UNIVERSITY OF ALBERTA

Deflection Routing In Two-Connected Mesh Networks

BY 

Yiqun Cai

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science.

DEPARTMENT OF COMPUTING SCIENCE

Edmonton, Alberta
Spring 1994

ISBN 0-612-11169-5

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: **Yiqun Cai**

TITLE OF THESIS: **Deflection Routing In Two-Connected Mesh Networks**

DEGREE: Master of Science

YEAR THIS DEGREE GRANTED: 1994

(Signed) Yiqun Cai

Yiqun Cai

Address:
Department of Computing Science
615 General Services Building
University of Alberta
Edmonton, Alberta
Canada T6G 2H1

Date: Jan 26 1994

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Deflection Routing In Two-Connected Mesh Networks** submitted by **Yiqun Cai** in partial fulfillment of the requirements for the degree of Master of Science.

. . . . . .

Dr. W. Dobosiewicz (Supervisor)

. . . . . .

Dr. P. Gburzynski (Co Supervisor)

. . . . . .

Dr. J. Harms (Examiner)

. . . . . .

Dr. W. Grover (External)

Dr. J. Culberson (Chair)

Date: Jan 26 1994 . . . . . .

To Chengyang

Abstract

Most traditional local area networks are based on linear topologies such as buses and rings. The throughput of these networks is traded for simple access to the networks, and is restricted by the linear topology regardless of the number of users attached. The linear architecture thus is not suitable for high speed local and metropolitan area networks.

The thesis describes another kind of network architectures — regular mesh networks. Mesh networks provide multiple paths between sources and destinations. The maximum throughput of a mesh network increases proportionally to $\sqrt{N}$, with N being the size of the network. The thesis provides an extensive study on two connected mesh networks. Two different deflection routing techniques are presented in the thesis. Synchronization methods are also studied for both slotted and non slotted mesh networks to achieve better routing performance of packets. And finally, three communication protocols using different medium access schemes and deflection strategies are developed and compared by simulations.

Acknowledgements

I would like to thank my supervisor, Dr. Wlodek Dobosiewicz for his time, energy, patience and insight he offered during the development of the thesis. He initiated my ideas of research, and his wisdom and guidance is much appreciated. Without his help, it would be impossible to complete the thesis.

I would also like to thank my co-supervisor, Dr. Pawel Gburzynski for his assistance in building and conducting my simulations.

Special thanks to members of my examining committee, Dr. Wayne Grover and Dr. Janelle Harms for their time and effort in reviewing my thesis and providing me with valuable comments, and Dr. Joe Culberson for chairing the committee.

Finally, I would like to thank my wife Chengyang for her support and encouragement throughout my graduate study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The past decade has seen the tremendous development of the technology of local area networks (LAN). Many of these networks are based on a linear topology — the shared transmission medium to which all nodes (or stations) are connected is in the shape logically equivalent to a bus or a ring[IEE85a, IEE85b, Ame87, IEE90, CO89]. Every node accesses the medium via an inexpensive interface, and is controlled by a simple medium access control (MAC) protocol. Examples include the widely deployed Ethernet network, the Token Ring network, etc.

A parameter that conveniently describes the span of a network is known as "a"[Kle75], defined as the ratio of the network-wide propagation delay to the time required to transmit a packet. In other words, "a" reflects the portion of the link bandwidth that a packet in transit takes. In early LANs such as Ethernet and Token Ring, at any moment, only one packet is allowed to be broadcast across the transmission medium. Therefore, these networks perform satisfactorily only when "a" is less than 1. When the geographical span of a network increases, or the transmission speed of signals (either electronic or optical signals) across the medium increases, the portion of the link bandwidth taken by a packet in transit decreases; as a result, "a" increases to more than 1. We use the term "big-a" to describe a network with "a" significantly larger than 1, say 10. In a big-a network, if there is still one packet allowed

to be broadcast over the network, the throughput[1] of the network drops significantly for there is only a small portion of the link bandwidth utilized.

Some recent high speed network architectures increase total throughput by facilitating multiple packet initiation and transmission in a "big-a" network[RB90]. For instance, FDDI uses a timed token-holding method in which the token is released immediately after transmission of a number of packets, therefore the protocol enables simultaneous coexistence of multiple packets across the network[Ros89, Ame87]. Another example is METARING where the register-insertion technique is employed so that packets can be inserted to the links at several nodes concurrently[C'O89, C'O90]. Using this technique, the throughput of METARING increases impressively[2]. However, these protocols are still *linear* in the sense that the total throughput is constrained regardless of the number of nodes connected to the network. The maximum throughput of FDDI is upper-bounded by the ratio of the total token holding time (ΣTHT) to the target token rotation time (TTRT), and the throughput of METARING is less than 8 in any circumstances.

Overall, for a linear network, the total throughput is constant, and throughput per user (node) decreases linearly as the number of users increases, which is not a desirable feature for a network serving a metropolitan area with many users exchanging information in large amounts.

When the number of nodes is sufficiently large, it is possible to increase the dimension of a network topology. One example is to rearrange nodes in a dual-ring network into a mesh of smaller but connected rings, where each node accesses two rings[Tod92, Max85b]. The rearrangement introduces a natural transmission concurrency — simultaneous transmissions take place in different rings. As a consequence, the total throughput can be increased (proportional to $\sqrt{\mathcal{N}}$ for $\mathcal{N}$ nodes) as the

[1]There are a number of ways of describing network throughput. In the thesis, we define the network throughput as the ratio of payload bits received to the transmission rate (bits per second).

[2]If not counting for the two counter-rotating full duplex rings, METARING is still able to achieve a throughput of 2 regardless of the geographical span of the network.

number of nodes increases. To a certain extent, the rearrangement eliminates the constraints of throughput imposed by linear topologies and protocols.

The thesis is devoted to the study of communication protocols for mesh networks. In a mesh network, every node has a number of identical input links as well as the same number of output links. Compared with a Token Ring network, multiple paths between a pair of source and destination nodes exist in a mesh network. The existence of the multiple paths decreases the average and maximum distance between nodes, reduces the fraction of network bandwidth taken by a packet in transit, and localizes load between frequently communicating nodes, thus increasing the throughput[Tod92, Max85b].

The existence of multiple paths, on the other hand, complicates the communication protocols, because an intermediate node has to match one of a number of output links for every incoming packet to be retransmitted. This function that a protocol performs is defined as the **routing** function in the thesis.

Many routing techniques have been discussed in literature. Among them, **deflection routing** becomes fascinating for its simplicity, efficiency, and so on[DTZ91, DTZ92, Max91, RL91, CL91b, Acs91]. **Deflection routing** works within a node in such a way that an incoming packet is routed to a less desirable output link if another packet was routed to its desirable link. Since incoming packets are granted higher retransmission priorities than locally sourced packets, queues of arriving packets will not accumulate given that there are the same number and capacity of input and output links within a node.

The efficient use of deflection routing depends on the options available to a routing node at the moment a routing decision is made for a packet. More routing options result in a better match between available output links to packets. Part of the routing options available depends on the number of packets and output ports simultaneously available for routing; this is referred to as **synchronization** in the thesis. In most of the previous research, it is assumed that packets arrive at the parallel input ports of

a node at a predetermined moment[Max85b, KY90]. The effect of **synchronization** on different **deflection** techniques has been overlooked and is discussed in the thesis together with the performance studies of protocols that use different medium access schemes and deflection technique.

The initial work towards the completion of the thesis comes from the study of Manhattan Street Network (MSN) proposed by Maxemchuk[Max85a, Max85b]. MSN is the most famous deflection mesh network architecture. The research of MSN by Maxemchuk and other scientists emphasized on the following areas:

- the superior performance exhibited by mesh networks, exemplified by MSN[Max85a, KM93, Max89];
- contention resolution rules[CL91a, CL91b];
- simplification of routing computation[Max87];
- problems with deflection routing[3][Max91];
- broadcasting in MSN[CA90];
- and bi-directional MSN[BC90].

Almost all the previous research on MSN uses the same deflection technique, and assumes a slotted access method. The study of the deflection routing is mostly with respect to the throughput performance, neglecting the fact that every deflection also causes a packet to stay longer in the network.

Our research is based on the previous study on MSN, and covers a much wider spectrum of problems from a more practical view. Besides the widely used distance-based deflection algorithm in MSN, we proposed a new position-based deflection algorithm for mesh networks, and compared their performances by simulation results. We also extend slotted protocols to non-slotted mesh networks. The performance of

[3]particularly *Random Deflection* as we define later in the thesis.

deflection routing is studied from both throughput and delay perspectives. Besides, in the thesis, we indicate that proper synchronization is critical to the performance of deflection routing, and we discuss extensively synchronization techniques.

The discussion in the thesis focuses on mesh networks that use 2×2 switches, i.e. there are two input links and two output links per switch. However, the synchronization techniques and the deflection strategies discussed can be extended to mesh networks with higher dimensions, where proper synchronization is even more important.

Chapter 2

Overview

Among many high speed network architectures that have been proposed for the next generation of LAN and MAN, deflection networks have received great attention because they offer multiple paths between sources and destinations, thus increasing the entire network throughput and improving reliability. The Manhattan Street Network (MSN) which was first proposed by Maxemchuk in 1985 is an example of a deflection network[Max85a, Max85b]. In the thesis, we denote deflection networks with node arrangement based on or derived from a regular mesh topology as **Deflection Mesh Networks** DMN. Before we study their performance, we briefly present in this chapter an overview of DMN. The architecture of DMN is described followed by a discussion of different medium access schemes and routing algorithms. Simulation environments and performance measures are summarized at last.

2.1 Architecture

A DMN is made up of a collection of nodes interconnected by point-to-point links[1]. The structure and implementation of nodes and links is briefly presented in this sec-

[1]The use of point-to-point links meets the current development stage of optical transmission using fiber as medium[Max85a].

tion. Detailed discussion is beyond the scope of the thesis. The topology parameters of a DMN is then studied with emphasis on the topological impact on the network throughput.

2.1.1 Node

A node in a DMN basically consists of a switch. The switch is responsible for accessing the links passing through the node. A node may also contain local components such as a work-station or a LAN that access the network via the switch. As shown in figure 2.1, besides the network and local interfaces, a switch has a switching device and some storage units.

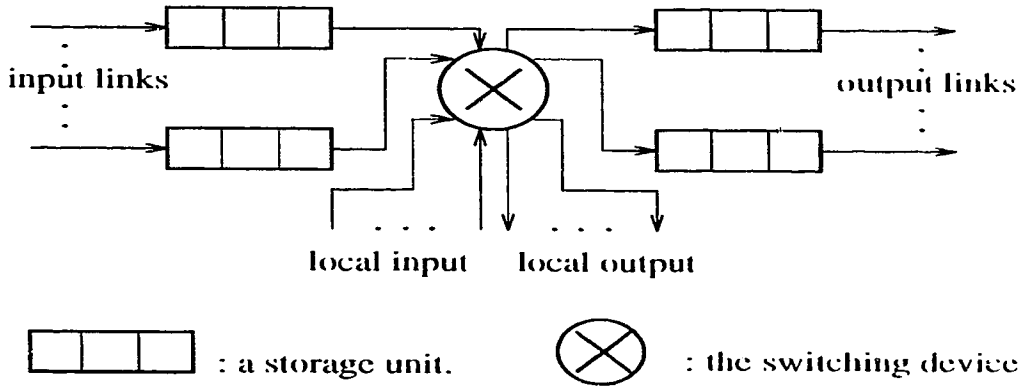


Figure 2.1: A conceptual switch structure.

Switch

A switch consists of network (and optionally local) interfaces and switching devices that actually perform the operation of routing packets. The network interfaces include a number of network input ports and the same number of network output ports. This number is denoted by k_n. A switch also has a number of local input ports, and the same number of local output ports. This number is denoted by k_l.

For simplicity, it is assumed in the thesis that each network input port of a switch is connected to exactly one input link, and each network output port is connected to one output link. k_n is defined as the *degree of connectivity* of a network. We investigate DMN that use 2×2 switches, i.e. for every switch, there are two network input ports and two output ports respectively. The resulting network is said to be *two-connected.*

The local input/output ports of a switch can be connected to a work-station, a LAN or some other devices (e.g. file servers, printers, etc) that generate or absorb traffic. If k_l is 0, the switch is not connected to any local stations or networks. Therefore the switch merely performs routing functions.

With the current level of technology, the processing of a switch is still in the electronic domain[JM93, CF93]. This is because:

1. it remains difficult to put complex logic into purely optical switches;

2. and this logic is needed to make decision related to routing.

Since the bandwidth of electronic processing and switching is relatively limited compared to that of optical transmission, an implementation of Electro-Optic switches was proposed in [CF93]. It uses optical delay lines to store and switch packets, eliminating the necessity of O/E and E/O conversion of conventional fiber-based networks, and thus improving the speed gap between electronic processing and optical transmission.

Compared to the ISO reference model, a switch performs the functions overlapping Data Link Layer and Network layer (3a)[Hal92]. It controls the access to the links as well as provides the routing decisions.

Storage Unit

Associated with each network input or output port of a switch is a storage unit. The storage unit is implemented as an electronic buffer or as a segmented optical

delay line. Storage units associated with input ports are necessary because they are used to delay incoming packets for extracting routing information and to synchronize incoming packets for routing. Storage units associated with output ports are optional. This storage is provided mainly for improving the routing quality, by decreasing the probability of a packet being deflected to the undesirable output link.

The use of electronic buffers enables random access of the electronic signals in the buffers. But it introduces severe overhead in a fiber-based network because of the O/E and E/O conversion. To improve the speed gap in a fiber-based network, the storage units represented in figure 2.1 are sometimes implemented as segments of short optical delay lines as discussed in [Max88, CF93]. The optical signals can be sensed at the end of each segment of delay lines. The details of the implementation are beyond the scope of the thesis. In the thesis, we use the term *buffer* to stand for implementation of either an electronic buffer or an segmented optical delay line.

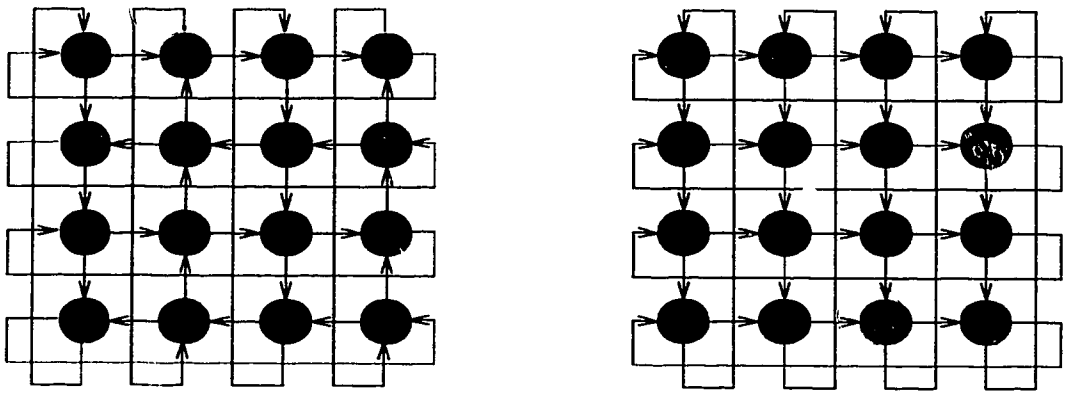
2.1.2 Links

A *link* represents a point-to-point channel that connects two adjacent nodes. A uni directional link only allows the transmission to take place from one end (a node) to the other at any moment, while a bidirectional link allows two-way transmission but at any moment, only one direction of transmission is allowed.

An assumption (*Link Assumption*) made in the thesis is that all the point-to-point links have the same transmission capacity, and they all have the same length measured in bits; except that the directions of the links may be different, all the links are identical.

Throughout the thesis, we are only interested in networks with "a" significantly greater than 1. Besides, we are only concerned with the situation where the propagation delay between two adjacent nodes is more significant than the buffering delay within a switch for a packet, and we count *hops* to study the delay of a packet in a network. Therefore, the actual length of a link is not of interest of study.

2.1.3 Regular Two-connected DMN



a. *b.*

Figure 2.2: Two examples of regular 4×4 DMN

In the thesis, we are interested in a regular two-connected DMN formed by a number of overlapping rings – visually, a collection of *horizontal* (or *row*) rings and a collection of *vertical* (or *column*) rings with every row ring and every column ring intersecting at a node as shown in figure 2.2 [2].

A ring (either a row ring or a column ring) is said to be *unidirectional* if it is made up of nodes connected via unidirectional links. Otherwise, it is said to be *bidirectional* if every link is a bidirectional one.

According to the direction of rings in a DMN, a DMN can be further categorized as:

1. *Bidirectional* DMN (BIDMN): all the rings – row rings and column rings are bidirectional.

[2] Although, from the point of view of topology, a DMN can be viewed as a different collection of rings each of which contains part of a row ring and part of a column ring as described above, categorizing rings into *row* and *column* two classes simplifies the discussion.

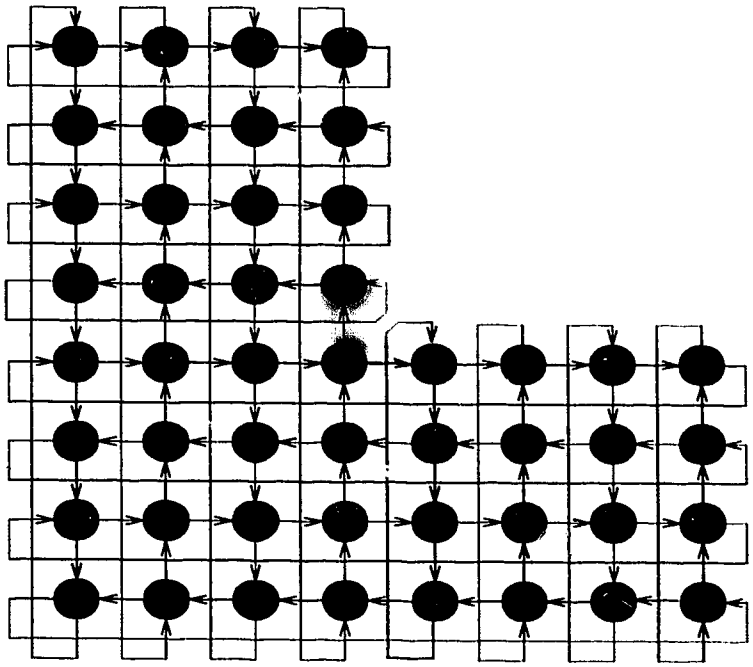


Figure 2.3: 48 Nodes

2. *Counter-Rotating* DMN (CTRDMN): all the adjacent row rings operate in opposite directions, and so do all the adjacent column rings.

3. *Co-Rotating* DMN (CODMN): all the row rings operate in the same direction, and all the column rings operate in the same direction.

4. *Arbitrary-Rotating* DMN (ARBDMN): there is no restriction on the direction of rings in a DMN. Each individual ring can be unidirectional or bidirectional. Every unidirectional ring can operate either clockwise or counter-clockwise.

Figure 2.2 presents two 4×4 DMN of unidirectional rings. Figure 2.2.a is a CTRDMN, and figure 2.2.b is a CODMN.

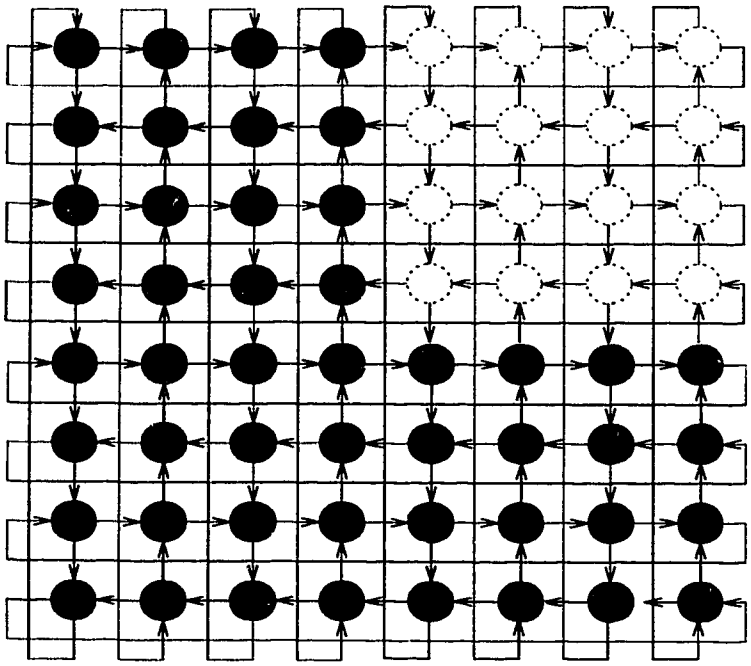


Figure 2.4: 48 Nodes + 16 Dummy Nodes

Two derivations from 8×8 CTRDMN

Besides DMN of regular topologies such as squares and rectangles, we also study two derivations from an 8×8 CTRDMN. In figure 2.3, there are 48 nodes arranged in an $\mathcal{L}$ shape. The upper right 16 nodes are disconnected from an 8×8 CTRDMN. This arrangement is denoted as $\mathcal{L}48$. In figure 2.4, 16 more nodes are added to $\mathcal{L}48$ to form a regular 8×8 CTRDMN. The difference is in $\mathcal{L}48$, all nodes generate and absorb traffic, but in $\mathcal{L}64$, the additional (visually the upper right) 16 nodes do not generate or absorb traffic. The 16 *dummy* nodes in $\mathcal{L}64$ only perform routing functions. Under uniform traffic assumption, two derivations study the performance of changing an asymmetrical topology into a symmetrical one but changing the distribution of traffic.

2.1.4 Topological Parameters

In this section, we present some topological parameters of DMN. These parameters, for example, average shortest distance and traffic matrix, help to describe the protocols and analyze their performance.

Size of a Network

The size of a DMN is denoted by $\mathcal{N}$. It is the total number of nodes in the DMN. We index a node in one of the following two ways:

1. A node is identified by the pair of row ring and column ring indices (α, β) from left to right, and from top to bottom, where $\alpha \geq 0$ and $\beta \geq 0$;
   or,
2. A node is identified by an integer i, $0 \leq i \leq \mathcal{N} - 1$.

Assuming γ to be the number of column rings, i is defined as:

$$i = \alpha * \gamma + \beta.$$

Distance Between Nodes

Assume every node is identified by an integer i as described above. Let $i \longrightarrow j$ denote that there is a link connecting node i and j, and the direction of the link is from i to j. A *path* from i to j, where $i \neq j$, denoted as $i \xrightarrow{+} j$, is defined as:

$$i \xrightarrow{+} j : i \longrightarrow j \cup \exists_k (i \xrightarrow{+} k \cap k \longrightarrow j).$$

Based on the *Link Assumption*, the length of a path (denoted as $\|i \xrightarrow{+} j\|$) is the number of links on the path. This number is called *hops*.

Since in a DMN, there exist multiple paths between two nodes. We define the distance between two nodes i and j: π_{ij} to be the shortest path from i to j, that is:

$$\pi_{ij} = min\{\|i \xrightarrow{+} j\|\}.$$

In a BIDMN, $\pi_{i,j}$ is the same as $\pi_{j,i}$ for all i and j, while in a unidirectional DMN, it is not always the case.

Diameter of a Network

The diameter ($\mathcal{D}$) of a DMN is the maximum value of $\pi_{i,j}$ for all i and j.

Traffic Matrix

We define $w_{i,j}$ as the weight of $\pi_{i,j}$ which is a real number satisfying:

$$\sum_{i=0}^{\mathcal{N}-1}\sum_{j=0}^{\mathcal{N}-1} w_{i,j} = 1 \tag{2.1}$$

where $0 \leq w_{i,j} \leq 1$, and $w_{i,j} = 0$ if $i = j$. $w_{i,j}$ specifies the frequency of transmission from i to j. $W = \{w_{i,j}\}$ is called the *traffic matrix*. Under uniform traffic, $w_{i,j} = \frac{1}{(\mathcal{N}-1)\times\mathcal{N}}$ for $i \neq j$, and the uniform traffic matrix is denoted by W_u.

Shortest Average Distance

The average shortest distance($\mathcal{ASD}_W$) of a DMN with respect to a traffic matrix $W = \{w_{i,j}\}$ is defined as:

$$\mathcal{ASD}_W = \sum_{i=0}^{\mathcal{N}-1}\sum_{j=0}^{\mathcal{N}-1} \pi_{i,j} \times w_{i,j} \tag{2.2}$$

It is difficult to derive a closed-form solution for $\mathcal{ASD}$ for an arbitrary DMN. However, some closed-form solutions have been provided for CTRDMN, CODMN and BIDMN for some particular node arrangements under uniform traffic assumption.

- For rectangular CTRDMN,[3] $\mathcal{N} = r \times s$, where $r, s \geq 2$,

$$\mathcal{ASD}_{W_u} = \begin{cases} \frac{N}{4}(r+s+4)-s-4 & \text{if } mod(s,2) = 1 \text{ and } mod(r,2) = 0 \\ \frac{N}{4}(r+s+4)-4 & \text{if } mod(s,2) = 0 \text{ and } mod(r,2) = 0 \\ \frac{N}{4}(r+s+4)-r-s-2 & \text{if } mod(s,2) = 1 \text{ and } mod(r,2) = 1 \end{cases} \tag{2.3}$$

[3]Since CTRDMN, CODMN and BIDMN are symmetrical from the topological point of view, r and s can stand for the number of row rings and column rings interchangeably.

The detailed derivation can be found in [CA90].

- For rectangular CoDMN, $\mathcal{N} = r \times s$,

$$\mathcal{ASD}_{W_u} = \frac{1}{\mathcal{N}^2}\sum_{i=0}^{r-1}\sum_{j=0}^{s-1}(i+j) = \frac{\mathcal{N}}{2(\mathcal{N}-1)}(r+s-2). \tag{2.4}$$

- For square BiDMN, $\mathcal{N} = r^2$,

$$\mathcal{ASD}_{W_u} = \begin{cases} \frac{r^3}{2(\mathcal{N}-1)} & \text{if } mod(r,2) = 0 \\ \frac{r}{2} & \text{if } mod(r,2) = 1 \text{ and } r \geq 3 \end{cases} \tag{2.5}$$

The derivation can be found in[BC90].

Table 2.1 is a comparison of $\mathcal{ASD}_{W_u}$ of CtrDmn, CoDmn and BiDmn with the exception of $\mathcal{L}64$ where only 48 nodes contribute traffic — a non-uniform traffic distribution. From the table, it is seen that BiDmn configurations have the smallest $\mathcal{ASD}_{W_u}$. The average shortest distance of CtrDmn is nearly 1 more than BiDmn but significantly smaller than that in the CoDmn for the same node arrangement.

2.1.5 Topological Effect on Throughput

We define the **network throughput** ($\mathcal{T}$) as the number of payload bits received per bit time over the network as a whole. Assume all the links have the same transmission capability, say M_l *bits per second* (*bps*) for each link, and the rate of the payload bits being received network-wide is M_b *bps*, the achieved network throughput is thus:

$$\mathcal{T} = \frac{M_b}{M_l}.$$

Based on the *Link Assumption*, since Dmn is two-connected, the total number of links is $2 \times \mathcal{N}$. The upper-bound of the network throughput is[DTZ91]:

$$\mathcal{T}_{max} = \frac{2 \times \mathcal{N}}{\mathcal{ASD}_W} \tag{2.6}$$

$\mathcal{ASD}_W$ is the average shortest distance under traffic matrix W.

