: An Interface for Message Passing on Adaptive Packet Routing Networks" *In Proceedings 1st Intl. Workshop, PCRCW '94, Seattle, USA, May 1994, Lecture Notes in Computer Science 853*, pp266-280, Springer-Verlag .

[Misi1991] J. Mišic, "Multicomputer Interconnection Network Based on a Star Graph", *Proceedings 24th Hawaii Int'l Conference on System Sciences*, vol. 2, pp. 373-381, 1991.

[NiMc1993] L. Ni and P. Mckinley, " A Survey of Wormhole Routing Techniques in Direct Networks", *IEEE Computer*, vol. 26, pp. 62-76, Feb. 1993.

[ReFeDoSh1997] J. Rexford, W. Feng, J. Dolter, and K. Shin, "PP-MESS-SIM: A Flexible and Extensible Simulator for Evaluating Multicomputer Networks", *IEEE Transactions on parallel and Distributed Systems*, vol. 8, no. 1, pp. 25-39, Jan. 1997.

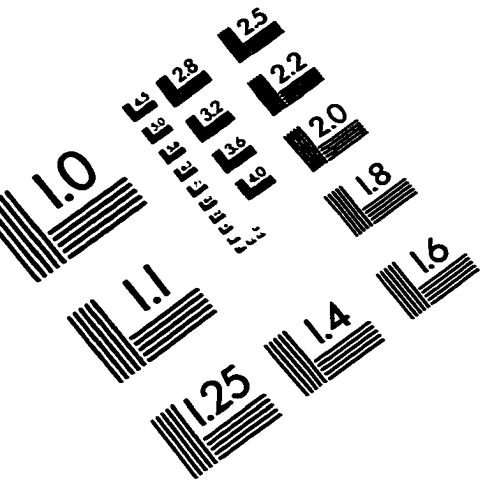
[RoMcCh1995] D. F. Robinson, P. K. Mckinley, and B. H.C. Cheng, "Optimal Multicast Communication in Wormhole-Routed Torus Network", *IEEE Transactions on parallel and Distributed Systems*, vol. 6, no. 10, pp. 1029-1041, Oct. 1995.

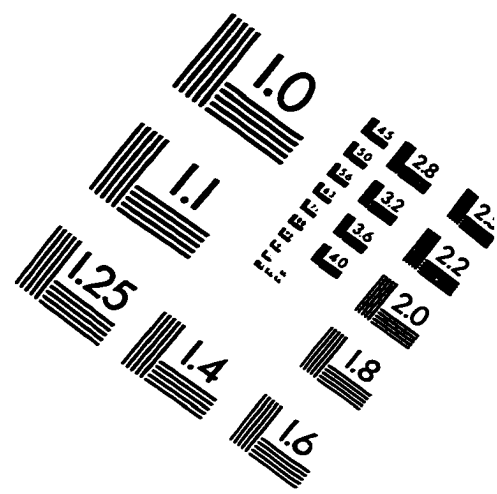
[ScTh1994] S. Scott and G. Thorson, "*Optimized Routing in the Cray T3D*", *In Proceedings 1st Intl. Workshop, PCRCW '94*, Seattle, USA, May 1994, Lecture Notes in Computer Science 853, pp281-294, Springer-Verlag.

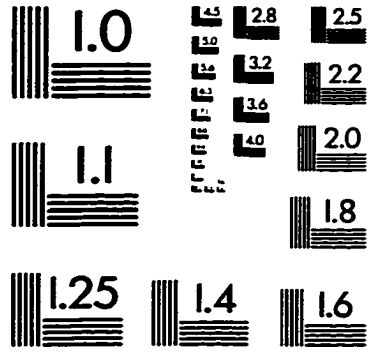
[ShDa1995] K. Shin and S. Daniel, " Analysis and Implementation of Hybrid Switching", *Proceedings of ISCA '95*, pp. 211-219, Santa Margherita Ligure, Italy, 1995.

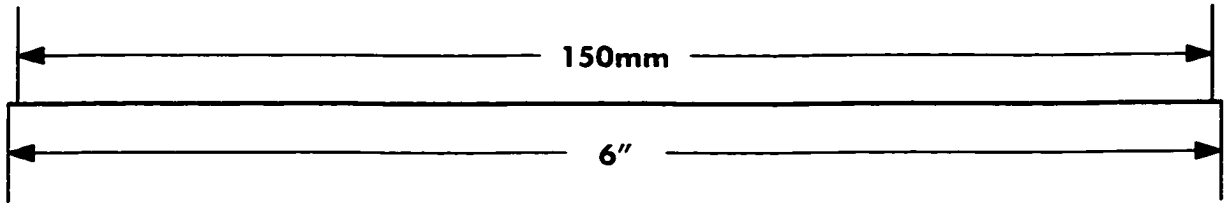
[UpVaMo1997] J. Upadhyay, V. Varavithya, and P. Mohapatra, "A Traffic-Balanced Adaptive Wormhole Routing Scheme for Two-Dimensional Meshes", *IEEE Transactions on Computers*, vol. 46, no. 2, pp. 190-197, Feb. 1997.

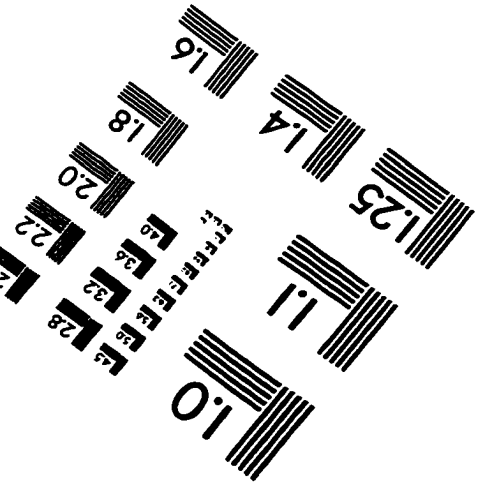
IMAGE EVALUATION
TEST TARGET (QA-3)

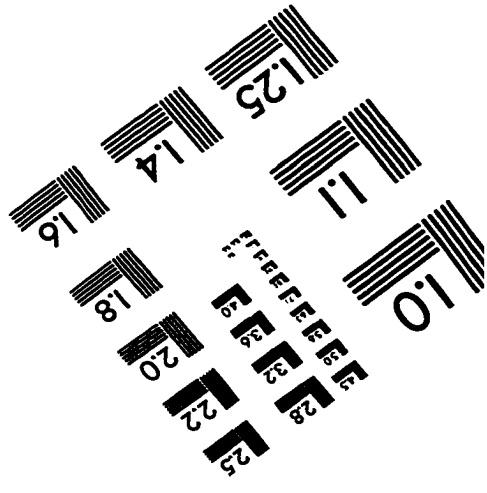












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989