	CTRDMN	BIDMN	CODMN
4x4	2.93	2.13	3.20
4x6	3.30	2.61	4.17
4x8	4.00	3.10	5.16
6x6	3.71	3.09	5.14
8x8	5.02	4.06	7.11
12x12	7.02	6.04	11.08
16x16	9.02	8.03	15.06
20x20	11.02	10.03	19.05
4x9	4.18	3.31	5.66
$\mathcal{L}61$	4.89	3.91	7.04
$\mathcal{L}48$	4.75	3.63	5.95

Table 2.1: Comparison of shortest average distances for different networks.

We find that from the expressions 2.3, 2.4 and 2.5, when N is sufficiently large, for square node arrangements under uniform traffic distribution, $\mathcal{ASD}_{W_u}$ of CTRDMN and BIDMN are approximately $\frac{\sqrt{N}}{2}$ and $\mathcal{ASD}_{W_u}$ of CODMN is approximately $\sqrt{N}$. From equation 2.6, it is found that under uniform traffic: (1) for the three DMN, the maximum throughput $\mathcal{T}_{max}$ increases proportionally to $\sqrt{N}$, and (2) CTRDMN and BIDMN yield close $\mathcal{T}_{max}$ which doubles that of CODMN for the same node arrangement.

If not considering the overhead introduced by transmitting the header of a packet, the maximum throughput can be achieved by using an infinitely large buffer in every intermediate switch so that every packet transmitted traverses along the route achieving shortest distance to its destination.

The equation 2.6 indicates that under the same network conditions – same size of the network and same traffic distribution, node arrangements that achieve smaller average shortest distance yield larger throughput. On the other hand, it also reveals that for any given node arrangement, traffic distributions achieving a smaller average shortest distance yield larger throughput too. Examples of such traffic distributions include the **communities-of-interest** traffic that favors communication between nodes that are relatively near. For instance, for a 6×6 CODMN, we define the following traffic matrix that favors local row-to-row and local column-to-column traffic:

$$W_{cof} = \begin{cases} w_{(i,j)} = \frac{14}{35\times5\times36} & \text{if } mod(i,6) = mod(j,6) \text{ or there exists an integer} \\ & k, 0 \le k \le 5 \text{ such that } k \times 6 \le i, j < (k+1) \times 6 \\ w_{(i,j)} = \frac{7}{35\times25\times36} & \text{Otherwise} \end{cases} \tag{2.7}$$

W_{cof} is a traffic distribution that for all messages[4] a node generates, 80% of the destination nodes are distributed uniformly among the nodes attached to the row or column ring to which the source node is connected. The rest of the destinations are uniformly distributed among nodes which are connected to different row and column

[4]We assume that none of the messages is a multi-cast or broadcast message.

rings. The corresponding average shortest path $\mathcal{ASD}_{W_{cf}}$ is 3.6. The maximum throughput achievable is thus to be 20. As a comparison, under uniform traffic, $\mathcal{ASD}_{W_u}$ of the same 6×6 CoDMN is 5.14, with the maximum achievable throughput of 14. The performance enhancement by introducing **communities-of-interest** is presented in more detail in chapter 5 by simulation results.

2.2 Medium Access Control

For a DMN, there are two different ways of accessing medium studied: slotted, and non-slotted with equal-size packets which are called *cells*. Therefore, there are two types of resulting networks – slotted DMN and non-slotted DMN. In the following discussion, we assume *destination removal* of full slots or cells, which means:

- for a full slot, the payload of the slot is removed by the destination of the slot so that the slot is regarded *empty* after its payload has been received;
- a cell is removed from the network by its destination.

2.2.1 Slotted DMN

In a slotted mesh network, there are a number of non-overlapping slots circulating in each ring. The end of one slot is immediately followed by the start of another slot so that there is no gap between a slot and its successor. The structure of a slot includes two parts: the *header* part, and the *payload* part. The *header* part contains controlling and routing informations, while the *payload* part carries actual data to be transmitted across the network.

One piece of control information contained in the *header* part of a slot is the *payload status* (PS) flag which is used to indicate whether the slot carries a payload or not. The flag is normally represented by a single bit with 1 standing for full and 0 for empty.

During the network initialization procedure, every slot transmitted has its PS flag set to 0. A switch which has some bits to transmit awaits an empty slot. Then, if the switch is permitted to fill the slot according to some rules, it sets the PS flag of that slot to 1 and fills the payload into the *payload* part of the slot. Otherwise, the switch just retransmits the slot. When the payload of a full slot is removed from the slot carrying it by a switch, the corresponding PS flag is reset to 0 so that the slot can be reused.

2.2.2 Non-slotted DMN

A non-slotted DMN of interest in the thesis assumes that all the packets transmitted in the network have the same length. They are called **cells**. As opposed to slots, the interval between two consecutive cells is not fixed, and structurally, the header of a cell does not include a PS flag.

Every outstanding cell in the network is delayed for a few bits for the purpose of routing when arriving at an intermediate switch. If a cell is addressed to the node containing the switch, the cell is removed from the network. Otherwise, the cell is retransmitted via one of the network output ports according to some routing rules.

When the network input ports of a switch sense a period of silence on the transmission links, or the switch removes cells from the network, the switch can insert new cells from its local source to the output links.

The insertion and retransmission of a cell is controlled by MAC-protocols. Sometimes, for the purpose of achieving a better routing, there will be a delay between the time a switch is allowed to insert or retransmit a cell and the time the operation actually commences. The delay operation involved is called *synchronization*, discussed in more detail in the next chapter.

2.3 Deflection Routing

In a two-connected DMN that uses 2×2 switches as shown in figure 2.1, a switch performs the routing function whenever there is at least one cell/slot seen at the network input ports as well as the local input ports. Routing priority is given to cells/slots coming from the network.

There are many factors considered for making the routing decision for a cell/slot. Such factors include the remaining distance to the destination, the previous routing especially deflection history, the actual time a cell/slot has spent in the network, etc. Based on the routing rules taking different factors into account, the two network output ports may have different preference for cells/slots to be routed. It is desirable that a cell/slot be routed at every intermediate node to its preferred port.

A cell/slot is not always routed to its preferred port, in the presence of one of the following conditions. The two conditions are defined as *Conflict Conditions*.

1. There is a cell/slot being transmitted at the preferred port of the cell/slot to be routed, and there is no buffer associated with the port.

2. There is a buffer associated with the preferred port, but it either

   - has reached its full capacity and can not accommodate any more cell/slots,
   - or the buffer can only store one cell/slot, but the cells/slot loses the competition for the preferred port with another cell/slot. Therefore, the *loser* can not be queued for its preferred port.

If one of the above happens, the cell/slot to be routed is forced to take the other port of the switch, which causes a full slot or a cell to traverse a longer route to its destination. This operation is defined as **deflection**.

There are many deflection approaches proposed and studied. In the thesis, we study two of them. One is a distance based deflection, named *Random Deflection*. It has been widely discussed[Max85b, BC90, CL91a, Max89, KM93]. The other is

a position (direction) based deflection, called *Vertical Deflection*, which is a new deflection technique proposed.

2.3.1 Random Deflection

Random Deflection is a distance based deflection. The deflection rules try to minimize the remaining distance of cells/slots to be routed.

There are two situations when a cell/slot is being routed: (1): both network output ports lead to the same distance to the destination, or (2): one port leads to a shorter one. The *preferred* port of the cell/slot is the one leading to a strictly shorter remaining distance to the destination. If there is no difference between two ports, the cell/slot doesn't have a preferred port, i.e. both ports qualify equally. For example, in a slotted network, an empty slot does not have a preferred port and can be routed to either of the ports.

In the case of routing decided for only one cell/slot at a time, the cell/slot will be routed to its preferred network output port unless one of the *Conflict Conditions* is true or it does not have a preferred port. For the first case, the cell/slot is deflected to another output port, while for the latter case, it is routed to one of the two ports with an even probability.

If a routing decision is made for two cells/slots at the same moment, exactly one of the following events takes place:

- the two cells/slots are routed to their own preferred ports, if they have disjoint preferred ports;
- exactly one cell/slot has a preferred port and is routed to that port, the other cell/slot is routed to the un-preferred port;
- the two cell/slots are routed to different ports with same probability if neither of them has a preferred port;

- in the case that two cells/slots have the same preferred port, and there is not enough spare storage capacity to buffer both of them, one of them (the *winner*) is selected at **random** for its preferred port, the other (the *loser*) is **deflected** to the un-preferred port.

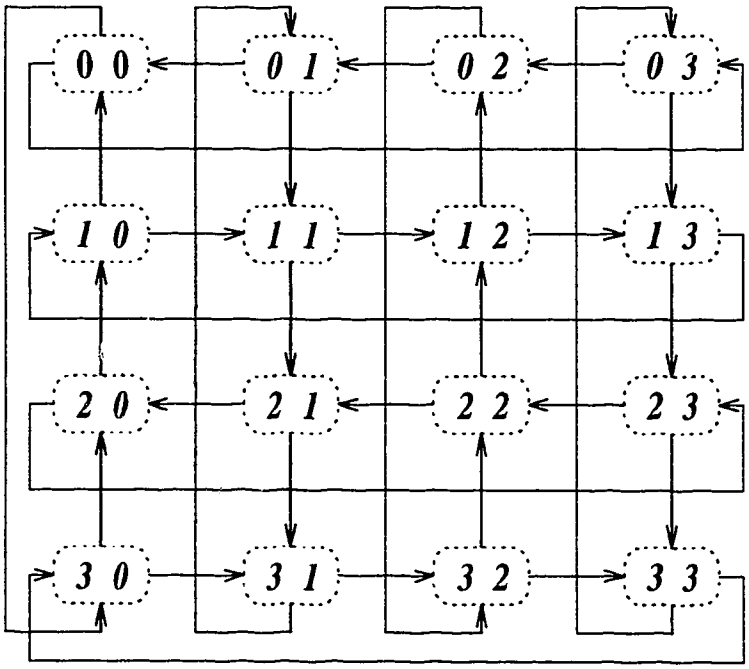


Figure 2.5: A 4×4 CTRDMN.

As described above, the deflection can take place at every direction: from row to column or from column to row. Consider the 4×4 CTRDMN presented in figure 2.5, the pair of numbers associated with each box stand for the indices of the node. A slot sourced at $\boxed{(0,0)}$ and destined at $\boxed{3,1}$ can take one of the following route:

1. $\boxed{(0,0)} \longrightarrow \boxed{(3,0)} \longrightarrow \boxed{(3,1)}$, which is the shortest route;

Destination Station	Shortest Distance	
	Through Row Port	Through Column Port
(0,1)	3	3
(0,2)	2	6
...	...	...

Table 2.2: A part of a routing look-up table in node (0,0).

2. (0,0) ⟶ (3,0) ⟶ (2,0) ⟶ (2,3) ⟶ (2,2) ⟶ (2,1) ⟶ (3,1), in which route, the slot is deflected at (3,0) so that it is forced to travel a longer route;

3. (0,0) ⟶ (0,3) ⟶ (1,3) ⟶ (1,0) ⟶ (0,0)..., which is the route that the slot is deflected at every node, and will never reach its destination;

4. ...

Routing uses a complicated selection mechanism to achieve symmetry and to avoid problems as deadlock, livelock, etc[Max91]. The major weakness of the deflection is that the route of a cell/slot is completely unpredictable.

Implementation

In order to implement the routing strategy, each switch has a look-up table (such as table 2.2) which shows the distances to other nodes in the network using each output port. Part of the look-up table in node (0,0) in Figure 2.5 is presented in table 2.2.

The advantage of the look-up table is easy to see: it is simple and affordable when the number of switches is moderate. It also provides accurate distance information for routing. But, it is very expensive to maintain and update the look-up table especially when there are frequent changes to the network.

In [Max87], Maxemchuk compared some distributed distance-based routing strategies. The idea was to first divide the whole network into four quadrants according to the relative address of a source to the destination nodes. Then the routing of a packet is decided according to the quadrant contains the transmitting switch. Routing does not use any global look-up table but requires a fair amount of computation to calculate the relative addresses and to divide the quadrants.

2.3.2 Vertical Deflection

Different from *Random Deflection*, *Vertical Deflection* is a position-based deflection. For a certain packet, deflections happen only at certain intermediate nodes, and directions of deflections are predetermined, say either from row to column or from column to row but not both. In the thesis, we assume "column-to-row" is used, i.e. every column-to-row retransmission causes a longer distance.

Taking the switch at node (α, β) in a slotted network as an example:

1. switch (α, β) sees a full slot at the row input port, and the destination of the slot has the same column index as the node has, i.e. β.

2. (α, β) removes the slot from the row ring and transmits it into the column ring.

3. when the next slot arrives from the column input port of (α, β), the switch transmits it via the row output port, if there is no more buffer capacity associated with the column port.

The third step in the above procedure causes a full slot or a cell coming from the column ring to be detoured to the row ring that leads to a longer route to its destination. It introduces a *column-to-row* **deflection**. With this approach, a slot propagates along its present row ring until it reaches its destination column, and a deflection is only triggered by a slot/cell coming from a row ring to its destination column ring. Routing ensures every full slot or cell to reach its destination along the

shortest column distance if the slot/cell is transmitted properly at the source node. For the same slot in figure 2.5, the routes it may choose to reach its destination $\boxed{(3,1)}$ is exactly one of the following four routes:

1. $\boxed{(0,0)} \longrightarrow \boxed{(0,3)} \longrightarrow \boxed{(0,2)} \longrightarrow \boxed{(0,1)} \longrightarrow \boxed{(1,1)} \longrightarrow \boxed{(2,1)} \longrightarrow \boxed{(3,1)}$, which is the shortest route. Compared to the shortest route using *Random Deflection*, it is much longer (4 more hops).

2. $\boxed{(0,0)} \longrightarrow ... \boxed{(0,1)} \longrightarrow \boxed{(1,1)} \longrightarrow \boxed{(1,2)} \longrightarrow ... \boxed{(1,1)} \longrightarrow \boxed{(2,1)} \longrightarrow \boxed{(3,1)}$ — in which route, the slot is deflected at $\boxed{(1,1)}$; the length of the route is 10.

3. a route that is similar to the second route, but instead of being deflected at $\boxed{(1,1)}$, the slot is deflected at $\boxed{(2,1)}$; the length of the route is also 10.

4. a route that the slot is deflected at both $\boxed{(1,1)}$ and $\boxed{(2,1)}$; the length of the route is 14.

To implement the routing, every switch should have the knowledge of the address of the nodes that are attached to the same column ring. This information is easier to maintain and update compared to the look-up table used in *Random Deflection*. It is also not necessary to perform any selection procedure at any switch. The deflection achieves a bounded end-to-end delay: assume there are i row rings, a full slot or a cell will traverse the length of at most $i-1$ row rings and one column ring (more exactly, less than $i-1$ hops in the column ring)[5]. However, drawbacks related to this deflection include:

1. average longer route to the destinations;

2. asymmetry existing among different traffic patterns, which reduces throughput.

[5]For square topologies, the maximum number of hops a slot will traverse is $\mathcal{N}-2$.

2.3.3 Conclusion

The two deflection routing strategies, either distance-based deflection or position-based deflection, have many advantages. By using deflection, some packets are forced to take longer routes to their destinations. When the load is light, deflection routing localizes the traffic between two frequently communicating nodes; when the load is heavy, it prevents buffers at nodes from overflowing so as to provide an unconventional mechanism of congestion control. In many cases, deflection routing outperforms store-and-forward routing[Max89].

2.4 Simulated Environment

Due to the difficulty of developing an analytical model, we use simulation as our principal technique to study the performance of mesh network protocols. The simulation package we use is SMURPH[Gbu93].

In SMURPH, a network is modeled by interconnected *stations*. Each *station* has a number of active *processes* that perform the function of receiving and transmitting *packets*. The protocols are specified as a collection of *finite state machines* carried out by *processes* in *stations*.

The following assumptions are made when conducting the simulations:

1. None of the messages generated is a broadcast or multi-cast one.

2. Payload bits are removed from the network by destination nodes.

3. A slot/cell is deflected to the longer route if it fails the competition for its preferred port or its preferred port is occupied.

4. A slot/cell contains a payload part of 400 bits and a header of 40 bits. Every slot/cell is delayed at least 40 bits at a switch.

5. The length of an individual ring is 1400 bits to make "a" large enough (above 10). In fact, all the protocols investigated have a common feature that their performance remains the same as long as the combined propagation delay (the ring latency delay and the sum of buffering delay at each switch) is longer than one slot.

Each simulation is repeated several times, normally from a shorter duration to a longer one to find out the steady measurement. When the measurement is acquired, it is confirmed by another simulation with a similar length[6].

2.5 Common Performance Measures

The following measures are considered in the thesis.

Network throughput

Network throughput ($\mathcal{T}$) is previously defined. The normalized throughput ($\mathcal{T}$) is defined as the percentage of the network throughput achieved ($\mathcal{T}$) to the maximum theoretical network throughput ($\mathcal{T}_{max}$):

$$\mathcal{T} = \frac{\mathcal{T}}{\mathcal{T}_{max}}. \tag{2.8}$$

Message Access Delay

It is the time measured in bits that elapses from the moment a message is queued to the moment the message is transmitted, excluding the transmission time.

Deflection Probability

This is the probability that a full slot or a cell will have to take a longer path to its destination. It is defined differently for each protocol as discussed in following

[6]The methods of conducting simulations are from [Jai91].

chapters.

Additional Hops

It is defined as the difference between the actual hops a slot or a cell traverses from its source to the destination and the distance between the same two nodes. To compare networks of different sizes, we use the measure of normalized additional hops, calculated as following: assume the traffic matrix is W, and there are totally $\mathcal{M}$ cells or full slots received. Let $\mathcal{AH}_k$ be the number of additional hops a cell or a full slot k traverses, the normalized additional hops ($\overline{\mathcal{AH}}$) is given by:

$$\overline{\mathcal{AH}} = \frac{\frac{1}{\mathcal{M}} \sum_{k=1}^{\mathcal{M}} \mathcal{AH}_k}{\mathcal{ASD}_w}$$

Chapter 3

Synchronization

In a two-connected DMN, every switch accesses a row ring and a column ring. The operation of the two rings are independent from each other, which results in the following two phenomena:

- cells/slots from row and column rings may arrive at the two network input ports of a switch at different moments;
- the length(boundaries) of the cells/slots from one ring may be different from those of the cells/slots from the other ring from the point of view of the switch receiving them.

In order to study the effect of the independent operation of the rings and ways to deal with it, we define the *Synchronous Condition* as: all switches use the same transmission rate.

In a *synchronous* DMN, all the cells/slots have the same length, which has a two-folded meaning: (1) the number of bits making up of a cell/slot is the same for all cells/slots; and (2) assuming a virtual global reference clock, the time required to receive a whole cell/slot is the same for every switch with respect to the global reference clock. At the receiving end of a link, once a switch sees the start of a cell/slot, it can predict the time when the end of the cell/slot is detected.

This chapter studies synchronization techniques of DMN. We first compare synchronizing techniques in a slotted network where all switches use the same transmission rate. Then, we study a method using excess bits to synchronize *slot rates* in an asynchronous slotted network. We also present techniques to synchronize retransmission of cells in a non-slotted networks.

3.1 Synchronous Slotted Networks

In a synchronous slotted DMN where all switches use the same transmission rate. Slots are continuously transmitted via the output ports of each switch. Although in ideal situations, row and column slots arrive at the same moment at the network interfaces of a switch without buffering, it is more likely in practice that the arrival times of row and column slots are different. Synchronization deals with the routing and retransmission of asynchronously incoming slots. Two different ways of synchronization are compared. They differ in whether a routing decision is made for two slots at the same moment or one slot at a time.

3.1.1 Aligning

This approach aims at aligning slots from row and column rings so that routing decisions can be made for two slots — one from each ring. It is done by the following during the initialization operation of a network:

1. When the network starts to operate, every switch continuously transmits empty slots via its two output ports. The transmission of the beginning of two slots takes place at the same moment as shown in figure 3.1.

2. When a switch detects that there is a slot (before the initialization procedures end, all slots are empty) arriving the switch from, say row ring, it buffers the incoming slot. The switch then retransmits the incoming slot via the row output

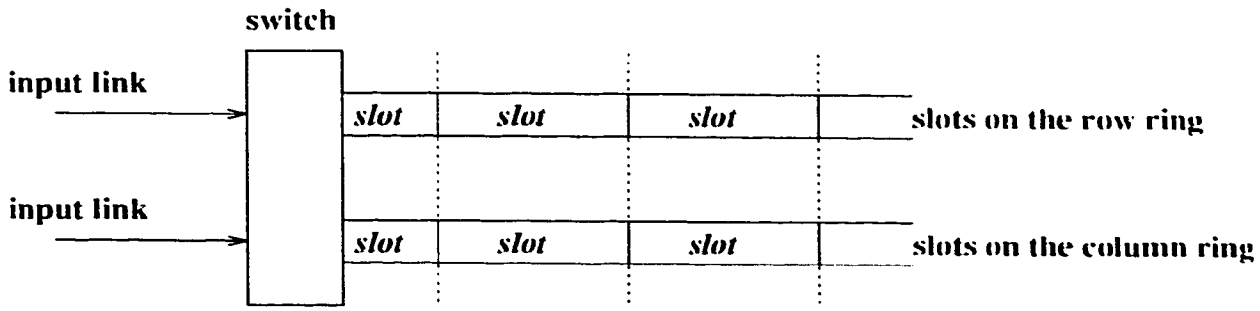


Figure 3.1: Transmissions of slots during initialization.

port immediately after the transmission of the current slot is completed at the same output port. The switch performs the same operation after the arrival of a slot from the column ring.

3. Once a switch starts to relay slots from an input port of the network, it does not insert any new slots into the network via the corresponding output port.

After the network enters regular operation, a slot from the row ring may arrive at a switch at a different moment from a slot from the column ring. But, after buffered for different duration, these two slots can be routed and retransmitted at the same moment as shown in figure 3.2.

Once a switch stops inserting new empty slots into the links, the duration of the switch buffers an incoming slot from one ring is the same for all slots arriving at the switch from the same ring. Assume $\mathcal{L}_r$ stands for the length of a ring measured in bits, $\tilde{\mathcal{N}}$ is the number of switches attached to that ring, $\mathcal{B}_j$ is the buffering time at switch $j, 1 \leq j \leq \tilde{\mathcal{N}}$ and $\mathcal{L}_s$ stands for the length of a slot in bits, the following relationship is satisfied, k is an integer:

$$k * \mathcal{L}_s = \mathcal{L}_r + \sum_{i=1}^{\tilde{\mathcal{N}}} \mathcal{B}_i. \tag{3.1}$$

The relationship is explained as: if a ring is extended to include the buffering at each switch attached to it, the ring can be divided into k continuous and non-

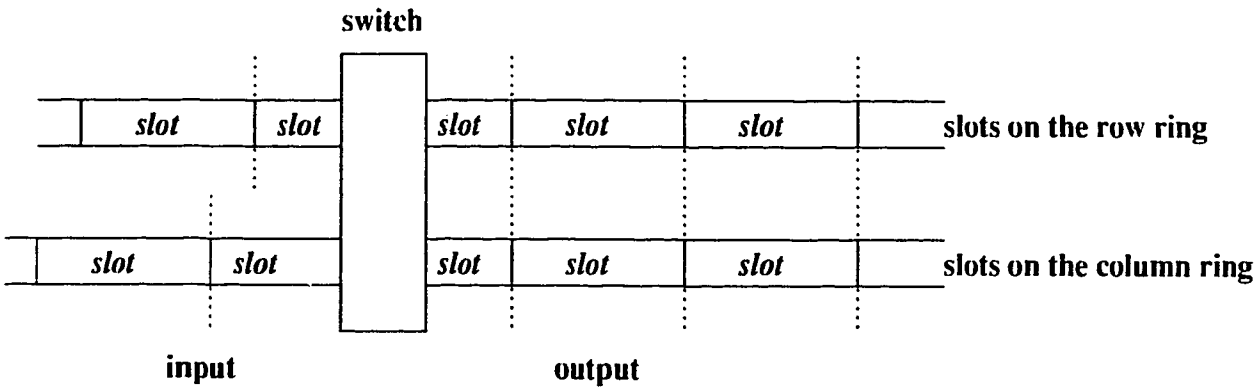


Figure 3.2: Transmissions of slots are synchronized after initialization.

overlapping slots during the regular operation of the network. $\mathcal{B}_j$ may differ among switches but remains unchanged for all slots arriving at switch j from the ring. Assume $\bar{\mathcal{B}}$ is the number of bits delayed at every switch for the purpose of extracting routing information, $\mathcal{B}_j$ is at least $\bar{\mathcal{B}}$ and at most $(\bar{\mathcal{B}} + \mathcal{L}_s)$.

3.1.2 Buffering

This approach differs from the *aligning* one in the sense that routing decisions can be made for only one slot at one moment. To implement this approach, it is necessary to introduce two buffers which are called *deflection buffers*. Figure 3.3 illustrates a logical connection between the storage units and switching devices. For simplicity, the local connections are not plotted. In the figure, B_h stands for the buffering of the header of every slot. RB and CB are two delaying buffers associated with the row and column input port respectively. They buffer slots which are subsequently retransmitted to the same ring from which the slots are previously received. RDB and CDB are two deflection buffers for slots that are routed to the other ring. All the buffers operate in a bit by bit fashion.

The *buffering approach* works based on the following rules:

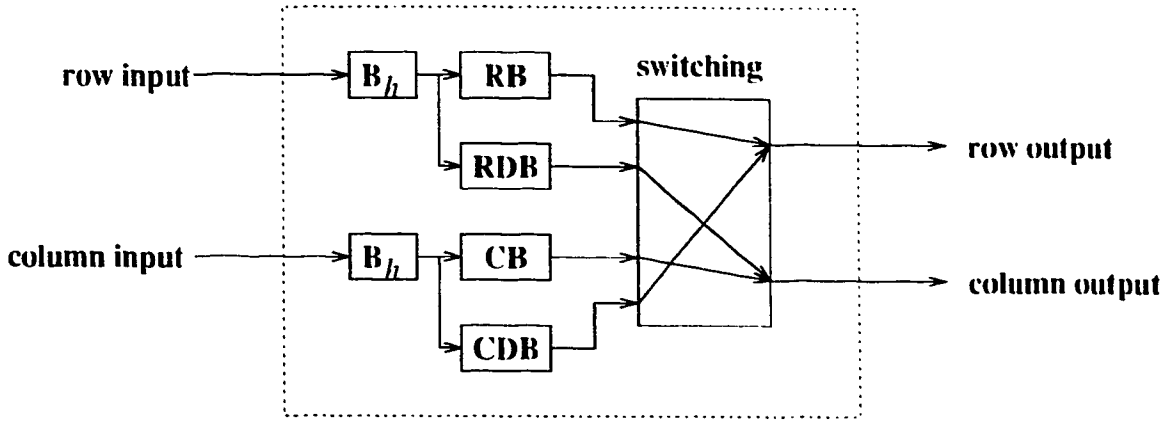


Figure 3.3: A switch with deflection buffers.

- *Priority:* Slots in deflection buffers are transmitted prior to the slots in delaying buffers. For example, if there is a slot in RDB and one in CB, the one in RDB will be transmitted immediately after the transmission of a slot at the column port and prior to the first slot in CB. Similarly, slots in CDB are transmitted earlier than slots in RB.

- *Occupancy:* A buffer, either a deflection buffer or a delaying buffer, is said to be *occupied* if the buffer reaches its maximum capacity before the last bit of a slot is shifted in. For example, if the maximum capacity of a buffer is one slot, and the whole slot buffered is not being transmitted, shifting any new slots to the buffer prior to the transmission of the buffered slot will overflow the buffer. Once the routing of a slot is decided, the slot, for instance, from the row ring, is shifted to either RDB or RB. If neither of the two buffers is occupied, the slot can be shifted to the buffer connected to its preferred port; otherwise, the slot is shifted to the non-occupied one. In the latter case, a **deflection** takes place if the slot has to travel a longer distance to its destination via the port.

- *Insertion:* It is possible that when a slot is shifted to a deflection buffer, say RDB, there is nothing to transmit along the ring from which the slot is previous received (that is the row ring in the case of RDB), which causes a period of

idling in the row ring, and in turn drops the global throughput. To prevent this happening, a switch inserts a slot via the row output port. The slot can be either empty or full according to the *Filling rule*.

- *Discarding:* Empty slots are not shifted to any of the buffers.
- *Filling:* When both buffers, one delaying buffer and one deflection buffers are empty, the switch generates an empty slot to be transmitted via the corresponding output port. The switch can fill the slot with proper payload bits if it has a nonempty local queue.

The buffering capacity required depends on deflection methods. As shown in chapter 5, when using *Vertical Deflection*, the duration of buffering can be minimized to less than the length of two slots. Besides, the duration of buffering of a slot in each buffer is fixed during the regular operation of the network.

3.1.3 Conclusion

It has been pointed above that for both *aligning* and *buffering* methods, the buffering delay of a slot to be retransmitted by a switch remains the same for all slots being retransmitted by that switch in the same buffer. The property is very useful because in the context of optical transmission and processing where buffers are implemented using segmented optical delay lines, the design of the sensor can be simplified to only have one tap.

Comparing the *aligning* and *buffering* methods, the *buffering* method introduces more overhead by shifting slots, but its routing control logic is simpler especially when the connectivity – the number of input and output ports is large. With *aligning* method, routing decision has to be made to match k incoming slots to k output ports. But, with *buffering*, the routing decision is distributed and made for a slot immediately after its routing information is extracted. Routing only performs a $1-to-k$ mapping – one slot is shifted to one of k delaying/deflection buffers, and a

subsequent $k - to - 1$ mapping. Though it has two stages (they can be pipelined if more than two slots are buffered), it is much easier to be implemented. At the current stage of optical technology, the *buffering* method has some practical advantage.

The choice of synchronization methods is also affected by the deflection strategy. For example, when using *Random Deflection* that minimizes the remaining distance, *aligning* is better than *buffering* in the sense that *Random Deflection* performs better in the presence of two slots to be routed at the same moment. This is similar to the comparison between *non-blocking* and *quasi-blocking* synchronization for non-slotted networks as revealed in section 4.2.

3.2 Asynchronous Slotted Networks

3.2.1 Distributed Transmission Rates

In many real situations, the *synchronous* condition does not hold for switches may have close but slightly different transmission rates (or **bit rate**, which is measured in bits per second). The difference is known as **clock tolerance**, denoted by α.

Suppose that the transmission rate (in bits per second) of the fastest switch is $\mathcal{R}_{max}$, and that of the slowest switch is $\mathcal{R}_{min}$. We define a (imaginary) nominal clock which has a rate $\mathcal{R}^* = (\mathcal{R}_{max} + \mathcal{R}_{min})/2$, and the maximum clock tolerance $\alpha = \mathcal{R}_{max} - \mathcal{R}^* = \mathcal{R}^* - \mathcal{R}_{min}$.

When $\alpha \neq 0$, the time a switch transmits a bit varies from switch to switch. Since the combined length of payload and header in bits remains the same for every slot transmitted by any switch (This length is denoted by σ), the time required for a switch with rate $\mathcal{R}$ to insert the header and payload of a slot is $\frac{\sigma}{\mathcal{R}}$.

A switch receives a slot using the rate the slot is transmitted, and the slot is retransmitted using the local transmission rate of the switch receiving it. There usually is a difference between the receiving rates and the transmitting rates of slots, and the difference will cause two problems described in the next section.

3.2.2 Two Problems

In this section, we present two problems caused by the existence of distributed transmission rates. We also provide some guidelines to deal with the problems.

Consider two adjacent switches, s_1 and s_2, with s_2 being the receiver of bits transmitted by s_1 via the link. Let $\mathcal{R}_1$ and $\mathcal{R}_2$ stand for the transmission rates of s_1 and s_2 respectively, $\mathcal{R}_1 \neq \mathcal{R}_2$.

Potential Problem #1 ($\mathcal{P}_1$): When $\mathcal{R}_1$ is less than $\mathcal{R}_2$...

When $\mathcal{R}_1$ is smaller, slots are retransmitted faster at s_2 than they are received. After s_2 has transmitted a certain number of bits, it may find that there is no bit to be retransmitted. If this happens, s_2 either generates a link or node failure message although there is not really such a failure existing, or s_2 changes its transmission rate during the middle of a slot, which causes the immediate successor of s_2 not to receive slots correctly.

To avoid the problem, we can establish a minimum number of bits of an incoming slot to be received before the slot is retransmitted. This threshold is denoted as Δ_1 in the thesis. The threshold is set so that a slot ready to be retransmitted should at least be buffered for Δ_1 bits. Therefore, after s_2 starts to insert a slot, the last bit of that slot arrives at s_2 prior to the moment when s_2 is supposed to complete the transmission of the slot. The problem can also be handled by introducing a *master clock* which aligns all the local transmission rates.

Potential Problem #2 ($\mathcal{P}_2$): When $\mathcal{R}_1$ is greater than $\mathcal{R}_2$...

When $\mathcal{R}_1$ is larger, it indicates that s_1 inserts bits faster than s_2 does. Since s_1 continuously transmits slots into the link, and these slots are retransmitted slower than they are received at s_2, buffers of s_2 tend to accumulate bits as s_2 continues its operation. After a sufficiently long operation, s_2 will run out of buffer space and start to lose slots at the input port.

To cope with the problem, s_1 and s_2 must synchronize the rates of slots being transmitted at the two switches. We define the **slot rate** as the rate at which a switch transmits slots. Switches with different *bit* rates are possible to have the same *slot* rate. It can be simply done as: assume the slowest switch (with transmission rate $\mathcal{R}_{min}$) needs $\frac{\sigma}{\mathcal{R}_{min}}$ to transmit a slot, while some other switch (with transmission rate $\mathcal{R}$) transmits a slot using $\frac{\sigma}{\mathcal{R}}$. Before transmitting the next slot, the switch (with rate $\mathcal{R}$) somehow wastes an amount of time equivalent to $\frac{\sigma}{\mathcal{R}_{min}} - \frac{\sigma}{\mathcal{R}}$. Based on this idea, all the switches can have the same slot rates. This approach is discussed in more detail in the latter sections.

3.2.3 Solutions

In this section, we discuss some possible solutions that can be adopted in the communication protocols to solve the above two problems. The *master clock* approach aligns local transmission clock rates; the use of a sufficiently large buffer at some switches adjusts the *slot rates* of switches; and the *appending excess bits* deal with individual transmission clock rates at each switch.

Master Clock

Introduction of a *master clock* is a standard solution to the asynchronism of a slotted network. The switch with the master clock transmits slots at regular intervals; other switches derive their local transmission rates according to the master clock to have the same transmission rate as the switch with the master clock, effectively reducing σ to 0. The network thus works as if it were synchronous. It is therefore not necessary to require any additional buffering (such as Δ_1) at the incoming buffers. However, a fatal disadvantage of the solution is the cost to handle the failure of the node with the master clock. Also, "big-a" networks would require an almost continuous flow of synchronization slots, wasting a large proportion of the network capacity.

Sufficient Buffering and Idling Symbols

This method is based on the fact that in a slotted ring network, at any moment during the regular operation, the number of outstanding slots (including slots carried by the medium and buffered at switches) remains unchanged for each ring. Therefore, the method aims at adjusting *slot rate* at each switch, which is done by maintaining a sufficiently large incoming buffer at each switch, especially at slower switches.

To deal with $\mathcal{P}_1$, the first bit of a slot is not transmitted by a switch until the last bit of the slot has been received by the switch. During the interval between the transmission of two consecutive slots (i.e. the interval between the transmission of the first bit of a slot and the last bit of its predecessor), a switch inserts **idling symbols** to the links.

As to $\mathcal{P}_2$, since $\mathcal{P}_2$ only happens with the slower switches, for simplicity, we take switch $s_{slowest}$ as an example. $s_{slowest}$ is a switch with the transmission rate of $\mathcal{R}_{min}$, and we assume the incoming buffer capacity of $s_{slowest}$ is k slots, supposing there are totally k slots in the ring.

Since there are totally k slots in the ring, and the buffer in $s_{slowest}$ is large enough to hold all these slots, slots won't be lost at $s_{slowest}$. On the other hand, when the buffer at $s_{slowest}$ accumulates bits, other switches start to reduce their slot rates since: (1) the rate they receive slots decreases, and (2) more idling symbols are received. As a result, the time that a switch transmits k slots is the same for all switches, equal to $\frac{k\sigma}{\mathcal{R}_{min}}$.

Obviously it is expensive to have a large buffer, and the maximum capacities required for the buffers may vary if the network conditions changes, which may cause the method not to work properly. These network conditions include the number the nodes on a ring, the length of the ring, and most important, the number of slots on each ring. The method also causes excessive buffering delays at slower switches.

Excess bits

This technique was first presented in [Max88]. It is very similar to the previous one, but much more astute. Instead of inserting idling symbols, it varies the total length of each slot dynamically: additional bits are appended to the end of a "regular" slot so that a slot now contains three parts: *header*, *payload* and *excess* as shown in figure 3.4.

The idea of the method is: if $\mathcal{R}_1$ is larger than $\mathcal{R}_2$, slots retransmitted by s_2 are shorter than they are received at the input port of s_2; otherwise, if $\mathcal{R}_1$ is smaller than $\mathcal{R}_2$, slots retransmitted by s_2 are longer. The combined length of the header and payload part of a slot remains the same at any moment, but the length of excess varies. Globally, the slowest switch transmits slots without excess bits and the fastest switch transmits slots with the maximum number of excess bits. In the next section, using this method in a slotted DMN to deal with the asynchronism is discussed in more detail.

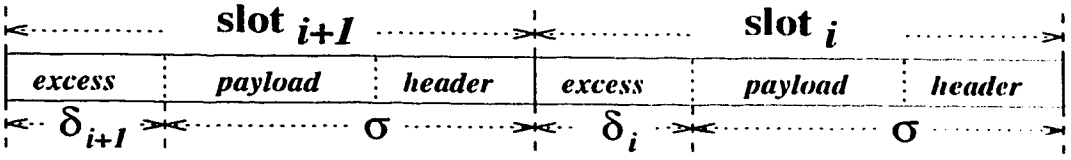


Figure 3.4: Slots with excess bits appended

With appending bits, the requirement of buffering at each switch is only dependent on the clock tolerance α, and is much less than the requirement in the method with a large buffer[1]. It is also insensitive to the change of network conditions.

The difference of three methods can be summarized as follows. Using the *master clock* approach, all the switches have the same derived transmission rates. By introducing a large buffer, *slot rates* of every k slot are the same for all switches. By

[1]One may view the "excess bits" method as one where the fiber link is used as a buffer (instead of buffering at switches).

appending excess bits, switches may have different transmission rates, but the slot rates, with respect to the nominal time required to transmit one slot, are approaching the same from switch to switch in the long run. The tuning with *appending bits* approach is much finer.

3.3 Appending Excess Bits in Asynchronous Slotted Network

In this section, we discuss the method of appending excess bits to a slot to deal with the asynchronism in more detail. We first study two thresholds Δ_1 and Δ_2 to handle problems $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. Then we present a description of protocol processes that carry out the operation in a two-connected slotted DMN.

With the method, every slot contains a number of excess bits after the header and the payload bits. Upon receiving a slot, the header and the payload bits (σ bits) of the slot are shifted bit by bit into an incoming buffer and excess bits are discarded.

The solution to problem $\mathcal{P}_1$ only concerns the length of header and payload bits buffered with respect to the clock tolerance, and the solution of problem $\mathcal{P}_2$ concerns the number of excess bits necessary to be appended. The next two sections describe the solutions to these two problems.

3.3.1 Problem $\mathcal{P}_1$ and The Solution: Δ_1

As we described earlier, the problem $\mathcal{P}_1$ can be solved by introducing a threshold Δ_1 on the incoming buffer: a slot is not retransmitted until Δ_1 bits of the slot has been buffered.

The value of Δ_1 is set to ensure that it takes the fastest switch longer to transmit σ bits into the network than it takes the slowest switch to transmit $\sigma - \Delta_1$ bits. The

relationship is expressed as in equations 3.2 and 3.3.

$$\frac{\sigma - \Delta_1}{\mathcal{R}^* - \alpha} \geq \frac{\sigma}{\mathcal{R}^* + \alpha} \tag{3.2}$$

yielding

$$\Delta_1 \geq \frac{2 \times \sigma \times \alpha}{\mathcal{R}^* + \alpha}. \tag{3.3}$$

Equations 3.2 and 3.3 require the following explanation. Assume the transmission rates of s_1 and s_2 are $\mathcal{R}_{min}$ and $\mathcal{R}_{max}$ respectively. Let s_2 be the immediate successor of s_1. For every slot s_2 receives from s_1, s_2 retransmits the slot after Δ_1 bits of the slot has been buffered. If Δ_1 satisfies equation 3.3, it is ensured that the last bit of the header and payload part of the slot inserted by s_1 is received prior to the completion of the transmission of the slot by s_2. Thus, problem $\mathcal{P}_1$ is solved.

During the initialization procedures, new empty slots are inserted to the network locally as will be described by the operation of protocol processes in section 3.3.3. After the network enters the regular operation, every slot to be retransmitted will be buffered at least Δ_1 bits.

3.3.2 Problem $\mathcal{P}_2$ and The Solution: Δ_2 and δ_{max}

To solve problem $\mathcal{P}_2$, we have to adjust the *slot rates* at switches by appending a proper number of excess bits to a slot. The solution of $\mathcal{P}_2$ depends on two parameters: first, the maximum length of excess bits that is sufficient for the slowest switch to operate; and second, the number of excess bits appended to a slot by an arbitrary switch so that the *slot rates* converge. The first parameter is denoted as δ_{max}. The second parameter is different from switch to switch, therefore, we introduce a universal threshold Δ_2 on the incoming buffers so that when the number of bits in the incoming buffer exceeds Δ_2, the switch should cease the current transmission, and starts the transmission of a buffered slot.

Length of Excess (δ)

The length of excess of a slot is denoted by δ; δ varies for a slot after it is transmitted by different switches. A switch transfers slots with more excess bits than its successor if it is *faster* than its successor, and vice versa.

Let δ_{max} denote the maximum number of excess bits needed. The fastest switch (with transmission rate $\mathcal{R}_{max} = \mathcal{R}^* + \alpha$) transmits slots of $\sigma_0 + \delta_{max}$ bits, and the slowest switch (with transmission rate $\mathcal{R}_{min} = \mathcal{R}^* - \alpha$) transmits slots of σ bits (i.e. without excess bits). In order for the slowest switch to transfer slots inserted by the fastest switch, the following relationship is to be satisfied:

$$\frac{\sigma + \delta_{max}}{\mathcal{R}^* + \alpha} = \frac{\sigma}{\mathcal{R}^* - \alpha} \tag{3.4}$$

The left hand of equation 3.4 is the nominal time spent to transmit a slot at the fastest switch, and the right hand is the nominal time required to insert a slot at the slowest switch. From equation 3.4, we have

$$\delta_{max} = \sigma \times \frac{2\alpha}{\mathcal{R}^* - \alpha} \tag{3.5}$$

The explanation of δ_{max} is that in order to prevent the slowest switch from running out of buffer space, the length of the excess bits of a slot transmitted by the fastest switch should be at least δ_{max} bits, so that the slowest switch can have enough bits to discard for increasing its *slot rate*. By discarding excess bits to make up the difference of transmission rates, the *slot rate* of the slowest switch can be the same as that of the fastest switch.

Threshold Δ_2

We have established the minimum number of excess bits that is required to be appended by the fastest switch. Now we establish Δ_2 to control the transmission of excess bits at different switches.

When the length of the buffer reaches Δ_2 bits, it means slots are being received faster than they are being transmitted, which may happen during either the initialization procedure or the regular operation of the network. During the initialization procedure, when a switch sees that the length of the buffer is below Δ_1, it inserts a new slot. The threshold Δ_2 is used to restrict the length of incoming buffer at the end of the transmission of the new slot[2]. On the other hand, during the regular operation of the network, it requires that the length of the buffer not exceed Δ_2 bits before the switch completes the transmission of the previous slots (*header* plus *payload*), which is assured if Δ_2 is guaranteed during the initialization procedure[3]. When the number of bits in the incoming buffer reaches Δ_2, the excess bits of the slot currently in transit is truncated, and the switch starts transmitting the buffered slot to increase the rate that the switch transmits slots.

To get Δ_2, we assume that the fastest switch can not transmit $(\Delta_2 - \Delta_1)$ bits when the slowest switch transmits $\delta_{max} + \sigma$ bits, which is what happens during the initialization procedure. Therefore:

$$\frac{\Delta_2 - \Delta_1}{\mathcal{R}^* + \alpha} = \frac{\sigma + \delta_{max}}{\mathcal{R}^* - \alpha}. \tag{3.6}$$

And the solution is

$$\Delta_2 = \Delta_1 + (\sigma + \delta_{max}) \times \frac{\mathcal{R}^* + \alpha}{\mathcal{R}^* - \alpha}. \tag{3.7}$$

3.3.3 Operation

To implement the above idea in a two-connected DMN, the operation can be carried out by two receiver processes (RP), a switch process (SP) and a transmitter process

[2]It should be made clear here that during the initialization procedure, when a switch inserts a new slot after it detects the incoming buffer is less than Δ_1, it appends the maximum length of excess bits δ_{max} to the slot. This is because at that moment, the switch does not know whether its transmission rate is *faster* or *slower* comparing to other's. Appending δ_{max} bits achieves a great flexibility.

[3]detailed derivation can be found in[Max88]

(XP) in every switch. Two RP's control the receiving of incoming slots and discarding excess bits. SP decides the routing of slots and synchronizes with XP during the transmission of slots. XP is responsible for inserting slots and appending excess bits to each slot. The synchronization of SP and XP is controlled by the following three signals:

1. $\mathcal{SIG}$(INSERT): with this signal, SP notifies XP to start transmitting slots buffered or inserting new slots.

2. $\mathcal{SIG}$(STOP): with this signal, SP asks XP to terminate its current transmission at both output ports. This signal is followed by a $\mathcal{SIG}$(INSERT) to notify XP to proceed to transmit the next two slots.

3. $\mathcal{SIG}$(EX): with this signal, XP notifies SP that a maximum number of excess bits have been appended to slots transmitted.

Receiver Processes(RP)

Each RP controls one network input interface. The function every RP performs is summarized as follows:

1. When the switch starts operation, RP clears the associated incoming buffer.

2. Upon receiving a slot from the incoming port, every bit making up the header and the payload of the slot is shifted one by one into the incoming buffer.

3. The bits of excess part of the slot are discarded.

4. If the switch is the destination of the payload of the slot, i.e., the receiver address of the payload matches the address of one of the stations attached to the switch, the payload of the slot is copied and the PS flag of the slot is cleared to be 0.

Switch Process(SP)

The function performed by SP includes the routing and transmission of slots.

1. When the network starts operation, SP generates two empty slots, and signals XP by $\mathcal{SIG}$(INSERT).

2. Upon notification by $\mathcal{SIG}$(EX) or one of the two incoming buffer has more than Δ_2 bits, SP does the following:

   (a) If SP is notified by $\mathcal{SIG}$(EX), it means that XP has transmitted the maximum number of excess bits of the slots, therefore,

      i. If none of the incoming buffer has more than Δ_1 bits, SP performs the same function as 1.

      ii. If only one of the incoming buffer has more than Δ_1 bits, SP removes the slot from the buffer and generates an empty slot, routes two slots and signals XP by $\mathcal{SIG}$(INSERT) to transmit two slots.

      iii. If both incoming buffers have more than Δ_1 bits, SP removes them from the buffers, routes them and signals XP by $\mathcal{SIG}$(INSERT) for transmission.

   In the above three situations, the first two of them happen only during the network initialization. Once the network enters regular operation, i.e. after both RP have heard transmission, no more empty slots are to be generated. When an empty slot is generated, SP does not fill it with payload because the whole network connectivity is yet unknown to SP.

   (b) If one of the incoming buffers has more than Δ_2 bits, SP signals XP by $\mathcal{SIG}$(STOP) to terminate the current transmission of slots.

      i. if both buffers have more than Δ_2 bits, SP removes one slot from each buffer, routes them and signals XP by $\mathcal{SIG}$(INSERT) for transmission.

ii. if only one buffer has more than Δ_2 bits, the slot is removed and routed with a slot either from the other buffer if the buffer has more than Δ_1 bits, or a generated empty slot if the other buffer has less than Δ_1 bits during the initialization of the network. SP then signals XP for transmission of the two slots.

Transmitter Process(XP)

XP performs functions of transmitting header and payload parts of slots and appending excess bits. Transmission of slots is triggered by $\mathcal{SIG}$(INSERT) and terminated by $\mathcal{SIG}$(STOP), summarized as follows:

Upon receiving $\mathcal{SIG}$(INSERT), XP starts to forward two slots at two output ports at the same moment using the same transmission rate. After the header and payload parts have been transmitted, XP starts to pad excess bits until it either receives a $\mathcal{SIG}$(STOP) or the maximum number of padding bits have been added to each slots, in which case, XP sends a $\mathcal{SIG}$(EX) to SP and terminates transmission.

In appendix A.1, we present a part of SMURPH source code that models the operation of the processes.

3.4 Asynchronism in Non-slotted Networks

Asynchronism in a non-slotted network is caused by two reasons: (1) the arrival of cells from a ring is intermittent, thus the interval between two consecutive cells, in the general case, is not predictable; (2) the distributed clocks have their own rates. This section studies possible ways of dealing with the asynchronism with emphasis on the first case.

3.4.1 Synchronizing Departure Times

In a non-slotted two-connected DMN, it is impossible to align a cell from one ring with a cell from the other ring within a switch for the arrival time of each cell is unknown to a switch at any moment. From the point of routing decisions being made, there are two practical ways of synchronization: *non-blocking* and *quasi-blocking*.

Non-Blocking

The method of *non-blocking* actually does not synchronize incoming cells. Every cell is routed individually and is subsequently retransmitted immediately if there is an available port.

For every switch, each incoming cell is first buffered long enough for extracting routing information. After that, if a cell is to be relayed, it is routed to the available port which may or may not be the cell's preferred port. In the case of both output ports being busy — which is the case when the cell arrives at the switch while the switch is in the middle of transmission of previous cells (e.g. locally inserted cells), the routing and transmission of the incoming cell is delayed until one of the output port becomes available. In order to minimize the delay of incoming cells, transmission of local cells does not take place unless there are not any incoming cells ready to be relayed. Therefore, as long as there is an available output port, an incoming cell will be relayed without any further delay of a predetermined length.

The technique is easy to implement, but it introduces unnecessary deflections that causes inefficiency as shown in figure 3.5.

In figure 3.5, C_1 and C_2 are two cells from the column ring, and R_1 and R_2 are two cells from the row ring. The sequence of the four cells arriving at the switch is R_1, C_1, R_2 and C_2, and the interval between two consecutive cells is less than a cell long. The preferred port of R_1, C_1 and C_2 is the column output port, and the preferred port of R_2 is row output port. Assume when R_1 is to be transmitted, both output ports are idle.

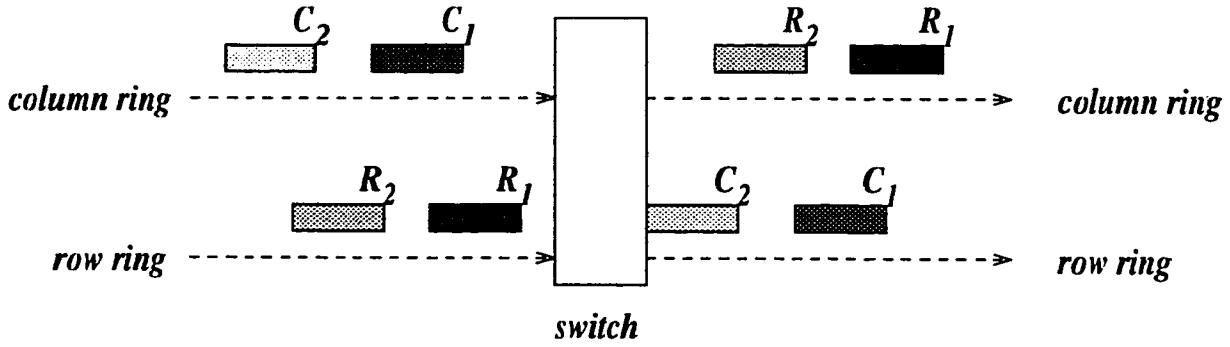


Figure 3.5: Non-blocking synchronization in a non-slotted network.

Using *non-blocking* synchronization, the sequence of retransmission is:

1. R_1 is routed to the column port;

2. C_1 is routed (deflected) to the row output port, because when C_1 is ready for transmission, R_1 is being transmitted via the column port;

3. R_2 is routed to the column port for C_1 is being transmitted via the row output port.

4. C_2 is routed to the row port for the similar reason as C_1.

In such a scenario, *non-blocking* causes 3 deflections – deflections of C_1, R_2 and C_2. But if the intervals between C_1 and C_2, and R_1 and R_2, are zero, with *aligning* introduced for slotted network, only one deflection is necessary. Compared to the *aligning* method, *non-blocking* causes three deflections. It is by no means efficient.

Store-and-Forward

Non-blocking method is easy to implement but causes unnecessary deflections. This is because routing decision is based only on one cell – the cell to be relayed.

Store-and-forward is another synchronization method which stems from slotted networks where slots from two rings are synchronized before routed. An incoming cell from one ring is stored in the incoming buffer until the arrival of a cell to the

other input port. Then routing decision is made for both cells in a way identical to slotted networks using *aligning* to synchronize.

Store-and-forward causes unreasonable delay under light load, because there are only a few transmissions occurring in the medium. It is also likely to incur *deadlock*. *Deadlock* appears when a switch buffers a cell, for example from the row ring, and waits for a cell from the column ring. Therefore, the switch blocks the transmission on the row ring. A number of switches in such a state may form a cycle that blocks the transmission of 4 or more rings, then *deadlock* is likely to spread to the whole network.

The potential cycle can be broken by introducing a time-out mechanism that leads to a *quasi-blocking* synchronization.

Quasi-Blocking

With this method, a time-out timer is added to the *store-and-forward* method so that a cell is buffered for a duration of up to a predetermined length which is usually equal to the length of a cell.

After a cell is ready for routing, it is first delayed for up to the length of a cell, during which:

- if there is a cell arriving at the other input port, and its routing information is already decoded, the switch makes the routing decision for both cells using the same way as using *aligning* in a slotted network.

- otherwise, at the end of the delay the cell is treated the same way as *non-blocking* synchronization does.

As will be compared in section 4.2, the routing of *quasi-blocking* outperforms *non-blocking* by a great amount in terms of throughput achieved. This is because *quasi-blocking* provides more information for making routing decision. Consider the same

incoming sequence of cells in figure 3.5, which is now plotted in figure 3.6, using *quasi-blocking* synchronization, what happens is:

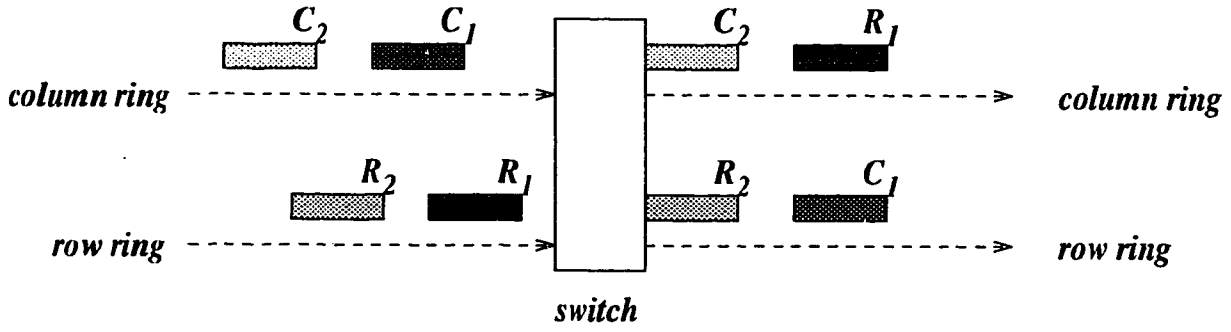


Figure 3.6: Quasi-blocking synchronization in a non-slotted network.

1. R_1 is delayed for the length of one cell before transmission.

2. By the end of the delay, C_1 is ready for transmission, then R_1 and C_1 will be transmitted via column and row port respectively (suppose R_1 wins the contention for the column port).

3. By the end of the delay of R_2, C_2 is ready for transmission, then R_2 and C_2 are routed to the row and column port respectively.

With *quasi-blocking*, there is only one deflection (deflection of C_1). The consequence of the synchronization using *non-blocking* and *quasi-blocking* are compared in figure 4.6 and figure 4.7, where the throughput by using *quasi-blocking* is much larger.

3.4.2 Distributed Clocks

In the previous two sections, *non-blocking* and *quasi-blocking* are discussed assuming a synchronous network where all switches use the same transmission rate. As in a slotted network, in real situations, it is always possible that there exists a small but

non-zero difference of local transmission rates. The speed of electronic signals or light may also vary in the transmission medium. But the resulted asynchronism is much less relevant than that in a slotted network, and the methods introduced above are still applicable.

This is because, first, in a non-slotted network, there is usually an idle interval between two consecutive cells anyway, regardless of clock rates. The idle interval serves naturally as a virtual buffer for *slower* switch to receive *faster* cells to solve $\mathcal{P}_2$. For *faster* switches (i.e. problem $\mathcal{P}_1$), buffering a few more bits than otherwise necessary allows to retransmit slower cells normally. Therefore, the above two methods can be used without significant modification.

Second, in an extreme case that a *faster* switch continuously inserts cells into the link to which its immediate and *slower* successor is connected, to prevent buffers of the slower switch from overflowing, traditional point-to-point flow/congestion control method is more worthwhile[4] than the way of handling asynchronism caused by difference in clock rates and link irregularity. It is different from a slotted network where such asynchronism can not be handled by flow/congestion control methods.

3.5 Conclusion

In a multi-connected network, synchronization deals with different arrival times of packets and non-zero clock tolerance. It is very important for routing which makes the decision to move a slot/cell from one ring to another. In the chapter, we have discussed methods of synchronizing flows of rings in a two-connected mesh networks. We have shown using *aligning* and *buffering* in a synchronous slotted network, and presented a method of dealing with distributed clock rates in an asynchronous network. Based on the approaches for the slotted network, we have also studied possible methods such as *quasi-blocking* in a non-slotted network where all the packets are of equal length.

[4]The detailed discussion is beyond the scope of the thesis.

All the methods are presented from the point of whether a routing decision is made for two slots/cells simultaneously or every slot/cell individually. These methods can be easily extended to mesh networks of higher connectivity.

Chapter 4

Two Random Deflection Protocols

In this chapter, we study the performance of two communication protocols of DMN. From the point of view of deflection routing, both protocols use a *Random Deflection* method: at every intermediate switch, a slot/cell is routed to the output port which minimizes the remaining distance to its destination measured by hops. Two protocols differ at the medium access methods: SRDP use slots while CRDP does not use slots but assumes there are cells — packets of equal length. The performance under uniform traffic is compared for square or rectangular node arrangements as well as the two derivations of 8×8 CTRDMN, $\mathcal{L}48$ and $\mathcal{L}64$.

4.1 SRDP – Slotted Random Deflection Protocol

We study the protocol for two-connected DMN where each switch has two input links and two output links. As stated earlier, the protocol (SRDP) basically performs two functions — it uses a slotted scheme to access the medium and it decides a one-hop-at-a-time routing for each full slot.

4.1.1 Operation

The operation of the protocol includes the insertion of payload into an empty slot, removal of payload bits from a slot, synchronization of two slots within a switch and deflection routing.

Insertion and Removal

It is simple to insert payload bits to an empty slot: after a switch detects there is an empty slot, the switch can inserts the payload bits — if there are some locally generated and queued — into the slot and set the *payload status* (PS) flag to be 1.

The protocol assumes destination removal strategy: when a switch detects that the payload of an incoming slot is addressed to the switch, or one of the stations attached to it, it copies the content of the slot. Then if the switch has some payload bits to transmit, it does so by inserting bits to the slot when retransmitting it without changing the PS flag; otherwise, if the switch does not want to transmit any payload bits, it retransmits the slot after clearing its PS flag to 0 to indicate an empty slot.

Every slot passing through a switch is buffered long enough so that after the addressing information is decoded, the switch is still able to change the PS flag and insert new payload bits into the slot. This improves the use of slots but introducing some more delay.

Synchronization

SRDP uses *aligning* method discussed in the last chapter to achieve synchronization of slots from row rings and the column rings. Slots from the row and column rings are delayed for a different amount of length before routing decision is made. The routing decision is made for two slots, one from row ring and one from column ring, at the same moment, and retransmission of two slots takes place at the same time via different output ports.

A non-zero clock tolerance is considered in the protocol. It is handled by appending excess bits. The maximum achievable throughput is smaller when appending bits to a slot. However, the reduction is not caused by the method of appending bits itself but by the fact of the distributed transmission rates which causes two problems that prevent the full utilization of the link bandwidth. On the other hand, the routing performance of using appending bits[1] is the same as a purely synchronized network for the fact that a routing decision is made for two slots, and slots in SRDP are the same as in a synchronous network in terms of routing information. The only difference between the two is the length of slots that doesn't affect routing.

Deflection

Random deflection is used in SRDP. The only output port minimizing the remaining distance to the destination is the preferred port of a slot. Every routing decision tries to route both slots to their preferred ports. If there is a conflict, the randomly selected *loser* is transmitted via the other port that causes a deflection.

The number of slots deflected can be decreased by introducing more buffer capacities at the output ports. However, in practice, deflection is not avoidable unless the output buffers have infinite capacity. *Random deflection* is able to take advantage of node arrangements having shorter $\mathcal{ASD}$ as well as communities-of-interest traffic, but it is impossible to predict the longest route a slot will travel from its source to the destination.

4.1.2 Performance

Simulation Parameters

The simulated environment is specified in section 2.4. In addition, since we consider a small clock tolerance, we further assume that:

[1] as well as adopting a master clock

1. the clock tolerance α is 1% of the nominal bit rate[2];

2. the maximum excess duration δ_{max} is 9 bits;

3. Δ_1 is 9 bits[3];

4. Δ_2 is 467 bits.

Besides the common measures defined in section 2.5, we define **Deflection Probability** as the probability of a full slot being deflected into a longer path to its destination. It is measured by the total number of full slots deflected to the total number of routing decision made. Since every routing decision is made for two slots, the deflection probability is below 0.5.

Since SRDP can take advantage of topology, we simulate SRDP for node arrangements of CTRDMN. The results presented in this section are for regular square or rectangular topology and uniform traffic.

Global Throughput and Message Access Delay

In figure 4.1, we plot message access delay against throughput ($\mathcal{T}$) achieved by SRDP. Figure 4.1 indicates that the maximum global throughput increases as the number of switches increases, which results in a larger number of rings. The global throughput is proportional to $\sqrt{\mathcal{N}}$ (roughly $(1.55 \pm 0.05) \times \sqrt{\mathcal{N}}$) where the number of switches is $\mathcal{N}$ (see table 4.1).

Deflection Probability

In Figure 4.2, we present the deflection probability as a function of normalized throughput ($\mathcal{T}$). When the throughput is fairly low, say below 40% of the maximum

[2]It is a fairly large tolerance in real life. But in fact, any smaller value of α yields similar results, in the sense that the throughput is a little (invisibly) higher, and the routing performance remains the same because tolerance is irrelevant to routing.

[3]Actually, every slot is buffered for 40 bits for extracting routing information.

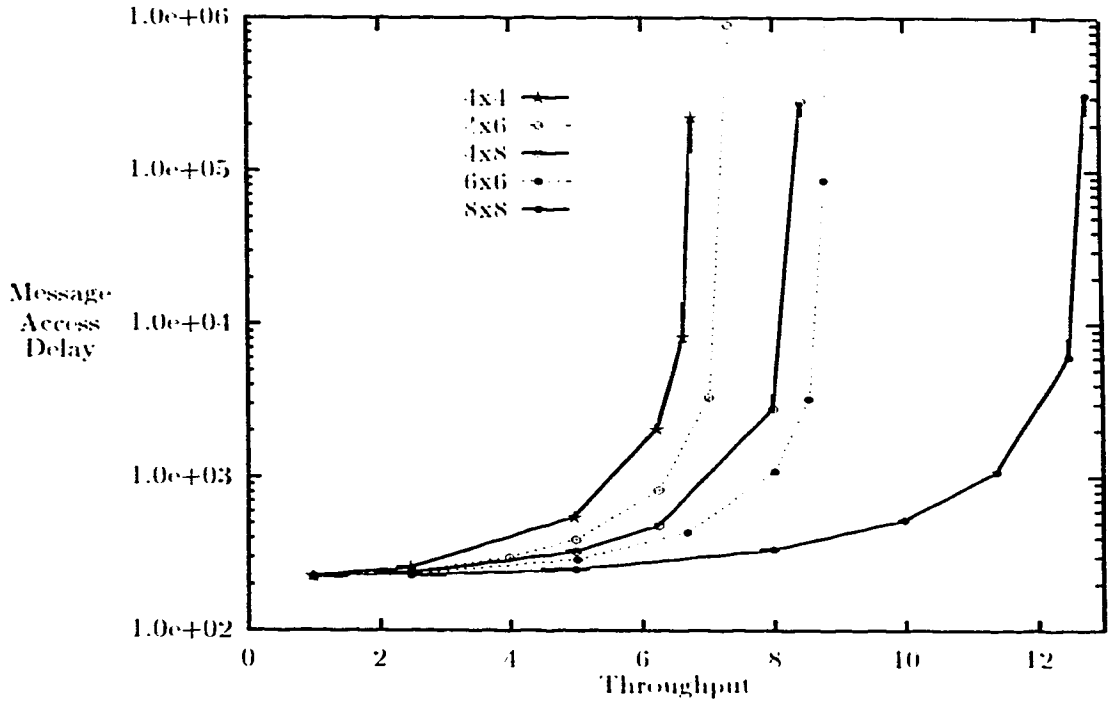


Figure 4.1: Performance of SRDP.

Topology	4×4	4×6	4×8	6×6	8×8
Throughput($\mathcal{T}$)	6.5	7.1	8.5	8.8	12.6
$\mathcal{T}/\sqrt{\mathcal{N}}$	1.6	1.5	1.5	1.5	1.6

Table 4.1: The maximum throughput of SRDP with respect to $\sqrt{\mathcal{N}}$.

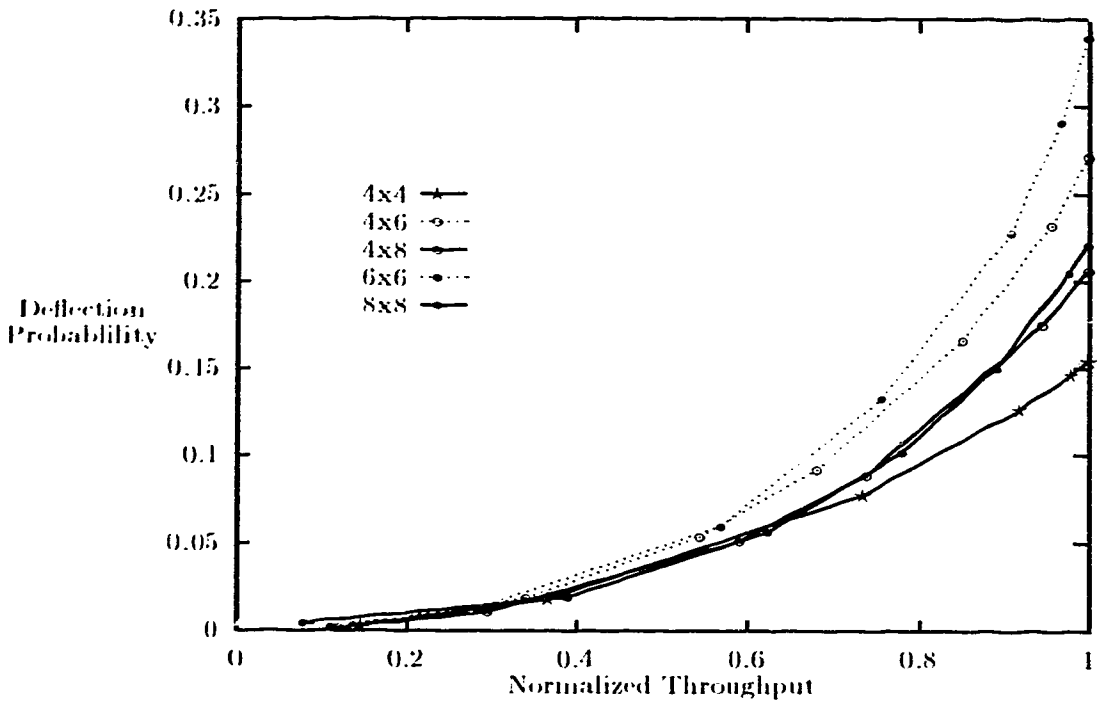


Figure 4.2: Deflection Probability of SRDP.

achievable throughput, the deflection probability is small for every node arrangement and the difference is invisible. But when the throughput increases, so does the deflection probability. When the maximum throughput is achieved, almost every slot is full. Therefore, deflection probabilities reach their maxima.

In the curves shown in Figure 4.2, it is observed that under heavy load, 6×6 topology has the largest deflection probability which is nearly 0.35 and 4×4 has the lowest. To explain this, in table 4.2, we give the probability of a slot having a preferred port when the slot is at the first hop to its destination.

Table 4.2 is constructed by calculating different hops to the destinations via row and column ports under uniform traffic assumption as illustrate in figure 4.3.

Assume the upper left switch is the source station, (0,0). In the figure, there are two numbers associated with each switch, represented by a box. The first is the

Topology	4×4	4×6	4×8	6×6	8×8
Preferred Ports	$\frac{6}{15}$	$\frac{14}{23}$	$\frac{14}{31}$	$\frac{26}{35}$	$\frac{38}{63}$

Table 4.2: The probability of a preferred port.

number of hops to the switches via the row port of (0,0), the second is the number of hops via the column port. In the figure, there are 9 stations each of which has the same distance from (0,0) via whatever port. The other 26 stations have different distances via different ports.

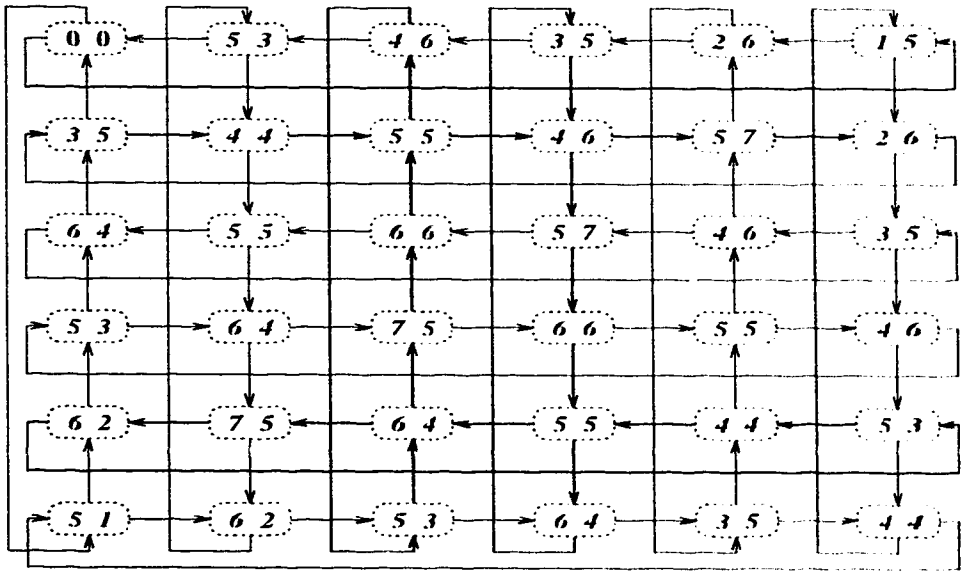


Figure 4.3: Port Preference of 6×6 under uniform traffic.

Figure 4.3 and table 4.2 reveal the probability of a slot having preferred port when the slot is just filled under uniform traffic, i.e. this is the probability of having preferred port in the first hop. The probability of having a preferred port increases as the slot advances towards the destination.

It is fairly difficult to derive an accurate analytical model to describe the relationship between the probability of having a preferred port and deflection probability. It

is even harder considering different load situations. The simulation results tell us that topologies with larger probabilities of having a preferred port at the first hop, such as 6×6 and 4×6, tend to incur larger deflection probabilities especially for heavy load.

Average Additional Hops

Another performance measure is the average length of routes a full slot traverses before reaching the destination. The length is measured in hops. The actual length includes two parts: a shortest distance which is decided only by the topology and traffic distribution, and an additional length caused by deflections.

In figure 4.4, the average additional hops between a pair of source and destination are normalized by the average shortest distance ($\mathcal{ASD}_{W_u}$), i.e. $\overline{\mathcal{AH}}$, and plotted as a function of normalized throughput for comparison. The figure shows that on average, a slot travels less than twice the shortest distance even for heavy loads, with slots in 4×4 travelling approximately 1.5 of the shortest distance and slots in 6×6 doubling the shortest distance.

Distribution of Additional Hops

In Figure 4.5, distribution of the additional hops is presented. The frequency is plotted against the additional hops a slot traverses before reaching the destination when maximum throughput is achieved. For node arrangements of CTRDMN, one deflection causes a slot to travel 2 or 4 additional hops. In 4×4, 4×8 and 8×8, one deflection causes a slot to travel exact 4 additional hops, while in 4×6 and 6×6, one deflection may causes 2 or 4 additional hops (See figure 4.3).

The curves in figure 4.5 illustrate that under heavy load, most (70%+) of the slots will reach their destinations with no more than 2 deflections. On the other hand, there are some slots that will reach their destination after traversing more than 10 times the average shortest distance. In a real network, these slots may be discarded by some

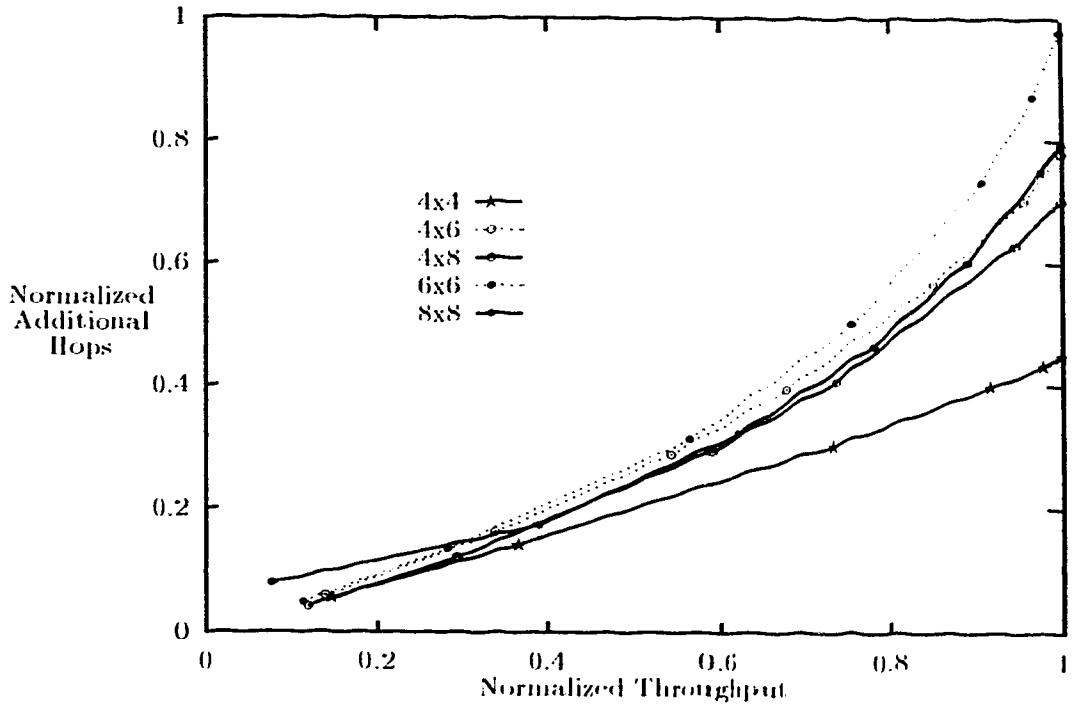


Figure 4.4: Average additional distance of SRDP.

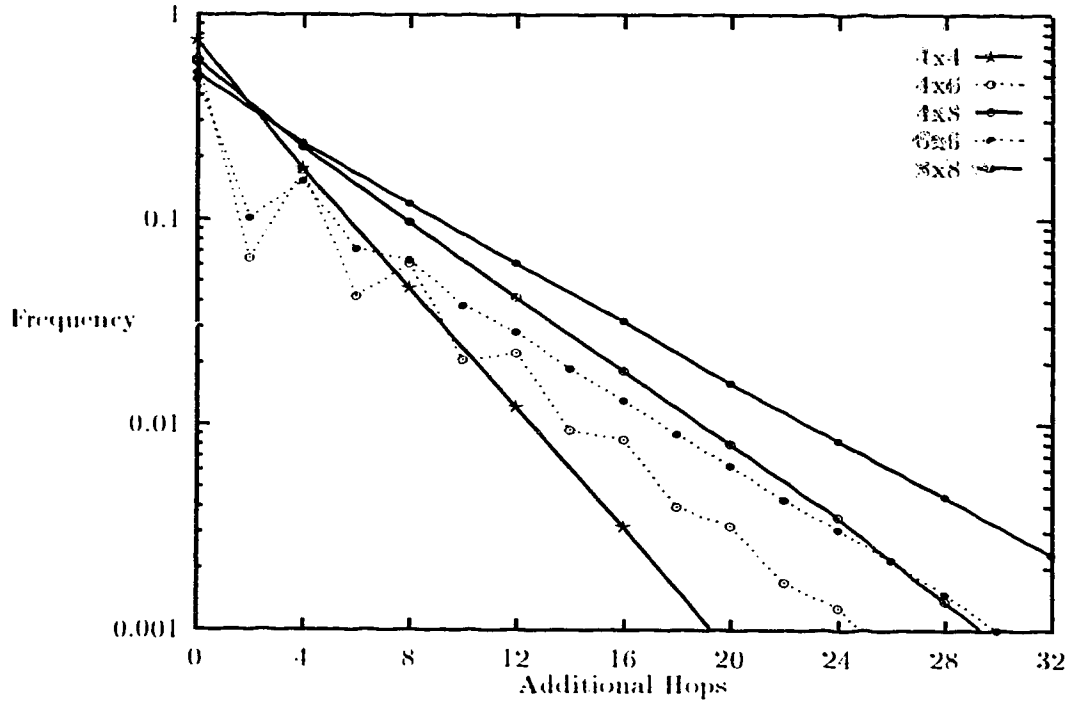


Figure 4.5: Distribution of additional hops under heavy load for SRDP.

intermediate nodes for their time-to-live in the network or, may cause a connection timed out. The drawback becomes even more serious if a multi packet message is to be transmitted across the network. In such a case, since packet loss is inevitable, it may take a very long time (possible *forever*) to receive the entire message. This problem prevents SRDP protocol from being a real life possibility.

4.1.3 Conclusion

The section studies a slotted protocol (SRDP) using *Random Deflection* to route slots. Every two slots are routed together. Each slot is routed to its preferred port which minimizes the remaining distance of the slot to its destination. If the preferred port is given access to another slot when the slot is ready for retransmission by a switch, the slot is routed to the other port which leads to a longer route to the slot's destination.

Both the operation and the performance of SRDP are presented. Most of the techniques used in SRDP, such as *aligning* synchronization, *Random Deflection*, and using appending bits, have been discussed in earlier chapters in detail. The description of the operation of the protocol is brief. As to the performance, a most significant feature of SRDP is that it yields increasing global throughput with the size of the network, which is a feature of mesh networks. One drawback of the protocol is it fails to guarantee an upper bound on the delay in the network.

4.2 CRDP – Non-Slotted Random Deflection Protocol

In this section, we study a non-slotted protocol CRDP that uses the same deflection routing strategy as SRDP, but does not use slots. Instead, the protocol assumes equal-size packets — cells, running in the network. Each cell contains a header and a payload part, but the header does not contain the *payload status* flag as in a slot. A switch willing to transmit inserts a cell into the link when it is allowed to do so.

Cells are removed from the rings by their destination switches. The next two sections provide description as well as the uniform performance of the protocol.

4.2.1 The Description of CRDP

The description of the protocol includes the deflection rules, synchronization of incoming cells, and the insertion of local cells.

Deflection

As in a slotted network, a switch buffers a certain number of bits of a cell when the cell is received by the switch. After that, the switch makes a routing decision for the cell to be retransmitted.

The deflection rules used in CRDP are similar to that of SRDP which is described in section 2.3.1. In the case when there are two cells, one from each ring, ready to be routed, the routing decision is made according to the same rules described in section 2.3.1. In the case that there is only one cell to be routed, the switch makes the routing decision for the cell individually. The routing rules for one-cell routing is:

1. If two output ports are available, the cell is routed to its preferred port if it has one, or to one of the two ports at random;

2. If there is another cell being transmitted at one of the output ports, the cell to be routed is retransmitted via the other port no matter whether the port is the cell's preferred port or not.

3. If there is no port available for transmitting the cell, the cell is delayed until one of the output port is available, and it is retransmitted via that port.

In general, when a cell can not be transmitted via its preferred port, the cell is deflected immediately to a longer route to its destination.

Synchronization

One feature of CRDP is that asynchronous cells instead of slots are used in the network. The gap between a cell and its predecessor is not fixed and, in general case, unpredictable. Therefore, when a switch makes a decision for a cell from one ring, for example row ring, the switch has no knowledge about the moment when the next cell arrives at the switch from the column ring. Routing decisions without knowledge of the cells from both rings cause unnecessary deflections, and in turn, decreases throughput.

This has been shown in section 3.4 where techniques of *non-blocking* and *quasi-blocking* synchronizing are compared. *Quasi-blocking* delays the moment of deciding routing for a cell for up to one cell long. Therefore, it decreases the number of deflections by considering the routing information of a cell from the other ring within that duration.

In CRDP, *quasi-blocking* is used to improve the quality of routing. Every cell to be routed within a switch is delayed for up to the length of a cell. By the end of the delay, if there is another cell coming from the other ring, the two cells are routed immediately. Otherwise, the delayed cell is removed from the buffer, routed and transmitted individually.

Local Cell Insertion

Using *quasi-blocking* in CRDP, a switch can insert a local cell into the link only when it detects that at most one incoming cell is being delayed for transmission and both output ports are idle. It follows the following procedures:

- If there is no incoming cell being delayed for transmission:

    1. If the switch has two locally generated cells, the switch can route and insert them into different links immediately.

2. Otherwise, if the switch only has one locally generated cell, the cell is delayed for up to the length of one cell before being inserted into the appropriated link. During the delay, there may be another cell ready for transmission. This cell can be either an incoming cell of the switch or a new generated local cell. In this case, the delayed local cell is routed and transmitted together with the incoming cell or the new locally generated cell.

- If there is an incoming cell being delayed for transmission, the switch routes the cell together with one local cell and transmit them onto the different links.

4.2.2 Uniform Performance of CRDP

As with SRDP, CRDP is simulated on node arrangements of CTRDMN. All transmission clocks in CRDP are assumed to have the same rate. One measure – probability of one-cell routing – is defined uniquely for CRDP to measure the ratio of the number of routing decision for only one cell to the total number of routing decisions.

Global Throughput and Message Access Delay

Figure 4.6 and 4.7 present different throughput performance of CRDP with different synchronization methods. Comparing these two figures, CRDP using *non-blocking* synchronization yields a much smaller throughput. The reason has been stated earlier. For the following figures about the performance of CRDP, we assume *quasi-blocking* is used to synchronize incoming cells.

Figure 4.7 presents the performance of message access delay as a function of global throughput achieved. Similar to the results of SRDP, global throughput increases with the size of the network and number of rings (see table 4.3). When the number of rings is the same, such as 4×8 and 6×6, the topology with smaller average shortest distance, such as squares, yields larger throughput. For light load, the message access delay is

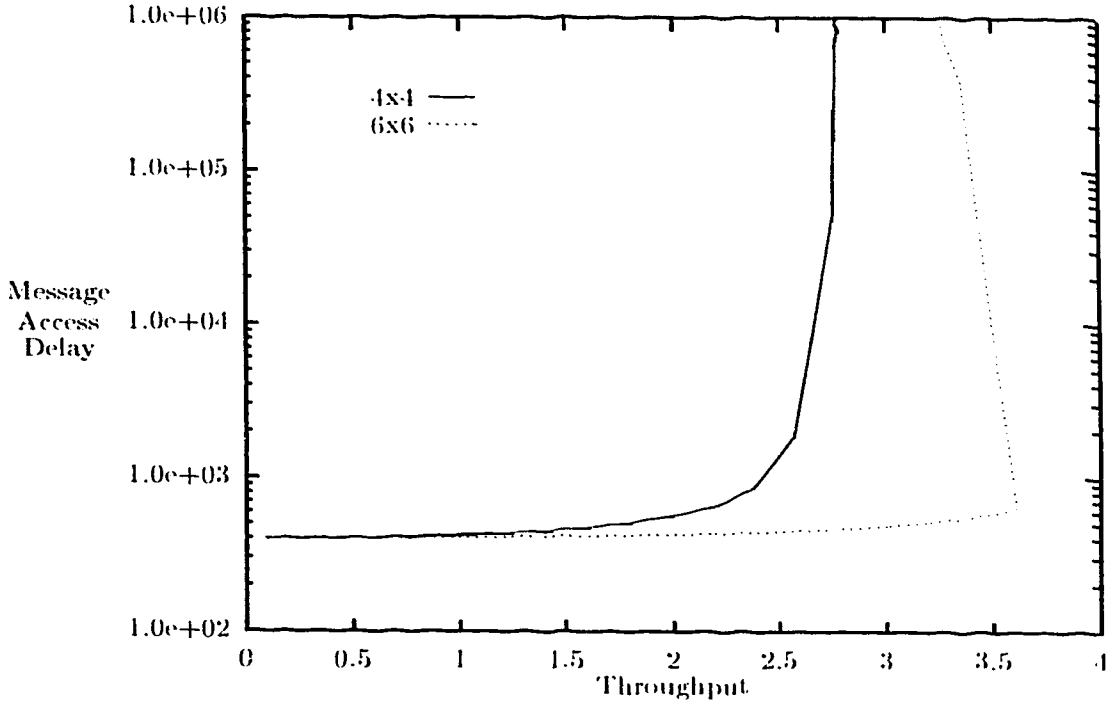


Figure 4.6: Performance of CRDP using *non-blocking* synchronization.

Topology	4×4	4×6	4×8	6×6	8×8
Throughput($\mathcal{T}$)	5.5	6.3	7.2	7.8	11.1
$\mathcal{T}/\sqrt{\mathcal{N}}$	1.4	1.3	1.3	1.3	1.4

Table 4.3: The maximum throughput of CRDP with respect to $\sqrt{\mathcal{N}}$.

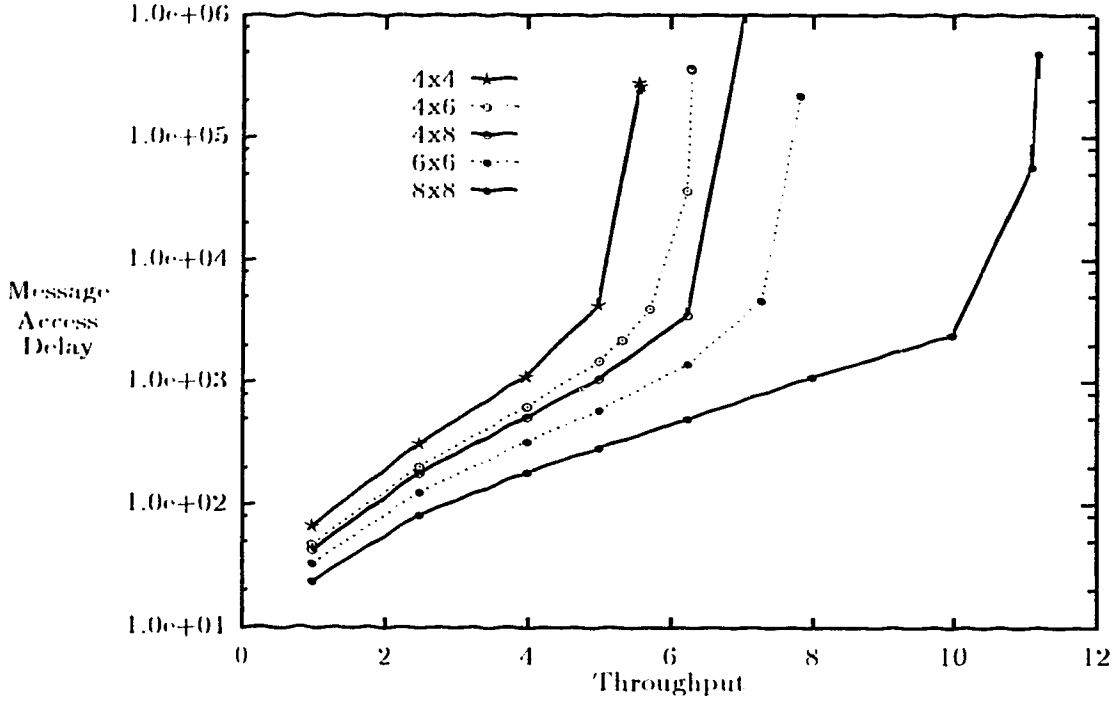


Figure 4.7: Performance of CRDP using *quasi-blocking*.

comparable to 0. And the node arrangements with more rings have lower message access delay.

Routing Probabilities

There are two routing probabilities of interest. One is the **one-cell routing probability** and the other is the **deflection probability** that reflects the probability that a cell has a preferred port, but it fails in the competition for that port and is forced to take a longer route to the destination. In the case of one-cell routing, the cell is always routed to its preferred port if it has one.

One-cell Routing Probability

The one-cell routing probability is plotted in figure 4.8 as a function of normalized

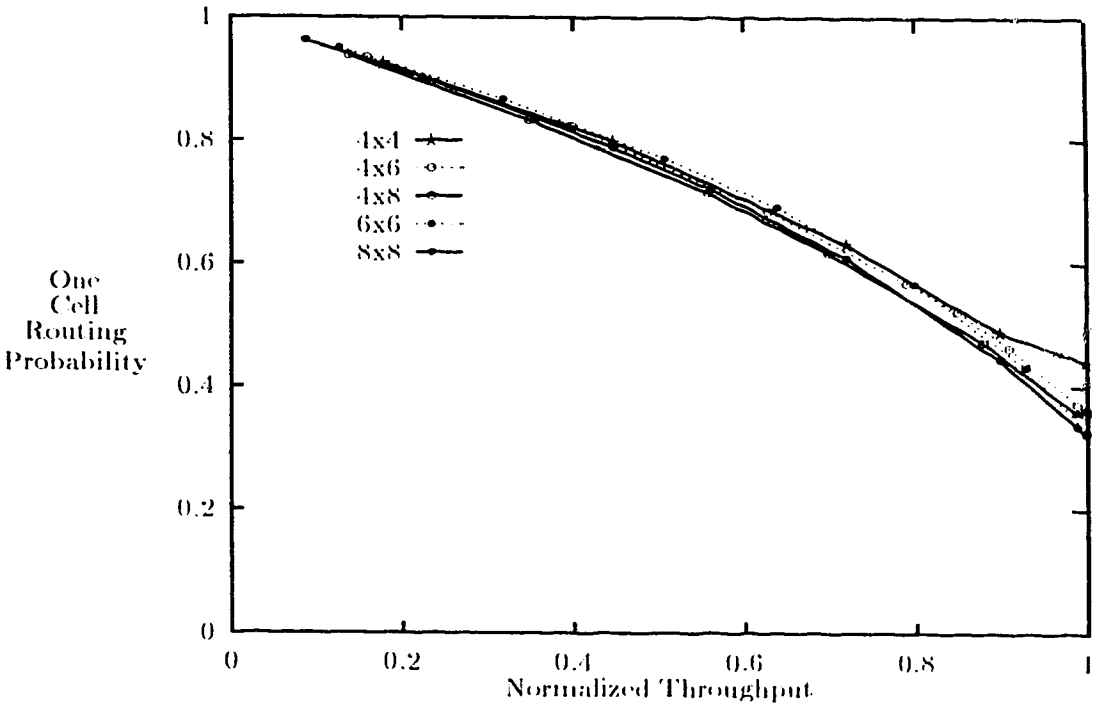


Figure 4.8: One-cell routing probability of CRDP.

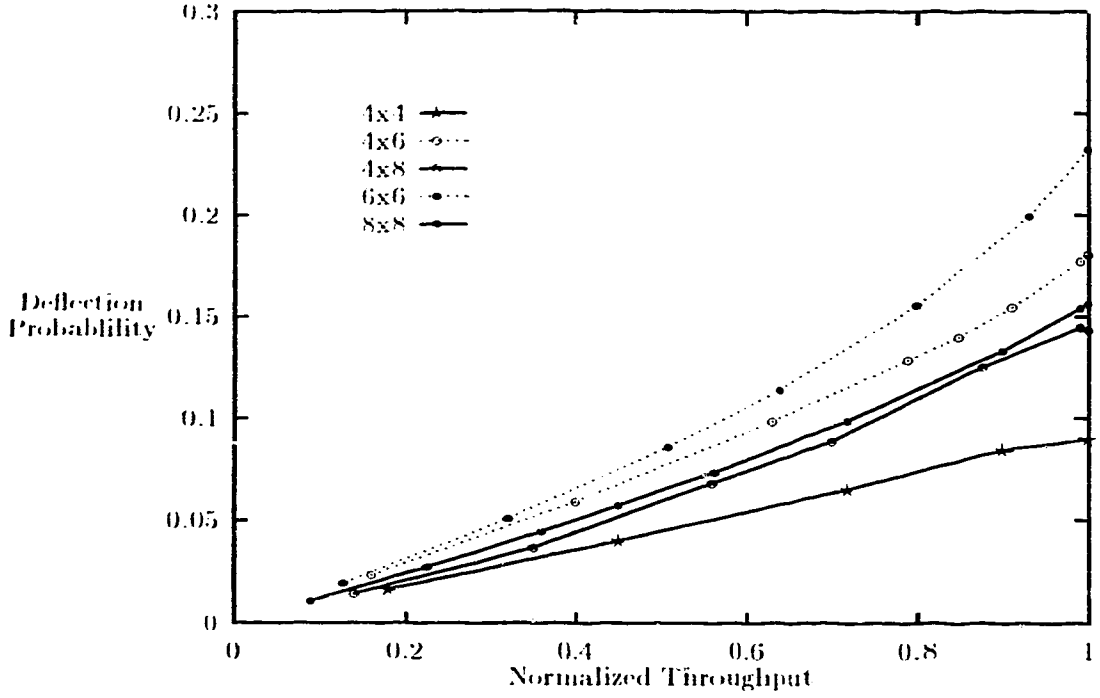


Figure 4.9: Deflection probability of CRDP.

throughput. It is very high for light load for there are only a few cells existing in the network. When the load increases, the probability drops: more of the cells are routed with their counterparts from the other rings. The difference of one-cell routing probability is quite small for different topologies. When the maximum throughput is achieved, there is a considerable number of packets, more than 30%, that are routed individually. The probability indicates that there are portions of links that do not carry cells. In other words, link bandwidth is not fully utilized. This contributes to the reason that the maximum throughput of CRDP is less than SRDP.

Deflection Probability

The deflection probability of CRDP is plotted in figure 4.9. Compared to the results of SRDP the values of SRDP are slightly lower. This is because every routing decision is made for two slots in SRDP, but in CRDP, some of the routing decisions are made

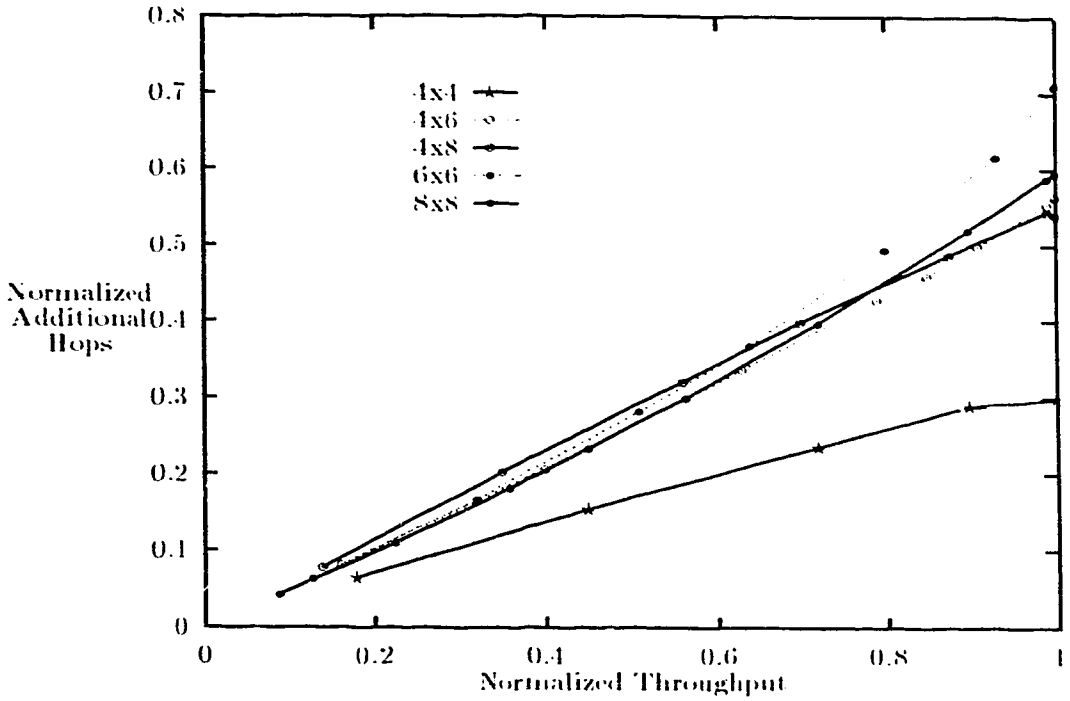


Figure 4.10: Additional Hops of CRDP.

for just one cell. As with SRDP, node arrangements bearing a smaller preferred port probability yield lower deflection probability. The results convince us that besides the load of the network, the preferred port probability is the most important factor contributing to the deflection probability.

Additional Hops and Their Distribution

The curves in figure 4.10 and 4.11 are the performance of normalized additional hops and its distribution of CRDP. Similar to SRDP, node arrangements having larger deflection probabilities also result in larger normalized additional hops. Compared to SRDP, the values of CRDP is a bit lower since fewer cells are deflected. Therefore, more cells arrive at their destination with none or one deflection.

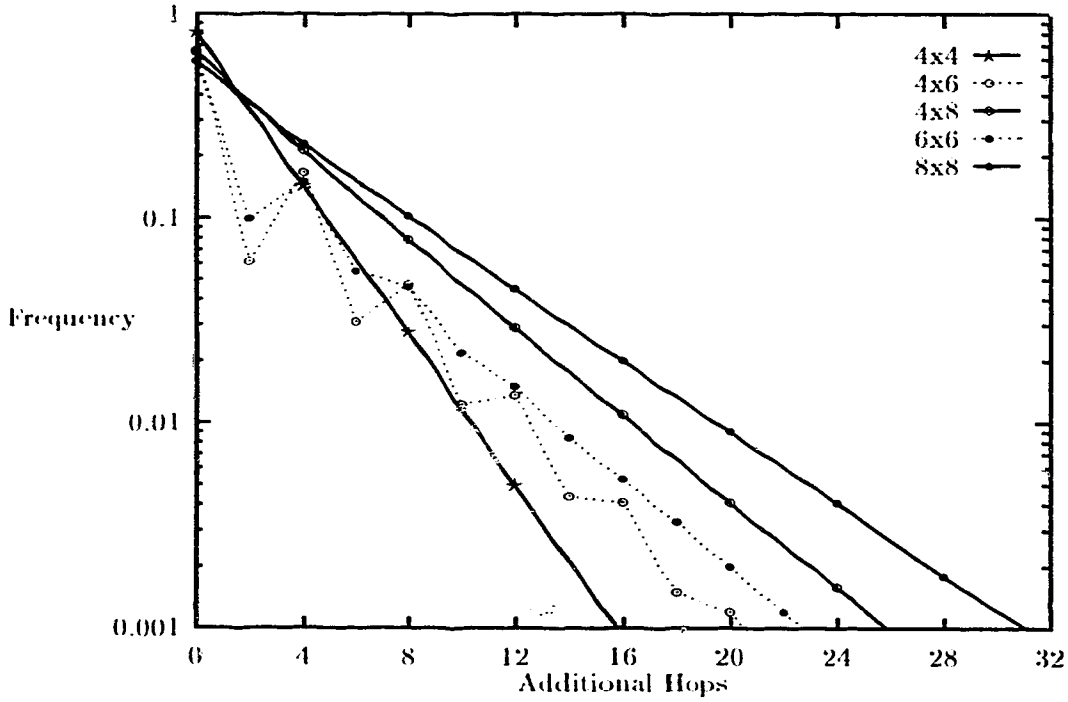


Figure 4.11: Distribution of Additional Hops for CRDP.

4.3 Performance of SRDP and CRDP for Irregular Topologies

In the last two sections, we have presented the operation of SRDP and CRDP as well as their performance under uniform traffic for square and rectangular node arrangements. In this section, we briefly give the performance of the protocols for irregular topologies.

We have presented two derivations of an 8×8 CTRDMN in figure 2.3 ($\mathcal{L}64$) and in Figure 2.4 ($\mathcal{L}64$). As we stated previously, $\mathcal{L}64$ is a way of tuning $\mathcal{L}48$ into a symmetrical configuration but with non-uniform traffic because some of the switches do not generate or absorb traffic.

In $\mathcal{L}48$, we assume uniform traffic so that every switch has the same weight probability of sending and receiving slots/cells. Since the topology is not symmetrical, the resulting routing distribution is not symmetrical either in the sense that the source-destination distance has geographical sense: some switches have smaller average source-destination distances than others.

In $\mathcal{L}64$, since we assume the 16 dummy switches do not generate or absorb any traffic, the absolute source-destination distance is the same for every switch if we do not consider the weight of traffic, but the traffic distribution among 64 switches is different.

The simulations are done for both SRDP and CRDP with the assumption that all the transmission rates are the same.

In figure 4.12, $\mathcal{L}48.slot$ stands for the results obtained for SRDP and $\mathcal{L}48.cell$ stands for the result obtained for CRDP. $\mathcal{L}64.slot$ and $\mathcal{L}64.cell$ are defined similarly. From the figure, we find that introducing 16 dummy switches increases throughput by about 20%. The achieved throughput for SRDP is quite close to that of the 8×8 configuration with SRDP, and so it is for CRDP. This is because by introducing dummy switches, the total number of links increases a lot, from 96 to 128, but the

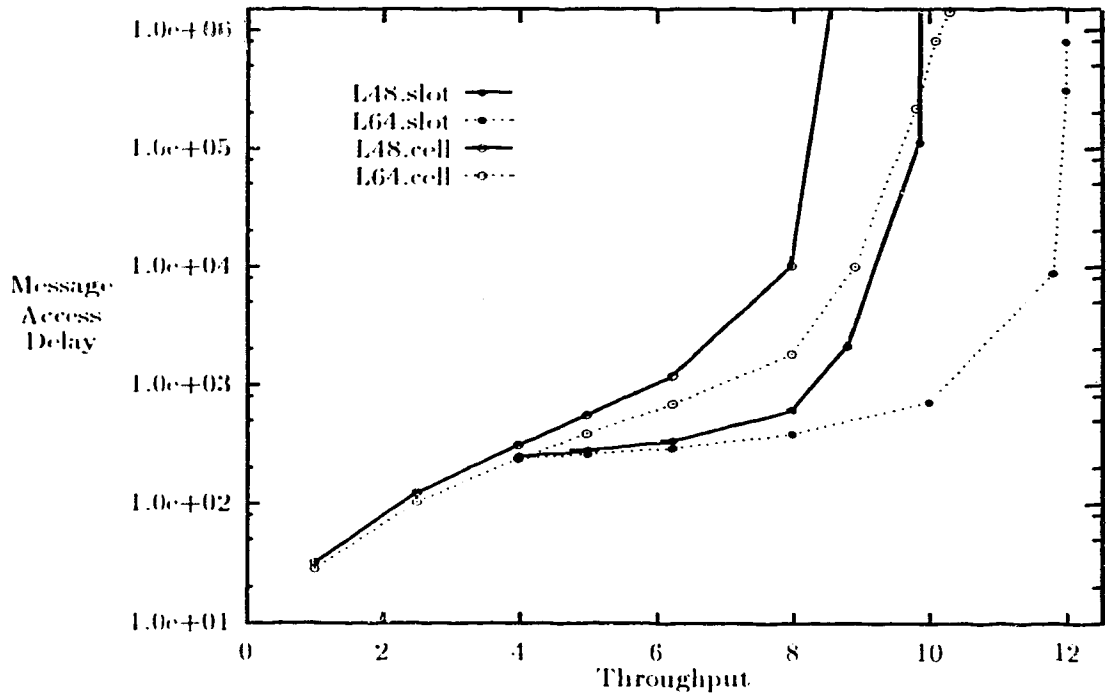


Figure 4.12: Performance of L-shaped network.

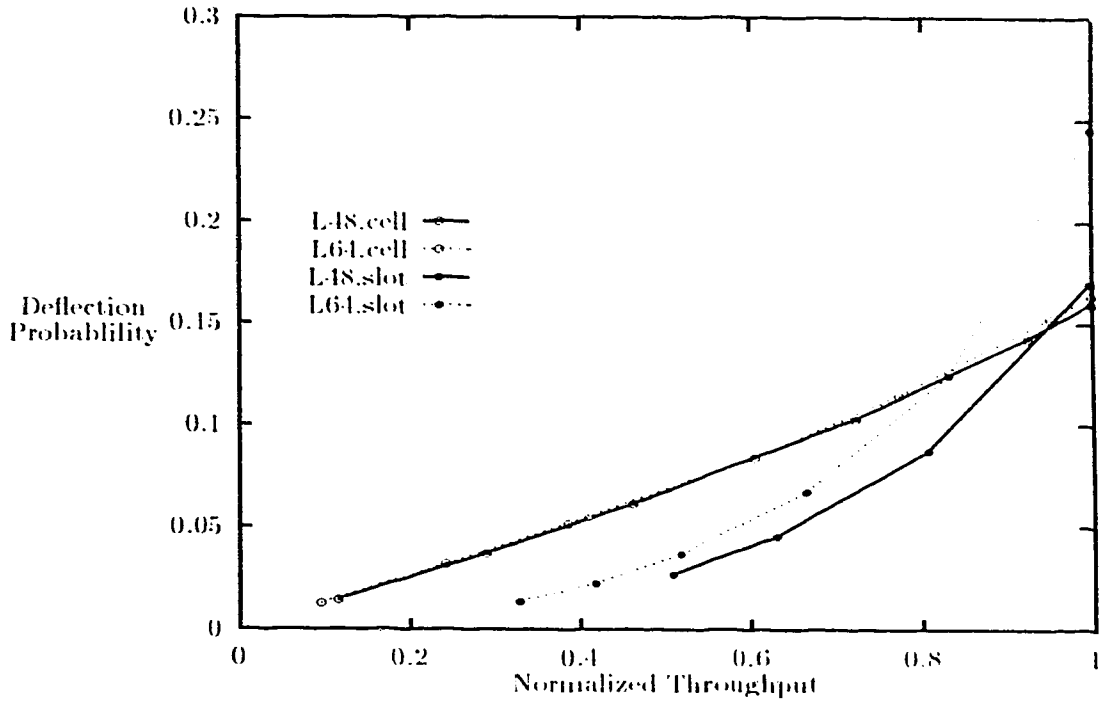


Figure 4.13: Deflection Probability of L-Shaped network.

average source-distance increases relatively little (Refer to table 2.1).

Figures 4.13 to 4.15 present other performance measures for $\mathcal{L}48$ and $\mathcal{L}64$. It is found that the additional throughput gained by adding 16 dummy switches is at the cost of incurring more deflections and increasing the end-to end delay in the network.

4.4 Conclusion

CRDP can be regarded as a non-slotted extension of SRDP. It improves the message access delay under light traffic load and decreases the deflection probability, thus decreasing the average end-to-end delay. The quality of routing is also improved in the sense that more cells reach their destinations by travelling shorter distances (comparing figure 4.5 and 4.11). On the other hand, it trades the global throughput

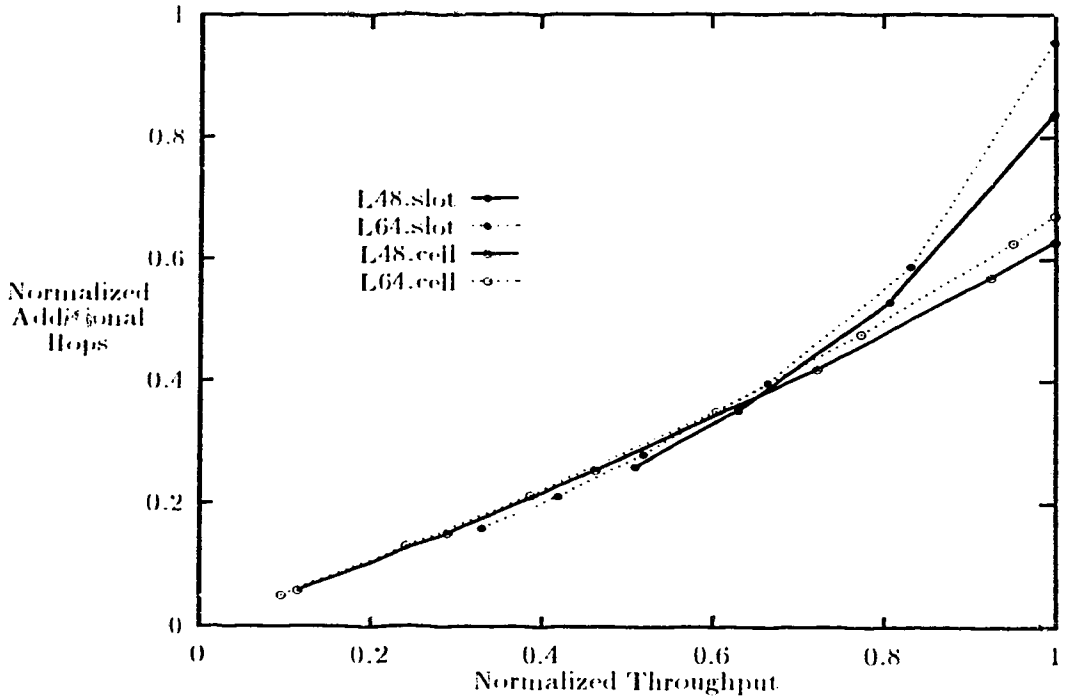


Figure 4.14: Normalized Additional Hops of L-Shaped network.

because the [illegible] can not utilize the whole link bandwidth. It also inherits the drawback of SRDP that the overall delay of a cell has no upper-bound.

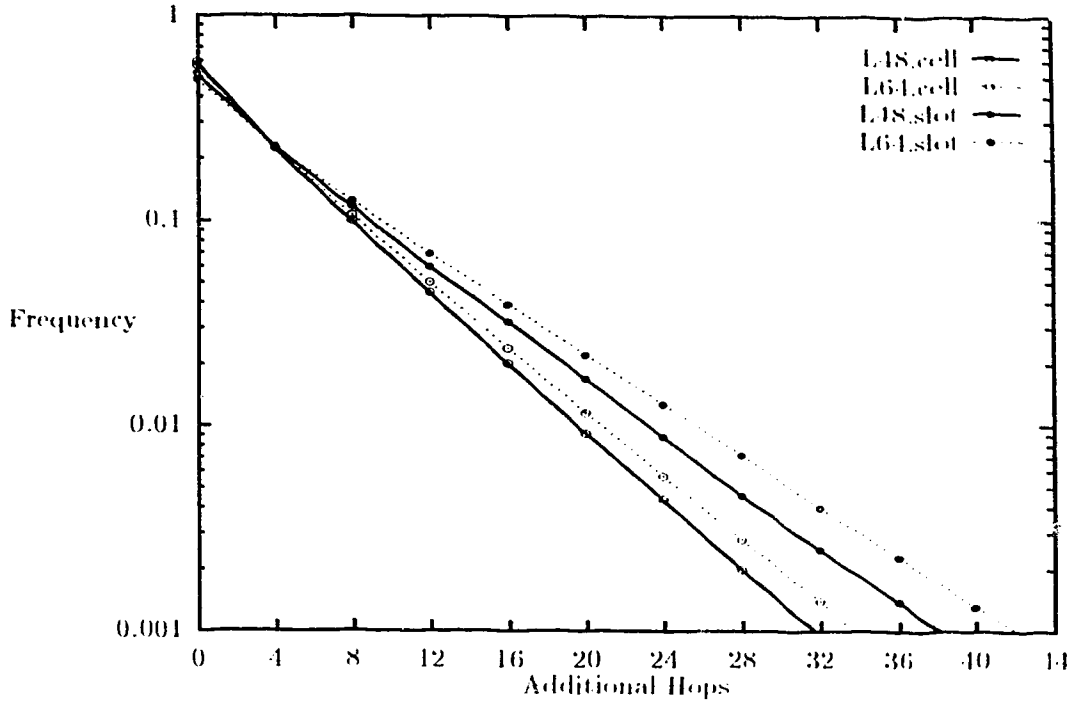


Figure 4.15: Distribution of Additional Hops of L-Shaped network under heavy load.

Chapter 5

A Vertical Deflection Protocol

In chapter 4, two protocols using *Random Deflection* are presented. Deflection in these two protocols takes place at every intermediate switch if there is a necessity. It is **symmetrical** in the sense that a cell/slot may be deflected to either a row ring or a column ring with the same probability. Since multiple paths exist between a pair of source and destination, it is impossible to predict along which route a cell/slot will reach its destination. Therefore, this property fails to guarantee a bounded delay between communicating switches.

In this chapter, we present a new slotted protocol SVDP which stands for *Slotted Vertical Deflection Protocol*. The protocol uses *Vertical Deflection* which means that only column-to-row deflections cause slots to travel a longer path to their destinations. The deflection is an asymmetrical one but ensures a bounded delay between any two switches.

In order to simplify the discussion, we define three traffic patterns:

- **row traffic:** a switch communicates with another one which is in the same row ring as the source.
- **column traffic:** a switch communicates with another one which is in the same column ring as the source.

- **remote traffic**: all other communications.

5.1 Operation of SVDP

SVDP uses a slotted access scheme. The structure of a slot in SVDP is the same as that of SRDP and so is the handling of the *payload status* (PS) flag. Different synchronization methods have been discussed in previous chapters. In SVDP, we adopt the *buffering* method to synchronize slots from different rings, although using of *aligning* method will yield the same performance except for the delay within a node.

5.1.1 Description of Protocol

SVDP is a slotted protocol. A number of non-overlapping slots circulate in each ring in the same way as in SRDP. These slots are not necessarily synchronized, which means that column and row slots arrive at a switch at different moments.

A switch willing to transmit a segment awaits an empty slot. Then if the empty slot is a column one (i.e. coming from the column ring), the switch fills it with payload of **column traffic**, otherwise, the switch fills it with payload of either **row traffic** or **remote traffic**.

Each switch buffers a certain number of bits of every slot passing through it, to decide how the slot should be routed. Assume that an intermediate switch $\boxed{(r,c)}$ receives a full slot from the row ring. Suppose that this slot is addressed to a switch $\boxed{(r_1,c)}$ attached to the same column ring. Switch $\boxed{(r,c)}$ transmits the row slot into the column ring immediately after it completes the transmission of the last column slot and deflects the next column slot to the row ring. Otherwise, the switch just relays the slots independently on both rings.

There may be a delay between the time a switch detects a row slot to be routed to the column ring and the time when this operation can actually take place without

damaging the column slot currently in transit. The delay turns into a gap in the slot stream in the row ring, because while the switch is waiting to insert the row slot into the column ring, it has nothing to relay to the row ring. Consequently, the next row slot will have to be buffered because the deflected column slot is being processed.

In order to synchronize the transmission of row and column slots in the above situations, the *buffering* method introduced in chapter 3 is adopted. Upon deflection, a switch moves the row slot to the RDB and generates an empty slot going into the row ring. When the column slot arrives at the switch, it is buffered in CDB prior to transmission into the row ring. Since a new slot is generated whenever a deflection takes place, to restrict the number of outstanding slots in the network to avoid running out of buffer space, only full slots are buffered and empty slots are discarded.

The collection of the processes that perform the function of the protocol is described in the next section.

5.1.2 Operation of Protocol Processes

The protocol of SVDP can be modeled by four interacting processes of two *receivers* and two *transmitters*. The synchronization of slots from row and column rings is carried out by $\mathcal{SIG}$(DEF) sent from *row receiver* to *column receiver*.

Row Receiver

1. Receive the header of a slot from the row ring.
2. A slot is discarded if it is empty. Otherwise, if it is addressed to one of the stations attached to the switch, the payload of the slot is copied, and the slot is discarded.
3. Otherwise, if the slot is a full slot but it is not addressed to the switch:
   (a) if the destination of the slot is attached to the same column ring as the switch does, the slot is shifted to RDB and a $\mathcal{SIG}$(DEF) is sent to *column*

receiver.

(b) otherwise, the slot is shifted into RB.

Column Receiver

1. Receive the header of a slot from the column ring.
2. A slot is discarded if it is empty. Otherwise, if it is addressed to one of the stations attached to the switch, the payload of the slot is copied, and the slot is discarded.
3. Otherwise, if the slot is a full slot but it is not addressed to the switch:
   (a) if a $\mathcal{SIG}$(DEF) is received, the slot is shifted to CDB;
   (b) otherwise, the slot is shifted to CB.

Row Transmitter

1. If CDB is not empty, transmit the slot in CDB via the row port.
2. Otherwise, if RB is not empty, transmit the slot in RB via the row port.
3. If both buffers are empty, a new slot is generated, filled with payload bits of row and rc[illegible]ffic and transmitted via the row port.

Column Transmitter

1. If RDB is not empty, transmit the slot in RDB via the column port.
2. Otherwise, if CB is not empty, transmit the slot in CB via the row port.
3. If both buffers are empty, a new slot is generated, filled with payload bits of column traffic and transmitted via the row port.

The SMURPH source code modelling the processes is provided in appendix A.2.

5.1.3 Length of Deflection Buffers

During synchronization, every slot is delayed for a period of time. The duration of buffering within each buffers, two deflection and two delaying buffers, is studied in this section. We assume that all switches use the same transmission rate.

When the network starts operation, all the buffers are cleared. The network starts by inserting empty slots via the output ports. The minimum requirement of the maximum capacity of RDB and CDB is two slots. It can be illustrated with the help of figure 5.1 which is similar to figure 3.3.

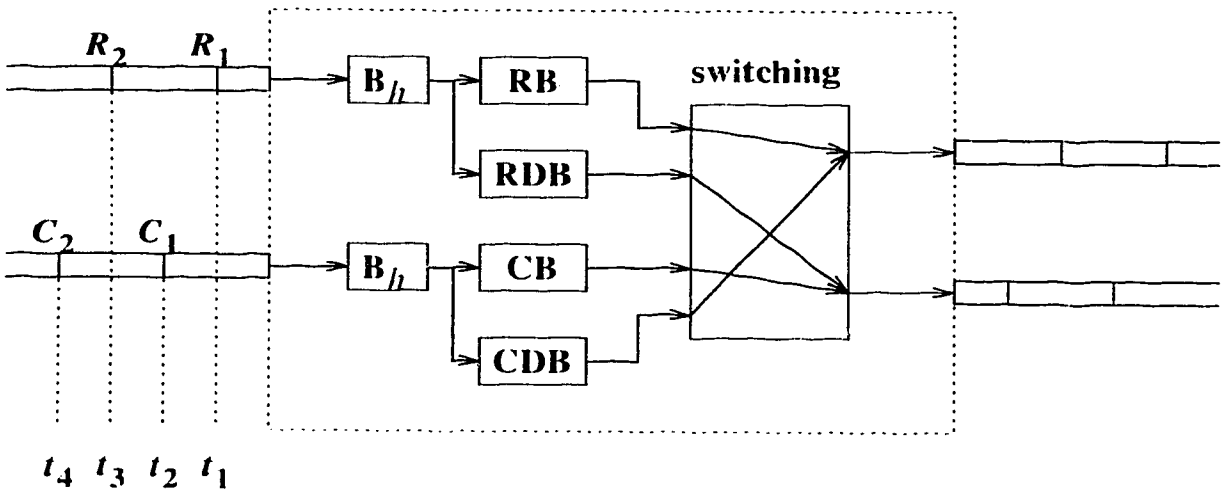


Figure 5.1: Operation of SVDP.

In figure 5.1, R_1 and R_2, and C_1 and C_2 are two consecutive slots coming from the row ring and the column ring respectively. t_i stands for the arriving time of each slots. For simplicity, t_i is measured by bits. We make the following assumptions:

- R_1 is the first row slot that is to be routed to the column ring at the switch. The general case of row slots is therefore same as R_2.

- When R_1 is ready for routing, RDB and CDB are empty. And RB and CB contain δ_r and δ_c bits respectively.
- The difference between t_{i+1} and t_i is strictly less than one slot.
- All of the slots are full slots but not addressed to the switch.

It is important to know that δ_r and δ_c are shorter than a slot. In a synchronous network, before any deflection take place, the row ring and the column ring at a switch are decoupled. The length of RB (CB) is equal to the remaining bits of a slot that being transmitted via the row (column) port when the first incoming row slot is ready for routing; the minimum requirement of the length of RB (CB) is shorter than a slot and satisfies equation 3.1. Since empty (as well as destined) slots are discarded before RB and CB, it is possible that when R_1 is ready for routing, RB or CB or both of them are empty but the output ports are transmitting slots. In this case, δ_r and δ_c are extended to stand for the length of the remaining bits of the row and column slots being transmitted respectively.

Delay of R_1 in RDB

R_1 is shifted to RDB after the routing information is extracted. R_1 is transmitted immediately after the transmission of the δ_r bits which is currently stored in CB — the remainder of the predecessor of C_1. Note that if R_1 were not routed to the column port, the *column transmitter* would transmit C_1 instead of R_1, therefore, the time R_1 is delayed in RDB is equal to $t_2 - t_1 + \delta_r$.

Delay in CDB

When R_1 is shifted into RDB, there are δ_r bits in RB waiting to be transmitted. There are two possibilities:

1. If $\delta_r \geq (t_2 - t_1)$, which means when C_1 is ready for routing, there is still a portion of the predecessor of R_1 being transmitted. In this case, C_1 is shifted

into CDB first, and is transmitted immediately after the transmission of the remaining $\delta_r - (t_2 - t_1)$ bits. Therefore, the delay of C_1 in CDB is $\delta_r - (t_2 - t_1)$.

2. Otherwise, $\delta_r < (t_2 - t_1)$, which indicates that before C_1 is shifted into CDB, the *row transmitter* has started the transmission of a locally generated slot (possibly empty). When C_1 is shifted into CDB, only $((t_2 - t_1) - \delta_r)$ of the new generated slot has been transmitted, therefore, C_1 has to be delayed for $t_3 - t_2 + \delta_r$ bits before its transmission commences. Since the difference between t_3 and t_1 is exactly one slot long, $t_3 - t_2 + \delta_r$ is thus equal to $\delta_r - (t_2 - t_1)$.

So, the delay of C_1 in CDB is $\delta_r - (t_2 - t_1)$.

Delay of R_2 and C_2

If R_2 is routed to the column port, since the transmission of R_1 has not finished, R_2 has to be buffered in RDB. When R_2 is ready for routing, the length of RDB — the portion of R_1 that has not been transmitted is $t_2 - t_1 + \delta_r$. Therefore, R_2 is also to be delayed for the same amount of time as R_1.

On the other hand, if R_2 is routed to the row port and when it is ready for routing, the transmission of C_1 has not finished. Since C_1 is delayed for $t_3 - t_2 + \delta_r$ bits, R_2 is delayed in RB for δ_r.

The action for C_2 is the same as for C_1 and is not repeated here.

Conclusion

From the above analysis, we draw a conclusion that the maximum delay in the deflection buffers — RDB and CDB can be minimized to less than two slots.

The above description of routing processes and length of delay is based on the assumption of the minimum buffering requirement. By increasing the delay in CB by numbers of slots, some slots from the column rings can be shifted into CB instead of CDB during a deflection, which results in a lower deflection probability.

5.1.4 *Filling Rule* of SVDP

When using *Vertical Deflection*, every deflection is triggered by a row slot that is routed to a column ring. It deflects a column slot to the row ring and consequently the column slot causes another deflection to go back to the right column ring if it is a full slot. It is apparent that if a remote slot is sent in a wrong column, i.e. the column of the destination is different from the column ring the slot is transmitted, there is a chance that the slot will be running along the column *forever* for it is not allowed to trigger a deflection to steer itself from the column ring onto the correct row ring.

To avoid the above happening, a restriction is applied when filling a slot. When a *column transmitter* generates an empty slot, it can only fill the slot with payload of column traffic. On the other hand, a *row transmitter* can fill a new generated slot with a payload of either row or remote traffic. It is useless for a *row transmitter* to fill an empty slot with column traffic payload for it is just a waste of bandwidth.

5.1.5 Asymmetry Among Traffic Patterns

Unlike *Random Deflection* used in SRDP and CRDP, *Vertical Deflection* causes a column slot to traverse a whole row ring before going back to the right column ring again. The protocol thus adds more load onto row rings even though only row and remote traffic is initiated on the row rings. As a result, row rings are saturated much earlier than column rings. For square and rectangular node arrangement under uniform traffic, the asymmetry is visible and is studied in the next section by performance measures of discriminating performance and slot utilizations.

5.1.6 Asynchronism

In SVDP, problems caused by the asynchronism of distributed transmission clock rates and link irregularity also exist. These problems have already been discussed earlier.

and can be solved by the similar approaches, such as using a master clock, increasing buffer capacities and using excess bits. As stated earlier, the routing performance of the protocol does not change with the master clock approach or appending bits.

5.2 Uniform Performance of SVDP

The performance of SVDP is presented below for square and rectangular node arrangements under uniform traffic. Unlike either SRDP or CRDP, SVDP is not able to take advantage of topologies of unidirectional rings, such as a shorter *ASD* brought by CTRDMN. The reasons are:

- For every full slot carrying column traffic, it is first transmitted into the column ring no matter how far (or near) the destination is. Similarly, every full slot of non-column traffic is first transmitted into the row ring anyway.
- Every deflection is only based on the knowledge of the index of a column ring.
- If a full slot is detoured from a column ring to a row ring, the slot traverses the whole row ring and goes back to the right column ring, regardless the fact that the distance traversed by the slot can be reduced by a row to column followed by a column-to-row deflection at some nearby switches.

In the simulations conducted, we assume nodes are arranged into a CODMN. In fact, the performance of SVDP remains the same for all such unidirectional meshes. In the figures presented in this section, 4×9 indicates a configuration that there are 4 switches in a row ring and 9 switches in a column ring.

Besides the measures introduced earlier, we measure the deflection probability of SVDP by dividing the number of deflected full column slots by the number of total full column slots. The column slots here include slots with a payload of either column traffic or remote traffic, but never row traffic.

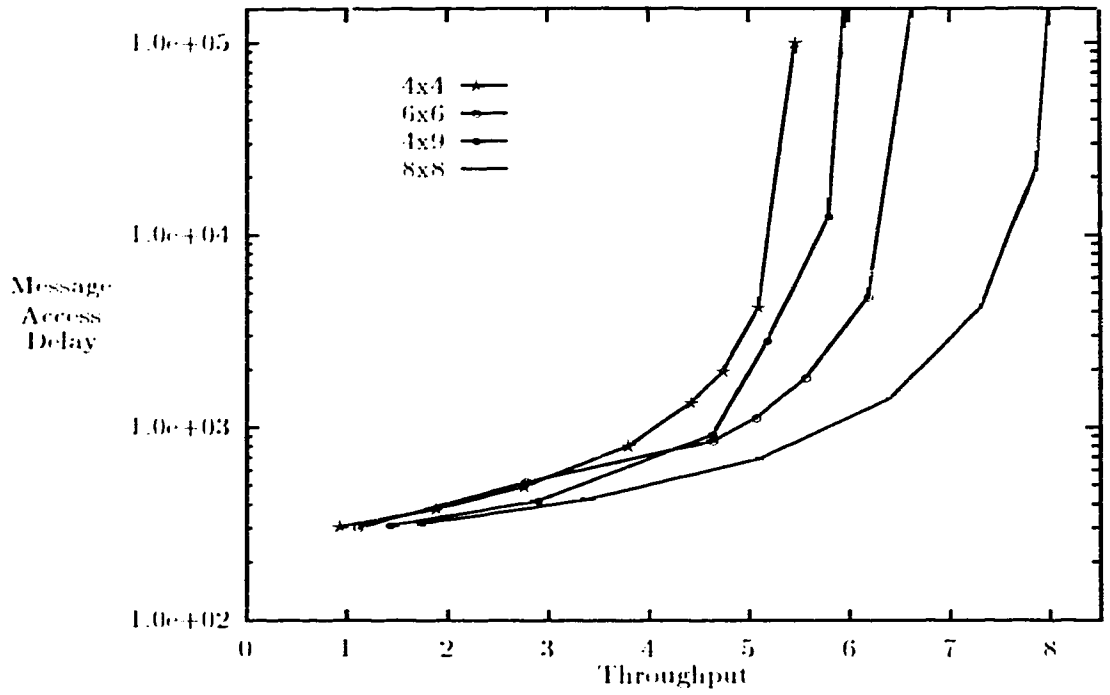


Figure 5.2: Performance of SVDP.

One more measure introduced to study the asymmetry of SVDP is the utilization of slots. The column slot utilization is measured by dividing the number of full slots by the number of total slots observed at the column rings. The row slot utilization is measured in a similar way.

5.2.1 Global Throughput and Message Access Delay

Figure 5.2 shows the throughput and message access delay performance under uniform traffic. Again, mesh networks prove to have a very useful feature that increasing the dimension of networks will increase the throughput. However, an exception is that 4×9 yields less throughput though this arrangement has more rings than 6×6. This is because: first, 4×9 has larger $\mathcal{ASD}_{W_u}$ and second, the amount of remote and column traffic of 4×9 is larger than 6×6, which potentially increases the number

Topology	4×4	6×6	4×8	8×8
Throughput ($\mathcal{T}$)	5.48	6.69	6.4	8.06
$\mathcal{T}/\sqrt{\mathcal{N}}$	1.37	1.12	1.02	1.01

Table 5.1: The maximum throughput of SVDP with respect to $\sqrt{\mathcal{N}}$.

of deflections under high load as shown in Figure 5.8. Table 5.1 describes the relationship between the uniform throughput ($\mathcal{T}$) and $\sqrt{\mathcal{N}}$. Let $c = \frac{\mathcal{T}}{\sqrt{\mathcal{N}}}$, the table shows that c decreases as $\mathcal{N}$ increases.

The throughput of SVDP is smaller than that of SRDP and CRDP for same node arrangements (consider table 5.1, 4.1 and 4.3). There are two reasons for this. The more important one is that *Random Deflection* used in SRDP and CRDP causes the penalty of deflection not to exceed 4 more hops, but *Vertical Deflection* used by SVDP causes a detoured column slot to travel a whole row ring before coming back to the right column. The difference does not decrease the throughput significantly of SVDP for 4×4, but does so for other node arrangements. As an indirect consequence, the slots can not be 100% utilized even when the maximum throughput is achieved. A second reason is that the impact of *Random Deflection* is local in the sense that the detoured slot/packet is capable of getting on the shortest path at the next switch, so SRDP and CRDP can take topological advantage such as shorter diameter in meshes with adjacent opposite-directed rings. But the impact of *Vertical Deflection* is global. Every detoured slot has only one correcting site which is the switch where the slot was previously deflected. Therefore, it can not take advantage of the topologies as SRDP and CRDP do.

Discriminating Performance

Figure 5.3 illustrates the asymmetry among different traffic patterns. In the simulation, we assume two message queues, one for column traffic and one for row and

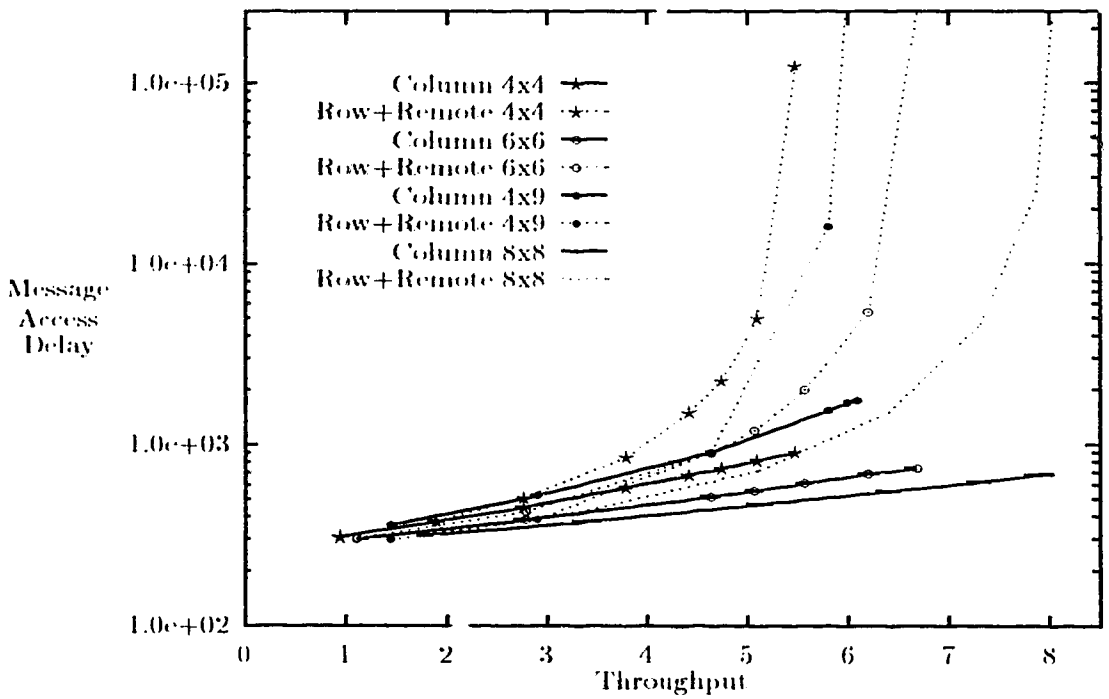


Figure 5.3: Discriminating Performance of SVDP.

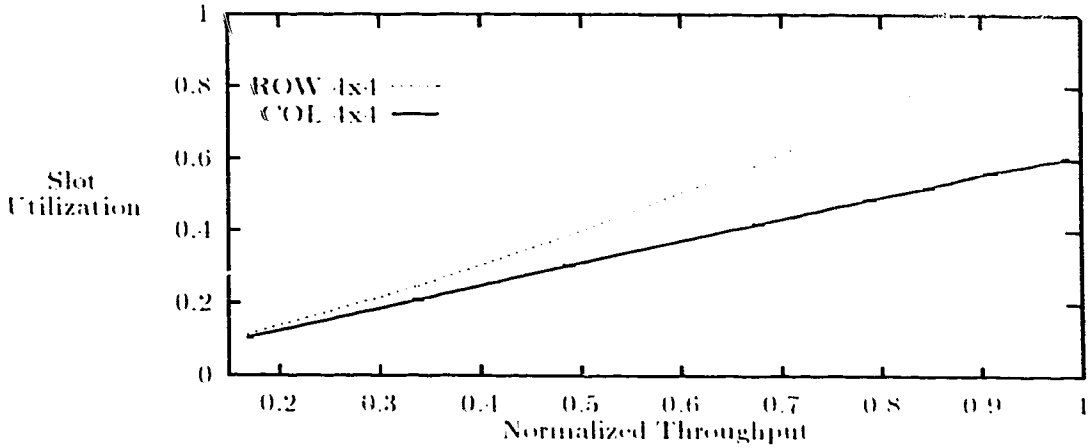


Figure 5.4: Slot Utilization of SVDP 4 × 4.

remote traffic. Figure 5.3 plots message access delay for messages of column traffic and non-column traffic respectively.

The curves indicate that row rings saturate much earlier than column rings, since the column message access delay is much lower than its counterpart. For light load, the two are close, because there are fewer deflections to detour full column slots. When the load increases, the message access delay of non-column traffic increases much faster while the message access delay of column traffic still maintains at comparatively low level.

Slot Utilization

Figures 5.4 to 5.7 plot the slot utilization as a function of normalized throughput. In all three figures, the column slot utilization increases almost linearly with the normalized throughput, while the row slot utilization increases much faster. When the network achieves the maximum throughput, almost all the row slots are full, but a large number of column slots are still observed empty. In contrast, in SRDP, almost all the slots are full for the maximum load.

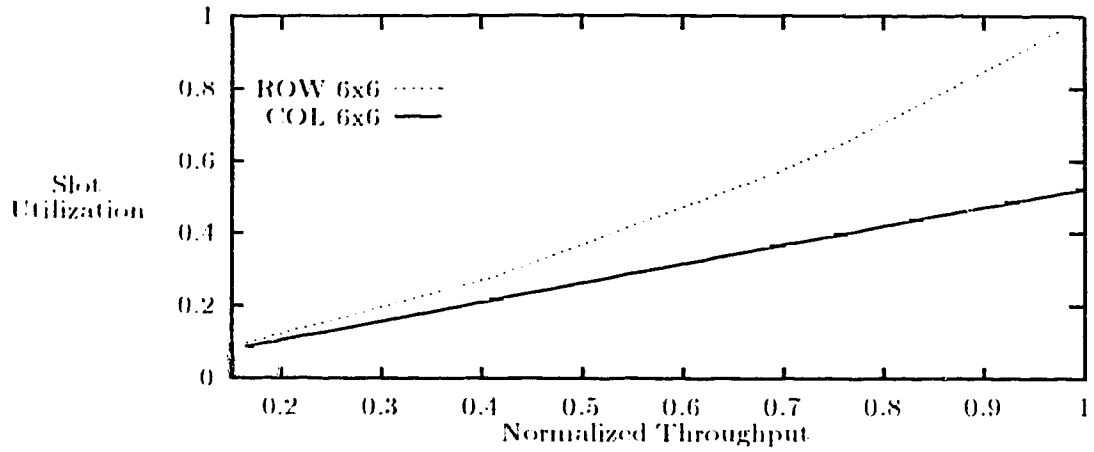


Figure 5.5: Slot Utilization of SVDP 6×6.

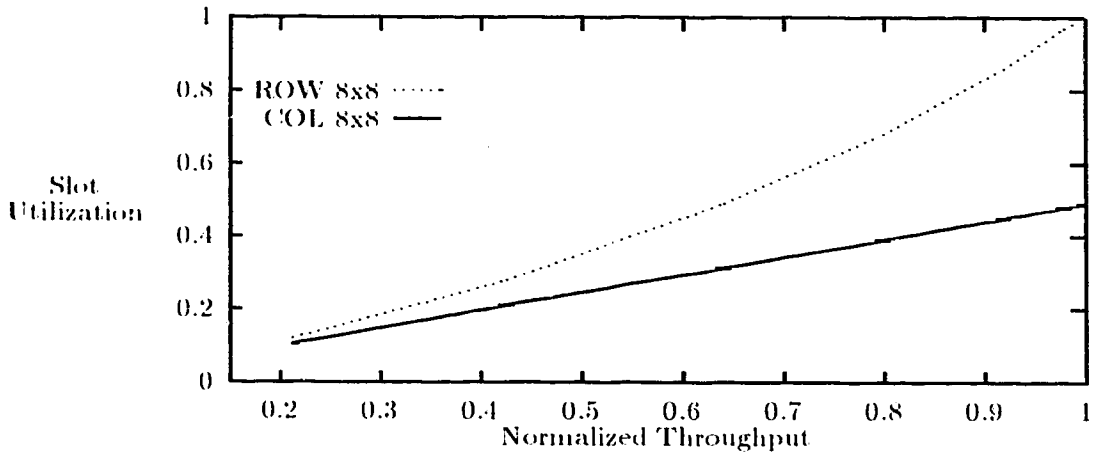


Figure 5.6: Slot Utilization of SVDP 8×8.

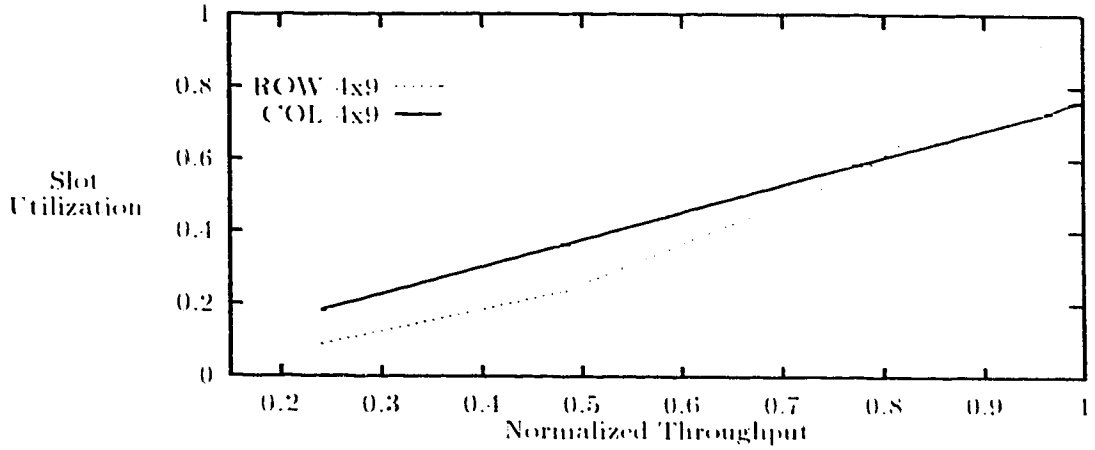


Figure 5.7: Slot Utilization of SVDP 4×9.

For square topologies, the column slot utilization is always lower than the row slot utilization under any load. But for 4×9, with 9 row rings and 4 column rings, for light load is low, since there is more column and remote traffic, the column utilization is a little higher. While the load increases, more column and remote traffic implies more deflections so that the utilization of row slots increases dramatically and exceeds the column slot utilization

5.2.2 Deflection Probability

The deflection probability is plotted in figure 5.8. There are very few deflections for light load because most of the column slots deflected are empty. As the load increases, more column slots become full and the deflection probability increases. The deflection probabilities of 4×4 and 6×6 are very close with the latter a little lower. But 4×9 has the highest probability. This is because 4×9 has the largest amount of remote and column traffic that potentially increases the chances of deflection. Besides, the column distance between a source and a destination of a column or a remote traffic in 4×9 is larger than that of 4×4 and 6×6, which again increases the chances of a

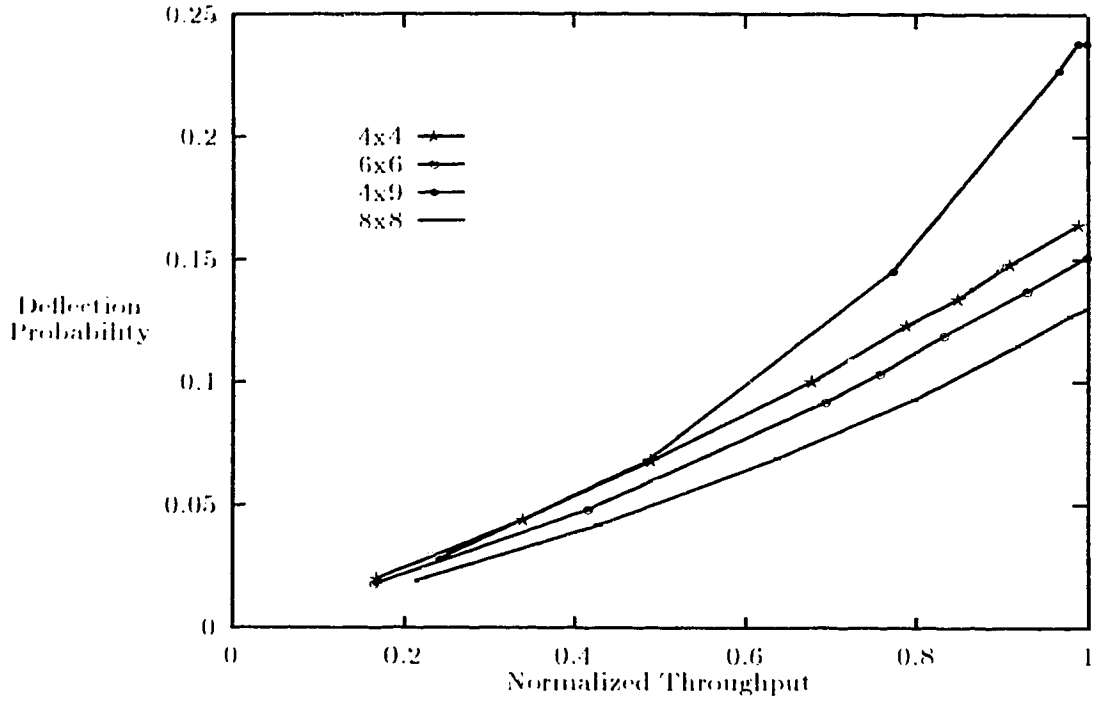


Figure 5.8: Deflection Probability of SVDP.

slot being detoured along its route to the destination.

Unlike SRDP and CRDP, the deflection probability is only decided by the amount of combined column and remote traffic and the column distance to the destination.

5.2.3 Additional Hops

From figure 5.9, when the load is heavy, it is found that the 4×4 arrangement has the fewest normalized additional hops and 4×9 has the most. In a 4×4 network, a deflection causes fewer penalty hops (2 less) than 6×6, although the deflection probability is slightly higher. But for 4×9, the deflection probability is significantly higher than 6×6 and 4×4, so that on average slots have to travel more hops (0.69×5.657) than 4×4 (0.32×3.2) and 6×6 (0.45×5.14).

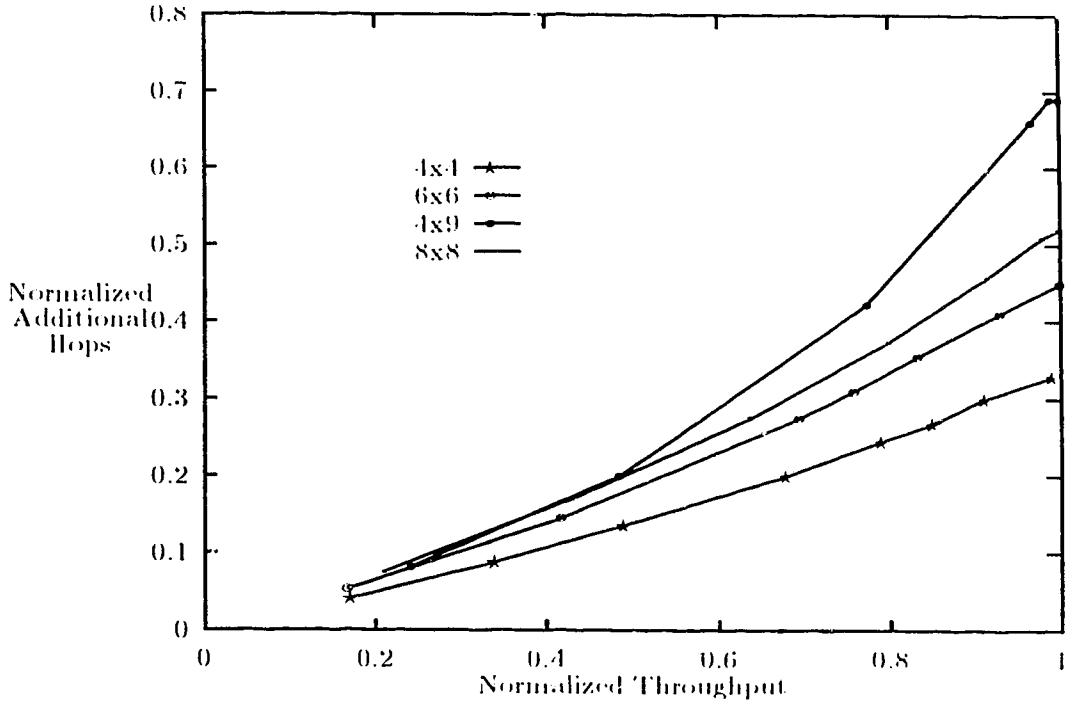


Figure 5.9: Additional Hops performance of SVDP.

5.3 Non-Uniform Performance of SVDP

As we described in previous sections, a major weakness of SVDP is the asymmetry existing among the different traffic patters. The asymmetry can be improved by a skewed traffic distribution or a rectangular node arrangement. The simulation results presented below describe the performance of SVDP under non-uniform traffic. The weight of three different traffic patterns is specified as: for any topology, the remote traffic weights 20%; the row and column traffic shares the rest 80% traffic and it is uniformly distributed among nodes. For example, for a 4×4 arrangement, suppose there is a message generator at a switch, for every message it generates, the message is destined to 9 remote switches with a 20% probability, to the 3 row switches and to the 3 column switches with an even 40% chance respectively.

The purpose of such a skewed traffic distribution is to investigate the performance of SVDP in the presence of communities-of-interest as well as the asymmetry among different traffic patterns under such a skewed distribution.

5.3.1 Global Throughput and Message Access Delay

Figure 5.10 shows that in the presence of communities-of-interest traffic distribution, mesh networks exhibit better performance. The throughput of SVDP under skewed traffic is much larger than that under uniform traffic. As an extreme, it is not difficult to imagine that when all the traffic only has local significance, i.e. either local row or local column, the combined throughput increases to the sum of the maximum throughput of all the row and column rings.

Figure 5.11 presents the discriminating message access delay performance of SVDP under skewed traffic. It is noticed that for 4×9 configuration, the non-column message access delay is lower than than column message access delay for most loads.

5.3.2 Slot Utilization

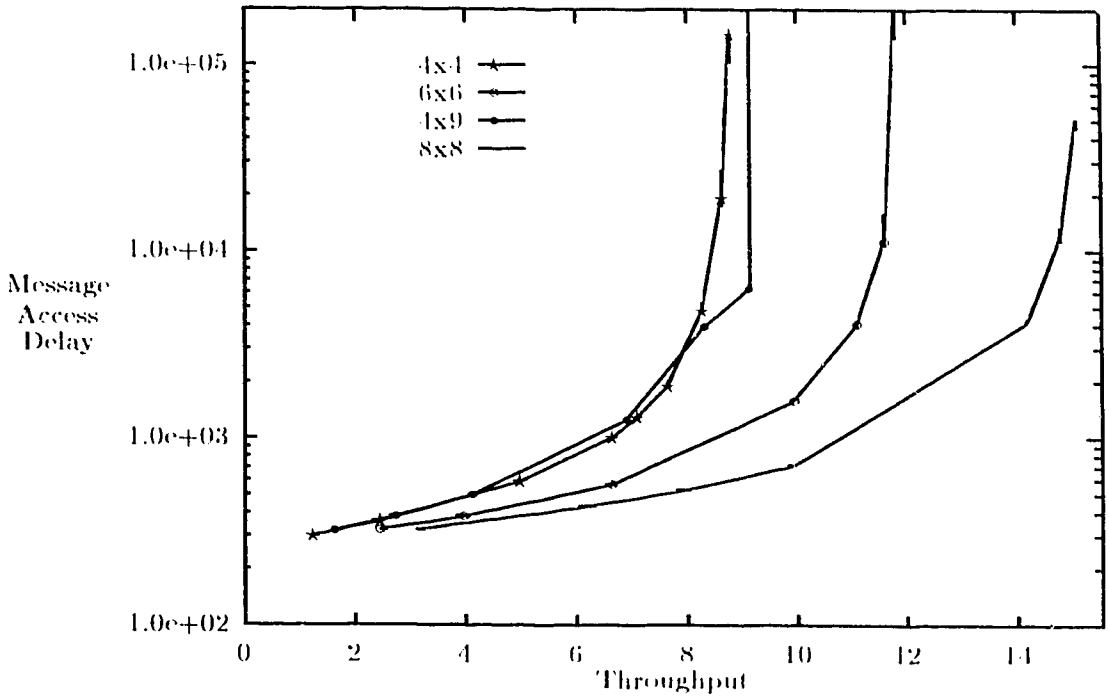


Figure 5.10: Skewed Performance of SVDP.

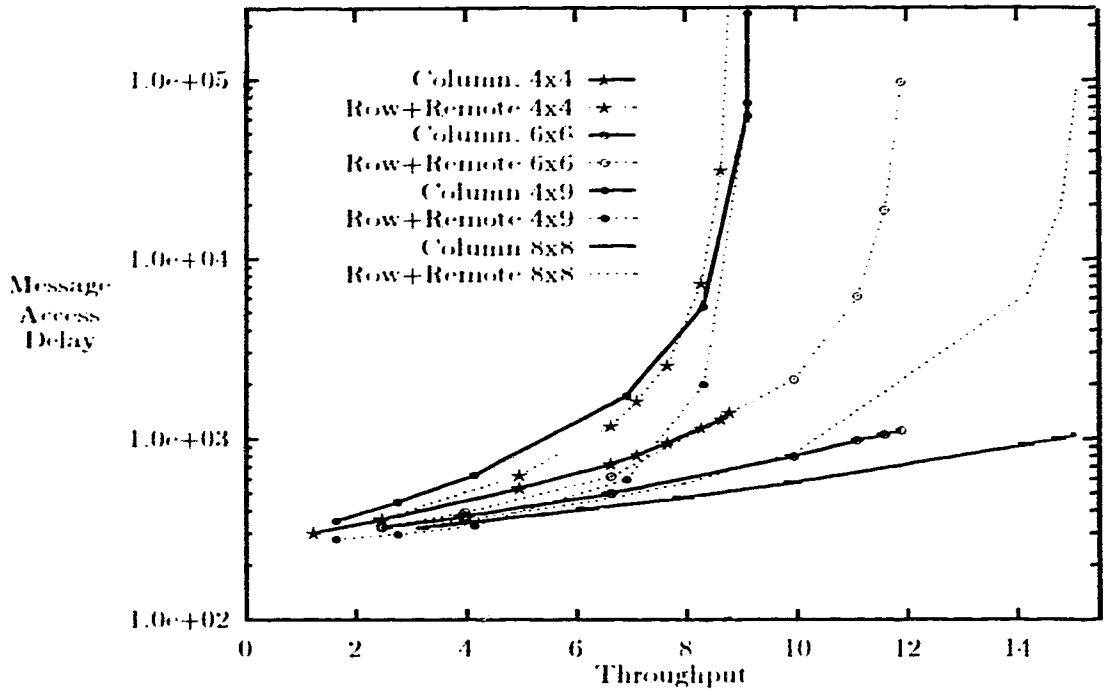


Figure 5.11: Discriminating Performance of SVDP.

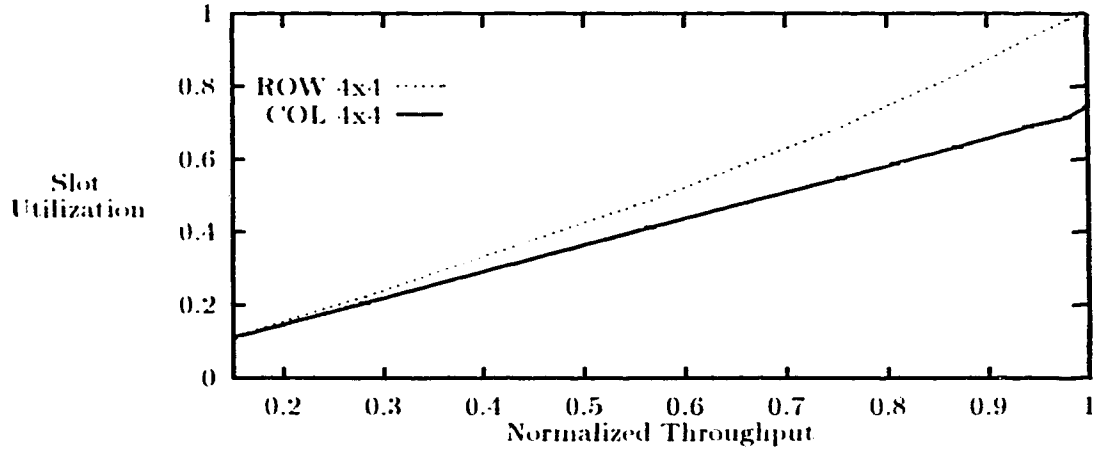


Figure 5.12: Slot Utilization of SVDP under skewed traffic, 4×4.

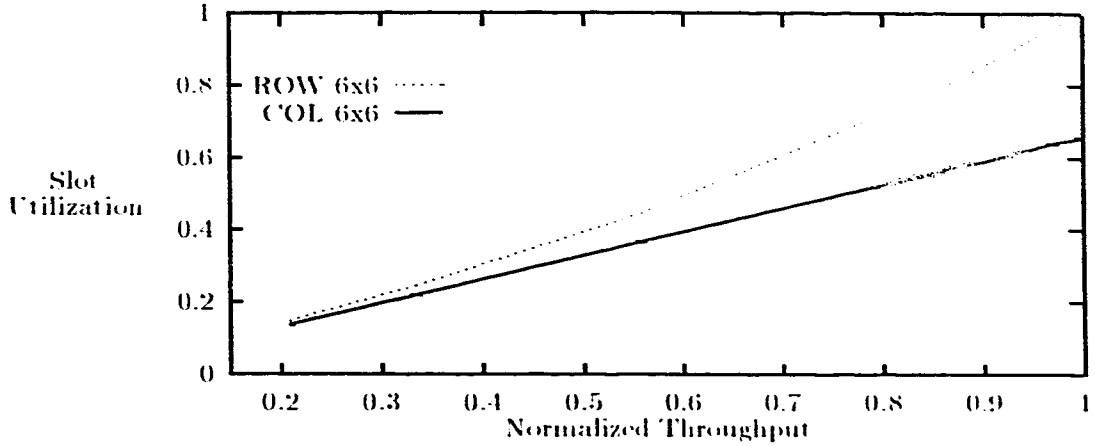


Figure 5.13: Slot Utilization of SVDP under skewed traffic, 6×6.

The slot utilization of SVDP under the skewed traffic distribution, as shown in figures 5.12, 5.13, 5.14 and 5.15, is similar to the uniform traffic case: the increment of column slot utilization is almost linear and the utilization of row slots increases faster, but the difference between utilization of row and column slots is much less significant especially when the maximum throughput is achieved.

Figure 5.15 shows that under such a skewed traffic, for a particular configuration such as 4×9, by increasing the number of row rings, it is possible to improve the balance between the load on row and column rings. In figure 5.15, the utilization of row slots is mostly less than the utilization of column slots. When the maximum throughput is achieved, almost all the slots – row slots and column slots in the network are utilized.

5.3.3 Deflection Probability and Additional Hops

The deflection probabilities shown in figure 5.16 are smaller than those under uniform traffic since there is less remote traffic that triggers deflections. The deflection probability of 4×9 is larger than the other two for the same reason as in the uniform

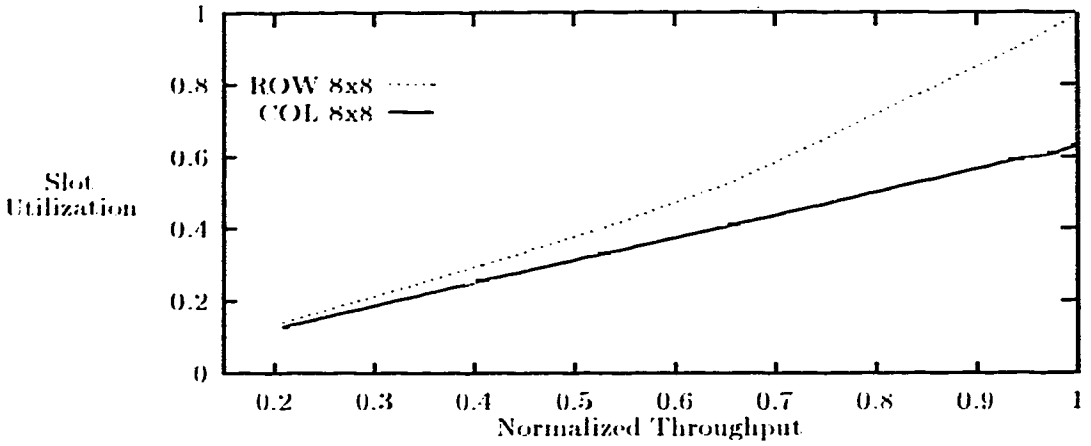


Figure 5.14: Slot Utilization of SVDP under skewed traffic, 8×8.

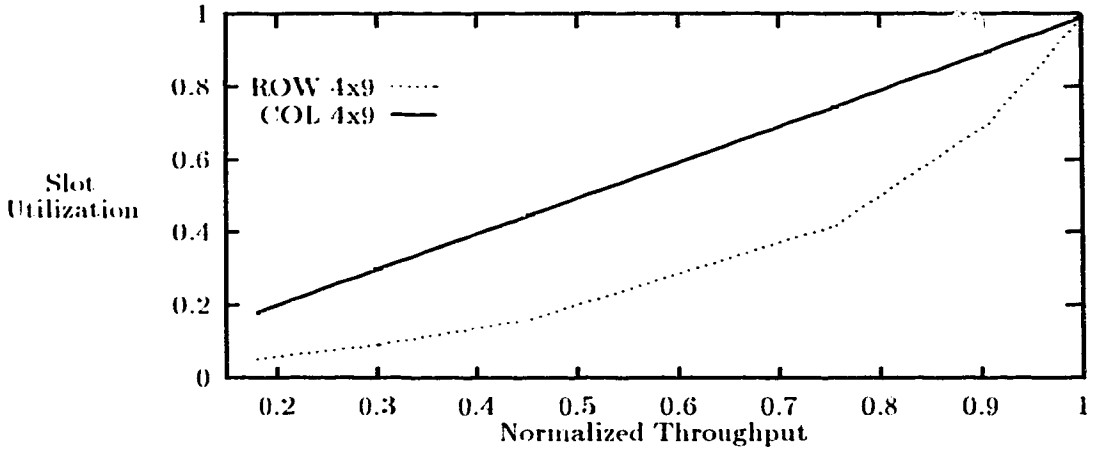


Figure 5.15: Slot Utilization of SVDP under skewed traffic, 4×9.

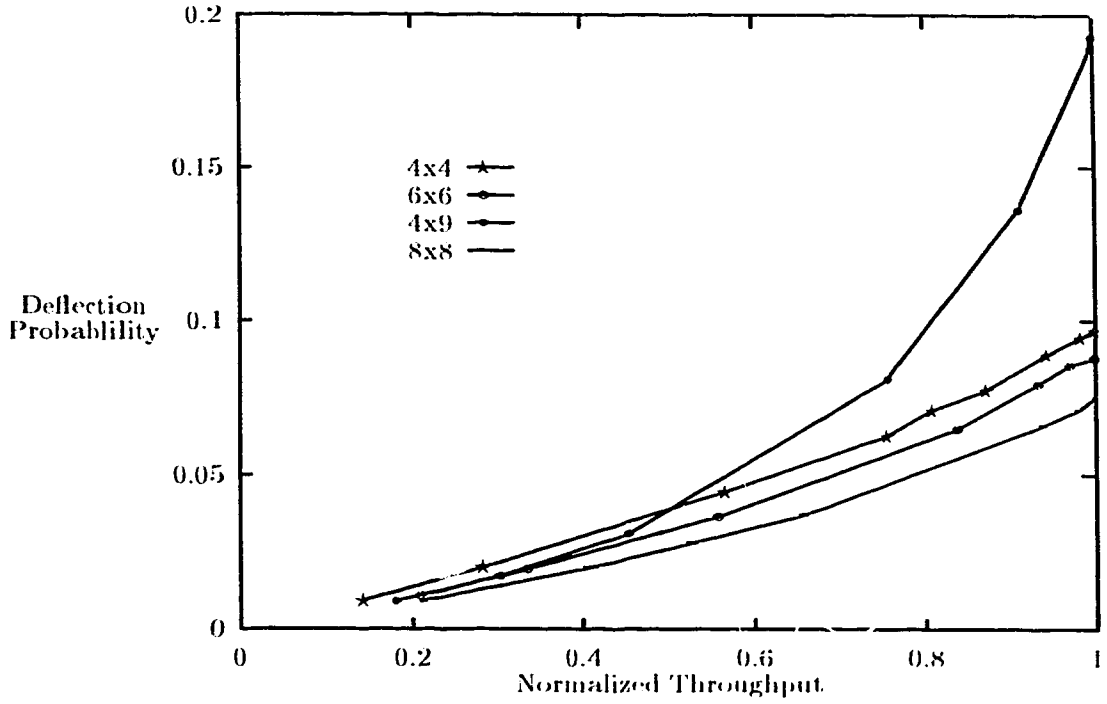


Figure 5.16: Deflection Probability of SVDP under skewed traffic.

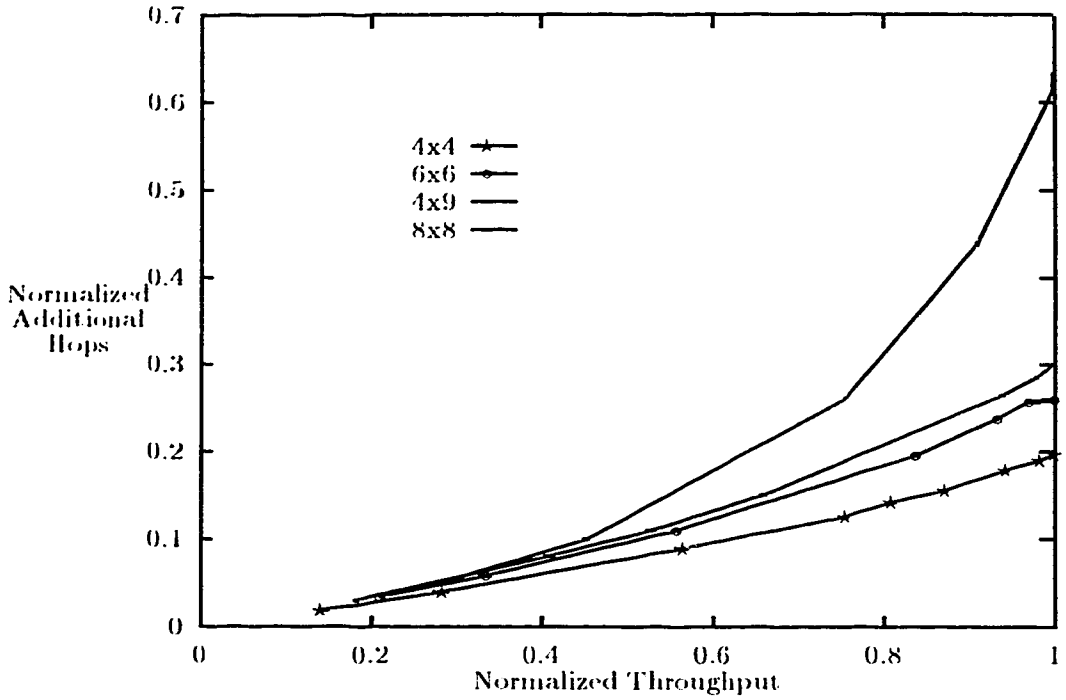


Figure 5.17: Average Additional Hops of SVDP under skewed traffic.

traffic case. On average, for these arrangements, the reduction of the deflection probability brought by the skewed traffic is about 5 percentage points.

Considering the normalized additional hops presented in figure 5.17, it is about 0.1 less than for the uniform traffic case for each configuration.

5.3.4 Conclusion

From the simulation results, we find that, under a skewed traffic distribution that favors local column and row traffic, the performance of SVDP is enhanced in the sense that the global throughput increases while the deflection probability drops and the balance among traffic patterns is improved.

5.4 Conclusion

This chapter proposes a new protocol of two-connected mesh networks. Compared to SRDP, the strength of the protocol includes:

1. The deflection strategy used by the protocol ensures a maximum distance between any pair of source and destination a slot traverse. Therefore, the protocol can be adopted for a real network.

2. The control of routing is much simpler. In SRDP (as well as CRDP), selection procedure is very important during routing to avoid deflection prejudice that causes deadlock and livelock[Max91]. In SVDP, the direction of deflection is predetermined so that the overhead related to performing selection is eliminated.

3. Routing of SVDP requires much less information than SRDP and CRDP. It is very useful when there is constant changes in the network.

On the other hand, SVDP introduces asymmetry among different traffic patterns. But the flaw can be improved by skewed traffic or adjusted topologies.

The performance of SVDP may also be enhanced by increasing buffer capacities at the switches, such as increasing the capacities of CB. Thus a switch will attempt to deflect with higher priority the column slots that are empty (e.g. by delaying the shifting of a full column slot to CBB). Inspecting figures illustrating slot utilizations (figure 5.4, 5.5, 5.6, 5.12, 5.13, and 5.14) tells that a large amount of column slots are empty, and deflecting these slots costs nothing but increasing the throughput.

Chapter 6

Summary

In the thesis, we have studied deflection routing protocols for two-connected mesh networks.

Two kinds of deflection strategies are compared: *Random Deflection* and *Vertical Deflection*. *Random Deflection* is distance-based. Packets are routed to their preferred ports which minimize the distances to their destinations. Each switch decides where to route a packet basing its decision solely on the relative location of the destination of the packet and possibly on contention for the output ports, but irrespective of the ring from which the packet is received. It is completely unknown, at the moment a packet is inserted to the network, which route a packet will take to reach its destination, and when and how the packet may be deflected at intermediate switches.

Vertical Deflection, on the other hand, is direction- or position-based. A packet is routed from one row ring to a column ring only at the switch where the two rings intersect. At other switches, a packet is routed to the same ring from which it is received. Routing depends more on the topologies than on the distances. Although it is hard to predict a deterministic route of a remote or a column packet, it is possible to know the column ring which the packet will *for sure* traverse, and the row rings which the packet will *probably* go through.

We also described synchronization techniques in the thesis. Synchronization is

used to improve the routing quality by getting more information of incoming packets from both rings. For slotted rings, *aligning* and *buffering* techniques are discussed. A method of appending excess bits is also studied for adjusting the *slot rate* in a slotted network with non-zero clock tolerance. For a non-slotted network where only equal size packets are used, it is indicated in the thesis that *quasi-blocking* works more efficiently than *non-blocking*[1].

Three protocols using different deflection and synchronization approaches are presented in the thesis. They are summarized as follows:

- SRDP is a slotted protocol that uses *Random Deflection* and *aligning* technique to synchronize incoming packets. The protocol also takes into account different clock rates at different switches.

- CRDP is a non-slotted protocol. The deflection and synchronization techniques used by CRDP are *Random Deflection* and *quasi-blocking* respectively.

- SVDP is a slotted protocol that employs *Vertical Deflection* and *buffering* synchronization.

In fact, for a slotted network, *buffering* and *aligning* work the same from the point of view of deflection[2] since the two synchronization techniques do not change the sequence of the packets arriving at a switch, thus packets deflected using one synchronization method will also be deflected using the other synchronization method.

With respect to performance, *Random Deflection*, since it is based on distance, yields a smaller average distance that a packet traverse to its destination. So, on average, protocols using *Random Deflection* are able to achieve larger throughput. In addition, since it is distance-based, it can be applied to irregular topologies where some row rings do not intersect some column rings. Protocols using *Random De-*

[1] In fact, *non-blocking* also has its applications where packets of variable lengths are used.

[2] although the requirements of buffer capacities are different.

flection are also more robust in the presence of node failure in which case, after the failing node is disconnected, the whole network connectivity can still be guaranteed.

Since *Vertical Deflection* is based on topology, it is not able to take the advantage of distance information of the node arrangement. In most cases, protocols using *Vertical Deflection* yield a smaller throughput[3]. It is also vulnerable during node failure, in which case, all the switches connected to the same row (or column) ring as the failing node is connected to can not send packets to each other, and other switches can not send **remote** packets to switches attached to the same column ring as the failing node is connected to. More connections are lost for protocols using *Vertical Deflection* if there is a node failure. Besides, the protocols can not work for topologies with some row rings not intersecting with some column rings.

As stated earlier, protocols using *Vertical Deflection* guarantee a maximum end-to-end delay of a packet once it is inserted into the network. The feature is useful in the sense that once a proper timeout interval is set, no packet will be discarded due to their **time-to-live** exceeds the limit. Protocols using *Random Deflection*, on the other hand, although achieving a shorter average end-to-end delay, fail to guarantee any bound and they are prone to packet loss. In table 6.1, assuming **time-to-live** is set so that no packet will be lost in SVDP, we give the rate of the packet loss in SRDP and CRDP for uniform traffic and square node arrangement when maximum throughput is achieved.

Many efforts have been put on the performance study of deflection protocols for mesh networks. Most of the studies rely on simulation models. To derive an analytical dependency for mesh protocols will be an interesting and challenging subject of future

[3]This can be seen by comparing the three tables 4.1, 4.3 and 5.1. The results of simulations are presented in the thesis given that a deflection takes place immediately after a packet's preferred port is occupied, i.e. the packet will not be buffered any longer for the availability of its preferred port. In addition, if assuming all deflections can be avoided by introducing a **LARGE** buffer, throughput of SVDP is at most half of SRDP for the shortest average paths of SVDP double their counterparts of SRDP for uniform traffic and square topologies.

Topologies	4×4	6×6	8×8
time-to-live (in *hops*)	14	31	62
Loss Rate of SRDP	4.4×10^{-3}	1.03×10^{-3}	1.4×10^{-5}
Loss Rate of CRDP	1.35×10^{-3}	2.56×10^{-4}	2×10^{-6}

Table 6.1: Packet loss rates of SRDP and CRDP with respect to the time-to-live bounds set by SVDP.

research. From the point of view of the protocols, since most deflection protocols share the same drawback that packets are not received in the order they are sent, an efficient deflection protocol that overcomes the drawback will be of interest and will put deflection protocols further to be considered for use in real world.

Bibliography

[Aca91] G. Albertengo et al. "Deflection Network: Principles, Implementation, Services", presented at CNR seminar "Broadband Communication Networks and Services", October 1991.

[Ame87] American National Standard. "FDDI Token ring Media Access Control (MAC)". ANSI X3.139, 1987.

[BC90] E. Borgonovo and E. Cadorin. "Locally-Optimal Routing in the Bidirectional Manhattan Network". In *IEEE INFOCOM*, pages 458–464, 1990.

[CA90] T.Y. Chung and D.P. Agrawal. "On the Network Characterization of and Optimal Broadcasting in the Manhattan Street Network". In *IEEE INFOCOM*, pages 465–472, 1990.

[CF93] I. Chlamtac and A. Fumagalli. "An Optical Switch Architecture for Manhattan Networks". *IEEE Journal on Selected Areas in Communications*, 11(4):550–559, May 1993.

[CL91a] A.K. Choudhury and V.O.K. Li. "Effect of Contention Resolution Rules on the Performance of Deflection Routing", pages 1706–1711, 1991.

[CL91b] A.K. Choudhury and V.O.K. Li. "Performance Analysis of Deflection Routing in the Manhattan Street Network", pages 1659–1665, 1991.

[CO89] I Cidon and Y. Ofek. "METARING – A ring with fairness and spatial reuse". Technical report, IBM T.J. Watson Research Center, 1989.

[CO90] I. Cidon and Y. Ofek. "A full-duplex ring with fairness and spatial reuse". In *IEEE INFOCOM*, pages 968–981, 1990.

[DTZ91] M. Decina, V. Trecordi, and G. Zanolini. "Throughput and packet loss in deflection routing multichannel metropolitan area networks", pages 1200–1208, 1991.

[DTZ92] M. Decina, V. Trecordi, and G. Zanolini. "Performance analysis of deflection routing in multichannel-metropolitan area networks". In *IEEE INFOCOM*, pages 2435–2443, 1992.

[Gbu93] P. Gburzynski. *Protocol modeling in SMURPH*. Prentice-Hall, 1993. (In preparation.).

[Hal92] F. Halsall. "*Data Communications, Computer Networks and Open Systems*". Addison-Wesley, 3 edition, 1992.

[IEE85a] IEEE Project 802.3. "ANSI/IEEE Std. 802.3: Carrier Sense Multiple Access with Collision Detection", 1985.

[IEE85b] IEEE Project 802.5. "ANSI/IEEE Std. 802.5: Token Ring Access Method and Physical Layer Specifications", 1985.

[IEE90] IEEE Project 802.6. "Proposed Standard, Distributed Queue Dual Bus Metropolitan Area Network". Doc. No. P802.6/D15, October 1990.

[Jai91] R. Jain. "*The Art of Computer Systems Performance Analysis*". John Wiley & Sons. Inc., 1991.

[JM93] A. Jajszczyk and H. T. Mouftah. "Photonic Packet Switching". *IEEE Communication Magazine*, 31(2):58–65, February 1993.

[Kle75] L. Kleinrock. "*Queuing Systems*". John Wiley & Sons Inc., 1975.

[KM93] R. Krishnan and N. F. Maxemchuk. "Is There Life Beyond Linear Topologies? A Comparison of DQDB and the Manhattan Street Network". In *Proceedings of IEEE INFOCOM*, pages 690–698, 1993.

[KY90] T. Kubo and K. Yoguchi. "HIGHWAY TRANSFER: A New Packet Forwarding Technique for Real-Time Applications". In *Proceedings of IEEE INFOCOM*, pages 403–408, 1990.

[Max85a] N. Maxemchuk. "Regular Mesh Topologies in Local and Metropolitan Area Networks". *AT&T Technical Journal*, 64(7):1659–1685, September 1985.

[Max85b] N. F. Maxemchuk. "The Manhattan Street Network". In *Proceedings of GLOBECOM*, pages 255–261, 1985.

[Max87] N. Maxemchuk. "Routing in the Manhattan Street Network". In *IEEE Transaction on Communications*, volume COM-35, pages 503–512, May 1987.

[Max88] N. Maxemchuk. "Distributed Clocks in Slotted Networks". In *IEEE INFOCOM*, pages 119–125, 1988.

[Max89] N. Maxemchuk. "Comparison of Deflection and Store-and-Forward Techniques in the Manhattan Street and Shuffle-Exchange Networks". In *IEEE INFOCOM*, pages 800–809, 1989.

[Max91] N. Maxemchuk. "Problems arising from deflection routing: Live-lock, Lock-out, Congestion and Message Reassembly". In *"High Capacity Local and Metropolitan Networks"*, pages 209–233. Springer Verlag, 1991.

[RB90] I. Rubin and J. E. Baker. "Media Access Control for High-Speed Local Area and Metropolitan Area Communications Networks". In *Proceedings of The IEEE*, volume 78, pages 168–203, January 1990.

[RL91] T. Robertazzi and A. Lazar. "Deflection Strategies for the Manhattan Street Network", pages 1652–1658, 1991.

[Ros89] F.E. Ross. "An Overview of FDDI: The Fiber Distributed Data Interface", *IEEE Journal on Selected Areas in Communications*, 7(7):1043–1051, September 1989.

[Tod92] T. D. Todd. "The Token Grid: Multidimensional Media Access For Local and Metropolitan Networks". In *Proceedings of IEEE INFOCOM*, pages 2415–2424, 1992.

Appendix A

Source Code

We provide the SMURPH source code for SRDP and SVDP. Since CRDP is very similar to SRDP, its code is not included.

A.1 Source Code for SRDP

The SMURPH source code of using appending bits to synchronize slots in SRDP is provided below. The code includes three parts: receiver process (RP), transmitter process (XP) and switch process (SP). Since there are two RP, one for each input port, and two XP, one for each output port. The following code just contains RP and XP for row rings. For column rings, the code is the same, except for the buffers and ports.

A.1.1 Receiver Process

```
RReceiver::perform {

  state Wait:
    /* Wait for the start of transmission of the next slot*/
    S->RIPort -> wait (BOT, Act);

  state Act:
```

```
    /*Detect the first bit of a slot*/
    S — RDelay.enter ( (SLOT *)ThePacket ):
    /*Shift the incoming slot into the buffer*/
    Timer — wait ( Delta1, Rec ):
    /*Delay the incoming slot for a number of bits*/

  state Rec:
    if ( (SLOT *)ThePacket — isMy () )
    /*If the switch is the destination of the slot*/
    S — RIPort — wait ( EOT, PRcv ):
    else
    /*Otherwise, wait for the end of the slot*/
    S — RIPort — wait ( EOT, Wait ):

  state PRcv:
    /*Receive the slot*/
    Client — receive ( (SLOT *)ThePacket, S—RIPort ):
    proceed Wait:

}
```

A.1.2 Transmitter Process

```
RXmitter::perform {

    SLOT *eSlot;

  state Init:
   /*Initialization, start to transmit empty slots*/
    S — RXMTE — wait ( NONEMPTY, Xmit_E);

  state Wait:
   /*Wait SIG(INSERT) signal*/
    S — RXDN — put ():
    S — RXMT — wait ( NONEMPTY, Rly ):

  state Xmit_E:
   /*Create empty slots*/
    eSlot = create SLOT;
    setFlag ( eSlot—Flags, EMPTY ):
    eSlot — fill(NONE, NONE, SSize+SFrame, SSize, NONE);
    S — ROBfr = *eSlot;
```

```
    delete(eSlot);
    proceed Rly;

  state Rly:
   /*Transmit empty slots*/
    S -- RXMT -- erase();
    S -- ROPort -- startTransfer( &(S -- ROBfr) );
    Timer -- wait ( ( SSize+SFrame ) * S -- ROPort -- get_TRate(),
          Xmit);

  state Xmit:
   /*Complete the transmission of header and payload*/
    S -- RDU -- put();
   /*Send a SIG(EX) and start appending excess bits*/
    Timer -- wait ( ALPHA * S -- ROPort -- get_TRate(), Done );
    S -- RSTOP -- wait ( NONEMPTY, Done );
   /*Wait for the SIG(STOP) signal */

  state Done:
   /*Either add the maximum number of excess bits */
   /*or be notified by SIG(STOP)*/
    S -- RSTOP -- erase();
    S -- ROPort -- stop();
    proceed Wait;
}
```

A.1.3 Switch Process

```
Switch::perform {

    SLOT *eSlot;

  state Init0:
   /*Decide a starting time*/
    Timer -- wait ( toss ( SSize+SFrame ) * ITUinBIT, Init );

  state Init:
   /*Notify transmitters to start inserting empty slots*/
    S -- RXMTE -- put();
    S -- CXMTE -- put();
    S -- RDU -- wait ( NONEMPTY, Wait );
    S -- CDU -- wait ( NONEMPTY, Wait );
```

```
state Wait:
  if ( S — RDelay . empty( ) ≠ 1 ) {
     maxRDly = LDLine ( S — RDelay );
      /*Find the number of bits stored in RDelay*/
     RDelt2 = ( d2 ≤ maxRDly
       ? 0
       : ( d2 — maxRDly ) );
   }
  else
  RDelt2 = 10 * d2 * ITUinBIT;

  if ( S — CDelay . empty( ) ≠ 1 ) {
     maxCDly = LDLine ( S — CDelay );
     CDelt2 = ( d2 ≤ maxCDly
       ? 0
       : ( d2 — maxCDly ) );
   }
  else
  CDelt2 = 10 * d2 * ITUinBIT;

  if ( ( RDelt2 == 0 ) || ( CDelt2 == 0 ) )
  proceed DELT2;
   else {
  Timer — wait ( ( RDelt2 ≤ CDelt2 ? RDelt2 : CDelt2 ), DELT2 );
  S — RXDN — wait (NONEMPTY, RDone);
  S — CXDN — wait (NONEMPTY, CDone);
   }

state DELT2:
 /*One buffer has reached the second threshold*/
  if ( S — ROPort — busy( ) )
  S — RSTOP — put( );
  if ( S — COPort — busy( ) )
  S — CSTOP — put( );

  S — RXDN — wait (NONEMPTY, RDone);
  S — CXDN — wait (NONEMPTY, CDone);

state RDone:
  S — CXDN — wait (NONEMPTY, Both);
```

```
state CDone:
  S — RXDN — wait (NONEMPTY, Both);

state Both:
 /*Route two slots*/
  S — RXDN — erase( );
  S — CXDN — erase( );
  S — RDU — erase( );
  S — CDU — erase( );

  route_r = ROW;
  route_c = COL;

  if ( ( S — RDelay . empty( ) ≠ 1 ) && ( LDLine ( S—RDelay ) ≥ d1 ) )
 /*Remove one slot from the buffer */
 S — RDelay . remove ( &(S — RSBfr) );
  else {
 /*Or insert a new empty slot*/
 eSlot = create SLOT;
 setFlag ( eSlot—Flags, EMPTY );
 eSlot — fill ( NONE, NONE, SSize+SFrame, SSize, NONE );
 S — RSBfr = *eSlot;
 delete(eSlot);
   }

  if ( ( S — CDelay . empty( ) ≠ 1 ) && ( LDLine ( S—CDelay) ≥ d1 ) )
 /*Remove one slot from the buffer*/
 S — CDelay . remove ( &(S — CSBfr) );
  else {
 /*Or insert a new empty slot*/
 eSlot = create SLOT;
 setFlag ( eSlot—Flags, EMPTY );
 eSlot — fill ( NONE, NONE, SSize+SFrame, SSize, NONE );
 S — CSBfr = *eSlot;
 delete(eSlot);
   }

  if ( S — RSBfr . isMy ( ) )
 /*Reuse received slot*/
 setFlag ( S — RSBfr . Flags, EMPTY );
```

```
 if ( S — CSBfr . isMy () )
/*Reuse received slot*/
setFlag ( S — CSBfr . Flags, EMPTY );

 if ( flagSet ( S — RSBfr . Flags, EMPTY ) )
if ( S — rReadyToTransmit () ) {
    /*Fill a slot*/
      S — RSBfr = S — RBuffer;
      S — RBuffer . release();
 }

 if ( flagSet ( S — CSBfr . Flags, EMPTY ) )
if ( S — cReadyToTransmit () ) {
   /*Fill a slot*/
      S — CSBfr = S — CBuffer;
      S — CBuffer . release();
 }

 route_r = S — route ( &(S — RSBfr) );
 route_c = S — route ( &(S — CSBfr) );
/*Route two slots*/

 switch ( route_r ) {
 case −1:   switch ( route_c ) {
          case −1:  route_r = ROW; route_c = COL; break;
          case COL: route_r = ROW; route_c = COL; break;
       case ROW: route_r = COL; route_c = ROW; break;
       }
       break;
 case COL:  switch ( route_c ) {
          case −1:  route_r = COL; route_c = ROW; break;
          case COL: /*Deflection happens*/
         switch ( toss(2) ){
            case 0: route_r = ROW; route_c = COL; break;
         case 1: route_r = COL; route_c = ROW; break;
         }
         break;
       case ROW: route_r = COL; route_c = ROW; break;
       }
       break;
 case ROW: switch ( route_c ) {
          case −1:  route_r = ROW; route_c = COL; break;
```

```
            case ROW: /*Deflection happens*/
           switch ( toss(2) ){
              case 0: route_r = ROW; route_c = COL; break;
           case 1: route_r = COL; route_c = ROW; break;
           }
           break;
         case COL: route_r = ROW; route_c = COL; break;
         }
         break;
    }

   /*Ready to transmit*/
    if ( route_r == ROW ) {
    S — ROBfr = S — RSBfr;
    S — COBfr = S — CSBfr;
   }
    else {
    S — ROBfr = S — CSBfr;
    S — COBfr = S — RSBfr;
   }

    S — RXMT — put ( );
    S — CXMT — put ( );
   /*Send SIG(INSERT)*/

    S — RDU — wait ( NONEMPTY, Wait );
    S — CDU — wait ( NONEMPTY, Wait );
}
```

A.2 Source Code for SVDP

The source code for SVDP includes four parts, two for receiver processes and two for transmitters.

A.2.1 Row Receiver

```
RReceiver::perform {

  state Wait:
    S—RIPort — wait (BOT, Act);
```

```
  state Act:
    S—RIBfr = *ThePacket;
    Timer — wait (recDelay, Rec);

  state Rec:
    if ( flagSet ( S—RIBfr.Flags, EMPTY ) ) {
   /*Discard empty slots*/
       ENTER_REQ;
       S—RIPort — wait(EOT, Wait);
     }
    else {
      if ( S—RIBfr . isMy( ) ) {
         ENTER_REQ;
         S—RIPort — wait (EOT, PRcv);
     }
       else {
         if ( (getC(ident(S)) == getC(S—RIBfr.Receiver)) &&
            getR(ident(S)) ≠ getR(S—RIBfr.Receiver)) {
         ENTER_CDB;
            S — SYN = 1;
         /*Send SIG(DEF)*/
               if ( S — ROPort — idle( ) )
                  S — XGAP.put( );
         /*Transmit an empty slot to remove gap*/
       }
         else
           ENTER_RB;
         S—RIPort — wait (EOT, Wait);
     }
   }

   state PRcv:
     Client — receive (&(S—RIBfr), ThePort);
     proceed Wait;

}
```

A.2.2 Column Receiver

```
CReceiver::perform {
```

```
  state Wait:
    S→CIPort → wait (BOT, Act);

  state Act:
    S→CIBfr = *ThePacket;
    Timer → wait (recDelay, Rec);

  state Rec:
    if ( flagSet ( S→CIBfr.Flags, EMPTY )) {
   /*Discard empty slots*/
        ENTER_CEQ;
        S→CIPort → wait (EOT, Wait);
      }
     else {

       if ( S→CIBfr . isMy() ) {
          ENTER_CEQ;
          S→CIPort → wait (EOT, PRcv);
     }
       else {
          if ( S → SYN == 1 ) {
      /*If receive SIG(DEF)*/
         ENTER_RDB;
       }
          else
             ENTER_CB;
          S→CIPort → wait (EOT, Wait);
   }

     }
  S → SYN = 0;

  state PRcv:
    Client → receive (&(S→CIBfr), ThePort);
    proceed Wait;
 }
```

A.2.3 Row Transmitter

```
RXmitter::perform {

  state Wait:
```

```
    S — XGAP . wait ( NONEMPTY, X_GAP );
    if ( ( S — RDB.empty( ) ≠ 1 ) || ( S — RB.empty( ) ≠ 1 ) )
      proceed Rel;
    else
   /*Insert new empty slots*/
      proceed Act;

  state Rel:
   /*Transmit slots in deflection buffer prior to delay buffer*/
    if ( S — RDB.empty( ) ≠ 1 )
   S — RDB.get ( &(S — ROBfr) );
    else
   S — RB.get ( &(S — ROBfr) );
    proceed Rly;

  state X_GAP:
   /*Remove gap by inserting empty slots*/
    S — XGAP . erase( );
    Slot_G = create Packet;
    Slot_G — fill (NONE, NONE, PFrame, PFrame, NONE );
    setFlag ( Slot_G — Flags, EMPTY );
    S — ROPort — transmit ( Slot_G, X_GAP_Done);

  state X_GAP_Done:
    S — GAP_F = 0;
    S — ROPort — stop( );
    proceed Wait;

  state Act:
    if ( S — rReadyToTransmit ( ) )
   /*Insert a new slot with non—column payload*/
      S — ROPort — transmit (S—RBuffer, TDone);
    else {
   /*Insert an empty slot*/
      S — REQ.get ( &(S — ROBfr) );
      setFlag ( S — ROBfr . Flags, EMPTY );
      S — ROBfr.Sender = S — ROBfr.Receiver = NONE;
      proceed Rly;
     }

  state Rly:
    S—ROPort — transmit (&(S—ROBfr), Done);
```

```
  state TDone:
    S→ROPort → stop ();
    S→RBuffer.release();
    proceed Wait;

  state Done:
    S→ROPort → stop ();
    proceed Wait;
}
```

A.2.4 Column Transmitter

```
CXmitter::perform {

  state Wait:
    if ( ( S → CDB.empty() ≠ 1 ) || ( S → CB.empty() ≠ 1 ) )
      proceed Rel;
    else
      proceed Act;

  state Rel:
    if ( S → CDB.empty() ≠ 1 )
   S → CDB.get ( &(S → COBfr) );
    else
   S → CB.get ( &(S → COBfr) );
    proceed Rly;

  state Act:
    if ( S → cReadyToTransmit () )
      S→COPort → transmit(&(S→CBuffer), TDone);
    else {
      S → CEQ.get ( &(S → COBfr) );
      setFlag ( S → COBfr . Flags, EMPTY );
      S → COBfr.Sender = S → COBfr.Receiver = NONE;
      proceed Rly;
     }

  state Rly:
    S→COPort → transmit (&(S→COBfr), Done);

  state TDone:
```

```
    S—COPort—stop();
    S—CBuffer.release();
    proceed Wait;

  state Done:
    S—COPort — stop ();
    proceed Wait;
}
```